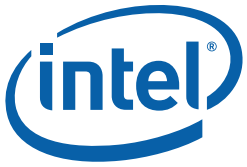


Intel® MPI Library for Linux* Reference Manual

Contents

Disclaimer and Legal Notices

Revision History

Document Number	Revision Number	Description	Revision Date
315399-001	3.1 Beta	Some new options and variables were added, three new sections “Statistics Gathering Mode”, “Unified Memory Management”, and “Integration into Eclipse* PTP” were created	/07/10/2007
315399-002	3.1	New names of variables were added, new section “Processor Information Utility” was added. Updated and reviewed for style	/10/02/2007

1 About this Document

This *Reference Manual* provides you with a complete command and tuning reference for the Intel MPI Library.

The Intel® MPI Library enables you to deliver maximum end user performance as soon as new processor and interconnect technology become available. The Intel® MPI Library is a multi-fabric message passing library that implements the Message Passing Interface, v2 (MPI-2) specification. It provides a standard library across Intel® platforms that:

- Delivers best in class performance for enterprise, divisional, departmental and workgroup high performance computing. Intel® MPI Library focuses on making applications perform better on IA based clusters.
- Enables to adopt MPI-2 functions as their needs dictate.

The Intel® MPI Library enables you to change or upgrade interconnects as the technology becomes available without major changes to the software or to the operating environment.

The library is included in the following kits:

- *Intel® MPI Library Runtime Environment* (RTO) has the tools you need to run programs including MPD daemons and supporting utilities, shared (.so) libraries, and documentation.
- *Intel® MPI Library Development Kit* (SDK) includes all of the Runtime Environment components plus compilation tools including compiler commands such as `mpicc`, include files and modules, static (.a) libraries, debug libraries, trace libraries, and test codes.

1.1 Intended Audience

This *Reference Manual* helps an experienced user understand full functionality of the Intel® MPI Library and get the best possible application performance.

1.2 Using Doc Type Field

This *Reference Manual* contains the following sections

Table 1-1 Document Organization

Section	Description
Section 1 About this Document	Section 1 introduces this document
Section 2 Command Reference	Section 2 describes options and variables for compiler commands, job startup commands and MPD daemon commands as well
Section 3 Tuning Reference	Section 3 describes environment variables used to influence program behavior and performance at run time
Section 4 Statistics Gathering Mode	Section 4 describes statistics about usage of MPI communication operations
Section 5 Unified Memory Management	Section 5 describes unified Intel memory management subsystem (`i_malloc`)

Section 6 Integration into Eclipse* PTP	Section 6 describes necessary steps for integration into Eclipse* Parallel Tools Platform

1.3 Conventions and Symbols

The following conventions are used in this document.

Table 1-2 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
NEW_VAR_NAME (OLD_VAR_NAME)	New variable name and old (deprecated) name variable name in parentheses

1.4 Related Information

The following related documents that might be useful to the user.

Product Web Site

Intel® MPI Library support

Intel® Cluster Tools Products

Intel® Software Development Products

2 Command Reference

2.1 Compiler Commands

(SDK only)

The following table lists available MPI compiler commands and the underlying compilers, compiler families, languages, and application binary interfaces (ABIs) that they support.

Table 2-1 Intel® MPI Library compiler drivers

Compiler command	Default compiler	Supported language(s)	Supported ABI(s)
Common compilers			
mpicc	gcc, cc	C	32/64 bit
mpicxx	g++	C/C++	32/64 bit
mpifc	gfortran	Fortran77/Fortran 95	32/64 bit
GNU* compilers versions 3 and higher			
mpigcc	gcc	C	32/64 bit
mpigxx	g++	C/C++	32/64 bit
mpif77	g77	Fortran 77	32/64 bit
mpif90	gfortran	Fortran 95	32/64 bit
Intel® Fortran, C++ compilers versions 8.0, 8.1, 9.0, 9.1, 10.0 or 10.1			
mpiicc	icc	C	32/64 bit
mpiicpc	icpc	C++	32/64 bit
mpiifort	ifort	Fortran77/Fortran 95	32/64 bit

NOTE:

- Compiler commands are available only in the Intel® MPI Library Development Kit.
- Compiler commands are in the *<installdir>*/bin directory. For Intel® 64 architecture in a 64-bit-enabled compiler, commands are in the *<installdir>*/bin64 directory and 32-bit compiler commands are in the *<installdir>*/bin directory.
- Ensure that the corresponding underlying compilers (32-bit or 64-bit, as appropriate) are already in your PATH.
- To port existing MPI-enabled applications to the Intel® MPI Library, recompile all sources.
- To display mini-help of a compiler command, execute it without any parameters.

2.1.1 Compiler Command Options

-mt_mpi

Use this option to link the thread safe version of the Intel® MPI library at the following levels: MPI_THREAD_FUNNELED, MPI_THREAD_SERIALIZED, or MPI_THREAD_MULTIPLE.

NOTE: If you specify either the -openmp or the -parallel options for the Intel® C Compiler, the thread safe version of the library is used.

NOTE: If you specify one of the following options for the Intel® Fortran Compiler, the thread safe version of the library is used:

- -openmp
- -parallel
- -threads
- -reentrancy
- -reentrancy threaded

-static_mpi

Use this option to link the Intel® MPI library statically. This option does not affect the default linkage method for other libraries.

-profile=<*profile_name*>

Use this option to specify an MPI profiling library. The profiling library is selected using one of the following methods:

- Through the configuration file <*profile_name*>.conf located in the <*installdir*>/etc directory (<*installdir*>/etc64 directory for Intel® 64 architecture in 64-bit mode). See Profiles for details.
- In the absence of the respective configuration file, by linking the library lib<*profile_name*>.so or lib<*profile_name*>.a located in the same directory as the Intel® MPI library.

-t or –trace

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

Use the -t=log or -trace=log option to link the resulting executable against the logging Intel® MPI and the Intel® Trace Collector libraries. The logging libraries trace internal Intel® MPI library states in addition to the usual MPI function calls.

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

-check

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

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

-dynamic_log

Use this option in combination with the -t option to link in the Intel® Trace Collector library dynamically. This option does not affect the default linkage method for other libraries.

Include $VT_ROOT/slib in the LD_LIBRARY_PATH environment variable to run the resulting programs.

-g

Use this option to compile a program in debug mode and link the resulting executable against the debugging version of the Intel® MPI library. See *Environment variables*, I_MPI_DEBUG for information on how to use additional debugging features with the -g builds.

-O

Use this option to enable optimization.

-echo

Use this option to display everything that the command script does.

-show

Use this option to learn how the underlying compiler is invoked. For example, use the following command to see the required compiler flags and options:

```
$ mpicc -show -c test.c
```

Use the following command to see the required link flags, options, and libraries:

```
$ mpicc -show -o a.out test.o
```

This is particularly useful for determining the command line for a complex build procedure that directly uses the underlying compilers.

-{cc,cxx,fc,f77,f90}=*<compiler>*

Use this option to select the underlying compiler.

For example, use the following command to select the Intel® C++ Compiler:

```
$ mpicc -cc=icc -c test.c
```

Make sure icc is in your path. Alternatively, you can specify the full path to the compiler.

-gcc-version=*<nnn>*

Use this option for compiler drivers mpicxx and mpiicpc when linking an application running in a particular GNU* C++ environment. The valid *<nnn>* values are:

<nnn> value	GNU* C++ version
320	3.2.x
330	3.3.x

340	3.4.x
400	4.0.x
410	4.1.x

By default, the library compatible with the detected version of the GNU* C++ compiler is used. Do not use this option if the GNU* C++ version is older than 3.2.

-compchk

Use this option to enable compiler setup checks. In this case each compiler command performs checks to ensure that the appropriate underlying compiler is set up correctly.

2.1.2 Configuration Files

You can create compiler configuration files using the following file naming convention:

<installdir>/etc/mpi*<compiler>*-*<name>*.conf

<installdir>/etc64/mpi*<compiler>*-*<name>*.conf for Intel® 64 architecture in 64-bit mode

where:

<compiler> = {cc,cxx,f77,f90}, depending on the language being compiled

<name> = name of underlying compiler with spaces replaced by hyphens

For example, the *<name>* value for cc -64 is cc--64

Source this file, if it exists, prior to compiling or linking to enable changes to the environment on a per-compiler-command basis.

2.1.3 Profiles

You can select a profile library through the -profile option of the Intel® MPI Library compiler drivers. The profile files are located in the *<installdir>*/etc directory (*<installdir>*/etc64 directory for Intel® 64 architecture in 64-bit mode). Intel® MPI Library comes with several predefined profiles for Intel® Trace Collector:

<installdir>/etc/vt.conf - regular Intel® Trace Collector library

<installdir>/etc/vtfs.conf - fail-safe Intel® Trace Collector library

<installdir>/etc/vtmc.conf – correctness checking Intel® Trace Collector library

You can also create your own profile as *<profile_name>*.conf

The following variables can be defined there:

PROFILE_PRELIB - libraries (and paths) to include before the Intel® MPI library

PROFILE_POSTLIB - libraries to include after the Intel® MPI library

PROFILE_INCPATHS - C preprocessor arguments for any include files

For instance, create a file /myprof.conf with the following lines:

PROFILE_PRELIB="-L*<path_to_myprof>*/lib -lmyprof"

```
PROFILE_INCPATHS="-I<paths_to_myprof>/include"
```

Use the command-line argument `-profile=myprof` for the relevant compile driver to select this new profile.

2.1.4 Environment Variables

I_MPI_{CC,CXX,FC,F77,F90}_PROFILE (MPI{CC,CXX,FC,F77,F90}_PROFILE)

Specify a default profiling library.

Syntax

`I_MPI_{CC,CXX,FC,F77,F90}_PROFILE=<profile_name>`

Deprecated Syntax

`MPI{CC,CXX,FC,F77,F90}_PROFILE=<profile_name>`

Arguments

<profile_name>	Specify a default profiling library

Description

Set this variable to select a specific MPI profiling library to be used by default. This has the same effect as if `-profile=<profile_name>` were used as an argument to `mpicc` or another Intel® MPI Library compiler driver.

I_MPI_TRACE_PROFILE

Specify a default profile for the `-trace` option.

Syntax

`I_MPI_TRACE_PROFILE=<profile_name>`

Arguments

<profile_name>	Specify a tracing profile name. The default value is `vt`

Description

Set this variable to select a specific MPI profiling library to be used with `-trace` option to `mpicc` or another Intel® MPI Library compiler driver.

The `I_MPI_{CC,CXX,F77,F90}_PROFILE` environment variable overrides `I_MPI_TRACE_PROFILE`.

I_MPI_CHECK_PROFILE

Specify a default profile for the `-check` option.

Syntax

`I_MPI_CHECK_PROFILE=<profile_name>`

Arguments

<profile_name>	Specify a checking profile name. The default value is `vtmc`

Description

Set this variable to select a specific MPI checking library to be used with the `-check` option to `mpicc` or another Intel® MPI Library compiler driver.

The `I_MPI_{CC,CXX,F77,F90}_PROFILE` environment variable overrides `I_MPI_CHECK_PROFILE`.

I_MPI_{CC,CXX,FC,F77,F90} (MPICH_{CC,CXX,FC,F77,F90})

Set the path/name of the underlying compiler to be used.

Syntax

`I_MPI_{CC,CXX,FC,F77,F90}=<compiler>`

Deprecated Syntax

`MPICH_{CC,CXX,FC,F77,F90}=<compiler>`

Arguments

<compiler>	Specify the full path/name of compiler to be used

Description

Set this variable to select a specific compiler to be used. Specify the full path to the compiler if it is not located in the search path.

NOTE: Some compilers may require additional command line options.

CFLAGS

Add additional `CFLAGS` to be used in compile and/or link steps.

Syntax

`CFLAGS=<flags>`

Arguments

<flags>	Specify the flags to be used in compile and/or link steps

Description

Set this variable to modify compiler behavior.

LDFLAGS

Set additional `LDFLAGS` to be used in the link step.

Syntax

`CFLAGS=<flags>`

Arguments

<flags>	Specify the flags to be used in the link step

Description

Set this variable to modify linker behavior.

I_MPI_ROOT

Set Intel® MPI Library installation directory path.

Syntax

`I_MPI_ROOT=<path>`

Arguments

<path>	Specify the installation directory of the Intel® MPI Library

Description

Set this variable to specify installation directory of the Intel® MPI Library.

VT_ROOT

Set Intel® Trace Collector installation directory path.

Syntax

`VT_ROOT=<path>`

Arguments

<path>	Specify the installation directory of the Intel® Trace Collector

Description

Set this variable to specify installation directory of the Intel® Trace Collector.

I_MPI_COMPILER_CONFIG_DIR

Set the location of the compiler configuration files.

Syntax

`I_MPI_COMPILER_CONFIG_DIR=<path>`

Arguments

<path>	Specify the location of the compiler configuration files. The default is *<installdir>*`/etc` or *<installdir>*`/etc64`

Description

Set this variable to change the default location of the compiler configuration files.

2.2 Job Startup Commands

mpiexec

Syntax

`mpiexec` *<g-options> <l-options> <executable>*

or

`mpiexec` *<g-options> <l-options> <executable> : *

<l-options> <executable>

or

`mpiexec -configfile` *<file>*

Arguments

<g-options>	Global options that apply to all MPI processes
<l-options>	Local options that apply to a single arg-set
<executable>	`./a.out` or `path/name` of the executable file
<file>	File with command-line options

Description

In the first command-line syntax, run the specified *<executable>* with the specified options. All global and/or local options apply to all MPI processes. A single arg-set is assumed. For example, the following command executes `a.out` over the specified *<# of processes>*:

```
$ mpiexec -n <# of processes> ./a.out
```

In the second command-line syntax, divide the command line into multiple arg-sets, separated by colon characters. All the global options apply to all MPI processes, but the various local options and *<executable>* can be specified separately for each arg-set. For example, the following command would run each given executable on a different host:

```
$ mpiexec -n 2 -host host1 ./a.out : \
          -n 2 -host host2 ./b.out
```

In the third command-line syntax, read the command line from specified *<file>*. For a command with a single arg-set, the entire command should be specified on a single line in *<file>*. For a command with multiple arg-sets, each arg-set should be specified on a single, separate line in *<file>*. Global options should always appear at the beginning of the first line in *<file>*.

MPD daemons must already be running in order for `mpiexec` to succeed.

NOTE: If "." is not in the path on all nodes in the cluster, specify *<executable>* as `./a.out` rather than `a.out`.

2.2.1 Global Options

-version or -V

Use this option to display Intel® MPI Library version information.

-tune

Use this option to optimize the Intel® MPI Library performance using the data collected by the `mpitune` utility. See Automatic Tuning Utility for more details.

-rdma

Use this option to select RDMA-capable network fabrics. This option is equivalent to the `-genv I_MPI_DEVICE rdma` option.

-RDMA

Use this option to select RDMA-capable network fabrics. The application will fail if no fabric found. This option is equivalent to `-genv I_MPI_DEVICE rdma -genv I_MPI_FALLBACK_DEVICE 0`.

-gm

Use this option to select Myrinet* GM* network fabric. This option is equivalent to `-genv I_MPI_DEVICE rdma:GmHca0`.

-GM

Use this option to select Myrinet GM network fabric. The application will fail if no fabric found. This option is equivalent to `-genv I_MPI_DEVICE rdma:GmHca0 -genv I_MPI_FALLBACK_DEVICE 0`.

-mx

Use this option to select Myrinet MX* network fabric. This option is equivalent to `-genv-I_MPI_DEVICE rdma:mx -genv I_MPI_RDMA_TINY_PACKET 1`.

-MX

Use this option to select Myrinet MX network fabric. The application will fail if no fabric found. This option is equivalent to `–genv I_MPI_DEVICE rdma:mx -genv I_MPI_RDMA_TINY_PACKET 1 -genv I_MPI_FALLBACK_DEVICE 0`.

-nolocal

Use this option to avoid running *<executable>* on the host where `mpiexec` is launched. This option is useful, for example, on clusters that deploy a dedicated master node for starting the MPI jobs, and a set of compute nodes for running the actual MPI processes.

-perhost <# of processes>

Use this option to place the indicated number of consecutive MPI processes on every host in group round robin fashion. The number of processes to start is controlled by the option `–n` as usual.

The `mpiexec` command controls how the ranks of the processes are allocated to the nodes in the cluster. By default, `mpiexec` uses group round-robin assignment of ranks to nodes, putting consecutive MPI processes on all processor cores.

In order to change this default behavior, set the number of processes per host using the `–perhost` option, and set the total number of processes by using the `–n` option (see *Local Options*). Then the first *<# of processes>* indicated by the `–perhost` option are executed on the first host, the next *<# of processes>* are executed on the next host, and so on.

-rr

Use this option to place consecutive MPI processes onto different host in round robin fashion. This option is equivalent to the `–perhost 1`.

-grr <# of processes>

Use this option to place the indicated number of consecutive MPI processes on every host in group round robin fashion. This option is equivalent to the `–perhost` *<# of processes>*.

-ppn <# of processes>

Use this option to place the indicated number of consecutive MPI processes on every host in group round robin fashion. This option is equivalent to the `–perhost` *<# of processes>*.

-machinefile <machine file>

Use this option to control the process placement through the *<machine file>*. The number of processes to start is controlled by the option `–n` as usual.

A machine file is a list of fully qualified or short host names, one name per line. Blank lines and lines that start with `'#'` as the first character are ignored.

By repeating a host name you will place additional processes on this host. You can also use the following format to avoid repetition of the same host name: *<host name>:<number of processes>*. For example, the following machine file:

```
host1
```

```
host1
host2
host2
host3
```

is equivalent to:

```
host1:2
host2:2
host3
```

-genv <ENVVAR> <value>

Use this option to set the <ENVVAR> environment variable to the specified <value> for all MPI processes.

-genvnone

Use this option to suppress the propagation of any environment variables to any MPI processes. The default is to propagate the entire environment from which mpiexec was called.

-g<l-option>

Use this option to apply the named local option <l-option> globally. See Local Options for a list of all local options.

(SDK only) -trace [<profiling_library>]

Use this option to profile your MPI application using the indicated <profiling_library>. If the <profiling_library> is not mentioned, the default profiling library libVT.so will be used.

Set the I_MPI_JOB_TRACE_LIBS environment variable to override the default profiling library.

NOTE: It is not necessary to link your application against the profiling library before execution.

(SDK only) -check [<checking_library>]

Use this option to check your MPI application using the indicated <checking_library>. If the <checking_library> is not mentioned, the default checking library libVTmc.so will be used.

Set the I_MPI_JOB_CHECK_LIBS environment variable to override the default checking library.

NOTE: It is not necessary to link your application against the checking library before execution.

-tv

Use this option to run <executable> under the TotalView* debugger. For example:

```
$ mpiexec -tv -n <# of processes> <executable>
```

See Environment Variables for information on how to select the TotalView* executable file.

-tvsu

Use this option to run <executable> for later attachment with the TotalView* debugger. For example:

```
$ mpiexec -tvsu -n <# of processes> <executable>
```

NOTE: To debug the running Intel® MPI job, attach the TotalView* to the Python instance that is running the `mpiexec` script.

-idb

Use this option to run *<executable>* under the Intel® Debugger. For example:

```
$ mpiexec -idb -n <# of processes> <executable>
```

Include the installation path of the Intel® Debugger in the `IDB_HOME` environment variable.

-idba <jobid>

Use this option to attach the Intel® Debugger to the existing *<jobid>*. For example:

```
$ mpiexec -idba <jobid>
```

-gdb

Use this option to run *<executable>* under the GNU* debugger. For example:

```
$ mpiexec -gdb -n <# of processes> <executable>
```

-gdba <jobid>

Use this option to attach the GNU* debugger to the existing *<jobid>*. For example:

```
$ mpiexec -gdba <jobid>
```

-a <alias>

Use this option to assign *<alias>* to the job.

-ordered-output

Use this option to avoid intermingling of data output by the MPI processes.

This option affects both standard output and error streams. For it to work, the last line output by each process must end with the end-of-line (‘\n’) character. Otherwise the application may stop responding.

-m

Use this option to merge output lines.

-l

Use this option to insert the MPI process rank at the beginning of all lines written to standard output.

-s <spec>

Use this option to direct standard input to the specified MPI processes.

<spec>	Define MPI process ranks
`all`	Use all processes
<l>,<m>,<n>	Specify an exact list and use processes *<l>*, *<m>* and *<n>* only. Default value is zero
<k>,<l>-<m>,<n>	Specify a range and use processes *<k>*, *<l>* through *<m>*, and *<n>*

-noconf

Use this option to disable processing of the `mpiexec` configuration files described in *Configuration Files*.

-ifhn <*interface/hostname*>

Use this option to specify the network interface for communication with the local MPD daemon. `<interface/hostname>` should be an IP address or a hostname associated with the alternative network interface.

-ecfn <*filename*>

Use this option to output xml exit codes to the file `<filename>`.

2.2.2 Local Options

-n <*# of processes*> or -np <*# of processes*>

Use this option to set the number of MPI processes to run the current arg-set.

-env <*ENVVAR*> <*value*>

Use this option to set the `<ENVVAR>` environment variable to specified `<value>` for all MPI processes in the current arg-set.

-envall

Use this option to propagate all environment variables in the current environment.

-envnone

Use this option to suppress the propagation of any environment variables to the MPI processes in the current arg-set. The default value is to propagate the entire environment from which `mpiexec` is called. This option does not affect the variables indicated in the `–[g]env` option(s).

-envlist <*list of env var names*>

Use this option to pass a list of environment variables with their current values.

-host <*nodename*>

Use this option to specify particular `<nodename>` on which the MPI processes in the current arg-set are to be run. For example, the following will run the executable `a.out` on host `host1` only:

```
$ mpiexec –n 2 –host host1 ./a.out
```

-configfile <*filename*>

Use this option to specify the file `<filename>` that contains command-line options. For example, the configuration file contains the following commands to run the executables `a.out` and `b.out` using the `rdssm` device over `host1` and `host2` respectively:

```
-host host1 -env I_MPI_DEBUG 2 -env I_MPI_DEVICE rdssm -n 2 ./a.out

-host host2 -env I_MPI_DEBUG 2 -env I_MPI_DEVICE rdssm -n 2 ./b.out
```

To launch the MPI application according to the above parameters, use:

```
mpiexec –configfile <filename>
```

-path <*directory*>

Use this option to specify the path to <*executable*> that is to be run in the current arg-set.

-wdir <*directory*>

Use this option to specify the working directory in which <*executable*> is to be run in the current arg-set.

-umask <*umask*>

Use this option to perform the 'umask <*umask*>' command for the remote process.

2.2.3 Configuration Files

The mpiexec configuration files specify the default options applied to all mpiexec commands.

If any of these files exist, their contents are prepended to the command-line options for mpiexec in the following order:

1. System-wide <*installdir*>/etc/mpiexec.conf. For Intel® 64 architecture in 64-bit mode the default location of the configuration file is the <*installdir*>/etc64 directory and in 32-bit mode is the <*installdir*>/etc directory.
2. User-specific $HOME/.mpiexec.conf
3. Session-specific $PWD/mpiexec.conf

You can override these files by defining environment variables and using command line options. You can skip these configuration files by using the mpiexec –noconf option.

You can create or modify these files. They contain mpiexec command-line options. Blank lines and lines that start with '#' are ignored. For example, to specify a default device, add the following line to the respective mpiexec.conf file:
-genv I_MPI_DEVICE <*device*>

2.2.4 Environment Variables

I_MPI_DEVICE

Select the particular network fabric to be used.

Syntax

I_MPI_DEVICE=<*device*>[:<*provider*>]

Arguments

<*device*>	One of {sock, shm, ssm}
sock	Sockets
shm	Shared-memory only (no sockets)
ssm	Combined sockets + shared memory (for clusters with SMP nodes)

<*device*>	One of {rdma, rdssm}
<*provider*>	Optional DAPL* provider name
rdma	RDMA-capable network fabrics including InfiniBand*, Myrinet* (via

	DAPL*)
rdssm	Combined sockets + shared memory + DAPL* (for clusters with SMP nodes and RDMA-capable network fabrics)

Description

Set this variable to select a specific fabric combination. If the I_MPI_DEVICE variable is not defined, Intel® MPI Library selects the most appropriate fabric combination automatically.

For example, to select shared-memory as the chosen fabric, use the following command:

```
$ mpiexec -n <# of processes> -env I_MPI_DEVICE shm <executable>
```

Use the <provider> specification only for the {rdma,rdssm} devices.

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

```
$ mpiexec -n <# of processes> -env I_MPI_DEVICE rdssm:OpenIB-cma <executable>
```

For these devices, if <provider> is not specified, the first DAPL* provider in /etc/dat.conf is used. If <provider> is set to none, the rdssm device establishes sockets connections between the nodes without trying to establish DAPL* connections first.

NOTE: If you build the MPI program using mpicc -g, the debug-enabled version of the library is used.

NOTE: If you build the MPI program using mpicc -t=log, the trace-enabled version of the library is used.

NOTE: The debug-enabled and trace-enabled versions of the library are only available when you use the Intel® MPI Library Development Kit.

I_MPI_FALLBACK_DEVICE

Set this environment variable to enable fallback to the available fabric. It is valid only for rdssm and rdma modes.

Syntax

I_MPI_FALLBACK_DEVICE=<arg>

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Fall back to the shared memory and/or socket fabrics if initialization of the DAPL* fabric fails. This is the default value
disable | no | off | 0	Terminate the job if the fabric selected by the I_MPI_DEVICE environment variable cannot be initialized

Description

Set this variable to control fallback to the available fabric.

If I_MPI_FALLBACK_DEVICE is set to enable and an attempt to initialize the specified fabric fails, the library falls back to the shared memory and/or socket fabrics. The exact combination of devices depends on the number of processes started per node. For example, the library can use only sockets or a mix of sockets plus shared memory (ssm) per node. This device ensures that the job will run but it may not provide the highest possible performance for the given cluster configuration.

If I_MPI_FALLBACK_DEVICE is set to disable and an attempt to initialize the specified fabric fails, the library terminates the MPI job.

I_MPI_DEBUG

Print out debugging information when an MPI program starts running.

Syntax

I_MPI_DEBUG=<*level*>

Arguments

<*level*>	Indicate level of debug information provided
0	Print no debugging information. This is default value
1	Output verbose error diagnostics
2	Confirm which I_MPI_DEVICE was used
3	Output effective MPI rank, pid and node mapping table
4	Print process pinning information
5	Print Intel MPI specific environment variables
> 5	Add extra levels of debug information

Description

Set this variable to control the output of the debugging information.

The I_MPI_DEBUG mechanism extends the MPICH2* MPICH_DBG_OUTPUT debug mechanism by overriding the current value and setting MPICH_DBG_OUTPUT=stdout.

Each printed line has the following format:

```
[Identifier] Debug message
```

Where

Debug message is a the message content,

Identifier is an identifier of MPI process that produced the message.

The identifier is a MPI process rank in case if <*level*> is an unsigned number or a rank#pid@hostname tuple in the case if a '+' sign is added in front of the <*level*> number. Here rank is a MPI process rank, pid is an UNIX process id, and hostname is a host name as defined at the process launch time. For example, the command:

```
$ mpiexec -n 1 -env I_MPI_DEBUG 2 ./a.out
```

produces the message:

```
[0] MPI startup(): shared memory data transfer mode
```

the command:

```
$ mpiexec -n 1 -env I_MPI_DEBUG +2 ./a.out
```

produces the message:

```
[0#1986@mpicluster001] MPI startup(): shared memory data transfer mode
```

NOTE: Compiling with `mpicc -g` causes considerable amount of additional debug information to be printed.

I_MPI_PERHOST

Define default settings for the `-perhost` option in the `mpiexec` command.

Syntax

`I_MPI_PERHOST=<value>`

Arguments

<value>	Define a value that is used for the `-perhost` option by default
`integer > 0`	Exact value for the option
`all`	All logical CPUs on a node
`allcores`	All cores (physical CPUs) on a node

Description

Set this variable to define the default setting for the `-perhost` option. If `-perhost` is explicitly presented in the command line `I_MPI_PERHOST` has no effect. The `-perhost` option is assumed with its value if `I_MPI_PERHOST` is defined.

NOTE: `I_MPI_PERHOST` is incompatible with the `mpiexec -host` option. The `I_MPI_PERHOST` environment variable will be ignored in this case.

I_MPI_NETMASK

Choose the network interface for MPI communication over sockets.

Syntax

`I_MPI_NETMASK=<arg>`

Arguments

<arg>	String parameter
<interface_name>	Name of the network interface, usually the UNIX* driver name followed by the unit number
<network_address>	Network address. The trailing zero bits imply netmask
<network_address/netmask>	Network address. The *<netmask>* value specifies the netmask length
<list of interfaces>	A colon separated list of network addresses and interface names

Description

Set this variable to choose the network interface for MPI communication over sockets. If a list of interfaces is specified, the first available interface on the node is used for communication.

Examples:

1. Use the following settings to select particular network interface for socket communications:
   `I_MPI_NETMASK=ib0`
2. Use the following settings to select particular network for socket communications. This setting implies the `255.255.0.0` netmask:
   `I_MPI_NETMASK=192.169.0.0`

3. Use the following settings to select particular network for socket communications with netmask set explicitly:
   I_MPI_NETMASK=192.169.0.0/24
4. Use the following settings to select specified network interfaces for socket communications:
   I_MPI_NETMASK=192.169.0.5/24:ib0:192.169.0.0

(SDK only) I_MPI_JOB_TRACE_LIBS

(MPIEXEC_TRACE_LIBS)

Choose the default libraries to preload through the -trace option.

Syntax

I_MPI_JOB_TRACE_LIBS=<*arg*>

Deprecated Syntax

MPIEXEC_TRACE_LIBS=<*arg*>

Arguments

<*arg*>	String parameter
a list of libraries	String containing the list of libraries to be preloaded. If the number of libraries is more than one then they must be blank separated

Description

Set this variable to allow choosing another library for preloading by the -trace option.

(SDK only) I_MPI_JOB_CHECK_LIBS

Choose the default libraries to preload through the -check option.

Syntax

I_MPI_JOB_CHECK_LIBS=<*arg*>

Arguments

<*arg*>	String parameter
a list of libraries	String containing the list of libraries to be preloaded. If the number of libraries is more than one then they must be blank separated

Description

Set this variable to allow choosing another library for preloading by the -check option.

I_MPI_JOB_STARTUP_TIMEOUT

Set the mpiexec job startup timeout.

Syntax

I_MPI_JOB_STARTUP_TIMEOUT=<*timeout*>

Arguments

<*timeout*>	Define mpiexec job startup timeout period in seconds
≥0	The default timeout value is 20 seconds

Description

Set this variable to make mpiexec wait the job to start in <timeout> seconds after its launch. The <timeout> value should be greater than zero. Otherwise, the variable setting is ignored and a warning message is printed. Setting this variable may make sense on large clusters with a lot of nodes where the job startup time may exceed the default value.

NOTE: Set the I_MPI_JOB_STARTUP_TIMEOUT variable in the shell environment before executing the mpiexec command. Do not use the -genv or -env options for setting the <timeout> value. Those options are used only for passing variables to the MPI process environment.

I_MPI_JOB_TIMEOUT

(MPIEXEC_TIMEOUT)

Set the mpiexec timeout.

Syntax

I_MPI_JOB_TIMEOUT=<timeout>

Deprecated Syntax

MPIEXEC_TIMEOUT=<timeout>

Arguments

<timeout>	Define mpiexec timeout period in seconds
≥0	The default timeout value is zero corresponding to no timeout

Description

Set this variable to make mpiexec terminate the job in <timeout> seconds after its launch. The <timeout> value should be greater than zero. Otherwise, the variable setting is ignored.

NOTE: Set the I_MPI_JOB_TIMEOUT variable in the shell environment before executing the mpiexec command. Do not use the -genv or -env options for setting the <timeout> value. Those options are used only for passing variables to the MPI process environment.

I_MPI_JOB_TIMEOUT_SIGNAL

(MPIEXEC_TIMEOUT_SIGNAL)

Define a signal number.

Syntax

I_MPI_JOB_TIMEOUT_SIGNAL=<number>

Deprecated Syntax

MPIEXEC_TIMEOUT_SIGNAL=<number>

Arguments

<number>	A number of the signal
> 0	The default value is 9 (SIGKILL)

Description

Define a signal number for killing the processes of the task if the timeout pointed to by I_MPI_JOB_TIMEOUT is over. If a wrong signal number not supported by the system is set, mpiexec prints a warning message and continues task termination using the default signal number 9 (SIGKILL).

I_MPI_JOB_SIGNAL_PROPAGATION

(MPIEXEC_SIGNAL_PROPAGATION)

Control signal propagation.

Syntax

I_MPI_JOB_SIGNAL_PROPAGATION=*<arg>*

Deprecated Syntax

MPIEXEC_SIGNAL_PROPAGATION=*<arg>*

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Turn on propagation. This is the default value
disable | no | off | 0	Turn off propagation

Description

Set this variable to control signal propagation. If it is turned on the signal is applied to all processes of the task. If signal propagation is disabled, the only process with rank #0 is killed with the given signal. The remaining processes are killed with the default signal 9 (SIGKILL).

NOTE: I_MPI_JOB_TIMEOUT_SIGNAL and I_MPI_JOB_SIGNAL_PROPAGATION can work independently.

I_MPI_OUTPUT_CHUNK_SIZE

Set the size of the stdout/stderr output buffer.

Syntax

I_MPI_OUTPUT_CHUNK_SIZE=*<size>*

Arguments

<size>	Define output chunk size in kilobytes
>0	The default chunk size value is 1 KB

Description

Set this variable to increase the size of the buffer used for intercept the stdout/stderr streams from processes. If the *<size>* value is not greater than zero, the variable setting is ignored and a warning message is displayed. Use this setting for applications that create significant amount of output from different processes. With the -ordered-output mpiexec option this setting helps to prevent the output from garbling.

NOTE: Set the I_MPI_OUTPUT_CHUNK_SIZE variable in the shell environment before executing the mpiexec command. Do not use the -genv or -env options for setting the *<size>* value. Those options are used only for passing variables to the MPI process environment.

I_MPI_PMI_EXTENSIONS

Turn on/off the use of the PMI extensions.

Syntax

I_MPI_PMI_EXTENSIONS=*<arg>*

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Turn on the PMI extensions
disable | no | off | 0	Turn off the PMI extensions

Description

The Intel® MPI Library automatically detects if your process manager supports the PMI extensions. The extensions substantially decrease task startup time but they must be handled by the process manager. Set I_MPI_PMI_EXTENSIONS to disable if your process manager does not support them.

I_MPI_JOB_FAST_STARTU

(I_MPI_PMI_FAST_STARTUP)

Turn on/off the new internal mpd algorithm intended for faster application startup.

Syntax

I_MPI_JOB_FAST_STARTUP=*<arg>*

Deprecated Syntax

I_MPI_PMI_FAST_STARTUP=*<arg>*

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Turn on the algorithm for fast PMI startup. This is the default value
disable | no | off | 0	Turn off the algorithm for fast PMI startup

Description

The new algorithm significantly decreases the application startup time. Some DAPL providers may be overloaded during startup of the large number of processes (greater then 512). To avoid this problem, turn off this algorithm by setting the I_MPI_JOB_FAST_STARTUP environment variable to disable.

I_MPI_DAT_LIBRARY

Select the particular DAT library to be used.

Syntax

I_MPI_DAT_LIBRARY=*<library>*

Arguments

<library>	Specify the exact library to be used instead of default libdat.so

Description

Set this variable to select a specific DAT library to be used. Specify the full path to the DAT library if it is not located in the dynamic loader search path.

NOTE: Use this variable only if you are going to utilize a DAPL provider.

TOTALVIEW

Select the particular TotalView* executable file to use.

Syntax

`TOTALVIEW=<path>`

Arguments

<path>	Path/name of the TotalView* executable file instead of the default `totalview`

Description

Set this variable to select a particular TotalView* executable file.

IDB_HOME

Set Intel® Debugger installation directory path.

Syntax

`IDB_HOME=<path>`

Arguments

<path>	Specify the installation directory of the Intel® Debugger

Description

Set this variable to specify installation directory of the Intel® Debugger.

I_MPI_TUNER_DATA_DIR

Set the alternate path to the automatic tuning utility output directory.

Syntax

`I_MPI_TUNER_DATA_DIR=<path>`

Arguments

<path>	Specify the automatic tuning utility output directory. The default value is *<mpiinstalldir>*`/etc` or *<mpiinstalldir>*`/etc`

Description

Set this variable to specify alternate tuned data files location. It makes sense if you want to use your own tuned data files from a separate directory.

2.3 Simplified Job Startup Command

mpirun

Syntax

`mpirun` *[<mpdboot options>] <mpiexec options>*

Arguments

<mpdboot options>	mpdboot options as described in the mpdboot command description below, except -n
<mpiexec options>	mpiexec options as described in the mpiexec section above

Description

Use this command to start an independent ring of mpd daemons, launch an MPI job, and shut down the mpd ring upon job termination.

The first non-mpdboot option (including -n or -np) delimits the mpdboot and mpiexec options. All options up to this point, excluding the delimiting option, are passed to the mpdboot command. All options from this point on, including the delimiting option, are passed to the mpiexec command.

All configuration files and environment variables applicable to the mpdboot and mpiexec commands are also pertinent to mpirun.

The set of hosts is defined by the following rules, which are checked in this order:

1. All host names from the mpdboot host file (either mpd.hosts or the file specified by the -f option).
2. All host names returned by the mpdtrace command, if there is an mpd ring running.
3. Local host (a warning is issued in this case).

The mpirun command also detects if the MPI job is submitted in a session allocated using a job scheduler like Torque*, PBS Pro*, LSF* or Parallelnavi* NQS*. In this case, the mpirun command extracts the host list from the respective environment and uses these nodes automatically according to the above scheme.

In such instances, you do not have to create the mpd.hosts file yourself. Just allocate the session you need using the particular job scheduler installed on your system, and use the mpirun command inside this session to run your MPI job.

See the product *Release Notes* for a complete list of the supported job schedulers.

2.4 MPD Daemon Commands

mpd

Start mpd daemon.

Syntax

```
mpd [ --help ] [ --host=<host> --port=<portnum> ] [ --noconsole ] \
    [ --trace ] [ --echo ] [ --daemon ] [ --bulletproof ] \
    [ --ifhn <interface/hostname> ] [ --listenport <listenport> ]
```

Arguments

[--help]	Display a help message
[-h <host> -p <portnum>] | [--host=<host> --port=<portnum>]	Specify the host and port to be used for entering an existing ring. The --host and --port options must be specified together
[-n] | [--noconsole]	Do not create console at startup

[-t] | [--trace]	Print internal MPD trace information
[-e] | [--echo]	Print a port number at startup to which other mpds may connect
[-d] | [--daemon]	Start mpd in daemon mode. By default, the interactive mode is enabled
[--bulletproof]	Turn MPD bulletproofing on
[--ifhn=<*interface/hostname*>]	Specify <*interface/hostname*> to use for MPD communications
[-l <*listenport*>] | [--listenport=<*listenport*>]	Specify the mpd listening port

Description

MPD is a process management system for starting parallel jobs. Before running a job, start mpd daemons on each host and connect them into a ring. Long parameter names may be abbreviated to their first letters by using only one hyphen and no equal sign. For example,

```
mpd -h masterhost -p 4268 -n
```

is equivalent to

```
mpd --host=masterhost --port=4268 -noconsole
```

If file named .mpd.conf is presented in the user's home directory, only user can have read and write privileges. The file must minimally contain a line with secretword=<*secretword*>. Create the mpd.conf file in the /etc directory instead of .mpd.conf in the root's home directory to run mpd as root. We do not recommend starting the MPD ring under the root account.

mpdboot

Start MPD ring.

Syntax

```
mpdboot  [ -n <#nodes> ] [ -f <hostsfile> ] [ -h ] [ -r <rshcmd> ] \
         [ -u <user> ] [ -m <mpdcmd> ] [ --loccons ] [ --remcons ] \
         [ -s ] [ -d ] [ -v ] [ -1 ] [ --ncpus=<ncpus> ] [ -o ]
```

or

```
mpdboot  [ --totalnum=<#nodes> ] [ --file=<hostsfile> ] [ --help ] \
         [ --rsh=<rshcmd> ] [ --user=<user> ] [ --mpd=<mpdcmd> ] \
         [ --loccons ] [ --remcons ] [ --shell ] [ --debug ] \
         [ --verbose ] [ -1 ] [ --ncpus=<ncpus> ] [ --ordered ]
```

Arguments

-h | --help	Display a help message
-d | --debug	Print debug information
-v | --verbose	Print extra verbose information. Show the <*rshcmd*> attempts

-n <#nodes> | --totalnum=<#nodes>	Number of nodes in mpd.hosts on which daemons are started
-r <rshcmd> | --rsh=<rshcmd>	Specify remote shell to start daemons and jobs. rsh is the default value
-f <hostsfile> | --file=<hostsfile>	Path/name of the file that has the list of machine names on which daemons are started
-1	Remove the restriction of starting only one mpd per machine
-m <mpdcmd> | --mpd=<mpdcms>	Specify the full path name of mpd on the remote hosts
-s | --shell	Specify shell
-u <user> | --user=<user>	Specify user
--loccons	Do not create local MPD consoles
--remcons	Do not create remote MPD consoles
--ncpus=<ncpus>	Indicate how many processors to use on the local machine (other nodes are listed in the hosts file)
-o | --ordered	Start all the mpd daemons in the exact order as specified in the mpd.hosts file

Description

Start the mpd daemons on the specified number of nodes by providing a list of node names in <mpd.hosts>.

The mpd daemons are started using the rsh command by default. If the rsh connectivity is not enabled, use the -r ssh option to switch over to ssh. Make sure that all nodes in the cluster can connect to each other via the rsh command without a password or, if the -r ssh option is used, via the ssh command without a password.

NOTE: The mpdboot command will spawn a MPD daemon on the host machine, even if the machine name is not listed in the mpd.hosts file.

mpdexit

Shut down a single mpd daemon.

Syntax

mpdexit <mpdid>

Arguments

<mpdid>	Specify the mpd daemon to kill

Description

Use this command to cause the single mpd daemon to exit. Use <mpdid> obtained via the mpdtrace -l command in the form <hostname>_<port number>.

mpdallexit

Shut down all mpd daemons on all nodes.

Arguments

This command takes no arguments.

Description

Use this command to shutdown all MPD rings.

mpdcleanup

Cleanup environment after `mpd` crash.

Syntax

```
mpdcleanup  [ -f <hostsfile> ] [ -r <rshcmd> ] [ -u <user> ] \
            [ -c <cleancmd> ] [ -a]
```

or

```
mpdcleanup  [ --file=<hostsfile> ] [ --rsh=<rshcmd> ] \
            [ --user=<user> ] [ --clean=<cleancmd> ] [ --all]
```

Arguments

`-f <hostsfile>` | `--file=<hostsfile>`	Specify the file containing a list of machines to clean up
`-r <rshcmd>` `--rsh=<rshcmd>`	Specify the remote shell to use
`-u <user>` | `--user=<user>`	Specify the user
`-c <cleancmd>` | `--clean=<cleancmd>`	Specify the command to use for removing the UNIX* socket. The default command is `/bin/rm -f`
`-a` | `--all`	Kill all `mpd` daemons related to the current settings of the `I_MPI_JOB_CONTEXT` environment variable on all hosts specified in the `<hostsfile>`

Description

Use this command to cleanup environment after mpd crash. It removes the UNIX* socket on local and remote machines or kills all `mpd` daemons related to the current environment controlled by the `I_MPI_JOB_CONTEXT` environment variable.

For instance, use the following command to remove the UNIX sockets on machines specified in the `hostsfile` file

```
mpdcleanup  --file=hostsfile --rsh=ssh
```

Use the following command to kill the `mpd` daemons on machines specified in the `hostsfile` file.

```
mpdcleanup  --file=hostsfile --all
```

mpdtrace

Determine whether `mpd` is running.

Syntax

```
mpdtrace [ -l ]
```

Arguments

[-l]	Show MPD identifiers instead of the hostnames

Description

Use this command to list hostnames or identifiers of mpd in the ring. The identifiers have the form *<hostname>_<port number>*.

mpdcheck

Check for configuration problems on the host or print configuration information about this host.

Syntax

```
mpdcheck [-v] [-l] [-h | --help]
mpdcheck -pc [-v] [-l]
mpdcheck -f <host_file> [-ssh] [-v] [-l]
mpdcheck -s [-v] [-l]
mpdcheck -c < server_host> <server_port> [-v] [-l]
```

Arguments

-pc	Print configuration information about a local host
-f *<host_file>*	Print information about hosts listed in *<host_file>*
-ssh	Cause the ssh tests on each remote host. Use in conjunction with the -f option
-s	Run mpdcheck as a server on one host
-c *<server_host>* *<server_port>*	Run mpdcheck as client on the other or same host. Connect to the *<server_host>* *<server_port>*
-l	Print diagnostic messages in extended format
-v	Print actions that mpdcheck is performing
-h	Print help

Description

Use this command to check configuration problems on the cluster nodes. Any output started with *** indicates a potential problem.

If you have problems running parallel jobs via mpd on one or more hosts, try to run the script once on each of those hosts.

mpdringtest

Test the MPD ring.

Syntax

```
mpdringtest [ <number of loops> ]
```

Arguments

<number of loops>	Number of loops

Description

Use this command to test how long it takes for a message to circle the mpd ring.

mpdlistjobs

List the running processes for a particular set of MPI jobs.

Syntax

```
mpdlistjobs [ -u <username> ] [ -a <jobalias> ] [ -j <jobid> ]
```

or

```
mpdlistjobs [ --user=<username> ] [ --alias=<jobalias> ]\
[ --jobid=<jobid> ]
```

Arguments

`-u <username>` | `--user=<username>`	List jobs of a particular user
`-a <jobalias>` | `--alias=<jobalias>`	List information about the particular job specified by `<jobalias>`
`-j <jobid>` | `--jobid=<jobid>`	List information about the particular job specified by `<jobid>`

Description

Use this command to list the running processes for a set of MPI jobs. All jobs for the current machine are displayed by default.

mpdsigjob

Apply a signal to a process running an application.

Syntax

```
mpdsigjob sigtype [-j <jobid> | -a <jobalias> ] [-s | -g ]
```

Arguments

`sigtype`	Specify the signal to send
`-a <jobalias>`	Send a signal to the job specified by `<jobalias>`
`-j <jobid>`	Send a signal to the job specified by `<jobid>`
`-s`	Deliver a signal to the single user process
`-g`	Deliver a signal to the group of processes. This is the default behavior

Description

Use this command to deliver a specific signal to the processes of a running job. The desired signal is the first argument. Specify only one of two options: `-j` or `-a`.

mpdkilljobs

Kill a job.

Syntax

```
mpdkilljobs [ <jobnum> ] [ -a <jobalias> ]
```

Arguments

`<jobnum>`	Kill the job specified by `<jobnum>`

-a <jobalias>	Kill the job specified by <jobalias>

Description

Use this command to kill the job specified by <jobnum> or by <jobalias>. Obtain <jobnum> and <jobalias> from the mpdlistjobs command. The <jobid> field has the following format: <jobnum>@<mpdid>.

mpdhelp

Print brief help concerning MPD commands.

Syntax

mpdhelp

Arguments

This command takes no arguments.

Description

Use this command to obtain a brief help message concerning MPD commands.

2.4.1 Configuration Files

$HOME/.mpd.conf

This optional configuration file contains the mpd daemon password. Create it before setting up the mpd daemons. Use it to control access to the daemons by various Intel® MPI Library users.

Syntax

The file has a single line:

secretword=<mpd password>

or

MPD_SECRETWORD=<mpd password>

Description

An arbitrary <mpd password> string only controls access to the mpd daemons by various cluster users. Do not use Linux* login passwords here.

Place the $HOME/.mpd.conf file on a network-mounted file system, or replicate this file so that it is accessible as $HOME/.mpd.conf on all nodes of the cluster.

When mpdboot is executed by some non-root <user>, this file should have user and ownership set to <user> and <<user>'s group> accordingly. The access permissions should set to 600 mode (only user have read and write privileges).

NOTE: MPD_SECRETWORD is a synonym for secretword.

mpd.hosts

This file has a list of node names which the mpdboot command uses to start mpd daemons.

Ensure that this file is accessible by the user who runs mpdboot on the node where the mpdboot command is actually invoked.

Syntax

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.

2.4.2 Environment Variables

PATH

Ensure that the PATH settings include the path to mpdboot and other mpd daemon commands.

NOTE: The <*installdir*>/bin directory (<*installdir*>/bin64 directory for Intel® 64 architecture in 64-bit mode) and the path to Python* version 2.2 or higher must be included in PATH in order for the mpd daemon commands to succeed.

I_MPI_JOB_CONFIG_FILE

(I_MPI_MPD_CONF)

Set the path/name of the mpd configuration file.

Syntax

I_MPI_JOB_CONFIG_FILE=<*path/name*>

Deprecated Syntax

I_MPI_MPD_CONF=<*path/name*>

Arguments

<*path/name*>	Absolute path of the MPD configuration file

Description

Set this variable to define the absolute path of the file that is used by the mpdboot script instead of the default value ${HOME}/.mpd.conf.

I_MPI_JOB_CONTEXT

(MPD_CON_EXT)

Set a unique name for the mpd console file. This enables you to run several mpd rings under the same user account.

Syntax

I_MPI_JOB_CONTEXT=<*tag*>

Deprecated Syntax

MPD_CON_EXT=<*tag*>

Arguments

<*tag*>	Unique MPD identifier

Description

Set this variable to different unique values to allow several MPD rings to co-exist. Each MPD ring is associated with a separate I_MPI_JOB_CONTEXT value. Once this variable is set, you can start one MPD ring and work with it without affecting other available MPD rings. Set the appropriate

I_MPI_JOB_CONTEXT value to work with a particular MPD ring. See *Simplified Job Startup Command* to learn about an easier way to run several Intel® MPI Library jobs at once.

I_MPI_JOB_TAGGED_PORT_OUTPUT

Turn on/off the use of the new tagged mpd port output.

Syntax

I_MPI_JOB_TAGGED_PORT_OUTPUT=<*arg*>

Arguments

<*arg*>	Binary indicator
enable | yes | on | 1	Turn on the tagged output
disable | no | off | 0	Turn off the tagged output. This is the default value

Description

The new tagged output format works at the mpdboot stage and prevents it from a failure if there is unexpected output from ssh. mpdboot sets this variable to 1 automatically. Set I_MPI_JOB_TAGGED_PORT_OUTPUT to disable if you do not want to use the new format.

2.5 Processor Information Utility

cpuinfo

Use the cpuinfo utility to display processor architecture information.

Syntax

cpuinfo

Description

The cpuinfo utility prints out processor architecture information that can be used for more suitable process pinning settings.

The output contains a number of tables:

1. General data.
   Architecture – one of “i686”, “x86_64”, “ia64”
   Hyperthreading – one of “enabled”, “disabled”, “not supported”
   Packages – the number of physical packages (sockets)
   Cores – the number of all cores
   Processors – the number of logical processors (cpu)
2. Processor identification table. The table represents three-level (thread, core, and package) identification of each logical processor.
   Thread – unique processor identifier within a core.
   Core – unique core identifier within a package.
   Package – unique package identifier within a node.
3. Processor placement table. The table represents a map of processor placement by packages and cores (inversion of previous processor identification table). Each entry contains:
   Package – a physical package identifier.
   Cores – a list of core identifiers that belong to this package.
   Processors – a list of processors that belong to this package. This list order directly corresponds to the core list. A group of processors enclosed in the brackets belongs to one core.
4. Cache sharing table. For each cache level the table contain:
   Size – cache size in bytes.

Processors – a list of processor groups (enclosed in the brackets) that shared this cache or “no sharing” otherwise.

NOTE: Only the architecture information is printed for Itanium-2 based machines.

Examples:

1. `cpuinfo` output for Dual-Core Intel® Xeon® Processor 5100 series:

```
Architecture  : x86_64
Hyperthreading: disabled
Packages   : 2
Cores      : 4
Processors : 4

=====  Processor identification  =====
Processor       Thread  Core    Package
0               0       0       0
1               0       0       3
2               0       1       0
3               0       1       3

=====  Processor placement  =====
Package Cores           Processors
0       0,1             0,2
3       0,1             1,3

=====  Cache sharing  =====
Cache   Size            Processors
L1      32  KB          no sharing
L2      4   MB          (0,2)(1,3)
```

2. `cpuinfo` output for Quad-Core Intel® Xeon® processor 5300 series:

```
Architecture  : x86_64
Hyperthreading: disabled
Packages   : 2
Cores      : 8
Processors : 8

=====  Processor identification  =====
Processor       Thread  Core    Package
0               0       0       0
1               0       2       0
2               0       0       1
3               0       2       1
4               0       1       0
5               0       3       0
6               0       1       1
7               0       3       1

=====  Processor placement  =====
Package Cores           Processors
0       0,2,1,3         0,1,4,5
1       0,2,1,3         2,3,6,7

=====  Cache sharing  =====
Cache   Size            Processors
```

```
L1      32  KB          no sharing
L2      4   MB          (0,4)(1,5)(2,6)(3,7)
```

3. `cpuinfo` output for machine with Hyper-Threading Technology enabled:

```
Architecture  : x86_64
Hyperthreading: enabled
Packages   : 2
Cores      : 2
Processors : 4

=====  Processor identification  =====
Processor       Thread  Core    Package
0               0       0       0
1               1       0       0
2               0       0       3
3               1       0       3

=====  Processor placement  =====
Package Cores           Processors
0       0               (0,1)
3       0               (2,3)

=====  Cache sharing  =====
Cache   Size            Processors
L1      16  KB          (0,1)(2,3)
L2      1   MB          (0,1)(2,3)
```

3 Tuning Reference

The Intel® MPI Library provides automatic tuning utility and many environment variables that can be used to influence program behavior and performance at run time. See the description of the automatic turning utility in section 3.1 and of the variables in sections 3.2-3.7.

3.1 Automatic Tuning Utility

mpitune

Use the `mpitune` utility to find optimal settings for the Intel® MPI Library on your cluster.

Syntax

```
mpitune  [ -e <envfile> ] [ -r <rulesfile> ] [ -f <hostsfile> ] \
         [ -w <workdir> ] [ -o <outputdir> ] [ -h ] [ -d ]
```

or

```
mpitune  [ --env <envfile> ] [ --rules <rulesfile> ] \
         [ --file <hostsfile> ] [ --wdir <workdir> ] \
         [ --outdir <outputdir> ] [ --help ] [ --debug ]
```

Arguments

`-h | --help`	Display help message
`-d | --debug`	Print debug information
`-e <envfile> | --env <envfile>`	Path/name of the file with hardware and software environment information. The `<installdir>/etc/env.xml` or `<installdir>/etc64/env.xml` is used by default
`-r <rulesfile> | --rules <rulesfile>`	Path/name of the file with tuning process rules. The `<installdir>/etc/rules.xml` is used by default
`-f <hostsfile> | --file <hostsfile>`	Path/name of the file that has a list of machine names that are used in tuning process. The `$PWD/mpd.hosts` is used by default
`-w <workdir> | --wdir <workdir>`	Specify the location of the benchmarking programs. The `<installdir>/bin` or `<installdir>/bin64` are used by default
`-o <outputdir> | --outdir <outputdir>`	Specify the output directory for the `mpiexec` configuration files. The `<installdir>/etc` or `<installdir>/etc64` is used by default

Description

The `mpitune` utility creates a set of Intel® MPI Library configuration files that contain optimal settings for a particular cluster. These configuration files are used automatically by the `mpiexec -tune` option.

Run this utility once after Intel® MPI Library installation and after every cluster configuration change (processor or memory upgrade, network reconfiguration, etc.). Always do this under the user account that was used for the Intel® MPI Library installation.

Make sure that the cluster is free of other jobs. Depending on the size of the cluster, this process may take more or less substantial time to complete.

3.2 Process Pinning

I_MPI_PIN

Turn on/off process pinning feature of the Intel® MPI Library.

Syntax

`I_MPI_PIN=<arg>`

Arguments

<arg>	Binary indicator
`enable | yes | on | 1`	Enable process pinning. This is the default value
`disable | no | off | 0`	Disable processes pinning

Description

Set this variable to turn off process pinning feature of the Intel® MPI Library.

I_MPI_PIN_MODE

Choose algorithm to be used for pinning processes

Syntax

`I_MPI_PIN_MODE=<pinmode>`

Arguments

<pinmode>	Select CPU pinning mode
`mpd`	Pin processes inside MPD. Default on SGI* Altix* platform
`lib`	Pin processes inside MPI library. Default on other platforms

Description

Set the `I_MPI_PIN_MODE` variable to choose the algorithm used for pinning process. This variable is valid only if `I_MPI_PIN` is enabled.

Set this variable to `lib` to make the Intel® MPI Library pin the processes. The already running MPI process pinned to CPU. There is no chance to co-locate the process CPU and its memory.

Set the `I_MPI_PIN_MODE` variable to `mpd` to make the `mpd` daemon pin processes via system specific means, if they are available. The pinning is done before the MPI process is started. It is possible to co-locate the process CPU and memory in this case. This pinning method has an advantage on the system with Non-Uniform memory Architecture (NUMA) like SGI* Altix*. Under NUMA, a processor can access its own local memory faster than non-local memory. To avoid additional overhead on the memory operations MPI process should be located in the local memory of the processor. Process pinning is performed if the operating system provides the necessary kernel interfaces.

NOTE: It is not recommended to change the default settings.

I_MPI_PIN_PROCESSOR_LIST

(I_MPI_PIN_PROCS)

Identify set of processors to be used for process pinning

I_MPI_PIN_PROCESSOR_LIST=<*proclist*>

Deprecated Syntax

I_MPI_PIN_PROCS=<*proclist*>

Arguments

<*proclist*>	Define mapping of process to CPU
all	Use all CPUs
allcores	Use all CPU cores (or physical CPU). This is the default value
<*l*>	Use only CPU number <*l*> (where <*l*> is 0,1, … , or total number of CPUs – 1)
<*l*>-<*m*>	Use CPUs from <*l*> to <*m*>
<*k*>,<*l*>-<*m*>,<*n*>	Use CPUs <*k*>, <*l*> through <*m*>, and <*n*>

Description

Set the I_MPI_PIN_PROCESSOR_LIST variable to define the set of processors. This variable is valid only if I_MPI_PIN is enabled.

If no set of CPUs is defined in the system, the number and order of the processors correspond to the output of the cat /proc/cpuinfo command. If a CPU set is defined in the system, the I_MPI_PIN_PROCESSOR_LIST value refers to the logical processors enabled in the current process set.

This variable does not influence the process placement that is controlled by the mpdboot and mpiexec commands. However, when this variable is defined and a process is placed upon the node, that process is bound to the next CPU out of the specified set.

For example, to pin the processes to the CPU0 and CPU3 on each node globally, use the following command:

```
$ mpirun -genv I_MPI_PIN_PROCESSOR_LIST 0,3 -n <# of processes> <executable>
```

To pin the processes to different CPUs on each node individually, use the following command:

```
$ mpirun -host host1 -env I_MPI_PIN_PROCESSOR_LIST 0,3 -n <# of processes> \
<executable> : -host host2 -env I_MPI_PIN_PROCESSOR_LIST 1,2,3 \

-n <# of processes> <executable>
```

To print extra debug information about the process pinning, use the following command:

```
$ mpirun -genv I_MPI_DEBUG 2 -m -host host1 \

-env I_MPI_PIN_PROCESSOR_LIST 0,3 -n <# of processes> <executable> : \

-host host2 -env I_MPI_PIN_PROCESSOR_LIST 1,2,3 \

-n <# of processes> <executable>
```

NOTE: The values of I_MPI_PIN_MODE and I_MPI_PIN_PROCESSOR_LIST can be defined locally, on a per host level, or globally, for the entire system.

3.3 Device Control

I_MPI_EAGER_THRESHOLD

Change the eager/rendezvous cutover point for all devices.

Syntax

I_MPI_EAGER_THRESHOLD=*<nbytes>*

Arguments

<nbytes>	Define eager/rendezvous cutover point
> 0	The default *<nbytes>* value is equal to 262 144 bytes

Description

Set this variable to control the point-to-point protocol switchover point. Data transfer algorithms are selected based on the following scheme:

- Messages shorter than or equal in size to *<nbytes>* are sent using the eager protocol.
- Larger messages are sent by using the more memory efficient rendezvous protocol.

I_MPI_INTRANODE_EAGER_THRESHOLD

Change the eager/rendezvous cutover point for intranode communication mode.

Syntax

I_MPI_INTRANODE_EAGER_THRESHOLD=*<nbytes>*

Arguments

<nbytes>	Define the threshold for DAPL* intranode communication
> 0	The default *<nbytes>* value is equal to 262 144 bytes

Description

Set this variable to change the threshold for communication within the node. Data transfer algorithms are selected based on the following scheme:

- Messages shorter than or equal in size to *<nbytes>* are sent using the eager protocol.
- Larger messages are sent by using the more memory efficient rendezvous protocol.

If I_MPI_INTRANODE_EAGER_THRESHOLD is not set, the value of I_MPI_EAGER_THRESHOLD is used.

I_MPI_WAIT_MODE

Turn on/off a wait mode.

Syntax

I_MPI_WAIT_MODE=<*arg*>

Arguments

<*arg*>	Binary indicator
enable | yes | on | 1	Turn on the wait mode
disable | no | off | 0	Turn off the wait mode. This is the default value

Description

Set this variable to control the wait mode. If this mode is enabled, the processes wait for receiving messages without polling of the fabric(s). This can save CPU time for other tasks.

NOTE: The wait mode supports the sock, shm and ssm devices.

NOTE: Use Native POSIX Thread Library with wait mode for shm devices.

NOTE: Use the following command to check what version of the thread library installed on your system:

```
getconf GNU_LIBPTHREAD_VERSION
```

I_MPI_SPIN_COUNT

Control the spin count value.

Syntax

I_MPI_SPIN_COUNT=<*scount*>

Arguments

<*scount*>	Define the loop spin count when polling fabric(s)
> 0	The default <*scount*> value is equal to 1 time for sock, shm, and ssm devices, and equal to 250 times for rdma and rdssm devices

Description

Set the spin count limit. The loop for polling the fabric(s) will spin <*scount*> times before freeing the processes if no incoming messages are received for processing. Smaller values for <*scount*> cause the Intel® MPI Library to release the processor more frequently.

Use the I_MPI_SPIN_COUNT environment variable for turning application performance. The best value for <*scount*> can be chosen on an experimental basis. It depends on the particular computational environment and application.

NOTE: Use the I_MPI_SPIN_COUNT environment variable with caution. Keep in mind that three different effects are possible: no effect, performance improvement, or performance degradation.

I_MPI_CACHE_BYPASS

Control a message transfer algorithm for shm device.

Syntax

I_MPI_CACHE_BYPASS=<*arg*>

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Enable message transfer bypass cache. This is the default value
disable | no | off | 0	Disable message transfer bypass cache

Description

Set this variable to control message transfer algorithm for shm device. By default messages greater than or equal in size to value specified by the I_MPI_CACHE_BYPASS_THRESHOLD environment variable are sent bypass cache. This feature is enabled on IA-32 and Intel® 64 architectures by default. It does not affect Itanium® 2-based systems.

I_MPI_CACHE_BYPASS_THRESHOLDS

Change messages copying algorithm cutover point.

Syntax

I_MPI_CACHE_BYPASS_THRESHOLD=*<nb_send>*,*[<nb_recv>*,*[<nb_send_l2>*,\

[<nb_recv_l2>,*[<nb_send_pk>*,*[<nb_recv_pk>]]]]]*

Arguments

<nb_send>	Define cutover point for sent messages when processes are pinned on cores without shared L2 cache and not located in the same physical processor package or when processes are not pinned
≥ 16384	The default *<nb_send>* value is 16384 bytes
<nb_recv>	Define cutover point for received messages when processes are pinned on cores without shared L2 cache and not located in the same physical processor package or when processes are not pinned
≥ 16384	The default *<nb_send>* value is ½ of the size of L2 cache
<nb_send_l2>	Define cutover point for sent messages when processes are pinned on cores with shared L2 cache
≥ 16384	Copying bypass cache is disabled
<nb_recv_l2>	Define cutover point for received messages when processes are pinned on cores with shared L2 cache
≥ 16384	The default *<nb_recv_l2>* value is ½ of the size of L2 cache
<nb_send_pk>	Define cutover point for sent messages when processes are pinned on cores without shared L2 cache but located in the same physical processor package
≥ 16384	Copying bypass cache is disabled
<nb_recv_pk>	Define cutover point for received messages when processes are pinned on cores without shared L2 cache but located in the same physical processor package
≥ 16384	The default *<nb_recv_pk>* value is ½ of the size of L2 cache

Description

Set this variable to control the switchover point for a message copying algorithm. Messages greater than or equal in size to defined thresholds value are copied bypass cache. “-1” value disables coping bypass cache. This variable is valid only if I_MPI_CACHE_BYPASS is enabled.

I_MPI_SHM_SINGLE_SEGMENT_THRESHOLD

(I_MPI_SHM_PROC_THRESHOLD)

Change the static/dynamic shared memory segment(s) allocation mode for the shm device.

Syntax

I_MPI_SHM_SINGLE_SEGMENT_THRESHOLD=<*nproc*>

Deprecated Syntax

I_MPI_SHM_PROC_THRESHOLD=<*nproc*>

Arguments

<*nproc*>	Define static/dynamic mode switch point for the shm device
> 0, < 90	The default <*nproc*> value is equal to 90

Description

Set this variable to change the allocation mode for the shm device.

The following modes are available for the allocation of the shared memory segment(s) for the shm device:

- If the number of processes started on the system is less than the value specified by <*nproc*>, the static mode is used. In that case only one common shared memory segment is allocated for all processes during the initialization stage.
- Otherwise, the dynamic mode is used and the shared memory segments are allocated for each connection individually.

NOTE: The dynamic connection establishment mode does not make sense when the static allocation mode is used. The I_MPI_DYNAMIC_CONNECTION environment variable is not applicable in this case.

3.4 RDMA and RDSSM Device Control

I_MPI_RDMA_TRANSLATION_CACHE

Turn on/off the use of a memory registration cache.

Syntax

I_MPI_RDMA_TRANSLATION_CACHE=<*arg*>

Arguments

<*arg*>	Binary indicator
enable | yes | on | 1	Turn on the memory registration cache. This is the default value
disable | no | off | 0	Turn off the memory registration cache

Description

Set this variable to turn on or off the memory registration cache.

The cache substantially increases performance but may lead to correctness issues in certain rare situations. See product Release Notes for further details.

I_MPI_RDMA_EAGER_THRESHOLD

(RDMA_IBA_EAGER_THRESHOLD)

Change the eager/rendezvous cutover point.

Syntax

I_MPI_RDMA_EAGER_THRESHOLD=<*nbytes*>

Deprecated Syntax

RDMA_IBA_EAGER_THRESHOLD=<*nbytes*>

Arguments

<*nbytes*>	Define eager/rendezvous cutover point
> 0	The default <*nbytes*> value is equal to 16 512 bytes

Description

Set this variable to control low-level point-to-point protocol switchover point. Data transfer algorithms for the rdma and rdssm devices are selected based on the following scheme:

- Messages shorter than or equal to <*nbytes*> are sent using the eager protocol through internal pre-registered buffers.
- Larger messages are sent by using the more memory efficient rendezvous protocol.

NOTE: This variable also determines the size of each pre-registered buffer. The higher it is the more memory is used for each established connection.

I_MPI_DYNAMIC_CONNECTION

(I_MPI_USE_DYNAMIC_CONNECTIONS)

Turn on/off the dynamic connection establishment.

Syntax

I_MPI_DYNAMIC_CONNECTION=<*arg*>

Deprecated Syntax

I_MPI_USE_DYNAMIC_CONNECTIONS=<*arg*>

Arguments

<*arg*>	Binary indicator
enable | yes | on | 1	Turn on the dynamic connection establishment. This is the default value
disable | no | off | 0	Turn off the dynamic connection establishment

Description

Set this variable to control dynamic connection establishment.

- If enabled, connections are established at the time of the first communication between each pair of processes. This is the default behavior.
- Otherwise, all connections are established upfront.

I_MPI_DYNAMIC_CONNECTION_MODE

(I_MPI_DYNAMIC_CONNECTIONS_MODE)

Choose the algorithm for establishing of the DAPL* connections.

Syntax

`I_MPI_DYNAMIC_CONNECTION_MODE=<arg>`

Deprecated Syntax

`I_MPI_DYNAMIC_CONNECTIONS_MODE=<arg>`

Arguments

<arg>	Mode selector
`reject`	Deny one of the two simultaneous connection requests. This is the default value
`disconnect`	Deny one of the two simultaneous connection requests after both connections have been established

Description

Set this variable to choose the algorithm for handling dynamically established connections for DAPL*-capable fabrics according to the following scheme:

- In the `reject` mode, one of the requests is rejected if two processes initiate the connection simultaneously.
- In the `disconnect` mode both connections are established, but then one is disconnected. The `disconnect` mode is provided to avoid a bug in certain DAPL* providers.

I_MPI_RDMA_SCALABLE_PROGRESS

Turn on/off scalable algorithm for RDMA read progress.

Syntax

`I_MPI_RDMA_SCALABLE_PROGRESS=<arg>`

Arguments

<arg>	Binary indicator
`enable | yes | on | 1`	Turn on scalable algorithm
`disable | no | off | 0`	Turn off scalable algorithm. This is the default value

Description

Set this variable to select scalable algorithm for RDMA read progress. In some cases it provides advantages for large number of processes.

I_MPI_INTRANODE_SHMEM_BYPASS

(I_MPI_USE_DAPL_INTRANODE)

Turn on/off the DAPL* intranode communication mode.

Syntax

`I_MPI_INTRANODE_SHMEM_BYPASS=<arg>`

Deprecated Syntax

`I_MPI_USE_DAPL_INTRANODE=<arg>`

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Turn on the DAPL* intranode communication
disable | no | off | 0	Turn off the DAPL* intranode communication. This is the default value

Description

Set this variable to specify the communication for the universal device within the node. If the DAPL* intranode communication mode is enabled, data transfer algorithms are selected based on the following scheme:

- Messages shorter than or equal in size to the threshold value of the I_MPI_INTRANODE_EAGER_THRESHOLD variable are transferred using shared memory.
- Large messages are transferred via the DAPL* layer.

NOTE: This variable is applicable only when shared memory and the DAPL* layer are turned on by setting the I_MPI_DEVICE environment variable to the rdssm value.

I_MPI_RDMA_EVD_POLLING

(I_MPI_RDMA_USE_EVD_FALLBACK)

Turn on/off the use of the Event Dispatcher (EVD) as a fallback method when polling for messages.

Syntax

I_MPI_RDMA_EVD_POLLING=*<arg>*

Deprecated Syntax

I_MPI_RDMA_USE_EVD_FALLBACK=*<arg>*

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Turn on the EVD based fallback
disable | no | off | 0	Turn off the EVD based fallback. This is the default value

Description

Set this variable to use the DAPL* Event Dispatcher (EVD) for detecting incoming messages.

Use this method instead of the default method of buffer polling if the DAPL* provider does not guarantee the delivery of the transmitted data in order from low to high addresses.

NOTE: The EVD method of message detection is typically substantially slower than the default algorithm.

I_MPI_RDMA_BUFFER_NUM

(NUM_RDMA_BUFFER)

Change the number of internal pre-registered buffers for each pair in a process group.

Syntax

I_MPI_RDMA_BUFFER_NUM=*<nbuf>*

Deprecated Syntax

NUM_RDMA_BUFFER=<*nbuf*>

Arguments

<*nbuf*>	Define the number of buffers for each pair in a process group
> 0	The default <*nbuf*> value ranges between 8 and 40 depending on the cluster size and platform

Description

Set this variable to change the number of internal pre-registered buffers for each pair in a process group.

NOTE: The more pre-registered buffers are available, the more memory is used for every established connection.

I_MPI_RDMA_BUFFER_SIZE

(I_MPI_RDMA_VBUF_TOTAL_SIZE)

Change the size of internal pre-registered buffers for each pair in a process group.

Syntax

I_MPI_RDMA_BUFFER_SIZE=<*nbytes*>

Deprecated Syntax

I_MPI_RDMA_VBUF_TOTAL_SIZE=<*nbytes*>

Arguments

<*nbytes*>	Define the size of pre-registered buffers
> 0	The default <*nbytes*> value is equal to 16 640 bytes

Description

Set this variable to define the size of the internal pre-registered buffer for each pair in a process group. The actual size is calculated by adjusting <*nbytes*> to align the buffer to an optimal value.

I_MPI_RDMA_BUFFER_ENLARGEMENT

(I_MPI_TWO_PHASE_BUF_ENLARGEMENT)

Turn on/off the use of two-phase buffer enlargement.

Syntax

I_MPI_RDMA_BUFFER_ENLARGEMENT =<*arg*>

Deprecated Syntax

I_MPI_TWO_PHASE_BUF_ENLARGEMENT=<*arg*>

Arguments

<*arg*>	Binary indicator
enable | yes | on | 1	Turn on the mode of using two-phase buffer enlargement
disable | no | off | 0	Turn off the mode of using two-phrase buffer enlargement. This is the default value

Description

Set this variable to control the use of the two-phase buffer enlargement according to the following algorithm:

- If enabled, small size internal pre-registered RDMA buffers are allocated and enlarged later if data size exceeds the threshold defined by `I_MPI_RDMA_BUFFER_ENLARGEMENT_THRESHOLD`
- Two-phase buffer enlargement is turned off by default.

I_MPI_RDMA_BUFFER_ENLARGEMENT_THRESHOLD

(I_MPI_RDMA_SHORT_BUF_THRESHOLD)

Change threshold for two-phase buffer enlargement mode.

Syntax

`I_MPI_RDMA_BUFFER_ENLARGEMENT_THRESHOLD=`*`<nbytes>`*

Deprecated Syntax

`I_MPI_RDMA_SHORT_BUF_THRESHOLD=`*`<nbytes>`*

Arguments

`<nbytes>`	Define the threshold for starting enlargement of the RDMA buffers
`> 0`	The default value is 580

Description

Set this variable to define the threshold for increasing the size of the two-phase RDMA buffers. This variable is valid only if `I_MPI_RDMA_BUFFER_ENLARGEMENT` is enabled.

I_MPI_RDMA_TINY_PACKET

Turn on/off the use of small packets.

Syntax

`I_MPI_RDMA_TINY_PACKET=`*`<arg>`*

Arguments

`<arg>`	Binary indicator
`enable | yes | on | 1`	Turn on the use of small packets
`disable | no | off | 0`	Turn off the use of small packets. This is the default value (the regular packet sizes is used instead)

Description

Set this variable to use the small packets for short messages. The regular packet sizes are used by default.

Certain DAPL* providers are sensitive to the packet size on certain hardware. Switching on small packets for short messages may increase performance in these cases.

I_MPI_RDMA_RNDV_WRITE

(I_MPI_USE_RENDEZVOUS_RDMA_WRITE)

Turn on/off the rendezvous RDMA Write protocol.

Syntax

I_MPI_RDMA_RNDV_WRITE=<arg>

Deprecated Syntax

I_MPI_USE_RENDEZVOUS_RDMA_WRITE=<arg>

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Turn on the RDMA Write rendezvous protocol
disable | no | off | 0	Turn off the RDMA Write rendezvous protocol. This is the default value (the RDMA Read protocol is used instead)

Description

Set this variable to select the RDMA Write-based rendezvous protocol.

Certain DAPL* providers have a slow RDMA Read implementation on certain platforms. Switching on the rendezvous protocol based on the RDMA Write operation may increase performance in these cases.

The Intel® MPI Library automatically switches to the rendezvous protocol based on RDMA Write operation if the DAPL* intranode communication is on and DAPL provider name contains substrings Openib or OpenIB. Set the I_MPI_RDMA_RNDV_WRITE to disable to avoid such behavior.

I_MPI_RDMA_CHECK_MAX_RDMA_SIZE

(I_MPI_DAPL_CHECK_MAX_RDMA_SIZE)

Check the value of the DAPL* attribute max_rdma_size.

Syntax

I_MPI_RDMA_CHECK_MAX_RDMA_SIZE=<arg>

Deprecated Syntax

I_MPI_DAPL_CHECK_MAX_RDMA_SIZE=<arg>

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Check the value of the DAPL* attribute max_rdma_size
disable | no | off | 0	Do not check the value of the DAPL* attribute max_rdma_size. This is the default value

Description

Set this variable to control message fragmentation according to the following scheme:

- If set to disable, the Intel® MPI Library does not take into account the value of the DAPL* attribute max_rdma_size for message fragmentation.
- If set to enable, the Intel® MPI Library fragments messages of size greater than the value of the DAPL* attribute max_rdma_size

I_MPI_RDMA_CONN_EVD_SIZE

(I_MPI_CONN_EVD_QLEN)

Define the event queue size of the DAPL* event dispatcher.

Syntax

`I_MPI_RDMA_CONN_EVD_SIZE=<size>`

Deprecated Syntax

`I_MPI_CONN_EVD_QLEN=<size>`

Arguments

<size>	Define the length of the event queue
> 0	The default value is queried from the DAPL provider

Description

Set this variable to define the event queue size of the DAPL event dispatcher. If this variable is set, the minimum value between *<size>* and the value obtained from the provider is used as the size of the event queue. The provider is required to supply a queue size that is at least equal to the calculated value, but it can also provide a larger queue size.

3.5 Collective Operation Control

Each collective operation in the Intel® MPI Library supports a number of communication algorithms. In addition to reasonable default settings, the library provides two ways to control the algorithm selection explicitly: the novel `I_MPI_ADJUST` environment variable family and the deprecated `I_MPI_MSG` environment variable family.

3.5.1 I_MPI_ADJUST family

I_MPI_ADJUST_*<opname>*

Control collective operation algorithm selection.

Syntax

`I_MPI_ADJUST_<opname>=<algid>[:<conditions>][;<algid>:<conditions>[…]]`

Arguments

<algid>	Algorithm identifier
≥ 0	The default value of zero selects reasonable default settings

<conditions>	A comma separated list of conditions. An empty list selects all message sizes and process combinations
<l>	Messages of size *<l>*
<l>-<m>	Messages of size from *<l>* to *<m>*, inclusive
<l>@<p>	Messages of size *<l>* and number of processes *<p>*

<l>-<m>@<p>-<q>	Messages of size from <l> to <m> and number of processes from <p> to <q>, inclusive

Description

Set this variable to select the desired algorithm(s) for the collective operation <opname> under particular conditions. Each collective operation has its own environment variable and algorithms (see Table 3-1).

Table 3-1 Environment variables, collective operations, and algorithms

Environment variable	Collective operation	Algorithms
I_MPI_ADJUST_ALLGATHER	MPI_Allgather	1. Recursive doubling algorithm 2. Bruck's algorithm 3. Ring algorithm 4. Topology aware Gatherv + Bcast algorithm
I_MPI_ADJUST_ALLGATHERV	MPI_Allgatherv	1. Recursive doubling algorithm 2. Bruck's algorithm 3. Ring algorithm 4. Topology aware Gatherv + Bcast algorithm
I_MPI_ADJUST_ALLREDUCE	MPI_Allreduce	1. Recursive doubling algorithm 2. Rabenseifner's algorithm 3. Reduce + Bcast algorithm 4. Topology aware Reduce + Bcast algorithm
I_MPI_ADJUST_ALLTOALL	MPI_Alltoall	1. Bruck's algorithm 2. Isend/Irecv + waitall algorithm 3. Pair wise exchange algorithm 4. Plum's algorithm
I_MPI_ADJUST_ALLTOALLV	MPI_Alltoallv	1. Isend/Irecv + waitall algorithm 2. Plum's algorithm
I_MPI_ADJUST_ALLTOALLW	MPI_Alltoallw	1. Isend/Irecv + waitall algorithm
I_MPI_ADJUST_BARRIER	MPI_Barrier	1. Dissemination algorithm 2. Recursive doubling algorithm 3. Topology aware dissemination algorithm 4. Topology aware recursive doubling algorithm

I_MPI_ADJUST_BCAST	MPI_Bcast	1. Binomial algorithm 2. Recursive doubling algorithm 3. Ring algorithm 4. Topology aware binomial algorithm 5. Topology aware recursive doubling algorithm 6. Topology aware ring algorithm
I_MPI_ADJUST_EXSCAN	MPI_Exscan	1. Partial results gathering algorithm 2. Partial results gathering regarding algorithm layout of processes
I_MPI_ADJUST_GATHER	MPI_Gather	1. Binomial algorithm 2. Topology aware binomial algorithm
I_MPI_ADJUST_GATHERV	MPI_Gatherv	1. Linear algorithm 2. Topology aware linear algorithm
I_MPI_ADJUST_REDUCE_SCATTER	MPI_Reduce_scatter	1. Recursive having algorithm 2. Pair wise exchange algorithm 3. Recursive doubling algorithm 4. Reduce + Scatterv algorithm 5. Topology aware Reduce + Scatterv algorithm
I_MPI_ADJUST_REDUCE	MPI_Reduce	1. Shumilin's algorithm 2. Binomial algorithm 3. Topology aware Shumilin's algorithm 4. Topology aware binomial algorithm
I_MPI_ADJUST_SCAN	MPI_Scan	1. Partial results gathering algorithm 2. Topology aware partial results gathering algorithm
I_MPI_ADJUST_SCATTER	MPI_Scatter	1. Binomial algorithm 2. Topology aware binomial algorithm
I_MPI_ADJUST_SCATTERV	MPI_Scatterv	1. Linear algorithm 2. Topology aware linear algorithm

The message size calculation rules for the collective operations are described in Table 3-2. Here, "n/a" means that the corresponding interval <*l*>-<*m*> should be omitted.

Table 3-2 Message Collective functions and

Collective function	Message size formula
MPI_Allgather	recv_count*recv_type_size
MPI_Allgatherv	total_recv_count*recv_type_size
MPI_Allreduce	count*type_size
MPI_Alltoall	send_count*send_type_size
MPI_Alltoallv	n/a
MPI_Alltoallw	n/a
MPI_Barrier	n/a
MPI_Bcast	count*type_size
MPI_Exscan	count*type_size
MPI_Gather	Ecv_count*recv_type_size if MPI_IN_PLACE is used, otherwise recv_count*recv_type_size
MPI_Gatherv	n/a
MPI_Reduce_scatter	total_recv_count*type_size
MPI_Reduce	count*type_size
MPI_Scan	count*type_size
MPI_Scatter	send_count*send_type_size if MPI_IN_PLACE is used, otherwise recv_count*recv_type_size
MPI_Scatterv	n/a

Examples:

1. Use the following settings to select second algorithm for MPI_Reduce operation:
   I_MPI_ADJUST_REDUCE=2
2. Use the following settings to define algorithms for MPI_Scatter operation:
   I_MPI_ADJUST_REDUCE_SCATTER=4:0-100,5001-10000;1:101-3200,2:3201-5000;3

   In this case algorithm four will be used for the message sizes from 0 up to 100 bytes and from 5001 to 10000 bytes, algorithm one will be used for the message sizes from 101 up to 3200 bytes, algorithm two will be used for the message sizes from 3201 up to 5000 bytes, and algorithm three will be used for all other messages.

3.5.2 I_MPI_MSG family

These variables are deprecated and supported mostly for backward compatibility. Use the I_MPI_ADJUST environment variable family whenever possible.

I_MPI_FAST_COLLECTIVES

Control default library behavior during selection of appropriate collective algorithm for specific execution situation.

Syntax

I_MPI_FAST_COLLECTIVES=<*arg*>

Arguments

<*arg*>	Binary indicator
enable | yes | on | 1	Fast collective algorithms are used. This is the default value
disable | no | off | 0	Slower and safer collective algorithms are used

Description

The Intel® MPI Library uses advanced collective algorithms designed for better application performance by default. The implementation makes the following assumptions:

- It is safe to utilize the flexibility of the MPI standard regarding the order of execution of the collective operations to take advantage of the process layout and other opportunities.
- There is enough memory available for allocating additional internal buffers.

Set the I_MPI_FAST_COLLECTIVES variable to disable if you need to obtain results that do not depend on the physical process layout or other factors.

NOTE: Some optimizations controlled by this variable are of an experimental nature. In case of failure, turn off the collective optimizations and repeat the run.

I_MPI_BCAST_NUM_PROCS

Control MPI_Bcast algorithm thresholds.

Syntax

I_MPI_BCAST_NUM_PROCS=<*nproc*>

Arguments

<*nproc*>	Define the number of processes threshold for choosing the MPI_Bcast algorithm
> 0	The default value is 8

I_MPI_BCAST_MSG

Control MPI_Bcast algorithm thresholds.

Syntax

I_MPI_BCAST_MSG=<*nbytes1,nbytes2*>

Arguments

<*nbytes1,nbytes2*>	Define the message size threshold range (in bytes) for choosing the MPI_Bcast algorithm
> 0 nbytes2 >= nbytes1	The default pair of values is 12288,524288

Description

Set these variables to control the selection of the three possible MPI_Bcast algorithms according to the following scheme (see Table 3-1 for algorithm descriptions):

1. The first algorithm is selected if the message size is less then <*nbytes1*>, or the number of processes in the operation is less than <*nproc*>.

2. The second algorithm is selected if the message size is greater than or equal to <nbytes1> and less than <nbytes2>, and the number of processes in the operation is a power of two.
3. If none of the above conditions is satisfied, the third algorithm is selected.

I_MPI_ALLTOALL_NUM_PROCS

Control MPI_Alltoall algorithm thresholds.

Syntax

I_MPI_ALLTOALL_NUM_PROCS=<nproc>

Arguments

<nproc>	Define the number of processes threshold for choosing the MPI_Alltoall algorithm
> 0	The default value is 8

I_MPI_ALLTOALL_MSG

Control MPI_Alltoall algorithm thresholds.

Syntax

I_MPI_ALLTOALL_MSG=<nbytes1,nbytes2>

Arguments

<nbytes1,nbytes2>	Defines the message size threshold range (in bytes) for choosing the MPI_Alltoall algorithm
> 0 nbytes2 >= nbytes1	The default pair of values is 256,32768

Description

Set these variables to control the selection of the three possible MPI_Alltoall algorithms according to the following scheme (see Table 3-1 for algorithm descriptions):

1. The first algorithm is selected if the message size is greater than or equal to <nbytes1>, and the number of processes in the operation is not less than <nproc>.
2. The second algorithm is selected if the message size is greater than <nbytes1> and less than or equal to <nbytes2>, or if the message size is less than <nbytes2> and the number of processes in the operation is less than <nproc>.
3. If none of the above conditions is satisfied, the third algorithm is selected.

I_MPI_ALLGATHER_MSG

Control MPI_Allgather algorithm thresholds.

Syntax

I_MPI_ALLGATHER_MSG=<nbytes1,nbytes2>

Arguments

<nbytes1,nbytes2>	Define the message size threshold range (in bytes) for choosing the MPI_Allgather algorithm
> 0 nbytes2 >= nbytes1	The default pair of values is 81920,524288

Description

Set this variable to control the selection of the three possible MPI_Allgather algorithms according to the following scheme (see Table 3-1 for algorithm descriptions):

1. The first algorithm is selected if the message size is less than *<nbytes2>* and the number of processes in the operation is a power of two.
2. The second algorithm is selected if the message size is less than *<nbytes1>* and number of processes in the operation is not a power of two.
3. If none of the above conditions is satisfied, the third algorithm is selected.

I_MPI_ALLREDUCE_MSG

Control MPI_Allreduce algorithm thresholds.

Syntax

I_MPI_ALLREDUCE_MSG=*<nbytes>*

Arguments

<nbytes>	Define the message size threshold (in bytes) for choosing the MPI_Allreduce algorithm
> 0	The default value is 2048

Description

Set this variable to control the selection of the two possible MPI_Allreduce algorithms according to the following scheme (see Table 3-1 for algorithm descriptions):

1. The first algorithm is selected if the message size is less than or equal *<nbytes>*, or the reduction operation is user-defined, or the count argument is less than the nearest power of two less than or equal to the number of processes.
2. If the above condition is not satisfied, the second algorithm is selected.

I_MPI_REDSCAT_MSG

Control the MPI_Reduce_scatter algorithm thresholds.

Syntax

I_MPI_REDSCAT_MSG=*<nbytes1,nbytes2>*

Arguments

<nbytes>	Define the message size threshold range (in bytes) for choosing the MPI_Reduce_scatter algorithm
> 0	The default value is 512,524288

Description

Set this variable to control the selection of the three possible MPI_Reduce_scatter algorithms according to the following scheme (see Table 3-1 for algorithm descriptions):

1. The first algorithm is selected if the reduction operation is commutative and the message size is less than *<nbytes2>*.
2. The second algorithm is selected if the reduction operation is commutative and the message size is greater than or equal to *<nbytes2>*, or if the reduction operation is not commutative and the message size is greater than or equal to *<nbytes1>*.
3. If none of the above conditions is satisfied, the third algorithm is selected.

I_MPI_SCATTER_MSG

Control MPI_Scatter algorithm thresholds.

Syntax

I_MPI_SCATTER_MSG=<*nbytes*>

Arguments

<*nbytes*>	Define the buffer size threshold range (in bytes) for choosing the MPI_Scatter algorithm
> 0	The default value is 2048

Description

Set this variable to control the selection of the two possible MPI_Scatter algorithms according to the following scheme (see Table 3-1 for algorithm descriptions):

1. The first algorithm is selected on the intercommunicators if the message size is greater than <*nbytes*>.
2. If the above condition is not satisfied, the second algorithm is selected.

I_MPI_GATHER_MSG

Control MPI_Gather algorithm thresholds.

Syntax

I_MPI_GATHER_MSG=<*nbytes*>

Arguments

<*nbytes*>	Define the buffer size threshold range (in bytes) for choosing the MPI_Gather algorithm
> 0	The default value is 2048

Description

Set this variable to control the selection of the two possible MPI_Gather algorithms according to the following scheme (see Table 3-1 for algorithm descriptions):

1. The first algorithm is selected on the intercommunicators if the message size is greater than <*nbytes*>.
2. If the above condition is not satisfied, the second algorithm is selected.

3.6 Miscellaneous

I_MPI_TIMER_KIND

Select the timer used by the MPI_Wtime and MPI_Wtick calls.

Syntax

I_MPI_TIMER_KIND=<*timername*>

Arguments

<*timername*>	Define the timer type
gettimeofday	If this setting is chosen, the MPI_Wtime and MPI_Wtick functions will

	work through the function `gettimeofday(2)`. This is the default value
`rdtsc`	If this setting is chosen, the `MPI_Wtime` and `MPI_Wtick` functions will use the high resolution RDTSC timer

Description

Set this variable to select either the ordinary or RDTSC timer.

NOTE: The resolution of the default `gettimeofday(2)` timer may be insufficient on certain platforms.

4 Statistics Gathering Mode

Intel® MPI Library has a built-in statistics gathering facility that collects essential performance data without disturbing the application execution. The collected information is output onto an easy to interpret text file. This section describes the environment variables used to control the built-in statistics gathering facility and provides example output files.

I_MPI_STATS

Control the statistics information output.

Syntax

I_MPI_STATS=<*level*>

Arguments

<*level*>	Indicate level of statistics information provided
0	Do not collect any statistics. This is the default value
1	Output the amount of data sent by each process
2	Output the number of calls and amount of transferred data
> 2	Output collective operation statistics for all communication contexts

Description

Set this variable to control the amount of the statistics information output onto the log file. No statistics are output by default.

I_MPI_STATS_SCOPE

Select the subsystem(s) for output statistics.

Syntax

I_MPI_STATS_SCOPE=<*subsystem*>[:<*ops*>][;<*subsystem*>[:<*ops*>][…]]

Arguments

<*subsystem*>	Define the target subsystem(s)
all	Collect statistics data for all operations. This is default value
coll	Collect statistics data for all collective operations
p2p	Collect statistics data for all point to point operations

<*ops*>	Define the target operations as a comma separated list
Allgather	MPI_Allgather
Allgatherv	MPI_Allgatherv
Allreduce	MPI_Allreduce

Alltoall	MPI_Alltoall
Alltoallv	MPI_Alltoallv
Alltoallw	MPI_Alltoallw
Barrier	MPI_Barrier
Bcast	MPI_Bcast
Exscan	MPI_Exscan
Gather	MPI_Gather
Gatherv	MPI_Gatherv
reduce_scatter	MPI_Reduce_scatter
Reduce	MPI_Reduce
Scan	MPI_Scan
Scatter	MPI_Scatter
Scatterv	MPI_Scatterv
Send	Standard transfers (MPI_Send, MPI_Isend, MPI_Send_init)
Bsend	Buffered transfers (MPI_Bsend, MPI_Ibsend, MPI_Bsend_init)
Csend	Point to point operations inside the collectives
Rsend	Ready transfers (MPI_Rsend, MPI_Irsend, MPI_Rsend_init)
Ssend	Synchronous transfers (MPI_Ssend, MPI_Issend, MPI_Ssend_init)

Description

Set this variable to select the target subsystem. All collective and point to point operations, including point to point support of the collectives are covered by default.

Examples:

1. The default settings are equivalent to:
   I_MPI_STATS_SCOPE=coll;p2p
2. Use the following settings to collect statistics for MPI_Bcast, MPI_Reduce, and all point to point operations:
   I_MPI_STATS_SCOPE=p2p;coll:bcast,reduce
3. Use the following settings to collect statistics for point to point operations inside the collectives:
   I_MPI_STATS_SCOPE=p2p:csend

I_MPI_STATS_FILE

Define the statistics output file name

Syntax

I_MPI_STATS_FILE=*<name>*

Arguments

<name>	Define the statistics output file name

Description

Set this variable to define the statistics output file. The stats.txt file is created in the current directory by default.

The statistics data is blocked and ordered according to the process ranks in the MPI_COMM_WORLD communicator. The timing data is presented in microseconds.

For example, with the following settings in effect

```
I_MPI_STATS=2
I_MPI_STATS_SCOPE=p2p;coll:bcast
```

the statistics output for a simple program that performs only one MPI_Bcast operation may look as follows:

```
Intel(R) MPI Library Version 3.1
____ MPI Communication Statistics ____

Stats level: 2
P2P scope: <FULL>
Collective scope: <Bcast>

~~~~ Process 0 of 2 on node host1

Data Transfers
Src    Dst    Volume(MB)  Transfers
----------------------------------------
000 --> 000  0.000000e+00      0
000 --> 001  9.384155e-02      6
========================================
Totals 9.384155e-02      6

Communication Activity
Operation    Volume(MB)  Calls
----------------------------------------
P2P
Csend        9.384155e-02      6
Send         0.000000e+00      0
Bsend        0.000000e+00      0
Rsend        0.000000e+00      0
Ssend        0.000000e+00      0
Collectives
Bcast        9.384155e-02      6
========================================

~~~~ Process 1 of 2 on node host1

Data Transfers
Src    Dst    Volume(MB)  Transfers
----------------------------------------
001 --> 000  0.000000e+00      0
```

```
001 --> 001  0.000000e+00      0
=========================================
Totals 0.000000e+00      0

Communication Activity
Operation    Volume(MB)  Calls
-----------------------------------------
P2P
Csend        0.000000e+00      0
Send         0.000000e+00      0
Bsend        0.000000e+00      0
Rsend        0.000000e+00      0
Ssend        0.000000e+00      0
Collectives
Bcast        9.384155e-02      6
=========================================

____ End of stats.txt file ____
```

5 Unified Memory Management

Intel® MPI Library provides a way to replace the memory management subsystem by a user defined package. The following function pointers may optionally be set by the user:

- `i_malloc`
- `i_calloc`
- `i_realloc`
- `i_free`

These pointers also affect the C++ new and delete operators.

The respective standard C library functions are used by default.

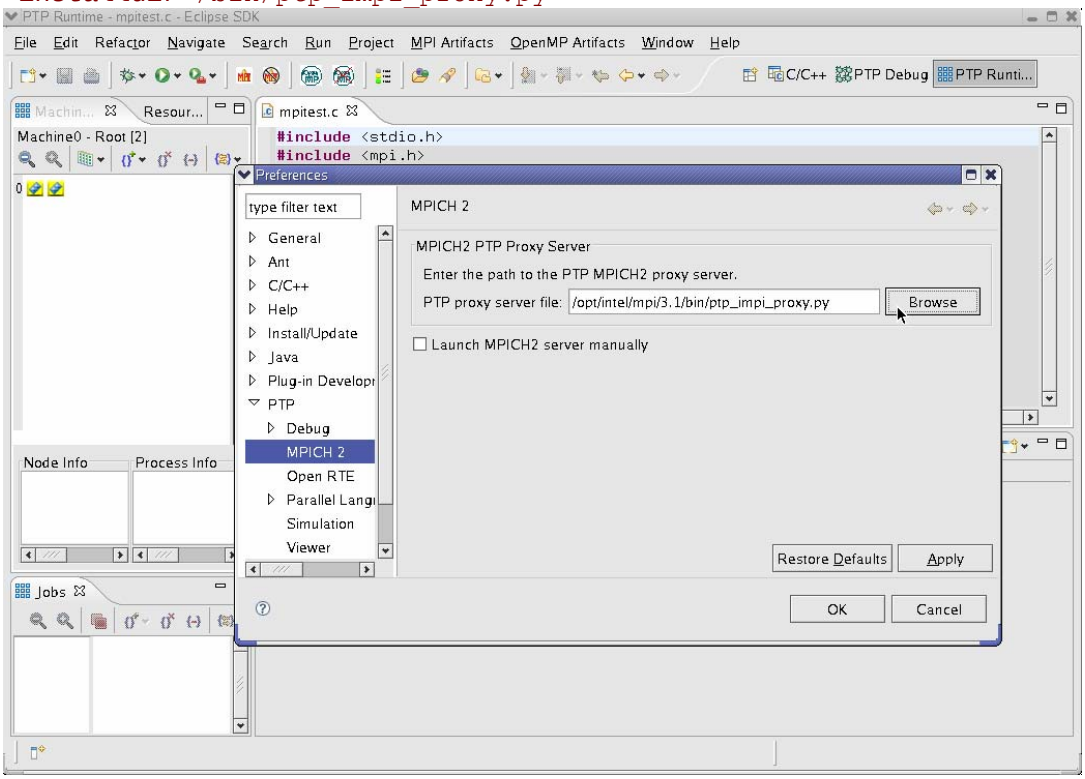
The following contrived source code snippet illustrates the usage of the unified memory subsystem:

```
#include <i_malloc.h>
#include <my_malloc.h>

int main( int argc, int argv )
{
    // override normal pointers
    i_malloc = my_malloc;
    i_calloc = my_calloc;
    i_realloc = my_realloc;
    i_free = my_free;

#ifdef _WIN32
    // also override pointers used by DLLs
    i_malloc_dll = my_malloc;
    i_calloc_dll = my_calloc;
    i_realloc_dll = my_realloc;
    i_free_dll = my_free;
#endif

    // now start using Intel(R) libraries
}
```

6 Integration into Eclipse* PTP

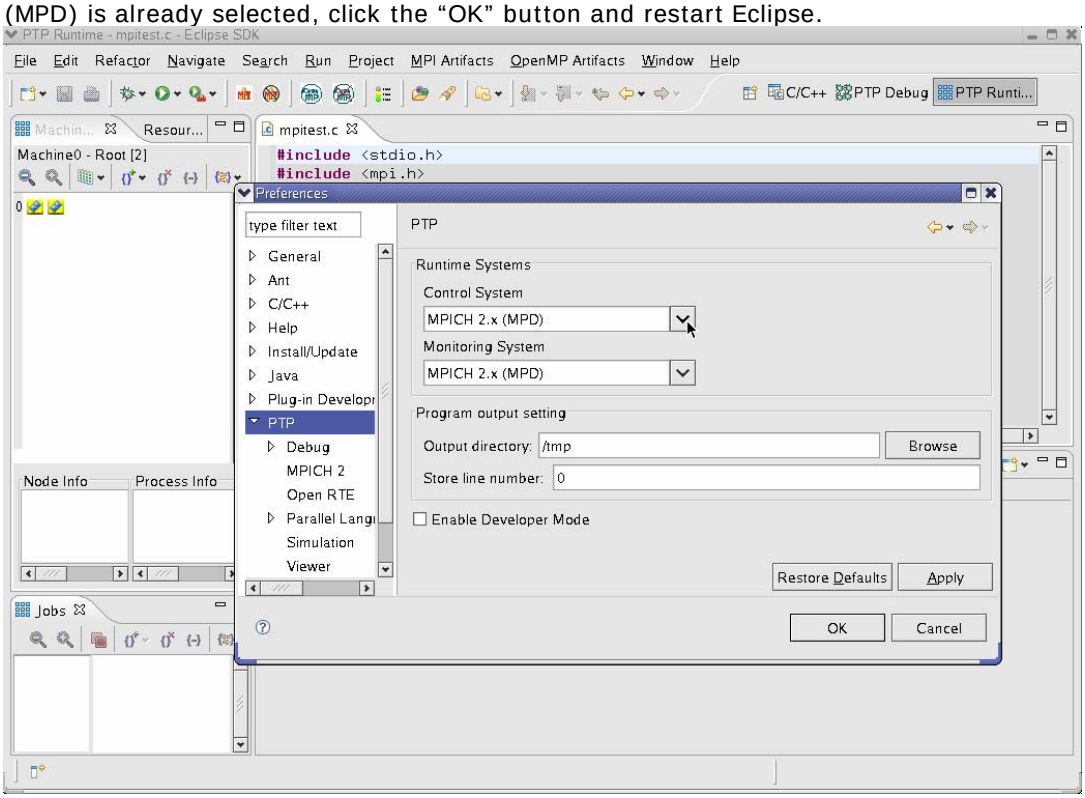
Intel® MPI Library can be used with Eclipse Parallel Tools Platform (PTP). You can launch parallel applications on the existing MPD ring from Eclipse PTP graphical user interface. The MPD ring must be started prior the PTP startup. Perform the following configuration steps to use PTP with the Intel® MPI Library:

1. Set the `PTPPATH` environment variable to specify location of the `ptplib.py` module.
2. Select Window->Preferences from the Eclipse main menu. Select PTP->MPICH 2 preference page.
3. Specify the full path to the `ptp_impi_proxy.py` file. Click the “Apply” button. For example `<installdir>/bin/ptp_impi_proxy.py`



4. Go to the PTP preference page.

5. Select MPICH2 (MPD) in both Control System and Monitoring System drop down menu. If MPICH2 (MPD) is already selected, click the “OK” button and restart Eclipse.



6. Switch to the PTP Runtime perspective.
7. In the Machines view you see cluster nodes, on which the MPD ring is currently working.
8. Refer to PTP User’s Guide for more information. The PTP documentation available at: http://www.eclipse.org/ptp/doc.php