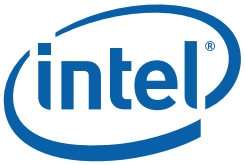


Intel® MPI Library for Linux* OS

Reference Manual

Document Number: 315399-008

Revision: 4.0

World Wide Web: http://www.intel.com

Contents

1 About this Document 6
1.1 Intended Audience 6
1.2 Using Doc Type Field 6
1.3 Conventions and Symbols 7
1.4 Related Information 7

2 Command Reference 8
2.1 Compiler Commands 8
2.1.1 Compiler Command Options 8
2.1.2 Configuration Files 11
2.1.3 Profiles 12
2.1.4 Environment Variables 12
2.2 Job Startup Commands 15
2.2.1 Extended Device Control Options 15
2.2.2 Global Options 17
2.2.3 Local Options 22
2.2.4 Configuration Files 23
2.2.5 Environment Variables 23
2.3 Simplified Job Startup Command 29
2.4 Experimental Scalable Process Management System (Hydra) 29
2.4.1 Global Options 30
2.4.2 Local Options 32
2.4.3 Environment Variables 33
2.5 Multipurpose Daemon Commands 36
2.5.1 Configuration Files 42
2.5.2 Environment Variables 43
2.6 Processor Information Utility 45

3 Tuning Reference 48
3.1 Automatic Tuning Utility 48
3.1.1 Cluster-specific Tuning 50
3.1.2 Application-specific Tuning 51
3.1.3 Tuning Utility Output 51
3.2 Process Pinning 51
3.2.1 Process Identification 51
3.2.2 Environment Variables 52
3.2.3 Interoperability with OpenMP* 57
3.3 Fabrics Control 63
3.3.1 Communication Fabrics Control 63
3.3.2 Shared Memory Control 69
3.3.3 DAPL-capable Network Fabrics Control 73
3.3.4 DAPL UD-capable Network Fabrics Control 80
3.3.5 TCP-capable Network Fabrics Control 85
3.3.6 TMI-capable Network Fabrics Control 86
3.3.7 OFA*-capable Network Fabrics Control 87
3.3.8 Failover Support in the OFA* Device 89
3.4 Dynamic Process Support 90
3.5 Collective Operation Control 91
3.5.1 I_MPI_ADJUST family 91
3.5.2 I_MPI_MSG family 94
3.6 Extended File System Support 98
3.6.1 Environment variables 99
3.7 Compatibility Control 99
3.8 Miscellaneous 100

4 Statistics Gathering Mode 101

5 Fault Tolerance 107
5.1 Environment Variables 107
5.2 Usage Model 108

6 ILP64 Support 109
6.1 Using ILP64 109
6.2 Known Issues and Limitations 109
7 Unified Memory Management 110
8 Integration into Eclipse* PTP 111
9 Glossary 113
10 Index 114

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
315399-003	3.1 build 038	Local options were added. Sections “Index”, “Glossary”, “Process Identification”, and “Interoperability with OpenMP*” were added	/03/05/2008
315399-004	3.2	Sections “Process pinning”, Automatic Tuning Utility, and “Statistic Gathering Mode” were updated	/09/05/2008
315399-005	3.2 Update 1	Section “ILP64 Support” was added, section “Interoperability with OpenMP*” was updated	/03/04/2009
315399-006	4.0 Engineering Preview	Sections “Processor Information Utility”, “Automatic Tuning Utility”, and “Device Control” were updated	/06/24/2009
315399-007	4.0 Beta	Sections “Processor Information Utility”, “Fabrics Control”, “Statistics Gathering Mode “, and “ILP64 Support” were updated. Section “Fault Tolerance” was added	/11/03/2009
315399-008	4.0	Sections “Fabrics Control”, “Environment Variables”, and “Job Startup Command” were updated. Sections “Experimental Scalable Process Management System (Hydra) “, and “Dynamic Process Support” were added.	/02/16/2009

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 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. The 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 to upgrade processors and interconnects as new technology becomes available, and achieves maximum application performance without changes to the software or to the operating environment.

The library is provided in the following kits:

- *The Intel® MPI Library Runtime Environment* (RTO) has the tools you need to run programs, including multipurpose daemon* (MPD) and supporting utilities, shared (.so) libraries, and documentation.
- *The Intel® MPI Library Development Kit* (SDK) includes all of the Runtime Environment components plus compilation tools, including compiler commands such as `mpiicc`, 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 the 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 1.2-1 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 how to obtain statistics of MPI communication operations
Section 5 ILP64 Support	Section 5 describes support provided for the ILP64 programming model

Section 6 Unified Memory Management	Section 6 describes the unified memory management subsystem (`i_malloc`)
Section 7 Integration into Eclipse* PTP	Section 7 describes the procedure for integration into Eclipse* Parallel Tools Platform
Section 8 Glossary	Section 8 explains basic terms used in this document
Section 9 Index	Section 9 references options and variable names

1.3 Conventions and Symbols

The following conventions are used in this document.

Table 1.3-1 Conventions and Symbols used in this Document

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

1.4 Related Information

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 2.1-1 The Intel® MPI Library Compiler Drivers

Compiler Command	Default Compiler	Supported Language(s)	Supported ABI(s)
Generic 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 Cersions 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 10.0, 10.1, 11.0, 11.1 and Higher			
mpiicc	icc	C	32/64 bit
mpiicpc	icpc	C++	32/64 bit
mpiifort	ifort	Fortran77/Fortran 95	32/64 bit

- Compiler commands are available only in the Intel® MPI Library Development Kit.
- Compiler commands are in the *<installdir>/<arch>*/bin directory. Where *<installdir>* refers to the Intel® MPI Library installation directory and *<arch>* is one of the following architectures:
  - ia32 – IA-32 architecture binaries
  - intel64 – Intel® 64 architecture binaries
- 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.

The MPI_THREAD_FUNNELED level is provided by default by the thread safe version of the Intel® MPI library.

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.

-static

Use this option to link the Intel® MPI library statically. This option is passed to a compiler.

-config=<*name*>

Use this option to source the configuration file. See *Configuration Files* for details.

-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>*/*<arch>*/etc. 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 mpiicc or another compiler script.

Use the -t=log or -trace=log option to link the resulting executable against the logging Intel® MPI Library 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 I_MPI_TRACE_PROFILE to the *<profile_name>* environment variable to specify another profiling library. For example, set I_MPI_TRACE_PROFILE to vtfs to link against the fail-safe version of the Intel® Trace Collector.

-check_mpi

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 `mpiicc` or another compiler script.

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

-ilp64

Use this option to enable ILP64 support. All integer arguments of the Intel MPI Library are treated as 64-bits values in this case.

NOTE: If you specify the `-i8` option for the Intel® Fortran Compiler, you still have to use the ILP64 option for linkage. See *ILP64 Support* for details.

-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.

-fast

Use this Intel compiler option to maximize speed across the entire program. This option forces static linkage method for the Intel® MPI Library.

NOTE: It works for `mpiicc`, `mpiicpc`, and `mpiifort` Intel compiler drivers only.

-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:

```
$ mpiicc -show -c test.c
```

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

```
$ mpiicc -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, 4.2.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.

-v

Use this option to print the compiler driver script version and its native compiler version.

2.1.2 Configuration Files

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

```
<installdir>/<arch>/etc/mpi<compiler>-<name>.conf
```

where:

`<arch>` = {ia32,intel64} for the IA-32, and the Intel® 64 architectures

`<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 these file or use the `-config` option, 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>/<arch>/etc` directory. The Intel® MPI Library comes with several predefined profiles for the 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 `mpiicc` 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 the -trace option to mpiicc 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_mpi 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_mpi option to mpiicc or another Intel® MPI Library compiler driver.

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

I_MPI_CHECK_COMPILER

Turn on/off compiler compatibility check.

Syntax

I_MPI_CHECK_COMPILER=*<arg>*

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Enable checking the compiler
disable | no | off | 0	Disable checking the compiler. This is the default value

Description

If I_MPI_CHECK_COMPILER is set to enable, the Intel MPI compiler drivers check the underlying compiler for compatibility. Normal compilation will be performed only if known version of underlying compiler is used.

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.

NOTE: The configuration file is sourced if it exists for a specified compiler. See *Configuration Files* for details.

I_MPI_ROOT

Set the 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 the 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 the 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 value is *<installdir>*/*<arch>*/etc

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 Extended Device Control Options

Use these options to select a specific fabric combination.

The exact combination of fabrics depends on the number of processes started per node.

If all processes start on one node, the Intel® MPI library uses `shm` intra-node communication regardless of the selected option from the list in this topic.

If the number of started processes is less than or equal to the number of available nodes, the library uses the first available fabric from the list of fabrics for inter-nodes communication.

For other cases, the library uses `shm` for intra-node communication, and the first available fabric from the list of fabrics for inter-nodes communication. See *I_MPI_FABRICS* and *I_MPI_FABRICS_LIST* for more details.

-rdma

Use this option to select an RDMA-capable network fabric for inter-nodes communication. The application attempts to use first available RDMA-capable network fabric from the list `dapl` or `ofa`. If no such fabric is available, other fabrics from the list `tcp` or `tmi` are used. This option is equivalent to the `-genv I_MPI_FABRICS_LIST dapl,ofa,tcp,tmi -genv I_MPI_FALLBACK 1` setting.

-RDMA

Use this option to select an RDMA-capable network fabric for inter-nodes communication. The application attempts to use first available RDMA-capable network fabric from the list `dapl` or `ofa`. The application fails if no such fabric is found. This option is equivalent to the `-genv I_MPI_FABRICS_LIST dapl,ofa -genv I_MPI_FALLBACK 1` setting.

-dapl

Use this option to select DAPL capable network fabric for inter-nodes communication. The application attempts to use DAPL capable network fabric. If no such fabric is available, another fabrics from the list `tcp,tmi` or `ofa` is used. This option is equivalent to the `-genv I_MPI_FABRICS_LIST dapl,tcp,tmi,ofa -genv I_MPI_FALLBACK 1` setting.

-DAPL

Use this option to select DAPL capable network fabric for inter-nodes communication. The application fails if no such fabric is found. This option is equivalent to the `-genv I_MPI_FABRICS_LIST dapl -genv I_MPI_FALLBACK 0` setting.

-ib

Use this option to select OFA capable network fabric for inter-nodes communication. The application attempts to use OFA capable network fabric. If no such fabric is available, another fabrics from the list `dapl,tcp` or `tmi` is used. This option is equivalent to the `-genv I_MPI_FABRICS_LIST ofa,dapl,tcp,tmi -genv I_MPI_FALLBACK 1` setting.

-IB

Use this option to select OFA capable network fabric for inter-nodes communication. The application fails if no such fabric is found. This option is equivalent to the `-genv I_MPI_FABRICS_LIST ofa -genv I_MPI_FALLBACK 0` setting.

-tmi

Use this option to select TMI capable network fabric for inter-nodes communication. The application attempts to use TMI capable network fabric. If no such fabric is available, another fabrics from the list `dapl,tcp` or `ofa` is used. This option is equivalent to the `-genv I_MPI_FABRICS_LIST tmi,dapl,tcp,ofa -genv I_MPI_FALLBACK 1` setting.

-TMI

Use this option to select TMI capable network fabric for inter-nodes communication. The application will fail if no such fabric is found. This option is equivalent to the `-genv I_MPI_FABRICS_LIST tmi -genv I_MPI_FALLBACK 0` setting.

-mx

Use this option to select Myrinet MX* network fabric for inter-nodes communication. The application attempts to use Myrinet MX* network fabric. If no such fabric is available, another fabrics from the list

dapl,tcp or ofa is used. This option is equivalent to the -genv I_MPI_FABRICS_LIST tmi,dapl,tcp,ofa -genv I_MPI_TMI_PROVIDER mx -genv I_MPI_DAPL_PROVIDER mx -genv I_MPI_FALLBACK 1 setting.

-MX

Use this option to select Myrinet MX* network fabric for inter-nodes communication. The application fails if no such fabric is found. This option is equivalent to the -genv I_MPI_FABRICS_LIST tmi -genv I_MPI_TMI_PROVIDER mx -genv I_MPI_FALLBACK 0 setting.

-psm

Use this option to select Qlogic* network fabric for inter-nodes communication. The application attempts to use Qlogic* network fabric. If no such fabric is available, another fabrics from the list dapl,tcp or ofa is used. This option is equivalent to the -genv I_MPI_FABRICS_LIST tmi,dapl,tcp,ofa -genv I_MPI_TMI_PROVIDER psm -genv I_MPI_FALLBACK 1 setting.

-PSM

Use this option to select Qlogic* network fabric for inter-nodes communication. The application fails if no such fabric is found. This option is equivalent to the -genv I_MPI_FABRICS_LIST tmi -genv I_MPI_TMI_PROVIDER psm -genv I_MPI_FALLBACK 0 setting.

-gm

Use this option to select Myrinet* GM* network fabric for inter-nodes communication. This option is equivalent to the -genv I_MPI_DEVICE rdssm:GmHca0 -genv I_MPI_FALLBACK_DEVICE 1 setting.

NOTE: This variable is deprecated and supported mostly for backward compatibility.

-GM

Use this option to select Myrinet* GM* network fabric for inter-nodes communication. The application fails if no such fabric is found. This option is equivalent to the -genv I_MPI_DEVICE rdssm:GmHca0 -genv I_MPI_FALLBACK_DEVICE 0 setting.

NOTE: This variable is deprecated and supported mostly for backward compatibility.

2.2.2 Global Options

-version or -V

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

-h or -help or --help

Use this option to display the mpiexec help message.

-tune [<*configuration_file*>]

Use this option to optimize the Intel® MPI Library performance using the data collected by the mpitune utility. If *<configuration_file>* is not mentioned, the best-fit tune options will be selected for the given configurations. Otherwise the given configuration file will be used.

The default location of the configuration files is *<installdir>/<arch>*/etc directory. Set the I_MPI_TUNER_DATA_DIR environment variable to override the default location.

See *Automatic Tuning Utility* for more details.

-nolocal

Use this option to avoid running *<executable>* on the host where the 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 total number of processes to start is controlled by the -n option 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.

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* for details. The first *<# of processes>* indicated by the -perhost option will be executed on the first host; the next *<# of processes>* will be executed on the next host, and so on.

See also the *I_MPI_PERHOST* variable.

-rr

Use this option to place consecutive MPI processes onto different host in round robin fashion. This option is equivalent to -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 -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 -perhost *<# of processes>*.

-machinefile *<machine file>*

Use this option to control the process placement through *<machine file>*. The total number of processes to start is controlled by the -n option 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 files:

```
host1
host1
host2
host2
host3
```

is equivalent to:

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

It is also possible to specify the network interface used for communication for each node: `<host name>:<number of processes> [ifhn=<interface_host_name>]`.

NOTE: The `-machinefile`, `-ppn`, `-rr`, and `-perhost` options are intended for process distribution. Do not use them simultaneously. Otherwise `-machinefile` will take precedence.

-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. During the application startup, the default value is the `-genvuser` option. The options `-genvnone, -genvuser, -genvall` have the lowest priority, `-genvlist, -genvexcl` have higher priority than the previous set. The `-genv` option has the highest priority. Local options have higher priority then the global options.

-genv <*ENVVAR*> <*value*>

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

-genvuser

Use this option to propagate all user environment variables to all MPI processes, with the exception of the following system variables: `$HOSTNAME, $HOST, $HOSTTYPE, $MACHTYPE, $OSTYPE`. This is the default setting.

-genvall

Use this option to enable propagation of all environment variables to all MPI processes.

-genvnone

Use this option to suppress propagation of any environment variables to any MPI processes.

(SDK only) -trace [<*profiling_library*>] or -t [<*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_mpi [<*checking_library*>]

Use this option to check your MPI application using the indicated `<checking_library>`. If `<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.

NOTE: Make sure that environment variable `TVDSVRLAUNCHCMD=ssh`, as the TotalView* uses `rsh` by default.

NOTE: The TotalView* debugger has a feature to displays the message queue state of your MPI program. To use the state display feature, do the following steps:

1. Run your `<executable>` with `-tv` option.

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

2. Answer **Yes** to the question about stopping the Python* job.

To display the internal state of the MPI library textually, select the **Tools** > **Message Queue** command. If you select the **Process Window Tools** > **Message Queue Graph** command, the TotalView* displays a window that shows a graph of the current message queue state. For more information, see TotalView*.

-tva <jobid>

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

```
$ mpiexec -tva <jobid>
```

-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 the standard output and standard error streams.

NOTE: For this option 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 the standard output.

-s <spec>

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

Arguments

<spec>	Define MPI process ranks
all	Use all processes
<l>,<m>,<n>	Specify an exact list and use processes <l>, <m> and <n> only. The 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 the section *Configuration Files.*

-ifhn <interface/hostname>

Use this option to specify the network interface for communication with the local MPD daemon. The <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>.

-configfile <filename>

Use this option to specify the file <filename> that contains command-line options. Blank lines and lines that start with '#' as the first character are ignored. 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 a MPI application according to the parameters above, use:

```
$ mpiexec -configfile <filename>
```

NOTE: This option may only be used alone. It terminates parsing of the `mpiexec` command line.

2.2.3 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.

-envuser

Use this option to propagate all user environment variables with the exception of the following variables: `$HOSTNAME, $HOST, $HOSTTYPE, $MACHTYPE, $OSTYPE`. This is the default setting.

-envall

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

-envnone

Use this option to suppress propagation of any environment variables to the MPI processes in the current arg-set.

-envlist <list of env var names>

Use this option to pass a list of environment variables with their current values. `<list of env var names>` is a comma separated list of variables to be sent into the processes. If this option is used several times in the command line, all variables listed in the arguments will be included into one list.

-envexcl <list of env var names>

Use this option to suppress propagation of the listed environment variables to the MPI processes in the current arg-set.

-host <nodename>

Use this option to specify a 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
```

-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.

2.2.4 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. The default location of the configuration file is the <installdir>/<arch>/etc.
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>
```

2.2.5 Environment Variables

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 the 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>] <message>
```

where <identifier> identifies the MPI process that produced the message, while <message> contains the debugging output.

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

For example, the following command:

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

may produce the following output:

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

while the command

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

may produce the following output:

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

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

I_MPI_PERHOST

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

Syntax

I_MPI_PERHOST=<value>

Arguments

<value>	Define the default process layout
<n> > 0	<n> processes per node
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 called in the command line, the I_MPI_PERHOST variable has no effect. The -perhost option assumes the value of the I_MPI_PERHOST variable if this variable 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.

(SDK only) I_MPI_JOB_TRACE_LIBS

(MPIEXEC_TRACE_LIBS)

Choose the libraries to preload through the –trace option.

Syntax

I_MPI_JOB_TRACE_LIBS=<arg>

Deprecated Syntax

MPIEXEC_TRACE_LIBS=<arg>

Arguments

<arg>	String parameter
<list>	Blank separated list of libraries to preload. The default value is vt

Description

Set this variable to choose an alternative library for preloading by the -trace option.

(SDK only) I_MPI_JOB_CHECK_LIBS

Choose the libraries to preload through the -check_mpi option.

Syntax

I_MPI_JOB_CHECK_LIBS=<arg>

Arguments

<arg>	String parameter
<list>	Blank separated list of libraries to preload. The default value is vtmc

Description

Set this variable to choose an alternative library for preloading by the -check_mpi 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
<n> $\geqslant$ 0	The default timeout value is 20 seconds

Description

Set this variable to make mpiexec wait for 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
<n> ⩾ 0	The default timeout value is zero, meaning 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 to be used when a job is terminated due to a timeout.

Syntax

I_MPI_JOB_TIMEOUT_SIGNAL=<number>

Deprecated Syntax

MPIEXEC_TIMEOUT_SIGNAL=<number>

Arguments

<number>	Define signal number
<n> > 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 signal number unsupported 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.

disable | no | off | 0	Turn off propagation. This is the default value

Description

Set this variable to control propagation of the signals (SIGINT, SIGTSTP, SIGCONT, SIGALARM, and SIGTERM) that may be received by the MPD daemons. If signal propagation is enabled, the received signal is sent to all processes of the MPI job. If signal propagation is disabled, all processes of the MPI job are stopped with the default signal 9 (SIGKILL).

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
<*n*> > 0	The default chunk size value is 1 KB

Description

Set this variable to increase the size of the buffer used to intercept the standard output and standard error streams from the 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 Intel® MPI Library Process Management Interface (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. If supported, the extensions substantially decrease task startup time. Set I_MPI_PMI_EXTENSIONS to disable if your process manager does not support these extensions.

I_MPI_JOB_FAST_STARTUP

(I_MPI_PMI_FAST_STARTUP)

Turn on/off the faster Intel® MPI Library process startup algorithm.

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 startup. This is the default value
disable | no | off | 0	Turn off the algorithm for fast startup

Description

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

TOTALVIEW*

Select a 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 the 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 the installation directory of the Intel® Debugger.

I_MPI_TUNER_DATA_DIR

Set an alternate path to the directory with the tuning configuration files.

Syntax

I_MPI_TUNER_DATA_DIR=<path>

Arguments

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

Description

Set this variable to specify an alternative location of the tuning configuration files.

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*, OpenPBS*, LSF*, Parallelnavi* NQS*, SLURM*, or Sun* Grid Engine*. 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 this case 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.

2.4 Experimental Scalable Process Management System (Hydra)

mpiexec.hydra

Use the experimental `mpiexec.hydra` utility to run Intel MPI application on a large cluster.

Syntax

`mpiexec.hydra <g-options> <l-options> <executable>`

or

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

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

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.hydra -f <hostsfile> –n <# of processes> ./a.out
```

<hostsfile> is the path/name of the file that has the list of machine names on which the application to run.

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:

```
$ cat hosts.file
host1:2
host2:2
$ mpiexec.hydra -f hosts.file –env <VAR1> <VAL1> –n 2 ./a.out : \
            -env <VAR2> <VAL2> -n 2 ./b.out
```

To start a job by mpiexec.hydra, the daemons are not required to be run before using the mpiexec.hydra command.

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

2.4.1 Global Options

-f *<hostsfile>*

Use this option to specify machine names to run application. List machine names line by line.

```
host1
host2
host3
```

Use colon with number of processes for required processes distribution across the machines.

```
host1:2
host2:3
```

Comments are started with #.

```
host1:2      # the first 2 processes will be run here
host2:3      # the rest 3 processes will be run on this host
```

See also the I_MPI_HYDRA_HOST_FILE variable.

-genv <ENVVAR> <value>

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

-genvall

Use this option to enable propagation of all environment variables to all MPI processes.

-genvnone

Use this option to suppress propagation of any environment variables to any MPI processes.

-genvlist <list of genv var names>

Use this option to pass a list of environment variables with their current values. <list of genv var names> is a comma separated list of variables to be sent into the processes.

-wdir <directory>

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

-pmi-connect <mode>

Use this option to choose the PMI connections method. Possible values are: mpich2, proxy, cache.

It is case sensitive.

- The mpich2 mode is the original mpich2 PMI connections mode. In this case, PMI connections are organized between MPI process and mpiexec.hydra.
- The proxy mode – PMI connections through pmi_proxy.
- The cache mode – PMI connections through pmi_proxy with PMI information caching on local pmi_proxies to minimize PMI requests.

The cache mode is the default PMI connections method.

See also the I_MPI_HYDRA_PMI_CONNECT variable.

2.4.1.1 Bootstrap Options

-bootstrap <bootstrap server>

Use this option to set bootstrap server to use. A bootstrap server is the basic remote node access mechanism that is provided on any system. Hydra supports multiple runtime bootstrap servers such as ssh, rsh, fork, and slurm to launch processes. The default bootstrap server is ssh.

See also the I_MPI_HYDRA_BOOTSTRAP variable.

-bootstrap-exec <bootstrap server>

Use this option to set executable bootstrap server to run. Possible values are ssh, rsh, fork, and slurm. The default bootstrap server is ssh.

See also the I_MPI_HYDRA_BOOTSTRAP_EXEC variable.

2.4.1.2 Communication Sub-system Options

-rmk <*RMK*>

Use this option to run the resource management kernel.

See also the *I_MPI_HYDRA_RMK* variable.

2.4.1.3 Other Options

-verbose

Use this option to print extra verbose information

See also the *I_MPI_HYDRA_DEBUG* variable.

-print-rank-map

Use this option to print rank mapping.

-print-all-exitcodes

Use this option to print exit codes of all processes.

2.4.2 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 the specified `<value>` for all MPI processes in the current arg-set.

-envall

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

See also the *I_MPI_HYDRA_ENV* variable.

-envnone

Use this option to suppress propagation of any environment variables to the MPI processes in the current arg-set.

-envlist <*list of env var names*>

Use this option to pass a list of environment variables with their current values. `<list of env var names>` is a comma separated list of variables to be sent into the processes.

2.4.3 Environment Variables

I_MPI_HYDRA_HOST_FILE

(HYDRA_HOST_FILE)

Set the hosts file to run the application.

Syntax

`I_MPI_HYDRA_HOST_FILE=<arg>`

Deprecated Syntax

`HYDRA_HOST_FILE=<arg>`

Arguments

<arg>	String parameter
<hostsfile>	Full or relative path to hosts file

Description

Set this variable to specify the hosts file.

I_MPI_HYDRA_DEBUG

(HYDRA_DEBUG)

Print out the debug information.

Syntax

`I_MPI_HYDRA_DEBUG=<arg>`

Deprecated Syntax

`HYDRA_DEBUG=<arg>`

Arguments

<arg>	Binary indicator
`0 | 1`	Turn on or off the debug output. The default value is `0`

Description

Set this variable to `1` to enable the debug mode and `0` to turn off the debug mode.

I_MPI_HYDRA_ENV

(HYDRA_ENV)

Set it to `all` to pass the environment.

Syntax

`I_MPI_HYDRA_ENV=<arg>`

Deprecated Syntax

`HYDRA_ENV=<arg>`

Arguments

<arg>	String parameter

all	To pass the launching node environment to the application processes

Description

By default, the launching node environment is passed to the executables as long as it does not overwrite any of the environment variables that have been preset by the remote shell.

I_MPI_MPIEXEC_TIMEOUT

(MPIEXEC_TIMEOUT)

Set the mpiexec timeout.

Syntax

I_MPI_MPIEXEC_TIMEOUT=*<timeout>*

Deprecated Syntax

MPIEXEC_TIMEOUT=*<timeout>*

Arguments

<timeout>	Define mpiexec timeout period in seconds
<n> ⩾ 0	The default timeout value is zero, which means 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.

I_MPI_HYDRA_BOOTSTRAP

(HYDRA_BOOTSTRAP)

Set the bootstrap server.

Syntax

I_MPI_HYDRA_BOOTSTRAP=*<arg>*

Deprecated Syntax

HYDRA_BOOTSTRAP=*<arg>*

Arguments

<arg>	String parameter
ssh | rsh | fork | slurm	The remote node access mechanism. The default is ssh

Description

Set this variable to specify the bootstrap server.

I_MPI_HYDRA_BOOTSTRAP_EXEC

(HYDRA_BOOTSTRAP_EXEC)

Set the bootstrap server to run application.

Syntax

I_MPI_HYDRA_BOOTSTRAP_EXEC=<arg>

Deprecated Syntax

HYDRA_BOOTSTRAP_EXEC=<arg>

Arguments

<arg>	String parameter
ssh | rsh | fork | slurm	The remote node access mechanism. The default is ssh

Description

Set this variable to specify the bootstrap server to run application.

I_MPI_HYDRA_RMK

(HYDRA_RMK)

Use the resource management kernel.

Syntax

I_MPI_HYDRA_DEMUX=<arg>

Deprecated Syntax

HYDRA_DEMUX=<arg>

Arguments

<arg>	String parameter
<rmk>	Resource management kernel

Description

Set this variable to use resource management kernel. In Intel® MPI Library 4.0, only pbs is supported.

I_MPI_HYDRA_PMI_CONNECT

Define the algorithm for Hydra PMI connections.

Syntax

I_MPI_HYDRA_PMI_CONNECT=<value>

Arguments

<value>	Define an algorithm for PMI connections with Hydra PMI
mpich2	Use the original mpich2 algorithm
proxy	Use the PMI connections through pmi_proxy.
cache	Minimize the PMI requests by caching PMI information into local pmi_proxy. The default is cache

Description

Use this variable to choose the PMI connections method. If -pmi-connect is explicitly presented in the mpiexec.hydra command line, I_MPI_HYDRA_PMI_CONNECT has no effect. The -pmi-connect option is assumed with its value if I_MPI_HYDRA_PMI_CONNECT is defined.

2.5 Multipurpose Daemon Commands

mpd

Start mpd daemon.

Syntax

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

Arguments

--help	Display a help message
-V | --version	Display the Intel® MPI Library version information
-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 a 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

Multipurpose daemon* (MPD) is the Intel® MPI Library 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 a file named .mpd.conf is presented in the user's home directory, only the 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  [ -h ] [ -V ] [ -n <#nodes>] [ -f <hostsfile> ] [ -r <rshcmd> ] \
         [ -u <user> ] [ -m <mpdcmd> ] [ --loccons ] [ --remcons ] \
         [ -s ] [ -d ] [ -v ] [ -1 ] [ --ncpus=<ncpus> ] [ -o ] \
         [ -b <maxbranch> ] [ -p ]
```

or

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

Arguments

-h | --help	Display a help message
-V | --version	Display Intel® MPI Library version information
-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. The default value is rsh
-f *<hostsfile>* | --file=*<hostsfile>*	Path/name of the file that has the list of machine names on which the daemons are started
-1	Remove the restriction of starting only one mpd per machine
-m *<mpdcmd>* | --mpd=*<mpdcms>*	Specify the full path name of the mpd on the remote hosts
-s | --shell	Specify the shell
-u *<user>* | --user=*<user>*	Specify the 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
-b *<maxbranch>* | --maxbranch=*<maxbranch>*	Use this option to indicate the maximum number of the mpd daemons to enter the mpd ring under another. This helps to control the parallelism of the mpd ring start. The default value is four

-p |--parallel-startup	Use this option to allow parallel fast starting of mpd daemons under one local root. No daemon checking is performed. This option also supports shells which do not transfer the output from the remote commands

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 through the rsh command without a password or, if the –r ssh option is used, through the ssh command without a password.

NOTE: The mpdboot command will spawn an 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 [--help] [-V] [--version] *<mpdid>*

Arguments

--help	Display a help message
-V | --version	Display Intel® MPI Library version information
<mpdid>	Specify the mpd daemon to kill

Description

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

mpdallexit

Shut down all mpd daemons on all nodes.

Syntax

mpdallexit [--help] [-V] [--version]

Arguments

--help	Display a help message
-V | --version	Display Intel® MPI Library version information

Description

Use this command to shutdown all MPD rings you own.

mpdcleanup

Cleanup the environment after an mpd crash.

Syntax

mpdcleanup [-h] [-V] [-f *<hostsfile>*] [-r *<rshcmd>*] [-u *<user>*] \

```
[ -c <cleancmd> ] [ -a]
```

or

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

Arguments

-h | --help	Display a help message
-V | --version	Display Intel® MPI Library version information
-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 <hostsfile>

Description

Use this command to cleanup the environment after an 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 the machines specified in the hostsfile file:

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

mpdtrace

Determine whether mpd is running.

Syntax

```
mpdtrace [ --help ] [ -V ] [ --version ] [ -l ]
```

Arguments

--help	Display a help message
-V | --version	Display Intel® MPI Library version information
-l	Show MPD identifiers instead of the hostnames

Description

Use this command to list the hostnames or identifiers of all mpds in the ring. The output 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 ] [ -V ] [ --version ]
mpdcheck –pc [ -v ] [ -l ]
mpdcheck -f <host_file> [ -ssh ] [ -v ] [ -l ]
mpdcheck –s [ -v ] [ -l ]
mpdcheck -c < server_host> <server_port> [ -v ] [ -l ]
```

Arguments

-h | --help	Display a help message
-V | --version	Display Intel® MPI Library version information
-pc	Print configuration information about a local host
-f *<host_file>*	Print information about the hosts listed in *<host_file>*
-ssh	Invoke testing of ssh 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 a client on the current or different host. Connect to the *<server_host> <server_port>*
-l	Print diagnostic messages in extended format
-v	Print the actions that mpdcheck is performing

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 through mpd on one or more hosts, try to run the script once on each of those hosts.

mpdringtest

Test the MPD ring.

Syntax

```
mpdringtest [ --help ] [ -V ] [ --version ] <number of loops>
```

Arguments

--help	Display a help message
-V | --version	Display Intel® MPI Library version information
<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 [ -h ] [ -V ] [ -u <username> ] [ -a <jobalias> ] [ -j <jobid> ]
```

or

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

Arguments

`-h | --help`	Display a help message
`-V | --version`	Display Intel® MPI Library version information
`-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 [ --help ] [ -V ] [ --version ] <sigtype> \
          [-j <jobid> | -a <jobalias> ] [-s | -g ]
```

Arguments

`--help`	Display a help message
`-V | --version`	Display Intel® MPI Library version information
`<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 a single user process
`-g`	Deliver a signal to a 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`.

mpdkilljob

Kill a job.

Syntax

`mpdkilljob [ --help ] [ -V ] [ --version ] [ <jobnum> ] [ -a <jobalias> ]`

Arguments

`--help`	Display a help message
`-V | --version`	Display Intel® MPI Library version information
`<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 [ -V ] [ --version ]`

Arguments

`-V | --version`	Display Intel® MPI Library version information

Description

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

2.5.1 Configuration Files

$HOME/.mpd.conf

This optional configuration file contains an `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* OS 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 be set to 600 mode (only user has 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.

2.5.2 Environment Variables

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 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 tagged output format works at the mpdboot stage and prevents a failure during startup due to unexpected output from a remote shell like 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.

I_MPI_MPD_CHECK_PYTHON*

Turn on/off the Python* versions check at the MPD ring startup stage.

Syntax

I_MPI_MPD_CHECK_PYTHON=<*arg*>

Arguments

<*arg*>	Binary indicator
enable | yes | on | 1	Check for Python version compatibility
disable | no | off | 0	Do not check the Python version compatibility. This is the default value

Description

Set this variable to enable compatibility checking of Python versions installed on the cluster nodes. This may lead to increased MPD ring startup time. The MPD behavior is undefined if incompatible Python versions are installed on the cluster.

If I_MPI_MPD_CHECK_PYTHON is set to enable and the compatibility check fails, mpdboot will exit abnormally and print a diagnostic message. An MPD ring will not be started.

I_MPI_MPD_RSH

Set the remote shell to start mpd daemons.

Syntax

I_MPI_MPD_RSH =<*arg*>

Arguments

<arg>	String parameter
<remoute shell>	The remote shell

Description

Set this variable to define the default setting for the `--rsh` *mpdboot* option. If `--rsh` is explicitly called in the command line, the `I_MPI_MPD_RSH` variable has no effect. The `--rsh` option assumes the value of the `I_MPI_MPD_RSH` variable if this variable is defined.

I_MPI_MPD_TMPDIR

TMPDIR

Set a temporary directory for the MPD subsystem.

Syntax

`I_MPI_MPD_TMPDIR=`*<arg>*

`TMPDIR=`*<arg>*

Arguments

<arg>	String parameter
<directory name>	A string that points to a scratch space location. The default value is `/tmp`

Description

Set one of these variables to specify an alternative scratch space location. The MPD subsystem creates its own files in the directory specified by these environment variables. If both variables point to valid directories, the value of the `TMPDIR` environment variable is ignored.

NOTE: The `mpd2.console_*` file full path length can be limited in some operating systems. You hit this limitation if you get the following diagnostic message: `socket.error: AF_UNIX path too long`. Decrease the length of the *<directory name>* string to avoid this issue.

NOTE: If *<arg>* points to a distributed file system (PANFS, PVFS, etc.), the `mpd` demons may not start. If this happens, set the `I_MPI_MPD_TMPDIR` and `TMPDIR` to point to a standard file system (ext2, ext3, NFS, etc.).

2.6 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 to define suitable process pinning settings. The output consists of a header and a number of tables. The output header includes the processor band name, the processor code name, and the processor architecture. The output includes the following tables:

- **Processor composition table**: describes the processor packages, cores, and threads.

  Processors (CPUs) – the number of software executive processor units.
  Packages (sockets) – the number of physical packages and corresponding sockets.
  Cores per package – the number of cores within each package.
  Threads per core – the number of processor units within each core. If the number equals one, Simultaneous Multi Threading (SMT) mode is disabled. If the number is larger than one, SMT mode is enabled.

- **Processor identification table**: identifies threads, cores, and packages of each logical processor accordingly.
  Thread Id – unique processor identifier within a core.
  Core Id – unique core identifier within a package.
  Package Id – unique package identifier within a node.

- **Processor placement table**: maps processor packages and cores. It is an inversion of the processor identification table. Each entry contains the information on packages, cores, and processors.
  Package Id – a physical package identifier.
  Cores Id – a list of core identifiers that belong to this package.
  Processors Id – a list of processors that belong to this package. This list order directly corresponds to the core list. A group of processors enclosed in brackets belongs to one core.

- **Cache sharing table**: lists information of sizes and processors groups, for each cache level.
  Size – cache size in bytes.
  Processors – a list of processor groups enclosed in the parentheses that shared this cache or **no sharing** otherwise.

NOTE: The architecture information is available on systems based on the IA-32 and Intel® 64 architectures.

Examples

1. `cpuinfo` output for Intel® Xeon® Processor 5400 series:

```
Intel(R) Xeon(TM) Processor (Intel64 Harpertown)

=====  Processor composition  =====
Processors(CPUs)  : 8
Packages(sockets) : 2
Cores per package : 4
Threads per core  : 1

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

=====  Placement on packages  =====
Package Id.  Core Id.  Processors
1            0,2,1,3   0,3,6,7
```

```
0            0,2,1,3   1,2,4,5

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

2. cpuinfo output for Intel® Core™ i7 processor with SMT support:

```
Intel(R) Core(TM) i7 Processor (Intel64 Bloomfield)

=====  Processor composition  =====
Processors(CPUs)  : 8
Packages(sockets) : 1
Cores per package : 4
Threads per core  : 2

=====  Processor identification  =====
Processor Thread Id. Core Id.  Package Id.
0         0          0         0
1         0          1         0
2         0          2         0
3         0          3         0
4         1          0         0
5         1          1         0
6         1          2         0
7         1          3         0
=====  Placement on packages  =====
Package Id   Core Id    Processors
0            0,1,2,3    (0,4)(1,5)(2,6)(3,7)

=====  Cache sharing  =====
Cache   Size            Processors
L1      32  KB         (0,4)(1,5)(2,6)(3,7)
L2      256 KB         (0,4)(1,5)(2,6)(3,7)
L3      8   MB         (0,1,2,3,4,5,6,7)
```

3 Tuning Reference

The Intel® MPI Library provides an automatic tuning utility and many environment variables that can be used to influence program behavior and performance at run time.

3.1 Automatic Tuning Utility

mpitune

Use the `mpitune` utility to find optimal settings for the Intel® MPI Library relevant to your cluster configuration or your application.

Syntax

```
mpitune  [ -h ] [ -V ] [ -hf <hostsfile> ] [ -od <outputdir> ] [ -d ] \
         [ -i <count> ] [ -s ] [ -a \"<application command line>\" ]
```

or

```
mpitune  [ --help ] [ --version] [ --host-file <hostsfile> ] \
         [ --output-directory <outputdir> ] [ --debug ] \
         [ --iterations <count>] [ --silent ] \
         [ --application \"<application command line>\" ]
```

Arguments

`-a \"<app_cmd_line>\"` | `--application \"<app_cmd_line>\"`	Switch on the application tuning mode. Quote the full command line as shown
`-of <file-name>` | `--output-file <file-name>`	Specify the application configuration file to be generated in the application-specific mode. By default, use the `$PWD/app.conf`
`-t \"<test_cmd_line>\"` | `--test \"<test_cmd_line>\"`	Replace the default Intel® MPI Benchmarks by the indicated benchmarking program in the cluster-specific mode. Quote the full command line as shown
`-cm` | `--cluster-mode {exclusive|full}`	Set the cluster usage mode `full` – maximum number of tasks will be executed. This is the default mode `exclusive` – only one task will be executed on the cluster at a time
`-d` | `--debug`	Print out the debug information
`-dl [d1[,d2…[,dN]]]` | `--device-list [d1[,d2,…[,dN]]]`	Select the device(s) you want to tune. By default, use all devices mentioned in the `<installdir>/<arch>/etc/devices.xml` file
`-fl [f1[,f2…[,fN]]]` | `--fabric-list [f1[,f2…[,fN]]]`	Select the fabric(s) you want to tune. By default, use all fabrics mentioned in the `<installdir>/<arch>/etc/fabrics.xml` file

-er | --existing-ring	Try to use an existing MPD ring. By default, create a new MPD ring
-hf <hostsfile> | --host-file <hostsfile>	Specify an alternative host file name. By default, use the $PWD/mpd.hosts
-h | --help	Display a help message
-hr | --host-range {min:max|min:|:max}	Set the range of hosts used for testing. The default minimum value is 1. The default maximum value is the number of hosts defined by the mpd.hosts or the existing MPD ring. The min: or :max format will use the default values as appropriate
-i <count> | --iterations <count>	Define how many times to run each tuning step. Higher iteration counts increase the tuning time, but may also increase the accuracy of the results. The default value is 3
--message-range {min:max|min:|:max}	Set the message size range. The default minimum value is 0. The default maximum value is 4194304 (4mb). By default, the values are given in bytes. They can also be given in the following format: 16kb, 8mb or 2gb. The min: or :max format will use the default values as appropriate
-od <outputdir> | --output-directory <outputdir>	Specify the directory name for all output files. By default, use the current directory. This directory should be accessible from all hosts
-pr {min:max|min:|:max} | --ppn-range {min:max|min:|:max} | --perhost-range {min:max|min:|:max}	Set the maximum number of processes per host. The default minimum value is 1. The default maximum value is the number of cores of the processor. The min: or :max format will use the default values as appropriate
-sf [file-path] | --session-file [file-path]	Continue the tuning process starting from the state saved in the file-path session file
-s | --silent	Suppress all diagnostic output
-td <dir-path> | --temp-directory <dir-path>	Specify a directory name for the temporary data. By default, use the $PWD/mpitunertemp. This directory should be accessible from all hosts
-tl <minutes> | --time-limit <minutes>	Set mpitune execution time limit in minutes. The default value is 0, which means no limitations
-mh | --master-host	Dedicate a single host to run mpitune
-V | --version	Print out version information

Deprecated Options

Deprecated Option	New Option
--outdir	-od | --output-directory
--verbose	-d | --debug
--file	-hf | --host-file
--logs	-lf | --log-file
--app	-a | --application

Description

Use the `mpitune` utility to create a set of Intel® MPI Library configuration files that contain optimal settings for a particular cluster or application. You can reuse these configuration files in the `mpiexec` job launcher by using the `-tune` option.

The MPI tuner utility operates in two modes:

- Cluster-specific, evaluating a given cluster environment using either the Intel® MPI Benchmarks or a user-provided benchmarking program to find the most suitable configuration of the Intel® MPI Library. This mode is used by default.
- Application-specific, evaluating the performance of a given MPI application to find the best configuration for the Intel® MPI Library for the particular application. Application tuning is enabled by the `--application` command line option.

3.1.1 Cluster-specific Tuning

Run this utility once after the Intel® MPI Library installation and after every cluster configuration change (processor or memory upgrade, network reconfiguration, etc.). Do this under the user account that was used for the Intel® MPI Library installation or set the tuner data directory with the `--output-directory` command.

If there are any configuration files in the `<installdir>/<arch>/etc` directory, the recorded Intel® MPI Library configuration settings will be used automatically by `mpiexec` with the `-tune` option.

For example:

1. Collect configuration settings for the cluster hosts listed in the `./mpd.hosts` file by using the Intel® MPI Benchmarks

   ```
   $ mpitune
   ```

2. Use the optimal recorded values when running on the cluster

   ```
   $ mpiexec -tune -n 32 ./myprog
   ```

The job launcher will find a proper set of configuration options based on the execution conditions: communication fabrics, number of hosts and processes, etc. If you have write access permission for `<installdir>/<arch>/etc`, all generated files will be saved in this directory; otherwise the current working directory will be used.

3.1.1.1 Replacing the Default Benchmark

This tuning feature is an extension of the cluster-specific tuning mode in which you specify a benchmarking application that will be used for tuning.

For example:

1. Collect the configuration settings for the cluster hosts listed in the `./mpd.hosts` file by using the desired benchmarking program

   ```
   $ mpitune --test \"benchmark -param1 -param2\"
   ```

2. Use the optimal recorded values for your cluster

   ```
   $ mpiexec -tune -n 32 ./myprog
   ```

3.1.2 Application-specific Tuning

Run the tuning process for any kind of MPI application by specifying its command line to the tuner. Performance is measured as inversed execution time of the given application. To reduce the overall tuning time, use the shortest representative application workload if applicable.

For example:

1. Collect configuration settings for the given application

```
$ mpitune --application \"mpiexec -n 32 ./myprog\" –of ./myprog.conf
```

2. Use the optimal recorded values for your application

```
$ mpiexec –tune ./myprog.conf -n 32 ./myprog
```

Based on the default tuning rules, the automated tuning utility evaluates a full set of the library configuration parameters to minimize the application execution time. By default, all generated files will be saved in the current working directory.

NOTE: The resulting application configuration file contains the optimal Intel® MPI Library parameters for this particular application only. If you want to tune the Intel® MPI Library for the same application in a different configuration (number of hosts, workload, etc.), you may need to rerun the automated tuning utility by using the desired configuration.

The automated tuning utility will overwrite the existing application configuration files by default. You should use a naming convention for your various application files to select the correct file when you need it.

3.1.3 Tuning Utility Output

Upon completion of the tuning process, the Intel® MPI Library tuning utility records the chosen values in the configuration file in the following format:

```
-genv I_MPI_DYNAMIC_CONNECTION 1
-genv I_MPI_ADJUST_REDUCE 1:0-8
```

The Intel MPI Library tuning utility ignores variables having no effect on the application when the difference between probes is at the noise level (1%). In this case, the utility does not set the variable and preserves the default library heuristics.

In the case of the tuning application having significant run-to-run performance variation, the Intel MPI Library tuning utility might select divergent values for the same variable under the same conditions. To improve decision accuracy, increase the number of iterations for each test run with the `-i` command line option. The default value for the iteration number is `3`.

3.2 Process Pinning

Use this feature to pin particular MPI process to a corresponding CPU and avoid undesired process migration. This feature is available on operating systems that provide the necessary kernel interfaces.

3.2.1 Process Identification

Two schemes are used to identify logical processors in a system:

1. System-defined logical enumeration

2. Topological enumeration based on three-level hierarchical identification through triplets (package/socket, core, thread)

The number of a logical CPU is defined as the corresponding position of this CPU bit in the kernel affinity bit-mask. Use the `cpuinfo` utility or the `cat /proc/cpuinfo` command to find out the logical CPU numbers.

Three-level hierarchical identification uses triplets that provide information about processor location and their order. The triplets are hierarchically ordered (package, core, and thread).

See example below for possible processor numbering where there are two sockets, four cores (two cores per socket), and eight logical processors (two processors per core).

NOTE: Logical and topological enumerations are not the same.

Table 3.2-1 Logical Enumeration

0	4	1	5	2	6	3	7

Table 3.2-2 Hierarchical Levels

Socket	0	0	0	0	1	1	1	1
Core	0	0	1	1	0	0	1	1
Thread	0	1	0	1	0	1	0	1

Table 3.2-3 Topological Enumeration

0	1	2	3	4	5	6	7

Use the `cpuinfo` utility to identify the correspondence between the logical and topological enumerations. See *Processor Information Utility* for more details.

3.2.2 Environment Variables

I_MPI_PIN

Turn on/off process pinning.

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 the process pinning feature of the Intel® MPI Library.

I_MPI_PIN_MODE

Choose the pinning method.

Syntax

I_MPI_PIN_MODE=<pinmode>

Arguments

<pinmode>	Choose the CPU pinning mode
mpd	Pin processes inside MPD. Default on the SGI* Altix* platform
lib	Pin processes inside the Intel MPI Library. Default on other platforms

Description

Set the I_MPI_PIN_MODE variable to choose the pinning method. This variable is valid only if the I_MPI_PIN environment variable is enabled.

Set this variable to lib to make the Intel® MPI Library pin the processes. In this mode 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 through system specific means, if they are available. The pinning is done before the MPI process launch. Therefore, it is possible to co-locate the process CPU and memory in this case. This pinning method has an advantage over a 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.

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

I_MPI_PIN_PROCESSOR_LIST (I_MPI_PIN_PROCS)

Define a processor subset and mapping rules for MPI processes pinning to separate processors of this subset.

Syntax

I_MPI_PIN_PROCESSOR_LIST=<value>

The variable value has three syntax forms:

1. <proclist>
2. [<procset>] [:[grain=<grain>] [,shift=<shift>]\
   [,preoffset=<preoffset>] [,postoffset=<postoffset>]
3. [<procset>][:map=<map>]

Deprecated Syntax

I_MPI_PIN_PROCS=<proclist>

NOTE: The postoffset keyword has offset alias.

NOTE: The second form of pinning procedure has three steps:

1. Cyclic shift of the source processor list on preoffset*grain value.
2. Round robin shift of the list derived on the first step on shift*grain value.
3. Cyclic shift of the list derived on the second step on the postoffset*grain value.

The result processor list is used for the consecutive mapping of MPI processes (i-th rank is mapped on the i-th list member).

NOTE: grain, shift, preoffset, and postoffset parameters have the unified style of setting.

Arguments

<proclist>	A comma-separated list of logical processor numbers and/or ranges of processors. Process with the i-th rank is pinned on the i-th processor in the list. The number should not exceed the amount of processors on a node
<l>	Processor with logical number *<l>*
<l>-<m>	Range of processors with logical numbers from *<l>* to *<m>*
<k>,<l>-<m>	Processors *<k>*, as well as *<l>* through *<m>*

<procset>	A processor is ordered according to the topological numeration. The default value is `allcores`
`all`	All logical processors. The power of this subset is equal to the number of CPU on a node
`allcores`	All logical processors that belong to different cores. A power of this subset is equal to the number of cores on a node. If Intel® Hyper-Threading Technology is disabled, `allcores` equals to `all`
`allsocks`	All logical processors that belong to different physical packages/sockets. The power of this subset is equal to the number of sockets on a node

<map>	Pattern used for the process placement
`bunch`	The processes are mapped in proportion on sockets as close as possible
`scatter`	The processes are mapped as remotely as possible not to share common resources: FSB, caches, core
`spread`	The processes are mapped consecutively with the possibility not to share common resources

<grain>	Specify pinning granularity cell for defined `procset`. Minimal `grain` is one element of `procset`. Maximal grain is a number of `procset` elements in a socket. The `grain` value must be multiple of the `procset` power. Otherwise, minimal grain is assumed. The default value is minimal `grain`
<shift>	Specify the round robin shift of the granularity cells along `procset`. `shift` is measured in the defined `grain` units. The `shift` value must be positive integer. Otherwise, no shift is performed. The default value is no shift
<preoffset>	Specify cyclic shift of `procset` on the `preoffset` value before the round robin shifting. The value is measured in the defined `grain` units. The `preoffset` value must be non negative integer. Otherwise, no shift is performed. The default value is no shift
<postoffset>	Specify cyclic shift of processor subset derived after round robin shifting on the `postoffset` value. The value is measured in the

	defined grain units. The postoffset value must be non-negative integer. Otherwise no shift is performed. The default value is no shift

<n>	Specify the explicit value of the corresponding parameter. <n> is non-negative integer
fine	Specify the minimal value of the corresponding parameter
core	Specify the parameter value equal to the amount of the corresponding parameter units contained in one core
cache1	Specify the parameter value equal to the amount of the corresponding parameter units that share L1 cache
cache2	Specify the parameter value equal to the amount of the corresponding parameter units that share L2 cache
cache3	Specify the parameter value equal to the amount of the corresponding parameter units that share L3 cache
cache	The largest value among cache1, cache2, and cache3
socket	Specify the parameter value equal to the amount of the corresponding parameter units contained in one physical package/socket
sock	sock = socket
half	Specify the parameter value equal to socket/2
mid	mid = half
third	Specify parameter value equal to socket/3
quarter	Specify parameter value equal to socket/4
octavo	Specify parameter value equal to socket/8

Description

Set the I_MPI_PIN_PROCESSOR_LIST variable to define the processor placement on processors. In order to avoid conflicts with shells, the variable value may be enclosed in quotes.

NOTE: This variable is valid only if I_MPI_PIN is enabled.

The I_MPI_PIN_PROCESSOR_LIST variable has three different variants:

1. Explicit processor list. This comma-separated list is defined in terms of logical processor numbers. Relative node rank of a process is an index to the list that is the i-th process pinned on i-th list member. This permits definition of any process placement on CPUs.

   For example, process mapping for I_MPI_PROCESSOR_LIST=p0,p1,p2,…,pn is as follows:

Rank on a node	0	1	2	…	n-1	N
Logical CPU	p0	p1	p2	…	pn-1	Pn

2. Grain/shift/offset mapping. This method provides cyclic shift of the defined grain along the processor list with step equal to shift*grain and a single shift on offset*grain at the end. This shifting action is repeated shift times.

For example: grain = 2 logical processors, shift = 3 grains, offset = 0.

Legend:

gray – MPI process grains

A) red – processor grains chosen on the 1st pass

B) cyan - processor grains chosen on the 2nd pass

C) green - processor grains chosen on the final 3rd pass

D) Final map table ordered by MPI ranks

A)

0 1			2 3			. . .	2n-2 2n-1		
0 1	2 3	4 5	6 7	8 9	10 11	. . .	6n-6 6n-5	6n-4 6n-3	6n-2 6n-1

B)

0 1	2n 2n+1		2 3	2n+2 2n+3		. . .	2n-2 2n-1	4n-2 4n-1	
0 1	2 3	4 5	6 7	8 9	10 11	. . .	6n-6 6n-5	6n-4 6n-3	6n-2 6n-1

C)

0 1	2n 2n+1	4n 4n+1	2 3	2n+2 2n+3	4n+2 4n+3	. . .	2n-2 2n-1	4n-2 4n-1	6n-2 6n-1
0 1	2 3	4 5	6 7	8 9	10 11	. . .	6n-6 6n-5	6n-4 6n-3	6n-2 6n-1

D)

0 1	2 3	…	2n-2 2n-1	2n 2n+1	2n+2 2n+3	…	4n-2 4n-1	4n 4n+1	4n+2 4n+3	…	6n-2 6n-1
0 1	6 7	…	6n-6 6n-5	2 3	8 9	…	6n-4 6n-3	4 5	10 11	…	6n-2 6n-1

3. Predefined scenario. In this case the most popular schemes of process pinning get unique names and these names are used for selection. Currently there are two such scenarios: `bunch` and `scatter`.

In the `bunch` scenario the processes are mapped proportionally to sockets as closely as possible. This makes sense for partial processor loading. In this case the number of processes is less than the number of processors.

In the `scatter` scenario the processes are mapped as remotely as possible to not share common resources: FSB, caches, cores.

In the example below there are two sockets, four cores per socket, one logical CPU per core, and two cores per shared cache.

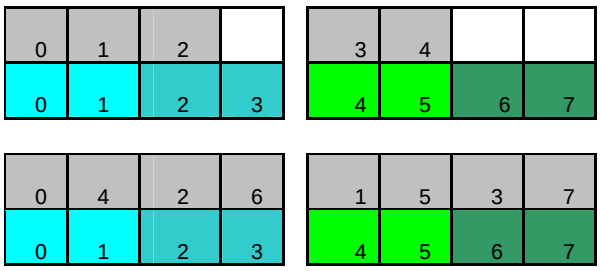
Legend:

gray – MPI processes

cyan – 1st socket processors

green – 2nd socket processors

The same color – processor pair share one cache



`bunch` scenario for 5 processors

`scatter` scenario for full loading

Examples

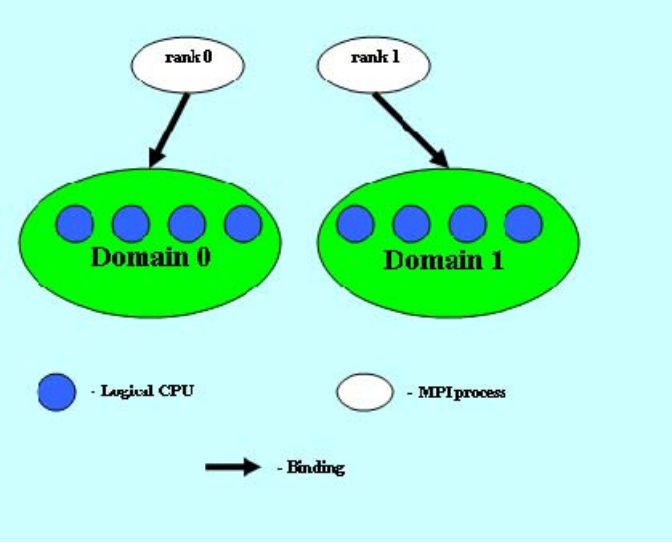
1. To pin the processes to the CPU0 and CPU3 on each node globally, use the following command:
```
$ mpirun -genv I_MPI_PIN_PROCESSOR_LIST 0,3 \
         -n <# of processes> <executable>
```
2. To pin the processes to different CPUs on each node individually (CPU0 and CPU3 on host1 and CPU0, CPU1 and CPU3 on host2), 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>
```
3. To print extra debug information about process pinning, use the following command:
```
$ mpirun –genv I_MPI_DEBUG 4 –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: If a number of processes is greater than a number of CPUs for pinning, a process list is wrapped on a processor list.

3.2.3 Interoperability with OpenMP*

I_MPI_PIN_DOMAIN

The Intel® MPI Library provides an additional environment variable to control process pinning for hybrid Intel MPI applications. The variable is used to define a number of non-overlapping subsets (domains) of logical processors on a node, and a set of rules on how MPI processes are bound to these domains by the following formula: **one MPI process per one domain**. See the picture below.



Picture 3.2-1 Domain Example

Each MPI process can create a number of children threads for running within the corresponding domain. The process threads can freely migrate from one logical processor to another within the particular domain. There are no any domains defined by default so they should be defined explicitly.

If the I_MPI_PIN_DOMAIN variable is defined, then the I_MPI_PIN_PROCESSOR_LIST variable setting is ignored.

If the I_MPI_PIN_DOMAIN variable is not defined, then MPI processes are pinned according to the current value of the I_MPI_PIN_PROCESSOR_LIST variable.

The I_MPI_PIN_DOMAIN variable has the following syntax forms:

1. Domain description through multi-core terms
2. Domain description through domain size and domain member layout
3. Explicit domain description through bit mask

Multi-core Shape

I_MPI_PIN_DOMAIN=*<mc-shape>*

<mc-shape>	Define domains through multi-core terms
core	Each domain consists of the logical processors that share a particular core. The number of domains on a node is equal to the number of cores on this node
socket | sock	Each domain consists of the logical processors that share a particular socket. The number of domains on a node is equal to the number of sockets on this node. The recommended value is socket
node	All logical processors on a node are arranged into a single domain
cache1	Logical processors that share a particular level 1 cache are arranged into a single domain
cache2	Logical processors that share a particular level 2 cache are arranged into a separate domain
cache3	Logical processors that share a particular level 3 cache are arranged into a separate domain
cache	The largest domain among cache1, cache2, and cache3 is selected

Explicit Shape

I_MPI_PIN_DOMAIN=*<size>[:<layout>]*

<size>	Define a number of logical processors in each domain (domain size)
omp	The domain size is equal to the OMP_NUM_THREADS environment variable value. If the OMP_NUM_THREADS environment variable is not set, each node is treated as a separate domain.
auto	The domain size is defined by the formula size=#cpu/#proc, where #cpu is the number of logical processors on a node, and #proc is the number of the MPI processes started on a node
<n>	The domain size is defined by the positive decimal number *<n>*

<layout>	Ordering of domain members. If *<layout>* is omitted then compact is assumed

platform	Domain members are ordered on the base of BIOS numbering (platform-depended numbering)
compact	Domain members are located as close to each other as possible in terms of common resources (cores, caches, sockets, etc.). This is the default value
scatter	Domain members are located as far away from each other as possible in terms of common resources (cores, caches, sockets, etc.)

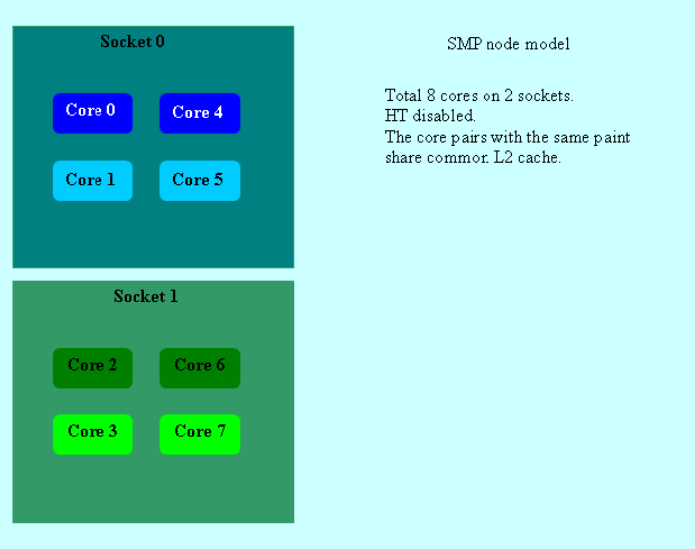
Explicit Domain Mask

I_MPI_PIN_DOMAIN=<*masklist*>

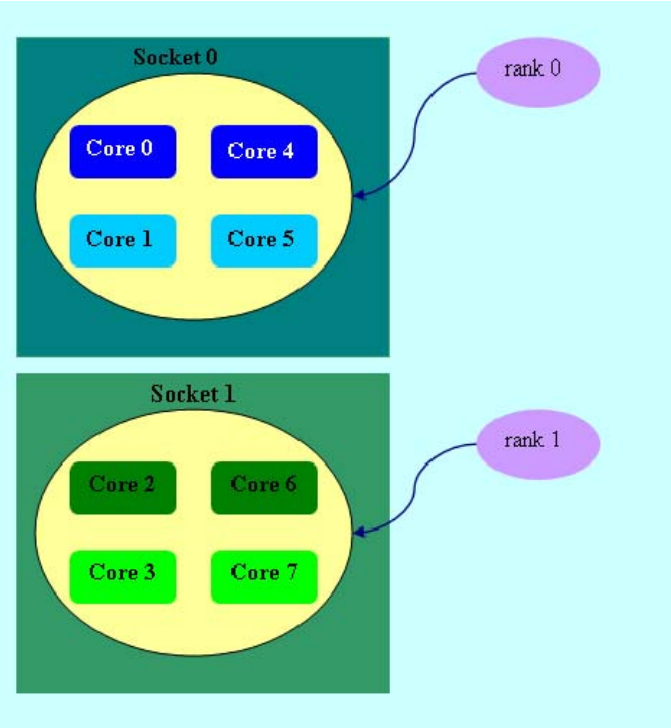
<*masklist*>	Define domains through the comma separated list of hexadecimal numbers (domain masks)
[m_1,…,m_n]	Each m_i number defines one separate domain. The following rule is used: the i^{th} logical processor is included into the domain if the corresponding m_i value is set to 1. All remaining processors are put into a separate domain. BIOS numbering is used

NOTE: In order to pin OpenMP processes/threads inside the domain the corresponding OpenMP feature (KMP_AFFINITY environment variable) may be used.

See the following model of the SMP node in the examples below:

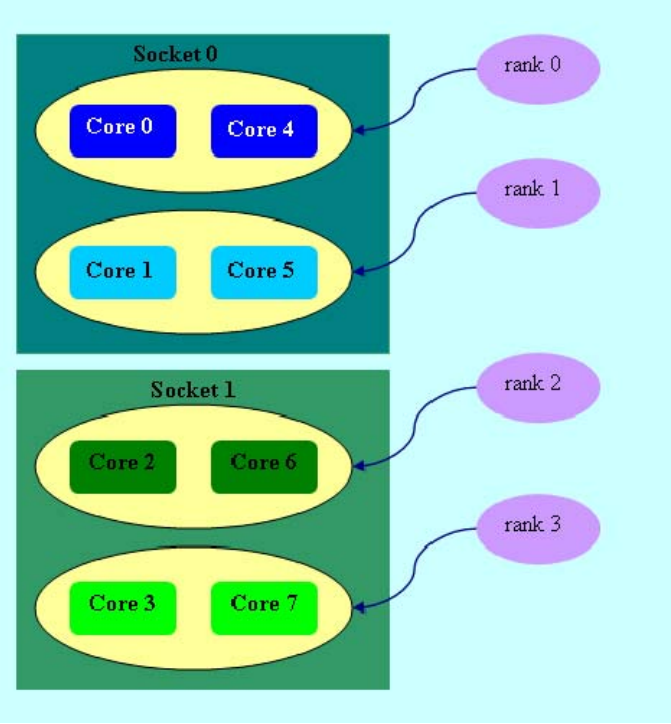


Picture 3.2-2 Model of Node



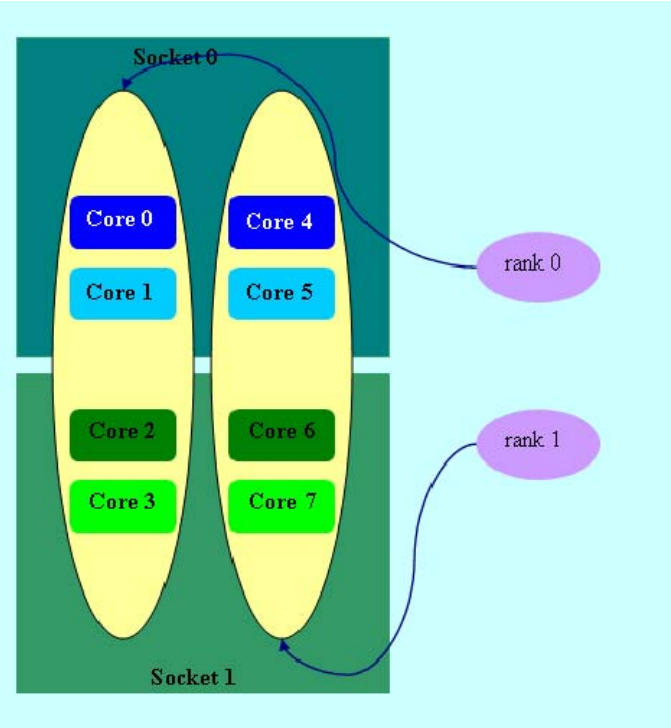
Picture 3.2-3 `mpiexec -n 2 -env I_MPI_PIN_DOMAIN socket ./a.out`

Two domains are defined according to the number of sockets. Process rank 0 can migrate on all cores on the 0-th socket. Process rank 1 can migrate on all cores on the first socket.



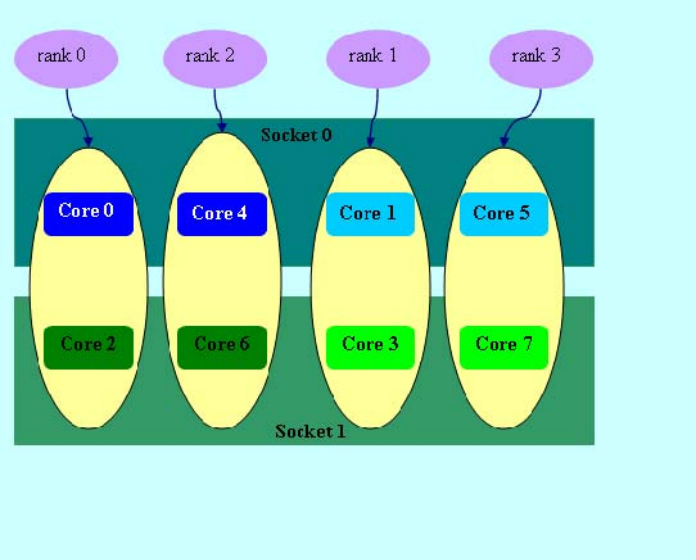
Picture 3.2-4 `mpiexec -n 4 -env I_MPI_PIN_DOMAIN cache2 ./a.out`

Four domains are defined according to the amount of common L2 caches. Process rank 0 runs on cores {0,4} that share L2 cache. Process rank 1 runs on cores {1,5} that share L2 cache as well, and so on.



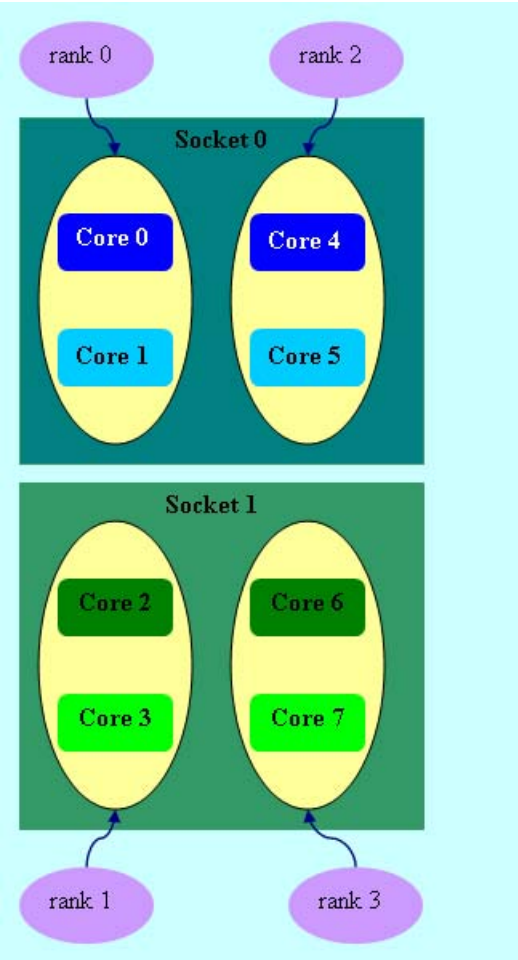
Picture 3.2-5 `mpiexec -n 2 -env I_MPI_PIN_DOMAIN 4:platform ./a.out`

Two domains with size=4 are defined. The fist domain contains {0,1,2,3} cores, and the second domain contains cores {4,5,6,7}. Domain members (cores) have consecutive numbering as defined by the platform option.



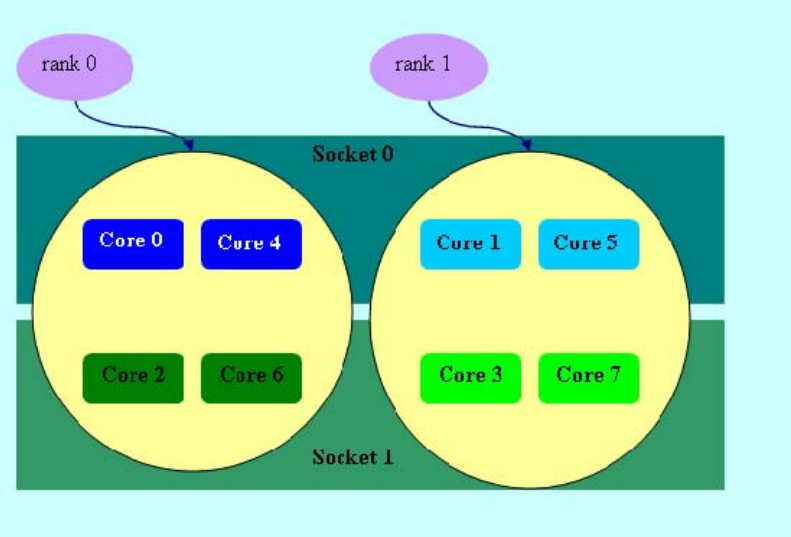
Picture 3.2-6 `mpiexec -n 4 -env I_MPI_PIN_DOMAIN auto:scatter ./a.out`

Domain size=2 (defined by the number of CPUs=8 / number pf process=4), scatter layout. Four domains {0,2}, {1,3}, {4,6}, {5,7} are defined. Domain members do not share any common resources.



Picture 3.2-7 `mpiexec -n 4 -env I_MPI_PIN_DOMAIN omp:platform ./a.out setenv OMP_NUM_THREADS=2`

Domain size=2 (defined by `OMP_NUM_THREADS=2`), platform layout. Four domains {0,1}, {2,3}, {4,5}, {6,7} are defined. Domain members (cores) have consecutive numbering.



Picture 3.2-8 mpiexec -n 2 -env I_MPI_PIN_DOMAIN [55,aa] ./a.out

The fist domain is defined by the 0x55 mask. It contains all cores with even numbers {0,2,4,6}. The second domain is defined by the 0xAA mask. It contains all cores with odd numbers {1,3,5,7}.

3.3 Fabrics Control

3.3.1 Communication Fabrics Control

I_MPI_FABRICS

(I_MPI_DEVICE)

Select the particular network fabrics to be used.

Syntax

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

Where `<fabric> := {shm, dapl, tcp, tmi, ofa}`

`<intra-node fabric> := {shm, dapl, tcp, tmi, ofa}`

`<inter-nodes fabric> := {dapl, tcp, tmi, ofa}`

Deprecated Syntax

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

Arguments

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

tmi	TMI-capable network fabrics including Qlogic*, Myrinet*, (through Tag Matching Interface)
ofa	OFA-capable network fabric including InfiniBand* (through OFED* verbs)

Correspondence with I_MPI_DEVICE

<device>	*<fabric>*
sock	tcp
shm	shm
ssm	shm:tcp
rdma	dapl
rdssm	shm:dapl
<provider>	Optional DAPL* provider name (only for the rdma and the rdssm devices) I_MPI_DAPL_PROVIDER=*<provider>* or I_MPI_DAPL_UD_PROVIDER=*<provider>*

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 the /etc/dat.conf file is used.

Description

Set this variable to select a specific fabric combination. If the requested fabric(s) is not available, Intel® MPI Library can fall back to other fabric(s). See *I_MPI_FALLBACK* for details. If the I_MPI_FABRICS variable is not defined, Intel® MPI Library selects the most appropriate fabric combination automatically.

The exact combination of fabrics depends on the number of processes started per node.

- If all processes start on one node, the library uses shm intra-node communication.
- If the number of started processes is less than or equal to the number of available nodes, the library uses the first available fabric from the fabrics list for inter-nodes communication.
- For other cases, the library uses shm for intra-node communication, and the first available fabric from the fabrics list for inter-nodes communication. See *I_MPI_FABRICS_LIST* for details.

NOTE: The combination of selected fabrics ensures that the job runs, but this combination may not provide the highest possible performance for the given cluster configuration.

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

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

To select shared-memory and DAPL–capable network fabric as the chosen fabric combination, use the following command:

```
$ mpiexec -n <# of processes> -env I_MPI_FABRICS shm:dapl <executable>
```

To enable Intel® MPI Library to select most appropriate fabric combination automatically, use the following command:

```
$ mpiexec -n <# of procs> -perhost <# of procs per host> <executable>
```

Set the level of debug information to 2 or higher to check which fabrics have been initialized. See I_MPI_DEBUG for details. For example:

```
[0] MPI startup(): shm and dapl data transfer modes
```

or

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

NOTE: If the I_MPI_FABRICS variable and the I_MPI_DEVICE variable are set at the same level (command line, environment, configuration files), the I_MPI_FABRICS environment variable has higher priority than the I_MPI_DEVICE variable.

I_MPI_FABRICS_LIST

Define a fabrics list.

Syntax

I_MPI_FABRICS_LIST=*<fabrics list>*

Where *<fabrics list>* := *<fabric>*,…,*<fabric>*

<fabric> := {dapl, tcp, tmi, ofa}

Arguments

<fabrics list>	Specify a fabrics list. The following list is the default value: dapl, tcp, tmi, and ofa

Description

Set this variable to define a list of fabrics. The library uses the fabrics list to choose the most appropriate fabrics combination automatically. For information on fabric combination, see I_MPI_FABRICS

For example, if I_MPI_FABRICS_LIST=dapl,tcp, I_MPI_FABRICS is not defined and the initialization of DAPL-capable network fabrics fails, the library falls back to TCP-capable network fabric. For information on fallback, see I_MPI_FALLBACK.

I_MPI_FALLBACK

(I_MPI_FALLBACK_DEVICE)

Set this environment variable to enable fallback to the first available fabric.

Syntax

I_MPI_FALLBACK=*<arg>*

Deprecated Syntax

I_MPI_FALLBACK_DEVICE=*<arg>*

Arguments

<arg>	Binary indicator

enable | yes | on | 1	Fall back to the first available fabric. This is the default value if I_MPI_FABRICS(I_MPI_DEVICE) environment variable is not set
disable | no | off | 0	Terminate the job if MPI can not initialize the one of the fabrics selected by the I_MPI_FABRICS environment variable. This is the default value if you set I_MPI_FABRICS(I_MPI_DEVICE) environment variable

Description

Set this variable to control fallback to the first available fabric.

If I_MPI_FALLBACK is set to enable and an attempt to initialize a specified fabric fails, the library uses the first available fabric from the list of fabrics. See *I_MPI_FABRICS_LIST* for details.

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

NOTE: If I_MPI_FABRICS is set and I_MPI_FALLBACK=enable, the library falls back to fabrics with higher numbers in the fabrics list. For example, if I_MPI_FABRICS=dapl, I_MPI_FABRICS_LIST=ofa,tmi,dapl,tcp, I_MPI_FALLBACK=enable and the initialization of DAPL-capable network fabrics fails, the library falls back to TCP-capable network fabric.

I_MPI_EAGER_THRESHOLD

Change the eager/rendezvous message size threshold for all devices.

Syntax

I_MPI_EAGER_THRESHOLD=*<nbytes>*

Arguments

<nbytes>	Set the eager/rendezvous message size threshold
> 0	The default *<nbytes>* value is equal to 262144 bytes

Description

Set this variable to control the protocol used for point-to-point communication:

- Messages shorter than or equal in size to *<nbytes>* are sent using the eager protocol.
- Messages larger than *<nbytes>* are sent using the rendezvous protocol. The rendezvous protocol uses memory more efficiently.

I_MPI_INTRANODE_EAGER_THRESHOLD

Change the eager/rendezvous message size threshold for intra-node communication mode.

Syntax

I_MPI_INTRANODE_EAGER_THRESHOLD=*<nbytes>*

Arguments

<nbytes>	Define the threshold for DAPL* intra-node communication
> 0	The default *<nbytes>* value is equal to 262144 bytes for all fabrics except shm. For shm, cutover point is equal to the value of I_MPI_SHM_CELL_SIZE environment variable

Description

Set this variable to change the protocol used for communication within the node:

- Messages shorter than or equal in size to <nbytes> are sent using the eager protocol.
- Messages larger than <nbytes> are sent using the rendezvous protocol. The rendezvous protocol uses the memory more efficiently.

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

I_MPI_INTRANODE_DIRECT_COPY

Turn on/off the intranode direct copy communication mode.

Syntax

I_MPI_INTRANODE_DIRECT_COPY=<arg>

Arguments

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

Description

Set this variable to specify the communication mode within the node. If the direct copy communication mode is enabled, data transfer algorithms are selected according to the following scheme:

- Messages shorter than or equal to the threshold value of the I_MPI_INTRANODE_EAGER_THRESHOLD variable are transferred using the shared memory.
- Messages larger than the threshold value of the I_MPI_INTRANODE_EAGER_THRESHOLD variable are transferred through the direct process memory access.

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 when more than one process runs per processor/core. Otherwise the value equals 250

Description

Set the spin count limit. The loop for polling the fabric(s) spins <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 tuning application performance. The best value for <scount> can be chosen on an experimental basis. It depends on the particular computational environment and application.

I_MPI_SCALABLE_OPTIMIZATION

(I_MPI_SOCK_SCALABLE_OPTIMIZATION)

Turn on/off scalable optimization of the network fabric communication.

Syntax

`I_MPI_SCALABLE_OPTIMIZATION=<arg>`

Deprecated Syntax

`I_MPI_SOCK_SCALABLE_OPTIMIZATION=<arg>`

Arguments

<arg>	Binary indicator
`enable | yes | on | 1`	Turn on scalable optimization of the network fabric communication. This is the default for 16 or more processes
`disable | no | off | 0`	Turn off scalable optimization of the network fabric communication. This is the default for less than 16 processes

Description

Set this variable to enable scalable optimization of the network fabric communication. In most cases, using optimization decreases latency and increases bandwidth for a large number of processes.

NOTE: Old notification `I_MPI_SOCK_SCALABLE_OPTIMIZATION` is equal to `I_MPI_SCALABLE_OPTIMIZATION` for `tcp` fabric.

I_MPI_WAIT_MODE

Turn on/off 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

Description

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

Use the Native POSIX Thread Library* with the wait mode for `shm` communications.

NOTE: To check which version of the thread library is installed, use the following command:

```
$ getconf GNU_LIBPTHREAD_VERSION
```

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 for 64 or more processes
disable | no | off | 0	Turn off the dynamic connection establishment. This is the default for less than 64 processes

Description

Set this variable to control dynamic connection establishment.

- If this mode is enabled, all connections are established at the time of the first communication between each pair of processes.
- If this mode is disabled, all connections are established upfront.

The default value depends on a number of processes in the MPI job. The dynamic connection establishment is off if a total number of processes is less than 64.

3.3.2 Shared Memory Control

I_MPI_SHM_CACHE_BYPASS

(I_MPI_CACHE_BYPASS)

Control the message transfer algorithm for the shared memory.

Syntax

I_MPI_SHM_CACHE_BYPASS=<*arg*>

Deprecated 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 enable/disable message transfer bypass cache for the shared memory. When enabled, the MPI sends the messages greater than or equal in size to the value specified by the I_MPI_SHM_CACHE_BYPASS_THRESHOLD environment variable through the bypass cache. By default, this feature is enabled on the IA-32 architecture and Intel® 64 architectures. This feature is not supported on IA-64 architecture systems.

I_MPI_SHM_CACHE_BYPASS_THRESHOLDS

(I_MPI_CACHE_BYPASS_THRESHOLDS)

Set the messages copying algorithm threshold.

Syntax

`I_MPI_SHM_CACHE_BYPASS_THRESHOLDS=<nb_send>,[<nb_recv>,[<nb_send_pk>,[<nb_recv_pk>]]]`

Deprecated Syntax

`I_MPI_CACHE_BYPASS_THRESHOLDS=<nb_send>,[<nb_recv>,[<nb_send_pk>,[<nb_recv_pk>]]]`

Arguments

<nb_send>	Set the threshold for sent messages in the following situations: Processes are pinned on cores that are not located in the same physical processor package Processes are not pinned
⩾ 0	The default *<nb_send>* value is 16,384 bytes
<nb_recv>	Set the threshold for received messages in the following situations: Processes are pinned on cores that are not located in the same physical processor package Processes are not pinned
⩾ 0	The default *<nb_recv>* value is 2,097,152 bytes
<nb_send_pk>	Set the threshold for sent messages when processes are pinned on cores located in the same physical processor package
⩾ 0	The default *<nb_send_pk>* value is -1 (copying bypass cache is disabled)
<nb_recv_pk>	Set the threshold for received messages when processes are pinned on cores located in the same physical processor package
⩾ 0	The default *<nb_recv_pk>* value is 2,097,152 bytes

Description

Set this variable to control the thresholds for the message copying algorithm. MPI copies messages greater than or equal in size to the defined threshold values so that the messages bypass the cache. The value of -1 disables cache bypass. This variable is valid only when I_MPI_SHM_CACHE_BYPASS is enabled.

I_MPI_SHM_LMT_BUFFER_NUM

(I_MPI_SHM_NUM_BUFFERS)

Change the number of shared memory buffers for Large Message Transfer (LMT) mechanism.

Syntax

I_MPI_SHM_LMT_BUFFER_NUM=<num>

Deprecated Syntax

I_MPI_SHM_NUM_BUFFERS=<num>

Arguments

<num>	The number of shared memory buffers for each process pair
> 0	The default value is 8

Description

Set this variable to define the number of shared memory buffers between each process pair.

I_MPI_SHM_LMT_BUFFER_SIZE

(I_MPI_SHM_BUFFER_SIZE)

Change the size of shared memory buffers for LMT mechanism.

Syntax

I_MPI_SHM_LMT_BUFFER_SIZE=<nbytes>

Deprecated Syntax

I_MPI_SHM_BUFFER_SIZE=<nbytes>

Arguments

<nbytes>	The size of shared memory buffers in bytes
> 0	The default <nbytes> value is equal to 32,768 bytes

Description

Set this variable to define the size of shared memory buffers for each pair of processes.

I_MPI_SHM_CELL_NUM

Change the number of shared memory cells.

Syntax

I_MPI_SHM_CELL_NUM=<num>

Arguments

<num>	The number of shared memory cells
> 0	The default value is 128

Description

Set this variable to define the number of shared memory cells.

I_MPI_SHM_CELL_SIZE

Change the size of shared memory cell.

Syntax

I_MPI_SHM_CELL_SIZE=<nbytes>

Arguments

<nbytes>	Size of shared memory cell in bytes
> 0	The default <nbytes> value is equal to 65,408 bytes

Description

Set this variable to define the size of shared memory cell.

I_MPI_SHM_FBOX_SIZE

Set the size of shared memory fastbox.

Syntax

I_MPI_SHM_FBOX_SIZE=<nbytes>

Arguments

<nbytes>	Size of shared memory fastbox in bytes
> 0	The default <nbytes> value is equal to 65,408 bytes

Description

Set this variable to define the size of shared memory fastbox.

I_MPI_SHM_SINGLE_SEGMENT_THRESHOLD

(I_MPI_SHM_PROC_THRESHOLD)

Change the 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 the threshold of allocation mode for the shm device
> 0	The default <nproc> value depends on the values of the I_MPI_SHM_NUM_BUFFERS and I_MPI_SHM_BUFFER_SIZE

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 static mode, only one common shared memory segment is allocated for all processes during the initialization stage.

- If the number of processes started on the system is more than the value specified by <nproc>, the dynamic mode is used. In dynamic mode, the shared memory segments are allocated for each connection.

The default value depends on the values of the I_MPI_SHM_NUM_BUFFERS and I_MPI_SHM_BUFFER_SIZE environment variables. It is equal to 90 in the case of default settings for I_MPI_SHM_NUM_BUFFERS and I_MPI_SHM_BUFFER_SIZE.

The I_MPI_DYNAMIC_CONNECTION environment variable is not applicable when MPI uses the static allocation mode.

I_MPI_SHM_BYPASS

(I_MPI_INTRANODE_SHMEM_BYPASS, I_MPI_USE_DAPL_INTRANODE)

Turn on/off the intra-node communication mode through network fabric along with shm.

Syntax

I_MPI_SHM_BYPASS=<arg>

Deprecated Syntaxes

I_MPI_INTRANODE_SHMEM_BYPASS=<arg>

I_MPI_USE_DAPL_INTRANODE=<arg>

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Turn on the intra-node communication through network fabric
disable | no | off | 0	Turn off the intra-node communication through network fabric. This is the default

Description

Set this variable to specify the communication mode within the node. If the intra-node communication mode through network fabric is enabled, data transfer algorithms are selected according to 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.
- Messages larger than the threshold value of the I_MPI_INTRANODE_EAGER_THRESHOLD variable are transferred through the network fabric layer.

NOTE: This variable is applicable only when shared memory and the network fabric are turned on either by default or by setting the I_MPI_FABRICS environment variable to shm:<fabric>. This mode is available only for dapl and tcp fabrics in MPI 4.0.

3.3.3 DAPL-capable Network Fabrics Control

I_MPI_DAPL_PROVIDER

Define the DAPL provider to load.

Syntax

I_MPI_DAPL_PROVIDER=<name>

Arguments

<name>	Define the name of DAPL provider to load

Description

Set this variable to define the name of DAPL provider to load. This name is also defined in the `dat.conf` configuration file. The DAPL provider name can be also specified inside `I_MPI_FABRICS` variable as `I_MPI_FABRICS=dapl` or `I_MPI_FABRICS=shm:dapl`.

I_MPI_DAT_LIBRARY

Select the DAT library to be used for DAPL provider.

Syntax

`I_MPI_DAT_LIBRARY=<library>`

Arguments

<library>	Specify the DAT library for DAPL provider to be used. Default values are `libdat.so` for DAPL 1.2 providers and `libdat2.so` for DAPL 2.0 providers

Description

Set this variable to select a specific DAT library to be used for DAPL provider. If the library is not located in the dynamic loader search path, specify the full path to the DAT library. This variable affects only on DAPL and DAPL UD capable fabrics.

I_MPI_DAPL_TRANSLATION_CACHE

(I_MPI_RDMA_TRANSLATION_CACHE)

Turn on/off the memory registration cache in the DAPL path.

Syntax

`I_MPI_DAPL_TRANSLATION_CACHE=<arg>`

Deprecated 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
`disable | no | off | 0`	Turn off the memory registration cache

Description

Set this variable to turn on/off the memory registration cache in the DAPL path.

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

I_MPI_DAPL_DIRECT_COPY_THRESHOLD

(I_MPI_RDMA_EAGER_THRESHOLD, RDMA_IBA_EAGER_THRESHOLD)

Change the threshold of the DAPL direct-copy protocol.

Syntax

I_MPI_DAPL_DIRECT_COPY_THRESHOLD=<*nbytes*>

Deprecated Syntaxes

I_MPI_RDMA_EAGER_THRESHOLD=<*nbytes*>

RDMA_IBA_EAGER_THRESHOLD=<*nbytes*>

Arguments

<*nbytes*>	Define the DAPL direct-copy protocol threshold
> 0	The default <*nbytes*> value is equal to 16456 bytes

Description

Set this variable to control the DAPL direct-copy protocol threshold. Data transfer algorithms for the DAPL-capable network fabrics are selected based on the following scheme:

- Messages shorter than or equal to <*nbytes*> are sent using the eager protocol through the internal pre-registered buffers. It involves additional calls of memcpy() function on sender and receiver sides. This approach is faster for short messages.
- Messages larger than <*nbytes*> are sent using the direct-copy protocol. It does not use any buffering but involves registration of memory on sender and receiver sides. The data is transferred directly from a sender to a receiver without calling memcpy()function. This approach is faster for large messages.

I_MPI_DAPL_DYNAMIC_CONNECTION_MODE

(I_MPI_DYNAMIC_CONNECTION_MODE, I_MPI_DYNAMIC_CONNECTIONS_MODE)

Choose the algorithm for establishing the DAPL* connections.

Syntax

I_MPI_DAPL_DYNAMIC_CONNECTION_MODE=<*arg*>

Deprecated Syntax

I_MPI_DYNAMIC_CONNECTION_MODE=<*arg*>

I_MPI_DYNAMIC_CONNECTIONS_MODE=<*arg*>

Arguments

<*arg*>	Mode selector
reject	Deny one of the two simultaneous connection requests. This is the default
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, if two processes initiate the connection simultaneously, one of the requests is rejected.
- 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_DAPL_SCALABLE_PROGRESS

(I_MPI_RDMA_SCALABLE_PROGRESS)

Turn on/off scalable algorithm for DAPL read progress.

Syntax

I_MPI_DAPL_SCALABLE_PROGRESS=<*arg*>

Deprecated 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 enable scalable algorithm for the DAPL read progress. In some cases, this provides advantages for systems with many processes.

I_MPI_DAPL_BUFFER_NUM

(I_MPI_RDMA_BUFFER_NUM, NUM_RDMA_BUFFER)

Change the number of internal pre-registered buffers for each process pair in the DAPL path.

Syntax

I_MPI_DAPL_BUFFER_NUM=<*nbuf*>

Deprecated Syntaxes

I_MPI_RDMA_BUFFER_NUM=<*nbuf*>

NUM_RDMA_BUFFER=<*nbuf*>

Arguments

<*nbuf*>	Define the number of buffers for each pair in a process group
> 0	The default value is 16

Description

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

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

I_MPI_DAPL_BUFFER_SIZE

(I_MPI_RDMA_BUFFER_SIZE, I_MPI_RDMA_VBUF_TOTAL_SIZE)

Change the size of internal pre-registered buffers for each process pair in the DAPL path.

Syntax

I_MPI_DAPL_BUFFER_SIZE=<*nbytes*>

Deprecated Syntaxes

I_MPI_RDMA_BUFFER_SIZE=<nbytes>

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 process pair in the DAPL path. The actual size is calculated by adjusting the <nbytes> to align the buffer to an optimal value.

I_MPI_DAPL_RNDV_BUFFER_ALIGNMENT

(I_MPI_RDMA_RNDV_BUFFER_ALIGNMENT, I_MPI_RDMA_RNDV_BUF_ALIGN)

Define the alignment of the sending buffer for the DAPL direct-copy transfers.

Syntax

I_MPI_DAPL_RNDV_BUFFER_ALIGNMENT=<arg>

Deprecated Syntaxes

I_MPI_RDMA_RNDV_BUFFER_ALIGNMENT=<arg>

I_MPI_RDMA_RNDV_BUF_ALIGN=<arg>

Arguments

<arg>	Define the alignment for the sending buffer
> 0 and a power of 2	The default value is 128

Set this variable to define the alignment of the sending buffer for DAPL direct-copy transfers. When a buffer specified in a DAPL operation is aligned to an optimal value, the data transfer bandwidth may be increased.

I_MPI_DAPL_RDMA_RNDV_WRITE

(I_MPI_RDMA_RNDV_WRITE, I_MPI_USE_RENDEZVOUS_RDMA_WRITE)

Turn on/off the RDMA Write-based rendezvous direct-copy protocol in the DAPL path.

Syntax

I_MPI_DAPL_RDMA_RNDV_WRITE=<arg>

Deprecated Syntaxes

I_MPI_RDMA_RNDV_WRITE=<arg>

I_MPI_USE_RENDEZVOUS_RDMA_WRITE=<arg>

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Turn on the RDMA Write rendezvous direct-copy protocol
disable | no | off | 0	Turn off the RDMA Write rendezvous direct-copy protocol

Description

Set this variable to select the RDMA Write-based rendezvous direct-copy protocol in the DAPL path. Certain DAPL* providers have a slow RDMA Read implementation on certain platforms. Switching on the rendezvous direct-copy protocol based on the RDMA Write operation can increase performance in these cases. The default value depends on the DAPL provider attributes.

I_MPI_DAPL_CHECK_MAX_RDMA_SIZE

(I_MPI_RDMA_CHECK_MAX_RDMA_SIZE)

Check the value of the DAPL attribute `max_rdma_size`.

Syntax

`I_MPI_DAPL_CHECK_MAX_RDMA_SIZE=<arg>`

Deprecated Syntax

`I_MPI_RDMA_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 this mode is enabled, the Intel® MPI Library fragmentizes the messages bigger than the value of the DAPL attribute `max_rdma_size`
- If this mode is disabled, the Intel® MPI Library does not take into account the value of the DAPL attribute `max_rdma_size` for message fragmentation

I_MPI_DAPL_MAX_MSG_SIZE

(I_MPI_RDMA_MAX_MSG_SIZE)

Control message fragmentation threshold.

Syntax

`I_MPI_DAPL_MAX_MSG_SIZE=<nbytes>`

Deprecated Syntax

`I_MPI_RDMA_MAX_MSG_SIZE=<nbytes>`

Arguments

<nbytes>	Define the maximum message size that can be sent through DAPL without fragmentation
`> 0`	If the `I_MPI_DAPL_CHECK_MAX_RDMA_SIZE` variable is enabled, the default *<nbytes>* value is equal to the `max_rdma_size` DAPL attribute value. Otherwise the default value is `MAX_INT`

Description

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

- If the I_MPI_DAPL_CHECK_MAX_RDMA_SIZE variable is set to disable, the Intel® MPI Library fragmentizes the messages whose sizes are greater than <nbytes>.
- If the I_MPI_DAPL_CHECK_MAX_RDMA_SIZE variable is set to enable, the Intel® MPI Library fragmentizes the messages whose sizes are greater than the minimum of <nbytes> and the max_rdma_size DAPL* attribute value.

I_MPI_DAPL_CONN_EVD_SIZE

(I_MPI_RDMA_CONN_EVD_SIZE, I_MPI_CONN_EVD_QLEN)

Define the event queue size of the DAPL event dispatcher for connections.

Syntax

I_MPI_DAPL_CONN_EVD_SIZE=<size>

Deprecated Syntaxes

I_MPI_RDMA_CONN_EVD_SIZE=<size>
I_MPI_CONN_EVD_QLEN=<size>

Arguments

<size>	Define the length of the event queue
> 0	The default value is 2*number of processes + 32 in the MPI job

Description

Set this variable to define the event queue size of the DAPL event dispatcher that handles connection related events. 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.

I_MPI_DAPL_SR_THRESHOLD

Change the threshold of switching send/recv to rdma path for DAPL wait mode.

Syntax

I_MPI_DAPL_SR_THRESHOLD=<arg>

Arguments

<nbytes>	Define the message size threshold of switching send/recv to rdma
>= 0	The default <nbytes> value is 256 bytes

Description

Set this variable to control the protocol used for point-to-point communication in DAPL wait mode:

- Messages shorter than or equal in size to <nbytes> are sent using DAPL send/recv data transfer operations.
- Messages greater in size than <nbytes> are sent using DAPL RDMA WRITE or RDMA WRITE immediate data transfer operations.

I_MPI_DAPL_SR_BUF_NUM

Change the number of internal pre-registered buffers for each process pair used in DAPL wait mode for send/recv path.

Syntax

I_MPI_DAPL_SR_BUF_NUM=<nbuf>

Arguments

<nbuf>	Define the number of send/recv buffers for each pair in a process group
> 0	The default value is 32

Description

Set this variable to change the number of the internal send/recv pre-registered buffers for each process pair.

I_MPI_DAPL_RDMA_WRITE_IMM

(I_MPI_RDMA_WRITE_IMM)

Enable/disable RDMA Write with immediate data InfiniBand (IB) extension in DAPL wait mode.

Syntax

I_MPI_DAPL_RDMA_WRITE_IMM=*<arg>*

Deprecated syntax

I_MPI_RDMA_WRITE_IMM=*<arg>*

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Turn on RDMA Write with immediate data IB extension
disable | no | off | 0	Turn off RDMA Write with immediate data IB extension

Description

Set this variable to utilize RDMA Write with immediate data IB extension. The algorithm is enabled if this environment variable is set and a certain DAPL provider attribute indicates that RDMA Write with immediate data IB extension is supported.

3.3.4 DAPL UD-capable Network Fabrics Control

I_MPI_DAPL_UD

Enable/disable using DAPL UD path.

Syntax

I_MPI_DAPL_UD=*<arg>*

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Turn on using DAPL UD IB extension
disable | no | off | 0	Turn off using DAPL UD IB extension. This is the default value

Description

Set this variable to enable DAPL UD path for transferring data. The algorithm is enabled if you set this environment variable and a certain DAPL provider attribute indicates that UD IB extension is supported.

I_MPI_DAPL_UD_PROVIDER

Define the DAPL provider to work with IB UD transport.

Syntax

I_MPI_DAPL_UD_PROVIDER=<*name*>

Arguments

<*name*>	Define the name of DAPL provider to load

Description

Set this variable to define the name of DAPL provider to load. This name is also defined in the dat.conf configuration file. The DAPL provider name can be also specified inside I_MPI_FABRICS variable as I_MPI_FABRICS=dapl or I_MPI_FABRICS=dapl:shm. Make sure that specified DAPL provider supports UD IB extension.

I_MPI_DAPL_UD_DIRECT_COPY_THRESHOLD

Change the message size threshold of the DAPL UD direct-copy protocol.

Syntax

I_MPI_DAPL_UD_DIRECT_COPY_THRESHOLD=<*nbytes*>

Arguments

<*nbytes*>	Define the DAPL UD direct-copy protocol threshold
> 0	The default <*nbytes*> value is equal to 16456 bytes

Description

Set this variable to control the DAPL UD direct-copy protocol threshold. Data transfer algorithms for the DAPL-capable network fabrics are selected based on the following scheme:

- Messages shorter than or equal to <*nbytes*> are sent using the eager protocol through the internal pre-registered buffers. It involves additional calls of memcpy() function on sender and receiver sides. This approach is faster for short messages.
- Messages larger than <*nbytes*> are sent using the direct-copy protocol. It does not use any buffering but involves registration of memory on sender and receiver sides. The data is transferred directly from a sender to a receiver without calling memcpy() function. This approach is faster for large messages.

I_MPI_DAPL_UD_RECV_BUFFER_NUM

Change the number of the internal pre-registered UD buffers for receiving messages.

Syntax

I_MPI_DAPL_UD_RECV_BUFFER_NUM=<*nbuf*>

Arguments

<*nbuf*>	Define the number of buffers for receiving messages
> 0	The default value is 256 + n*4 where n is a total number of process in MPI job

Description

Set this variable to change the number of the internal pre-registered buffers for receiving messages. These buffers are common for all connections or process pairs.

NOTE: The pre-registered buffers use up memory, however they help avoid the loss of packets.

I_MPI_DAPL_UD_SEND_BUFFER_NUM

Change the number of internal pre-registered UD buffers for sending messages.

Syntax

`I_MPI_DAPL_UD_SEND_BUFFER_NUM=<nbuf>`

Arguments

<nbuf>	Define the number of buffers for sending messages
> 0	The default value is `256 + n*4` where n is a total number of process in MPI job

Description

Set this variable to change the number of the internal pre-registered buffers for sending messages. These buffers are common for all connections or process pairs.

NOTE: The pre-registered buffers use up memory, however they help avoid the loss of packets.

I_MPI_DAPL_UD_ACK_SEND_POOL_SIZE

Change the number of ACK UD buffers for sending messages.

Syntax

`I_MPI_DAPL_UD_ACK_SEND_POOL_SIZE=<nbuf>`

Arguments

<nbuf>	Define the number of ACK UD buffers for sending messages
> 0	The default value is `128`

Description

Set this variable to change the number of the internal pre-registered ACK buffers for sending service messages. These buffers are common for all connections or process pairs.

I_MPI_DAPL_UD_ACK_RECV_POOL_SIZE

Change the number of ACK UD buffers for receiving messages.

Syntax

`I_MPI_DAPL_UD_ACK_RECV_POOL_SIZE=<nbuf>`

Arguments

<nbuf>	Define the number of ACK UD buffers for receiving messages
> 0	The default value is `512+n*4`, where n is total number of process in MPI job

Description

Set this variable to change the number of the internal pre-registered ACK buffers for receiving service messages. These buffers are common for all connections or process pairs.

I_MPI_DAPL_UD_TRANSLATION_CACHE

Turn on/off the memory registration cache in the DAPL path.

Syntax

I_MPI_DAPL_UD_TRANSLATION_CACHE=<*arg*>

Arguments

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

Description

Set this variable to turn off the memory registration cache in the DAPL UD path.

Using the cache substantially improves performance but may lead to correctness issues in certain rare situations. See product *Release Notes* for further details.

I_MPI_DAPL_UD_REQ_EVD_SIZE

Define the event queue size of the DAPL UD event dispatcher for sending data transfer operations.

Syntax

I_MPI_DAPL_UD_REQ_EVD_SIZE=<*size*>

Arguments

<*size*>	Define the length of the event queue
> 0	The default value is 2000

Description

Set this variable to define the event queue size of the DAPL event dispatcher that handles completions of sending DAPL UD data transfer operations (DTO). 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.

I_MPI_DAPL_UD_CONN_EVD_SIZE

Define the event queue size of the DAPL UD event dispatcher for connections.

Syntax

I_MPI_DAPL_UD_CONN_EVD_SIZE=<*size*>

Arguments

<*size*>	Define the length of the event queue
> 0	The default value is 2*number of processes + 32

Description

Set this variable to define the event queue size of the DAPL event dispatcher that handles connection related events. 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.

I_MPI_DAPL_UD_RECV_EVD_SIZE

Define the event queue size of the DAPL UD event dispatcher for receiving data transfer operations.

Syntax

I_MPI_DAPL_UD_RECV_EVD_SIZE=<*size*>

Arguments

<*size*>	Define the length of the event queue
> 0	The default value depends on the number UD and ACK buffers

Description

Set this variable to define the event queue size of the DAPL event dispatcher that handles completions of receiving DAPL UD data transfer operations (DTO). 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.

I_MPI_DAPL_UD_RNDV_MAX_BLOCK_LEN

Define maximum size of block that is passed at one iteration of DAPL UD direct-copy protocol.

Syntax

I_MPI_DAPL_UD_RNDV_MAX_BLOCK_LEN=<*nbytes*>

Arguments

<*arg*>	Define maximum size of block that is passed at one iteration of DAPL UD direct-copy protocol
> 0	The default value is 65536

Set this variable to define the maximum size of memory block that is passed at one iteration of DAPL UD direct-copy protocol. If the size of message in direct-copy protocol is greater than given value, the message will be divided in several blocks and passed in several operations.

I_MPI_DAPL_UD_RNDV_BUFFER_ALIGNMENT

Define the alignment of the sending buffer for the DAPL UD direct-copy transfers.

Syntax

I_MPI_DAPL_UD_RNDV_BUFFER_ALIGNMENT=<*arg*>

Arguments

<*arg*>	Define the alignment of the sending buffer
> 0 and a power of 2	The default value is 16

Set this variable to define the alignment of the sending buffer for DAPL direct-copy transfers. When a buffer specified in a DAPL operation is aligned to an optimal value, this may increase data transfer bandwidth.

I_MPI_DAPL_UD_RNDV_COPY_ALIGNMENT_THRESHOLD

Define threshold where alignment is applied to send buffer for the DAPL UD direct-copy transfers.

Syntax

I_MPI_DAPL_UD_RNDV_COPY_ALIGNMENT_THRESHOLD=<*nbytes*>

Arguments

<*nbytes*>	Define send buffer alignment threshold
> 0 and a power of 2	The default value is 32768

Set this variable to define the threshold where the alignment of the sending buffer is applied in DAPL direct-copy transfers. When a buffer specified in a DAPL operation is aligned to an optimal value, this may increase data transfer bandwidth.

3.3.5 TCP-capable Network Fabrics Control

I_MPI_TCP_NETMASK

(I_MPI_NETMASK)

Choose the network interface for MPI communication over TCP-capable network fabrics.

Syntax

I_MPI_TCP_NETMASK=<*arg*>

Arguments

<*arg*>	Define the network interface (string parameter)
<*interface_mnemonic*>	Mnemonic of the network interface: ib or eth
ib	Select IPoIB*
eth	Select Ethernet. This is the default value
<*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 TCP-capable network fabrics. If you specify a list of interfaces, the first available interface on the node will be used for communication.

Examples

- Use the following setting to select the IP over InfiniBand* (IPoIB) fabric:
  I_MPI_TCP_NETMASK=ib
- Use the following setting to select the specified network interface for socket communications:
  I_MPI_TCP_NETMASK=ib0
- Use the following setting to select the specified network for socket communications. This setting implies the 255.255.0.0 netmask:
  I_MPI_TCP_NETMASK=192.169.0.0

- Use the following setting to select the specified network for socket communications with netmask set explicitly:
  `I_MPI_TCP_NETMASK=192.169.0.0/24`
- Use the following setting to select the specified network interfaces for socket communications:
  `I_MPI_TCP_NETMASK=192.169.0.5/24:ib0:192.169.0.0`

3.3.6 TMI-capable Network Fabrics Control

I_MPI_TMI_LIBRARY

Select the TMI library to be used.

Syntax

`I_MPI_TMI_LIBRARY=<library>`

Arguments

<library>	Specify a TMI library to be used instead of the default `libtmi.so`

Description

Set this variable to select a specific TMI library. Specify the full path to the TMI library if the library does not locate in the dynamic loader search path.

I_MPI_TMI_PROVIDER

Define the name of the TMI provider to load.

Syntax

`I_MPI_TMI_PROVIDER=<name>`

Arguments

<name>	name of the TMI provider to load

Description

Set this variable to define the name of the TMI provider to load. The name must also be defined in the tmi.conf configuration file.

I_MPI_TMI_PROBE_INTERVAL

Define the frequency that the TMI module probes the internal control messages.

Syntax

`I_MPI_TMI_PROBE_INTERVAL=<value>`

Arguments

<value>	Define the frequency that the TMI module probes the internal control messages
`integer > 0`	Exact value for the option

Description

Set this variable to define how often the TMI module should probe for incoming internal control messages. A larger value means less frequent probes. The value `1` means that a probe happens each time the TMI module is polled for progress. The default setting is `20`.

Reducing the probe frequency helps improve the performance when there are a large number of unexpected messages. The trade-off is longer response time for the internal control messages. In MPI 4.0, the internal control messages only affect the MPI functions for one-sided operations (RMA).

3.3.7 OFA*-capable Network Fabrics Control

I_MPI_OFA_NUM_ADAPTERS

Set the number of connection adapters.

Syntax

I_MPI_OFA_NUM_ADAPTERS=<*arg*>

Arguments

<*arg*>	Define the maximum number of connection adapters used
>0	Use the specified number of adapters. The default value is 1

Description

Set the number of the used adapters. If the number is greater than the available number of adapters, all the available adaptors are used.

I_MPI_OFA_ADAPTER_NAME

Set the name of adapter that is used.

Syntax

I_MPI_OFA_ADAPTER_NAME=<*arg*>

Arguments

<*arg*>	Define the name of adapter
Name	Use the specified adapter. By default, any adapter can be used

Description

Set the name of adaptor to be used. If the adapter with specified name does not exist, the library returns error. This has effect only if I_MPI_OFA_NUM_ADAPTERS=1.

I_MPI_OFA_NUM_PORTS

Set the number of used ports on each adapter.

Syntax

I_MPI_OFA_NUM_PORTS=<*arg*>

Arguments

<*arg*>	Define the number of ports that are used on each adapter
> 0	Use the specified number of ports. The default value is 1

Description

Set the number of used ports on each adaptor. If the number is greater than the available number of ports, all the available ports are used.

I_MPI_OFA_NUM_RDMA_CONNECTIONS

Set the maximum number of connections that use the rdma exchange protocol.

Syntax

I_MPI_OFA_NUM_RDMA_CONNECTIONS=<*num_conn*>

Arguments

<*num_conn*>	Define the maximum number of connections that use the rdma exchange protocol
>= 0	Create the specified number of connections that use the rdma exchange protocol. The rest processes use the send/ receive exchange protocol
-1	Create $\log_2$(number of processes) rdma connections
>= number of processes	Create rdma connections for all processes. This is the default value

Description

There are two exchange protocols between two processes: send/receive and rdma. This variable specifies the maximum amount of connections that use rdma protocol.

RDMA protocol is faster but requires more resources. For a large application, you can limit the number of connections that use the rdma protocol so that only processes that actively exchange data use the rdma protocol.

I_MPI_OFA_SWITCHING_TO_RDMA

Set the number of messages that a process should receive before switching this connection to RDMA exchange protocol.

Syntax

I_MPI_OFA_SWITCHING_TO_RDMA=<*number*>

Arguments

<*number*>	Define the number of messages that the process receives before switching to use the rdma protocol
>= 0	If this process receives <*number*> of messages, start using the rdma protocol

Description

Count the number of messages received from the specific process. The connection achieved the specified number tries to switch to rdma protocol for exchanging with that process. The connection will not switch to rdma protocol if the maximum number of connections that use the rdma exchange protocol defined in I_MPI_OFA_NUM_RDMA_CONNECTIONS has been reached.

I_MPI_OFA_RAIL_SCHEDULER

Set the method of choosing rails for short messages.

Syntax

I_MPI_OFA_RAIL_SCHEDULER=<*arg*>

Arguments

<arg>	Mode selector
ROUND_ROBIN	Next time use next rail
FIRST_RAIL	Always use the first rail for short messages
PROCESS_BIND	Always use the rail specific for process

Description

Set the method of choosing rails for short messages. The algorithms are selected according to the following scheme:

- In the ROUND_ROBIN mode, the first message is sent using the first rail; the next message is sent using the second rail, and so on.
- In the FIRST_RAIL mode, the first rail is always used for short messages.
- In the PROCESS_BIND mode, the process with the smallest rank uses the first rail, and the next uses the second rail.

I_MPI_OFA_TRANSLATION_CACHE

Turn on/off the memory registration cache.

Syntax

I_MPI_OFA_TRANSLATION_CACHE=*<arg>*

Arguments

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

Description

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

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

3.3.8 Failover Support in the OFA* Device

The Intel® MPI Library recognizes the following events to detect hardware issues:

- IBV_EVENT_QP_FATAL Error occurred on a QP and it transitioned to error state
- IBV_EVENT_QP_REQ_ERR Invalid request local work queue error
- IBV_EVENT_QP_ACCESS_ERR Local access violation error
- IBV_EVENT_PATH_MIG_ERR A connection failed to migrate to the alternate path
- IBV_EVENT_CQ_ERR CQ is in error (CQ overrun)
- IBV_EVENT_SRQ_ERR Error occurred on an SRQ
- IBV_EVENT_PORT_ERR Link became unavailable on a port
- IBV_EVENT_DEVICE_FATAL CA is in FATAL state

Intel® MPI Library stops using port or whole adapter for communications if one of these issues is detected. The communications will be continued through the available port or adapter if application is running in the multi-rail mode. The application will be aborted if no healthy ports/adapters are available.

Intel® MPI Library also recognizes the following event

- `IBV_EVENT_PORT_ACTIVE` Link became active on a port

The event indicates that the port can be used again and is enabled for communications.

3.4 Dynamic Process Support

The Intel® MPI Library provides support for the MPI-2 process model what allows creation and cooperative termination of processes after an MPI application has started. It provides

- a mechanism to establish communication between the newly created processes and the existing MPI application
- a process attachment mechanism to establish communication between two existing MPI applications even when one of them does not spawn the other

The existing MPD ring (see *mpdboot* for details) is used for the placement of the spawned processes in the round robin fashion. The first spawned process is placed after the last process of the parent group. A specific network fabric combination is selected using the usual fabrics selection algorithm (see *I_MPI_FABRICS* and *I_MPI_FABRICS_LIST* for details).

For example, to run a dynamic application, use the following commands:

```
$ mpdboot -n 4 -r ssh

$ mpiexec -n 1 -gwdir <path_to_executable> -genv I_MPI_FABRICS shm:tcp
<spawn_app>
```

In the example, the `spawn_app` spawns 4 dynamic processes. If the `mpd.hosts` contains the following information,

```
host1
host2
host3
host4
```

the original spawing process is placed on `host1`, while the dynamic processes is distributed as follows: 1 – on `host2`, 2 – on `host3`, 3 – on `host4`, and 4 – again on `host1`.

To run a client-server application, use the following commands on the server host:

```
$ mpdboot -n 1

$ mpiexec -n 1 -genv I_MPI_FABRICS shm:dapl <server_app> > <port_name>
```

and use the following commands on the intended client hosts:

```
$ mpdboot -n 1

$ mpiexec -n 1 -genv I_MPI_FABRICS shm:dapl <client_app> < <port_name>
```

To run a simple MPI_COMM_JOIN based application, use the following commands on the intended host:

```
$ mpdboot -n 1 -r ssh

$ mpiexec -n 1 -genv I_MPI_FABRICS shm:ofa <join_server_app> <
<port_number>

$ mpiexec -n 1 -genv I_MPI_FABRICS shm:ofa <join_client_app> <
<port_number>
```

3.5 Collective Operation Control

Each collective operation in the Intel® MPI Library supports a number of communication algorithms. In addition to highly optimized 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. They are described in the following sections.

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 the reasonable 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 below.

Table 3.5-1 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 5. Binomial gather + scatter algorithm 6. Topology aware binominal gather + scatter algorithm 7. Ring 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 5. Binominal gather + scatter algorithm 6. Topology aware binominal gather + scatter 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 7. Shumilin's bcast 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 3. Shumilin's 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 3. Shumilin's 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 the table below. Here, "n/a" means that the corresponding interval *<l>-<m>* should be omitted.

Table 3.5-2 Message Collective Functions

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	recv_count*recv_type_size if MPI_IN_PLACE is used, otherwise send_count*send_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 the second algorithm for MPI_Reduce operation:
   I_MPI_ADJUST_REDUCE=2
2. Use the following settings to define the algorithms for MPI_Reduce_scatter operation:
   I_MPI_ADJUST_REDUCE_SCATTER=4:0-100,5001-10000;1:101-3200,2:3201-5000;3

In this case. algorithm 4 will be used for the message sizes from 0 up to 100 bytes and from 5001 to 10000 bytes, algorithm 1 will be used for the message sizes from 101 up to 3200 bytes, algorithm 2 will be used for the message sizes from 3201 up to 5000 bytes, and algorithm 3 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 the default library behavior during selection of the most appropriate collective algorithm.

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 3.5-1 for algorithm descriptions):

1. The first algorithm is selected if the message size is less than <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 3.5-1 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 3.5-1 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 3.5-1 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 pair of values 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 3.5-1 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 3.5-1 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 3.5-1 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 Extended File System Support

The Intel® MPI Library provides loadable shared modules to provide native support for the following file systems:

- Panasas* ActiveScale* File System (PanFS)
- Parallel Virtual File System, Version 2 (Pvfs2)
- Lustre* File System

Set the I_MPI_EXTRA_FILESYSTEM environment variable to on to enable parallel file system support. Set the I_MPI_EXTRA_FILESYSTEM_LIST environment variable to request native support for the specific file system. For example, to request native support for Panasas* ActiveScale* File System, do the following:

```
$ mpiexec -env I_MPI_EXTRA_FILESYSTEM on \
  -env I_MPI_EXTRA_FILESYSTEM_LIST panfs -n 2 ./test
```

3.6.1 Environment variables

I_MPI_EXTRA_FILESYSTEM

Turn on/off native parallel file systems support.

Syntax

`I_MPI_EXTRA_FILESYSTEM=<arg>`

Arguments

<arg>	Binary indicator
`enable | yes | on | 1`	Turn on native support for the parallel file systems
`disable | no | off | 0`	Turn off native support for the parallel file systems. This is the default value

Description

Set this variable to enable parallel file system support. The `I_MPI_EXTRA_FILESYSTEM_LIST` environment variable must be set to request native support for the specific file system.

I_MPI_EXTRA_FILESYSTEM_LIST

Select specific file systems support.

Syntax

`I_MPI_EXTRA_FILESYSTEM_LIST=<fs>[, <fs>, ... , <fs>]`

Arguments

<fs>	Define a target file system
`panfs`	Panasas* ActiveScale* File System
`pvfs2`	Parallel Virtual File System, Version 2
`lustre`	Lustre* File System

Description

Set this variable to request support for the specific parallel file system. This variable is handled only if the `I_MPI_EXTRA_FYLESYSTEM` is enabled. The Intel® MPI Library will try to load shared modules to support the file systems specified by `I_MPI_EXTRA_FILESYSTEM_LIST`.

3.7 Compatibility Control

The Intel® MPI Library 4.0 implements the MPI-2.1 standard. The following MPI routines are changed:

- `MPI_Cart_create`
- `MPI_Cart_map`

- MPI_Cart_sub
- MPI_Graph_create

If your application depends on the strict pre-MPI-2.1 behavior, set the environment variable I_MPI_COMPATIBILITY to 3.

I_MPI_COMPATIBILITY

Select the runtime compatibility mode.

Syntax

I_MPI_COMPATIBILITY=<*value*>

Arguments

<*value*>	Define compatibility mode
3	Enable the Intel® MPI Library 3.x compatibility mode
4	Disable the Intel® MPI Library 3.x compatibility mode and enable the Intel® MPI Library 4.0 compatible mode. This is the default value

Description

Set this variable to choose the Intel® MPI runtime compatible mode.

3.8 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

The 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 a 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 statistics collection.

Syntax

`I_MPI_STATS=[n-] m`

Arguments

`n, m`	Possible stats levels of the output information
`1`	Output the amount of data sent by each process
`2`	Output the number of calls and amount of transferred data
`3`	Output statistics combined according to the actual arguments
`4`	Output statistics defined by a buckets list
`10`	Output collective operation statistics for all communication contexts

Description

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

NOTE: `n, m` represent the positive integer numbers define range of output information. The statistics from level `n` to level `m` inclusive are output. Omitted *n* value assumes to be 1.

I_MPI_STATS_SCOPE

Select the subsystem(s) to collect statistics for.

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 the 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. This internal operation serves all 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 to collects statistics for. All collective and point-to-point operations, including the point-to-point operations performed inside 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 the 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 the point-to-point operations inside the collectives:
   I_MPI_STATS_SCOPE=p2p:csend

I_MPI_STATS_BUCKETS

Identify a list of ranges for message sizes and communicator sizes that will be used for collecting statistics.

Syntax

I_MPI_STATS_BUCKETS=<msg>[@<*proc*>][,<msg>[@<*proc*>]]…

Arguments

<msg>	Specify range of message sizes in bytes
<l>	Single value of message size
<l>-<m>	Range from *<l>* to *<m>*

<proc>	Specify range of processes (ranks) for collective operations
<p>	Single value of communicator size
<p>-<q>	Range from *<p>* to *<q>*

Description

Set the I_MPI_STATS_BUCKETS variable to define a set of ranges for message sizes and communicator sizes.

Level 4 of the statistics provides profile information for these ranges.

If I_MPI_STATS_BUCKETS variable is not used, then level 4 statistics is not gathered.

If a range is omitted then the maximum possible range is assumed.

Examples

To specify short messages (from 0 to 1000 bytes) and long messages (from 50000 to 100000 bytes), use the following setting:

```
-env I_MPI_STATS_BUCKETS 0-1000,50000-100000
```

To specify messages that have 16 bytes in size and circulate within four process communicators, use the following setting:

```
-env I_MPI_STATS_BUCKETS "16@4"
```

NOTE: When the *@* symbol is present, the variable value must be enclosed in quotes.

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=4
I_MPI_STATS_SCOPE=p2p;coll:allreduce
```

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

```
Intel(R) MPI Library Version 4.0
____ MPI Communication Statistics ____

Stats level: 4
P2P scope:< FULL >
Collectives scope:< Allreduce >

~~~~ Process 0 of 2 on node svlmpihead01 lifetime = 414.13

Data Transfers
Src     Dst    Amount(MB)   Transfers
----------------------------------------
000 --> 000    0.000000e+00 0
000 --> 001    7.629395e-06 2
========================================
Totals         7.629395e-06 2

Communication Activity
Operation      Volume(MB)   Calls
----------------------------------------
P2P
Csend          7.629395e-06 2
Send           0.000000e+00 0
Bsend          0.000000e+00 0
Rsend          0.000000e+00 0
Ssend          0.000000e+00 0
Collectives
Allreduce       7.629395e-06      2
========================================

Communication Activity by actual args
P2P
Operation      Dst    Message size Calls
-----------------------------------------
Csend
1       1      4            2
Collectives
Operation      Context      Algo   Comm size    Message size Calls  Cost(%)
-------------------------------------------------------------------------
Allreduce
1              0            1      2            4            2       44.96
```

```
================================================================================

~~~~ Process 1 of 2 on node svlmpihead01 lifetime = 306.13

Data Transfers
Src     Dst    Amount(MB)   Transfers
----------------------------------------
001 --> 000    7.629395e-06 2
001 --> 001    0.000000e+00 0
========================================
Totals         7.629395e-06 2

Communication Activity
Operation      Volume(MB)   Calls
----------------------------------------
P2P
Csend          7.629395e-06 2
Send           0.000000e+00 0
Bsend          0.000000e+00 0
Rsend          0.000000e+00 0
Ssend          0.000000e+00 0
Collectives
Allreduce       7.629395e-06      2
========================================

Communication Activity by actual args
P2P
Operation      Dst    Message size Calls
-----------------------------------------
Csend
1       0      4            2
Collectives
Operation      Context      Comm size    Message size Calls  Cost(%)
------------------------------------------------------------------------
Allreduce
1       0      2            4            2      37.93
========================================================================

____ End of stats.txt file ____
```

In the example above all times are measured in microseconds. The message sizes are counted in bytes. **MB** means megabyte equal to 2^{20} or 1 048 576 bytes. The process life time is calculated as a stretch of time between `MPI_Init` and `MPI_Finalize`. The **Algo** field indicates the number of

algorithm used by this operation with listed arguments. The **Cost** field represents a particular collective operation execution time as a percentage of the process life time.

5 Fault Tolerance

Intel® MPI Library provides extra functionality to enable fault tolerance support in the MPI applications. The MPI standard does not define behavior of MPI implementation if one or several processes of MPI application are abnormally aborted. By default, Intel® MPI Library aborts the whole application if any process stops.

Set the environment variable `I_MPI_FAULT_CONTINUE` to `on` to change this behavior. For example,

```
$ mpiexec -env I_MPI_FAULT_CONTINUE on -n 2 ./test
```

An application can continue working in the case of MPI processes an issue if the issue meets the following requirements:

- An application sets error handler `MPI_ERRORS_RETURN` to communicator `MPI_COMM_WORLD` (all new communicators inherit error handler from it)
- An application uses master-slave model and the application will be stopped only if the master is finished or does not respond
- An application uses only point-to-point communication between a master and a number of slaves. It does not use inter slave communication or MPI collective operations.
- Handle a certain MPI error code on a point-to-point operation with a particular failed slave rank for application to avoid further communication with this rank. The slave rank can be blocking/non-blocking send, receive, probe and test,
- Any communication operation can be used on subset communicator. If error appears in collective operation, any communication inside this communicator will be prohibited.
- Master failure means the job stops.

5.1 Environment Variables

I_MPI_FAULT_CONTINUE

Turn on/off support for fault tolerant applications.

Syntax

`I_MPI_FAULT_CONTINUE=<arg>`

Arguments

<arg>	Binary indicator
`enable | yes | on | 1`	Turn on support for fault tolerant applications
`disable | no | off | 0`	Turn off support for fault tolerant applications. This is default value

Description

Set this variable to provide support for fault tolerant applications.

5.2 Usage Model

An application sets MPI_ERRORS_RETURN error handler and checks return code after each communication call. If a communication call does not return, MPI_SUCCESS destination process should be marked unreachable and exclude communication with it. For example:

```
if(live_ranks[rank]) {
      mpi_err = MPI_Send(buf, count, dtype, rank, tag, MPI_COMM_WORLD);
      if(mpi_err != MPI_SUCCESS) {
            live_ranks[rank] = 0;
      }
}
```

In the case of non-blocking communications, errors can appear during wait/test operations.

6 ILP64 Support

The term *ILP64* means that int, long, and pointer data entities all occupy 8 bytes. This differs from the more conventional LP64 model in which only long and pointer data entities occupy 8 bytes while int entities stay at 4 byte size. More information on the historical background and the programming model philosophy can be found for example in http://www.unix.org/version2/whatsnew/lp64_wp.html

6.1 Using ILP64

Use the following options to enable the ILP64 interface

- Use the Fortran compiler driver option `-i8` for separate compilation and the `-ilp64` option for separate linkage. For example,

```
$ mpiifort -i8 -c test.f
$ mpiifort -ilp64 -o test test.o
```

- Use the `mpiexec -ilp64` option to preload the ILP64 interface. For example,

```
$ mpiexec -ilp64 -n 2 ./myprog
```

6.2 Known Issues and Limitations

- Datatype counts and other arguments with values larger than 2^{31}-1 are not supported.
- Special MPI types `MPI_FLOAT_INT`, `MPI_DOUBLE_INT`, `MPI_LONG_INT`, `MPI_SHORT_INT`, `MPI_2INT`, `MPI_LONG_DOUBLE_INT`, `MPI_2INTEGER` are not changed and still use a 4-byte integer field.
- Predefined communicator attributes `MPI_APPNUM`, `MPI_HOST`, `MPI_IO`, `MPI_LASTUSEDCODE`, `MPI_TAG_UB`, `MPI_UNIVERSE_SIZE`, and `MPI_WTIME_IS_GLOBAL` are returned by the functions `MPI_GET_ATTR` and `MPI_COMM_GET_ATTR` as 4-byte integers. The same holds for the predefined attributes that may be attached to the window and file objects.
- Do not use the `-i8` option to compile MPI callback functions, such as error handling functions, user-defined reduction operations, etc.
- You have to use a special ITC library if you want to use the Intel® Trace Collector with the Intel MPI ILP64 executable files. If necessary, the Intel MPI `mpiifort` compiler driver will select the correct ITC library automatically.
- Use the `mpif.h` file instead of the MPI module in Fortran90* applications. The Fortran module supports 32-bit `INTEGER` size only.
- There is currently no support for C and C++ applications.

7 Unified Memory Management

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

- 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:

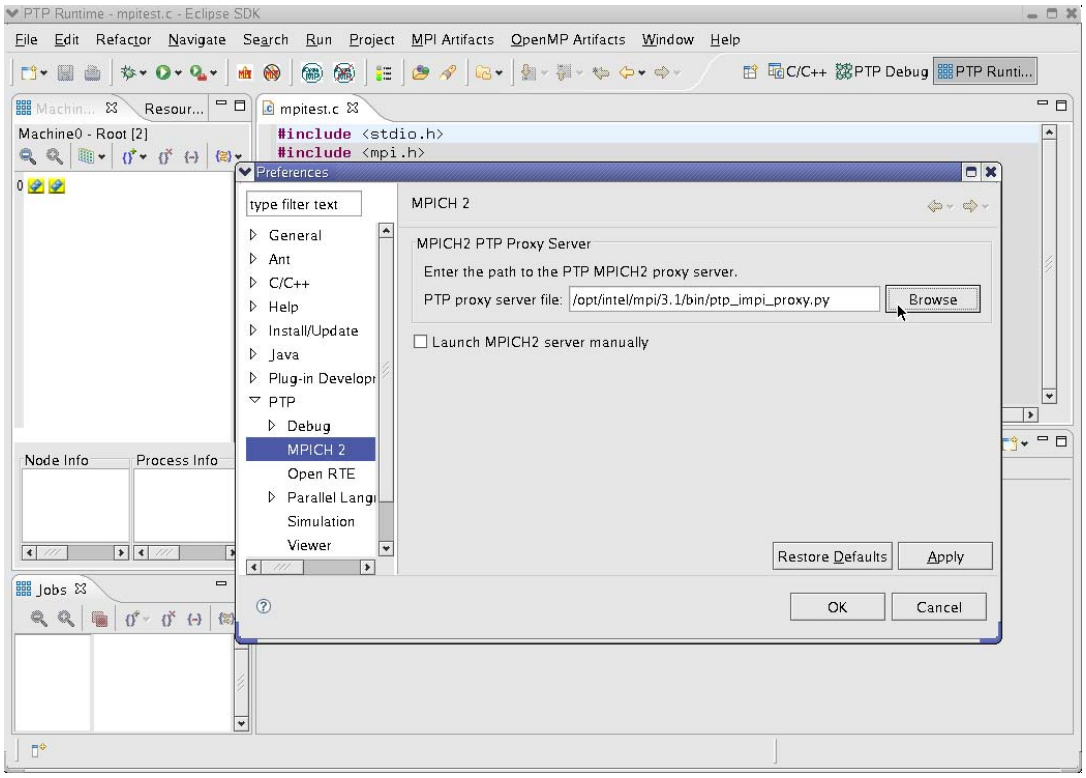
```
#include <i_malloc.h>
#include <my_malloc.h>

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
}
```

8 Integration into Eclipse* PTP

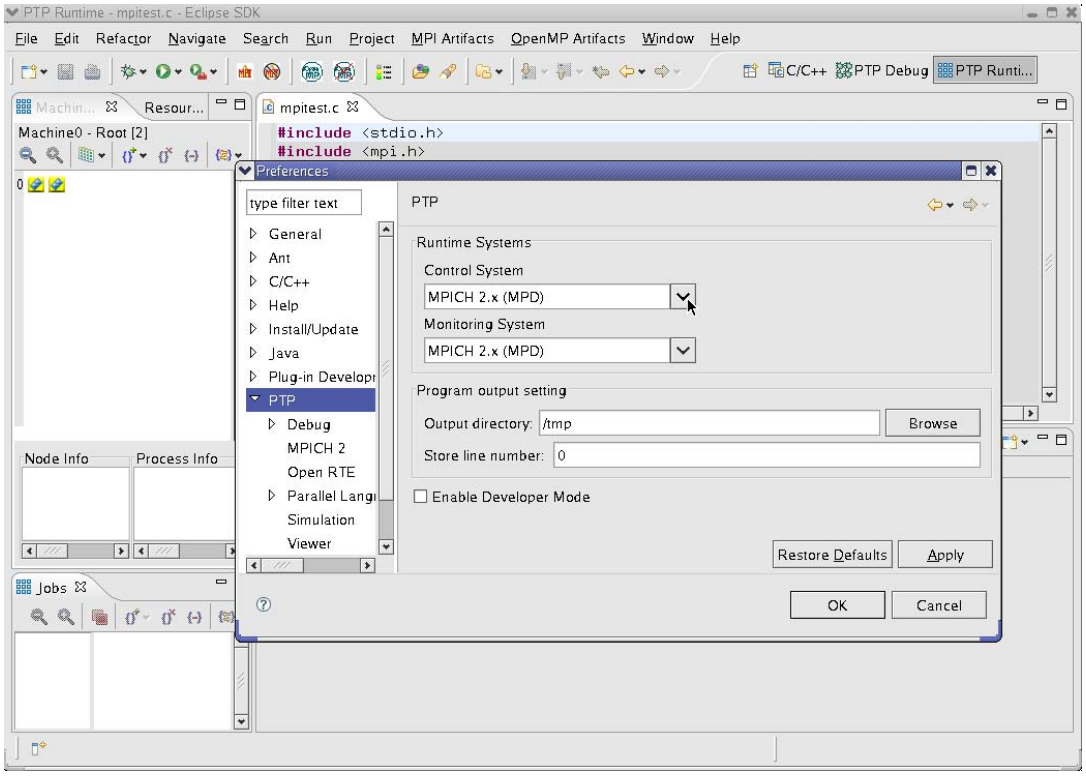
The Intel® MPI Library can be used with the Eclipse* Parallel Tools Platform (PTP). You can launch parallel applications on the existing MPD ring from the Eclipse PTP graphical user interface. The MPD ring must be started prior to the PTP startup.

Perform the following configuration steps to use PTP with the Intel® MPI Library:

1. Set the `PTPPATH` environment variable to specify the 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, for example, `<installdir>/bin/ptp_impi_proxy.py`. Click the **Apply** button.



4. Go to the PTP preference page.
5. Select MPICH2* (MPD) in both Control System and Monitoring System drop down menus. 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 will see the cluster nodes on which the MPD ring is currently working.
8. Refer to the PTP User's Guide for more information. The PTP documentation is available at: *http://www.eclipse.org/ptp/doc.php*

9 Glossary

hyper-threading technology	A feature within the IA-32 family of processors, where each processor core provides the functionality of more than one logical processor.
logical processor	The basic modularity of processor hardware resource that allows a software executive (OS) to dispatch task or execute a thread context. Each logical processor can execute only one thread context at a time.
multi-core processor	A physical processor that contains more than one processor core.
multi-processor platform	A computer system made of two or more physical packages.
processor core	The circuitry that provides dedicated functionalities to decode, execute instructions, and transfer data between certain sub-systems in a physical package. A processor core may contain one or more logical processors.
physical package	The physical package of a microprocessor capable of executing one or more threads of software at the same time. Each physical package plugs into a physical socket. Each physical package may contain one or more processor cores.
processor topology	Hierarchical relationships of “shared vs. dedicated” hardware resources within a computing platform using physical package capable of one or more forms of hardware multi-threading.

10 Index

$HOME/.mpd.conf, 42
(I_MPI_RDMA_RNDV_WRITE, 77
-{cc,cxx,fc,f77,f90}=<*compiler*>, 11
-a <*alias*>, 21
-check_mpi, 10
–check_mpi [<*checking_library*>], 19
-compchk, 11
-config=<*name*>, 9
-configfile <*filename*>, 21
cpuinfo, 46
-dynamic_log, 10
-ecfn <*filename*>, 21
-echo, 10, 36
Eclipse Parallel Tools Platform, 111
-env <*ENVVAR*> <*value*>, 22, 32
-envall, 22, 32
-envexcl <*list of env var names*>, 22
-envlist <*list of env var names*>, 22, 31, 32
-envnone, 22, 32
-envuser, 22
-fast, 10
-g, 10, 19, 24, 41
-gcc-version=<*nnn*>, 11
-gdb, 20
-gdba <*jobid*>, 20
-genv <*ENVVAR*> <*value*>, 19, 31
-genvall, 19, 31
-genvnone, 19, 31
-genvuser, 19
-gm, 17
-GM, 17
-grr <*# of processes*>, 18
-h, 17, 36, 37, 38, 39, 40, 41, 48, 49
--help, 17, 36, 37, 38, 39, 40, 41, 42, 48, 49
–help, 17
-host <*nodename*>, 22
I_MPI_{CC,CXX,FC,F77,F90}, 12, 13
I_MPI_{CC,CXX,FC,F77,F90}_PROFILE, 12
I_MPI_ADJUST_<*opname*>, 91
I_MPI_ALLGATHER_MSG, 96
I_MPI_ALLREDUCE_MSG, 97
I_MPI_ALLTOALL_MSG, 96
I_MPI_ALLTOALL_NUM_PROCS, 96
I_MPI_BCAST_MSG, 95
I_MPI_BCAST_NUM_PROCS, 95
I_MPI_CACHE_BYPASS, 69, 70
I_MPI_CACHE_BYPASS_THRESHOLDS, 70
I_MPI_CHECK_COMPILER, 13
I_MPI_CHECK_PROFILE, 10, 13
I_MPI_COMPATIBILITY, 100
I_MPI_COMPILER_CONFIG_DIR, 14
I_MPI_CONN_EVD_QLEN, 79
I_MPI_DAPL_BUFFER_NUM, 76
I_MPI_DAPL_BUFFER_SIZE, 76
I_MPI_DAPL_CHECK_MAX_RDMA_SIZE, 78, 79
I_MPI_DAPL_CONN_EVD_SIZE, 79
I_MPI_DAPL_DIRECT_COPY_THRESHOLD, 74, 75
I_MPI_DAPL_MAX_MSG_SIZE, 78
I_MPI_DAPL_PROVIDER, 64, 73
I_MPI_DAPL_RDMA_RNDV_WRITE, 77
I_MPI_DAPL_RNDV_BUFFER_ALIGNMENT, 77
I_MPI_DAPL_SCALABLE_PROGRESS, 76
I_MPI_DAPL_TRANSLATION_CACHE, 74
I_MPI_DAPL_UD, 64, 80, 81, 82, 83, 84
I_MPI_DAPL_UD_ACK_RECV_POOL_SIZE, 82
I_MPI_DAPL_UD_ACK_SEND_POOL_SIZE, 82
I_MPI_DAPL_UD_CONN_EVD_SIZE, 83
I_MPI_DAPL_UD_DIRECT_COPY_THRESHOLD, 81
I_MPI_DAPL_UD_PROVIDER, 64, 81
I_MPI_DAPL_UD_RECV_BUFFER_NUM, 81
I_MPI_DAPL_UD_RECV_EVD_SIZE, 84
I_MPI_DAPL_UD_REQ_EVD_SIZE, 83
I_MPI_DAPL_UD_RNDV_BUFFER_ALIGNMENT, 84
I_MPI_DAPL_UD_SEND_BUFFER_NUM, 82
I_MPI_DAPL_UD_TRANSLATION_CACHE, 83
I_MPI_DAT_LIBRARY, 74
I_MPI_DEBUG, 10, 21, 23, 24
I_MPI_DEVICE, 16, 17, 21, 23, 63, 64
I_MPI_DYNAMIC_CONNECTION, 68, 69, 73, 75
I_MPI_DYNAMIC_CONNECTION_MODE, 75
I_MPI_EAGER_THRESHOLD, 66, 67
I_MPI_EXTRA_FILESYSTEM, 98, 99
I_MPI_EXTRA_FILESYSTEM_LIST, 98, 99
I_MPI_FABRICS, 63, 64, 65, 66, 74, 81
I_MPI_FABRICS_LIST, 64, 65, 66, 90
I_MPI_FALLBACK, 64, 65, 66, 90
I_MPI_FALLBACK_DEVICE, 16, 17, 65, 66
I_MPI_FAST_COLLECTIVES, 94, 95
I_MPI_GATHER_MSG, 98
I_MPI_INTRANODE_DIRECT_COPY, 67
I_MPI_INTRANODE_EAGER_THRESHOLD, 66, 67, 73
I_MPI_INTRANODE_SHMEM_BYPASS, 73
I_MPI_JOB_CHECK_LIBS, 19, 25
I_MPI_JOB_CONFIG_FILE, 43

I_MPI_JOB_CONTEXT, 39, 43, 44
I_MPI_JOB_FAST_STARTUP, 27, 28
I_MPI_JOB_SIGNAL_PROPAGATION, 26
I_MPI_JOB_STARTUP_TIMEOUT, 25
I_MPI_JOB_TAGGED_PORT_OUTPUT, 44
I_MPI_JOB_TIMEOUT, 25, 26
I_MPI_JOB_TIMEOUT_SIGNAL, 26
I_MPI_JOB_TRACE_LIBS, 19, 24, 33, 34, 35
I_MPI_MPD_CHECK_PYTHON, 44
I_MPI_MPD_TMPDIR, 45
I_MPI_NETMASK, 85, 86
I_MPI_OFA_NUM_ADAPTERS, 87
I_MPI_OFA_NUM_PORTS, 87
I_MPI_OFA_NUM_RDMA_CONNECTIONS, 88
I_MPI_OFA_RAIL_SCHEDULER, 88
I_MPI_OFA_SWITCHING_TO_RDMA, 88
I_MPI_OUTPUT_CHUNK_SIZE, 27
I_MPI_PERHOST, 24, 35, 45
I_MPI_PIN, 52, 53, 55
I_MPI_PIN_DOMAIN, 57
I_MPI_PIN_MODE, 52, 53
I_MPI_PIN_PROCESSOR_LIST, 55
I_MPI_PMI_EXTENSIONS, 27
I_MPI_RDMA_BUFFER_NUM, 76, 79
I_MPI_RDMA_BUFFER_SIZE, 76, 77
I_MPI_RDMA_CHECK_MAX_RDMA_SIZE, 78
I_MPI_RDMA_CONN_EVD_SIZE, 79
I_MPI_RDMA_MAX_MSG_SIZE, 78
I_MPI_RDMA_RNDV_BUF_ALIGN, 77
I_MPI_RDMA_RNDV_BUFFER_ALIGNMENT, 77, 84, 85
I_MPI_RDMA_RNDV_WRITE, 77
I_MPI_RDMA_SCALABLE_PROGRESS, 76
I_MPI_RDMA_TRANSLATION_CACHE, 74, 89
I_MPI_RDMA_VBUF_TOTAL_SIZE, 76, 77
I_MPI_RDMA_WRITE_IMM, 80
I_MPI_REDSCAT_MSG, 97
I_MPI_ROOT, 14
I_MPI_SCALABLE_OPTIMIZATION, 68
I_MPI_SCATTER_MSG, 98
I_MPI_SHM_BUFFER_SIZE, 71
I_MPI_SHM_BYPASS, 73
I_MPI_SHM_CACHE_BYPASS, 69
I_MPI_SHM_CELL_NUM, 71
I_MPI_SHM_CELL_SIZE, 66, 71, 72
I_MPI_SHM_FBOX_SIZE, 72
I_MPI_SHM_LMT_BUFFER_NUM, 70, 71
I_MPI_SHM_LMT_BUFFER_SIZE, 71
I_MPI_SHM_NUM_BUFFERS, 70, 71
I_MPI_SHM_SINGLE_SEGMENT_THRESHOLD, 72
I_MPI_SOCK_SCALABLE_OPTIMIZATION, 68
I_MPI_SPIN_COUNT, 67
I_MPI_STATS, 101, 102, 103
I_MPI_STATS_BUCKETS, 102, 103
I_MPI_STATS_FILE, 103
I_MPI_STATS_SCOPE, 101, 102, 103
I_MPI_TCP_NETMASK, 85
I_MPI_TIMER_KIND, 100
I_MPI_TMI_LIBRARY, 86
I_MPI_TMI_USE_IPROBE, 86
I_MPI_TRACE_PROFILE, 9, 12, 13
I_MPI_TUNER_DATA_DIR, 28
I_MPI_USE_DAPL_INTRANODE, 73
I_MPI_USE_DYNAMIC_CONNECTIONS, 68, 69
I_MPI_USE_RENDEZVOUS_RDMA_WRITE, 77
I_MPI_WAIT_MODE, 68
-ib, 16
-IB, 16
-idb, 20
IDB_HOME, 20, 28
-idba <*jobid*>, 20
-ifhn <*interface/hostname*>, 21, 36
-ilp64, 10
-l, 21, 36, 39, 40
-m, 21, 37
-machinefile <*machine file*>, 18
mpd, 29, 36, 37, 38, 39, 40, 41, 42, 43, 44, 49, 53
mpd.hosts, 37, 38
mpdallexit, 38
mpdboot, 29, 36, 37, 38, 43, 44
mpdcheck, 40
mpdcleanup, 38, 39
mpdexit, 38
mpdhelp, 42
mpdkilljob, 42
mpdlistjobs, 41, 42
mpdringtest, 40
mpdsigjob, 41
mpdtrace, 29, 38, 39
mpiexec, 15, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 29, 30, 34, 50, 64, 65, 90, 91
mpirun, 29
mpitune, 17, 29, 48, 50
-mt_mpi, 8
-mx, 16, 17
-MX, 17
-n <*# of processes*> or -np <*# of processes*>, 22, 32
-noconf, 21
-nolocal, 18
NUM_RDMA_BUFFER, 76
-O, 10
-ordered-output, 21
PATH, 8, 10
-path <*directory*>, 22

-perhost <*# of processes*>, 18
-ppn <*# of processes*>, 18
-profile=<*profile_name*>, 9, 12
-rdma, 16
-RDMA, 16
-rr, 18
-s <*spec*>, 21
-show, 10
-static, 9
-static_mpi, 9
-t or –trace, 9
TMPDIR, 45
TOTALVIEW, 28
-trace [<*profiling_library*>] or -t [<*profiling_library*>], 19
-tune, 17, 50
-tv, 19
-tva <*jobid*>, 20
-tvsu, 20
-umask <*umask*>, 23
-v, 11, 37, 40
-version or -V, 17, 30
VT_ROOT, 9, 10, 14
-wdir <*directory*>, 22, 31