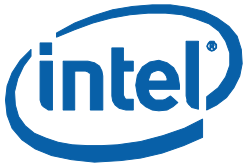


Intel® MPI Library for Windows* OS

Reference Manual

Document Number: 315399-011

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

Contents

1. Introduction 6
1.1. Introducing Intel® MPI Library 6
1.2. Intended Audience 6
1.3. What's New 7
1.4. Notational Conventions 7
1.5. Related Information 7
2. Command Reference 9
2.1. Compiler Commands 9
2.1.1. Compiler Command Options 10
2.1.2. Configuration Files 12
2.1.3. Profiles 12
2.1.4. Environment Variables 13
2.2. Job Startup Command 15
2.2.1. Global Options 16
2.2.2. Local Options 19
2.2.3. Environment Variables 20
2.3. Simple Multi-Purpose Daemon* 27
2.4. Scalable Process Management System (Hydra) 30
2.4.1. Hydra Service 30
2.4.2. Job Startup Commands 31
2.4.3. Global Options 31
2.4.4. Local Options 41
2.4.5. Extended Device Control Options 42
2.4.6. Environment Variables 43
2.5. Integration with Microsoft* HPC Job Scheduler 50
2.6. Integration with PBS Pro* Job Scheduler 51
2.7. Hetero Operating System Cluster Support 51
2.8. Processor Information Utility 52
3. User Authorization 56
3.1. Overview 56
3.2. Installation 56
3.2.1. Active Directory* Setup 57
3.3. Environment Variables 58
4. Tuning Reference 59
4.1. Using mpitune Utility 59
4.1.1. Cluster Specific Tuning 64
4.1.2. Application Specific Tuning 65
4.1.3. Tuning Utility Output 66
4.2. Process Pinning 66
4.2.1. Processor Identification 66
4.2.2. Environment Variables 67
4.2.3. Interoperability with OpenMP* API 74
4.3. Fabrics Control 85
4.3.1. Communication Fabrics Control 85
4.3.2. Shared Memory Control 91
4.3.3. DAPL-capable Network Fabrics Control 97

4.3.4. TCP-capable Network Fabrics Control 106
4.4. Collective Operation Control 108
4.4.1. I_MPI_ADJUST Family 108
4.5. Miscellaneous 115
4.5.1. Compatibility Control 115
4.5.2. Dynamic Process Support 116
4.5.3. Statistics Gathering Mode 117
4.5.4. ILP64Support 138
4.5.5. Unified Memory Management 139
4.6. Secure Loading of Dynamic Link Libraries* 140
5. Graphical Utilities 142
6. Glossary 146
7. Index 147

Disclaimer and Legal Notices

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors which may cause deviations from published specifications.

MPEG-1, MPEG-2, MPEG-4, H.261, H.263, H.264, MP3, DV, VC-1, MJPEG, AC3, AAC, G.711, G.722, G.722.1, G.722.2, AMRWB, Extended AMRWB (AMRWB+), G.167, G.168, G.169, G.723.1, G.726, G.728, G.729, G.729.1, GSM AMR, GSM FR are international standards promoted by ISO, IEC, ITU, ETSI, 3GPP and other organizations. Implementations of these standards, or the standard enabled platforms may require licenses from various entities, including Intel Corporation.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Intel, the Intel logo, BlueMoon, BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino Inside, Cilk, Core Inside, E-GOLD, Flexpipe, i960, Intel, the Intel logo, Intel AppUp, Intel Atom, Intel Atom Inside, Intel Core, Intel Inside, Intel Insider, the Intel Inside logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel Sponsors of Tomorrow., the Intel Sponsors of Tomorrow. logo, Intel StrataFlash, Intel vPro, Intel XScale, Intel True Scale Fabric, InTru, the InTru logo, the InTru Inside logo, InTru soundmark, Itanium, Itanium Inside, MCS, MMX, MPSS, Moblin, Pentium, Pentium Inside, Puma, skoool, the skoool logo, SMARTi, Sound Mark, Stay With It, The Creators Project, The Journey Inside, Thunderbolt, Ultrabook, vPro Inside, VTune, Xeon, Xeon Phi, Xeon Inside, X-GOLD, XMM, X-PMU and XPOSYS are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Microsoft, Windows, and the Windows logo are trademarks, or registered trademarks of Microsoft Corporation in the United States and/or other countries.

Java is a registered trademark of Oracle and/or its affiliates.

Bluetooth is a trademark owned by its proprietor and used by Intel Corporation under license.

Intel Corporation uses the Palm OS* Ready mark under license from Palm, Inc.

OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

1. Introduction

This *Reference Manual* provides you with command and tuning reference for the Intel® MPI Library. The *Reference Manual* contains the following sections

Document Organization

Section	Description
Section 1 Introduction	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 User Authorization	Section 3 describes different user authorization methods
Section 4 Tuning Reference	Section 4 describes environment variables used to influence program behavior and performance at run time
Section 5 Graphical Utilities	Section 5 describes graphical user interface (GUI) utilities distributed with the Intel® MPI Library
Section 6 Glossary	Section 6 explains basic terms used in this document
Section 7 Index	Section 7 references options and environment variables names

1.1. Introducing Intel® MPI Library

The Intel® MPI Library is a multi-fabric message passing library that implements the Message Passing Interface, v3.0 (MPI-3.0) specification. It provides a standard library across Intel® platforms that enable adoption of MPI-3.0 functions as their needs dictate.

The Intel® MPI Library enables developers to change or to upgrade processors and interconnects as new technology becomes available 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 scalable process management system (Hydra*) and supporting utilities, dynamic (.dll) 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, debug libraries, program database (`.pdb`) files, and test codes.

1.2. Intended Audience

This *Reference Manual* helps an experienced user understand the full functionality of the Intel® MPI Library.

1.3. What's New

This document reflects the updates for Intel® MPI Library 5.0 Update 3 release for Windows* OS:

The following latest changes in this document were made:

- Update the components list for *Intel® MPI Library Development Kit* (SDK) in the Introducing Intel® MPI Library section.
- Update the Compiler Commands topic in the Command Reference section.
- Update the description of `-configfile` in the Global Options topic of the Job Startup Command section.
- Update the arguments for `I_MPI_DEBUG` and `I_MPI_TUNER_DATA_DIR` in the Environment Variables topic of the Job Startup Command section.
- Update the descriptions of environment variables `I_MPI_DAPL_DIRECT_COPY_THRESHOLD`, `I_MPI_DAPL_BUFFER_NUM` and `I_MPI_DAPL_BUFFER_SIZE` in the DAPL-capable Network Fabrics Control topic.
- Update the IPM Statistics Format topic in the Statistics Gathering Mode section.
- Update the Region Control topic in the Statistics Gathering Mode section.

1.4. Notational Conventions

The following conventions are 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

1.5. 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

This section provides information on different command types and how to use these commands:

- Compiler commands
- Job startup command
- Simple multi-purpose daemon*
- Scalable Process Management System (Hydra)
- Hetero Operating System Cluster Support
- Processor information utility

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	Underlying Compiler	Supported Language(s)	Supported ABI(s)
Common Compilers			
`mpicc.bat`	`cl.exe`	C	64 bit
`mpicxx.bat`	`cl.exe`	C++	64 bit
`mpifc.bat`	`ifort.exe`	Fortran 77/Fortran 95	64 bit
Microsoft* Visual C++* Compilers			
`mpicl.bat`	`cl.exe`	C/C++	64 bit
Intel® Fortran, C++ Compilers Versions 13.1 through 14.0 and Higher			
`mpiicc.bat`	`icl.exe`	C	64 bit
`mpiicpc.bat`	`icl.exe`	C++	64 bit
`mpiifort.bat`	`ifort.exe`	Fortran 77/Fortran 95	64 bit

- Compiler commands are available only in the Intel® MPI Library Development Kit.
- Compiler commands are in the *<installdir>\<arch>\bin* directory. For the Intel® 64 architecture in 64-bit-enabled compiler, commands are in the *<installdir>\intel64\bin* directory.

- The environment settings can be established by sourcing the `<installdir>\intel64\bin\mpivars.bat` script. If you need to establish environment settings for different library configurations, you can pass one of the following arguments to the `mpivars.bat` script to switch to corresponding configurations:
  - `debug`
  - `release`
  - `debug_mt`
  - `release_mt`

  Multi-threaded optimized library is chosen by default.

- Ensure that the corresponding underlying compiler (64-bit, as appropriate) is 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

-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>.lib` 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.

To use this option, include the installation path of the Intel® Trace Collector in the `VT_ROOT` environment variable.

-check_mpi

Use this option to link the resulting executable against the Intel® Trace Collector correctness checking library.

To use this option, include the installation path of the Intel® Trace Collector in the `VT_ROOT` environment variable.

-ilp64

Use this option to enable partial ILP64 support. All integer arguments of the Intel MPI Library are treated as 64-bit 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.

NOTE:

The debugging version of the Intel® MPI Library is built without optimization. See *I_MPI_LINK* option for details about choosing a version of Intel® MPI Library.

-link_mpi=<*arg*>

Use this option to always link the specified version of the Intel® MPI Library. See the *I_MPI_LINK* environment variable for detailed argument descriptions. This option overrides all other options that select a specific library, such as `-mt_mpi` and `-Zi`

/Zi or /Z7 or /ZI/debug

Use these options 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 `/Zi`, `/Z7`, `/ZI` or `debug` builds.

NOTE:

The `/ZI` option is only valid for C/C++ compiler.

-O

Use this option to enable compiler optimization.

Setting this option triggers a call to the `libirc` library. Many of those library routines are more highly optimized for Intel microprocessors than for non-Intel microprocessors.

-echo

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

-show

Use this option to learn how the underlying compiler is invoked, without actually running it. For example, use the following command to see the required compiler flags and options:

```
> mpiicc.bat -show -c test.c
```

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

```
> mpiicc.bat -show -o a.exe test.obj
```

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

-show_env

Use this option to see the environment settings in effect when the underlying compiler is invoked.

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

Use this option to select the underlying compiler.

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

```
> mpiicc.bat -cc=icl.exe -c test.c
```

For this to work, `icl.exe` should be in your path. Alternatively, you can specify the full path to the compiler.

NOTE:

This option works only with the `mpiicc.bat` and the `mpifc.bat` commands.

-v

Use this option to print the compiler driver script version.

2.1.2. Configuration Files

You can create Intel® MPI Library compiler configuration files using the following file naming convention:

*<installdir>**<arch>*\etc\mpi*<compiler>*-*<name>*.conf

where:

<compiler> = {cc,fc}, depending on the language being compiled

<name> = name of underlying compiler

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

To enable changes to the environment based on the compiler command, you need to source this file before compiling or linking.

2.1.3. Profiles

You can select a profile library through the `-profile`option of the Intel® MPI Library compiler drivers.

You can also create your own profile as *<installdir>**<arch>*\etc*<profile_name>*.conf

The following environment 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 `<installdir>\<arch>\etc\myprof.conf` with the following lines:

```
SET PROFILE_PRELIB=<path_to_myprof>\lib\myprof.lib
SET 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

Specify a default profiling library.

Syntax

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

Arguments

`<profile_name>`	Specify a default profiling library.

Description

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

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>
```

Arguments

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

Description

Set this environment 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.

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 environment 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 environment 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 environment variable to change the default location of the compiler configuration files.

I_MPI_LINK

Select a specific version of the Intel® MPI Library for linking.

Syntax

`I_MPI_LINK=<arg>`

Arguments

<arg>	Version of library
opt	The optimized, single threaded version of the Intel® MPI Library
opt_mt	The optimized, multithreaded version of the Intel MPI Library
dbg	The debugging, single threaded version of the Intel MPI Library
dbg_mt	The debugging, multithreaded version of Intel MPI Library

Description

Set this variable to always link against the specified version of the Intel® MPI Library.

2.2. Job Startup Command

mpiexec.smpd

Syntax

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

or

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

or

```
mpiexec.smpd -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.exe or path\name of the executable file
<file>	File with command-line options

Description

By using the first command-line syntax, you can start all MPI processes of the *<executable>* with the single arg-set. For example, the following command executes a.out over the specified *<# of processes>*:

```
> mpiexec.smpd -n <# of processes> a.exe
```

By using the second command-line syntax, you can start several MPI programs or the same MPI program with different arg-sets. For example, the following command would run each given executable on a different host:

```
> mpiexec.smpd  -n 2 -host host1 a.exe : \
-n 2 -host host2 b.exe
```

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>*.

Simple multi-purpose daemon (SMPD) service must already be running in order for `mpiexec.smpd` to succeed.

NOTE:

If path to executable is not in the `PATH` on all nodes in the cluster, specify *<executable>* as *<path>*`\a.exe` rather than `a.exe`.

2.2.1. Global Options

-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.

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 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>*`]`.

-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 executable files a.exe and b.exe using the dapl fabric over host1 and host2 respectively:

```
-host host1 -env I_MPI_DEBUG 2 -env I_MPI_FABRICS shm:dapl -n 2 ./a.exe

-host host2 -env I_MPI_DEBUG 2 -env I_MPI_FABRICS shm:dapl -n 2 ./b.exe
```

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

```
> mpiexec.smpd -configfile <filename>
```

NOTE:

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

-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.

NOTE:

Local options have higher priority than global options:

- The -genv option has the highest priority.
- The options -genvlist, -genvexcl have lower priority than the -genv option.
- The options -genvnone, -genvuser, -genvall have the lowest priority,.

-l

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

-tune

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.

For the Intel® 64 architecture in 64-bit mode, the default location of the configuration files is in the *<installdir>*\intel64\etc directory. Set the I_MPI_TUNER_DATA_DIR environment variable to override the default location.

See *Automatic Tuning Utility* for more details.

-p *<port>* or -port *<port>*

Use this option to specify the SMPD port mpiexec.smpd should connect to. This option can be useful if SMPD is using a non-default port number.

-hosts n *<host1> <num_proc1> <host2> <num_proc2> ... <hostn> <num_procn>*

Use this option to specify a particular hosts list and a number of processes on each of the MPI processes in the current arg-set are to be run. For example, the following command line will run the executable `a.exe` on the hosts `host1` and `host2`. Two processes will be run on host1 and one process on host 2 respectively:

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

-logon

Use this option to prompt for your account name and password.

-delegate

Use this option to enable the domain-based authorization with the delegation ability.

-impersonate

Use this option to enable the limited domain-based authorization. You will not be able to open files on remote machines or access mapped network drives.

-pwdfile *<filename>*

Use this option to read the account and password from the file specified. Put account on the first line and password on the second one.

-nopopup_debug

Use this option to disable the system popup dialog if the process crashes.

-exitcodes

Use this option to print the process exit codes when each process exits.

-verbose

Use this option to redirect the `smpd` output to `stdout`.

-localroot

Use this option to launch the root process directly from `mpiexec.smpd` if the host is local. You can use this option to launch GUI applications. The interactive process should be launched before any other process in a job.

Example:

```
> mpiexec.smpd -n 1 -host <host2> -localroot interactive.exe : -n 1 -host
<host1> background.exe
```

-localonly

Use this option to run an application on the local node only. If you use this option only for the local node, the smpd service is not required.

-register [-user n]

Use this option to encrypt the user name and password to the registry.

-remove [-user n]

Use this option to delete the encrypted credentials from the registry. If no user index is specified, all entries are removed.

-validate [-user n] [-host hostname]

Use this option to validate the encrypted credentials for the current or the specified host. You can set a specific user index when using this option. If no user index is specified, the index `0` is used as the default value.

-timeout *<seconds>*

Use this option to set timeout for the job.

-whoami

Use this option to print the current user name.

-h or -help or --help

Use this option to display the `mpiexec.smpd` help message.

2.2.2. Local Options

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

Use this option to set the number of MPI processes to run with 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.

-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.

-envuser

Use this option to propagate all user environment variables to all MPI processes with the exception of the following environment variables:

```
%ALLUSERSPROFILE%, %APPDATA%, %CommonProgramFiles%, %CommonProgramFiles
(x86)%, %COMPUTERNAME%, %HOMEDRIVE%, %HOMEPATH%, %NUMBER_OF_PROCESSORS%, %OS%,
%PROCESSOR_ARCHITECTURE%, %PROCESSOR_IDENTIFIER%, %PROCESSOR_LEVEL%,
%PROCESSOR_REVISION%, %ProfilePath%, %ProgramFiles%, %ProgramFiles (x86)%,
%SystemDrive%, %SystemRoot%, %TEMP%, %TMP%, %USERDNSDOMAIN%, %USERDOMAIN%,
%USERNAME%, %USERPROFILE%.
```

This is the default setting.

-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 to run MPI processes in the current argument set. For example, the following command runs the executable `a.exe` on host `host1` only:

```
> mpiexec.smpd -n 2 -host host1 ./a.exe
```

-path *<directory>*

Use this option to specify the path to the *<executable>* to run. The separator is **;**

-dir *<directory>* or -wdir *<directory>*

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

-map *<drive:\\host\share>*

Use this option to create network mapped drive on nodes before starting *<executable>*. Network drive will be automatically removed after the job completion.

-mapall

Use this option to request creation of all user created network mapped drives on nodes before starting *<executable>*. Network drives will be automatically removed after the job completion.

2.2.3. Environment Variables

I_MPI_DEBUG

Print out debugging information when an MPI program starts running.

Syntax

```
I_MPI_DEBUG=<level>[,<flags>]
```

Arguments

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

<flags>	Comma-separated list of debug flags
pid	Show process id for each debug message
tid	Show thread id for each debug message for multithreaded library
time	Show time for each debug message
datetime	Show time and date for each debug message
host	Show host name for each debug message
level	Show level for each debug message
scope	Show scope for each debug message
line	Show source line number for each debug message
file	Show source file name for each debug message
nofunc	Do not show routine name
norank	Do not show rank
flock	Synchronize debug output from different process or threads
nobuf	Do not use buffered I/O for debug output

Description

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

NOTE:

NOTE: Set the same *<level>* value for all ranks.

You can specify the output file name for debug information by setting the I_MPI_DEBUG_OUTPUT environment variable.

To simplify process identification, add the + or – sign in front of the numerical value for I_MPI_DEBUG. This setting produces debug output lines prefixed with the MPI process rank, a Windows* OS process id, and a host name as defined at the process launch time. For example, the command:

```
> mpiexec.smpd -n <# of processes> -env I_MPI_DEBUG +2 a.exe
```

or

```
> mpiexec.smpd -n <# of processes> -env I_MPI_DEBUG +2,pid,host a.exe
```

may produce the following output:

```
[rank#pid@hostname]Debug message
```

NOTE:

Compiling with mpiicc.bat /Zi, /Z7 or /Z7 causes considerable amounts of additional debug information to be printed.

I_MPI_DEBUG_OUTPUT

Set output file name for debug information.

Syntax

```
I_MPI_DEBUG_OUTPUT =<arg>
```

Arguments

<arg>	String value
stdout	Output to stdout - default value
stderr	Output to stderr
<file_name>	Specify the output file name for debug information

Description

Set this environment variable if you want to split output of debug information from the output produced by an application. If you use format like *%r*, *%p* or *%h*, rank, pid or host name is added to the file name accordingly.

I_MPI_PRINT_VERSION

Print library version information.

Syntax

`I_MPI_PRINT_VERSION=<arg>`

Arguments

<arg>	Binary indicator
`enable | yes | on | 1`	Print library version information.
`disable | no | off | 0`	No action. This is the default value.

Description

Set this environment variable to enable/disable printing of Intel® MPI library version information when an MPI application starts running.

I_MPI_NETMASK

Choose the network interface for MPI communication over sockets.

Syntax

`I_MPI_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
<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 or interface mnemonics

Description

Set this environment variable to choose the network interface for MPI communication over sockets in the `sock` and `ssm` communication modes. If you specify a list of interfaces, the first available interface on the node will be used for communication.

Examples

1. Use the following setting to select the IP over InfiniBand* (IPoIB) fabric:

```
I_MPI_NETMASK=ib

I_MPI_NETMASK=eth
```

2. Use the following setting to select a particular network for socket communications. This setting implies the `255.255.0.0` netmask:

   ```
   I_MPI_NETMASK=192.169.0.0
   ```

3. Use the following setting to select a particular network for socket communications with netmask set explicitly:

   ```
   I_MPI_NETMASK=192.169.0.0/24
   ```

4. Use the following setting to select the specified network interfaces for socket communications:

   ```
   I_MPI_NETMASK=192.169.0.5/24:ib0:192.169.0.0
   ```

NOTE:

If the library cannot find any suitable interface by the given value of `I_MPI_NETMASK`, the value will be used as a substring to search in the network adapter's description field. And if the substring is found in the description, this network interface will be used for socket communications. For example, if `I_MPI_NETMASK=myri` and the description field contains something like Myri-10G adapter, this interface will be chosen.

I_MPI_JOB_TIMEOUT

(MPIEXEC_TIMEOUT)

Set the `mpiexec.smpd` timeout.

Syntax

`I_MPI_JOB_TIMEOUT=<timeout>`

Deprecated Syntax

`MPIEXEC_TIMEOUT=<timeout>`

Arguments

<timeout>	Define `mpiexec.smpd` timeout period in seconds
<n> >= 0	The default timeout value is zero, meaning no timeout

Description

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

NOTE:

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

I_MPI_SMPD_VERSION_CHECK

Set this environment variable to enable strong SMPD version check. This configuration is valid only for Windows* OS.

Syntax

`I_MPI_SMPD_VERSION_CHECK=`*<arg>*

Arguments

<arg>	Binary indicator
`enable | yes | on | 1`	Check the version and fail in case of mismatch. This is the default value
`disable | no | off | 0`	Only print a warning message in case of a version mismatch and continue working

Description

Set this environment variable to control SMPD version check. The Intel® MPI Library terminates application if the versions mismatch is found. To disable SMPD version check set the `I_MPI_SMPD_VERSION_CHECK` environment variable to `disable`.

I_MPI_DAT_LIBRARY

Select a particular DAT library to be used.

Syntax

`I_MPI_DAT_LIBRARY=`*<library>*

Arguments

<library>	Specify the exact library to be used instead of the default dat.dll

Description

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

NOTE:

Use this environment variable only if you are going to utilize a DAPL* provider.

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>*\intel64\etc

Description

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

I_MPI_PLATFORM

Select the intended optimization platform.

Syntax

`I_MPI_PLATFORM=<platform>`

Arguments

<platform>	Intended optimization platform (string value)
auto[:min]	Optimize for the oldest supported Intel® Architecture Processor across all nodes. This is the default value
auto:max	Optimize for the newest supported Intel® Architecture Processor across all nodes
auto:most	Optimize for the most numerous Intel® Architecture Processor across all nodes. In case of a tie, choose the newer platform
uniform	Optimize locally. The behavior is unpredictable if the resulting selection differs from node to node
none	Select no specific optimization
htn | generic	Optimize for the Intel® Xeon® Processors 5400 series and other Intel® Architecture processors formerly code named Harpertown
nhm	Optimize for the Intel® Xeon® Processors 5500, 6500, 7500 series and other Intel® Architecture processors formerly code named Nehalem
wsm	Optimize for the Intel® Xeon® Processors 5600, 3600 series and other Intel® Architecture processors formerly code named Westmere

`snb`	Optimize for the Intel® Xeon® Processors E3-1200 series and other Intel® Architecture processors formerly code named Sandy Bridge
`ivb`	Optimize for the Intel® Xeon® Processors E3-1225V2, E3-1275V2 series and other Intel® Architecture processors formerly code named Ivy Bridge
`knc`	Optimize for the Intel® Xeon® Processors (codename: Knights Corner). If Intel Xeon Phi coprocessor is present on the cluster, the value is chosen by default.

Description

Set this variable to use the predefined platform settings. It is available for both Intel® and non-Intel microprocessors, but it may utilize additional optimizations for Intel microprocessors than it utilizes for non-Intel microprocessors.

NOTE:

The values `auto:min`, `auto:max` and `auto:most` may increase the MPI job startup time.

I_MPI_PLATFORM_CHECK

Turn on/off the optimization setting similarity check.

Syntax

`I_MPI_PLATFORM_CHECK=<arg>`

Argument

`<arg>`	Binary indicator
`enable | yes | on | 1`	Turns on the optimization platform similarity check. This is the default value
`disable | no | off | 0`	Turns off the optimization platform similarity check

Description

Set this variable to check the optimization platform settings of all processes for similarity. If the settings are not the same on all ranks, the library terminates the program. Disabling this check may reduce the MPI job startup time.

2.3. Simple Multi-Purpose Daemon*

smpd

Simple multi-purpose daemon.

Syntax

```
smpd.exe    [ -h ] [ --help ] [ -port <port> ] [ -d ] \
            [ -install | -regserever ] [ -start ] [ -stop ] \
            [ -shutdown <hostname> ] [ -status <hostname> ] \
            [ -restart <hostname> ] [ -anyport ] [ -hosts ] [ -sethosts ] \
            [ -set <option_name> <option_value> ] [ -get <option_name> ] \
            [ -traceon <logfilename> [ <hostA> <hostB> ... ]] \
            [ -traceoff [ <hostA> <hostB> ...]] \
            [ -remove | -unregister | -uninstall ] [ -register_spn ]\
            [ -remove_spn ] [ -V ] [ -version ]
```

Arguments

`-h | --help`	Display a help message
`-p <port> | -port <port>`	Specify the port that `smpd` is listening on
`-d | -debug`	Start `smpd` in debug mode
`-install -regserver`	Install the `smpd` service
`-start`	Start the `smpd` service
`-stop`	Stop the `smpd` service
`-shutdown <hostname>`	Shutdown `smpd` on specified `<hostname>`
`-status <hostname>`	Get the `smpd` status on specified `<hostname>`
`-restart <hostname>`	Restart `smpd` on specified `<hostname>`
`-anyport`	Use any port for listening
`-hosts`	Get the `smpd` ring list
`-sethosts`	Set the `smpd` ring from specified hosts. This settings impact all users
`-set <option_name> <option_value>`	Register the `<option_name>` key to the `HKEY_LOCAL_MACHINE` registry key
`-get <option_name>`	Get the value of the registered `<option_name>` key from the `HKEY_LOCAL_MACHINE` registry key
`-traceon <logfilename> <hostA> <hostB> ...`	Restart `smpd` and store the output into provided `<logfilename>`

`-traceoff <hostA> <hostB>`	Restart `smpd` without logging the output
`-remove | -unregserver | -uninstall`	Remove `smpd` service
`-register_spn`	Register Service Principle Name (SPN) in the Windows* domain for the cluster node on which this command is executed
`-remove_spn`	Remove SPN from the Windows* domain for the cluster node on which this command is executed
`-V`	Get the Intel® MPI Library version info
`-version`	Display the `smpd` version information

Description

Simple Multipurpose Daemon* (SMPD) is the Intel® MPI Library process management system for starting parallel jobs. Before running a job, start `smpd` service on each host and connect them into a ring.

Use the `smpd.exe` command to install, uninstall, start or stop SMPD service.

Examples:

1. Use the following command to install SMPD service:

```
> smpd.exe -install
```

NOTE:

This command must be run by a user with administrator privileges. After that all users will be able to launch MPI jobs using `mpiexec`.

2. Use the following command to start the SMPD service in debug mode:

```
> smpd.exe -d
```

NOTE: Note:

NOTE: Simple Multipurpose Daemon* (SMPD) has been deprecated since Intel® MPI Library 5.0 release. You can use scalable process management system (Hydra) to start parallel jobs.

2.4. Scalable Process Management System (Hydra)

Hydra on Windows* OS inherits the same environment variables and options as for Hydra on Linux* OS. This section describes the specific options.

2.4.1. Hydra Service

hydra_service

Hydra Service agent.

Syntax

```
hydra_service.exe [ -install | -regserever ] [ -start ] [ -stop ] \
                  [ -remove | -unregister | -uninstall ] [ -register_spn ]
```

Arguments

`-install` `-regserver`	Install the `hydra` service
`-start`	Start the `hydra` service
`-stop`	Stop the `hydra` service
`-shutdown <hostname>`	Shutdown the `hydra` service on the specified `<hostname>`
`-status <hostname>`	Get the `hydra` status on the specified `<hostname>`
`-restart <hostname>`	Restart the `hydra` service on the specified `<hostname>`

`-remove`
`-unregserver`
`-uninstall`
	Remove the `hydra` service

`-register_spn`	Register service principle name (SPN) in the Windows* domain for the cluster node on which this command is executed
`-remove_spn`	Remove SPN from the Windows* domain for the cluster node on which this command is executed

Description

Hydra service agent is a part of the Intel® MPI Library process management system for starting parallel jobs. Before running a job, start the service on each host.

Examples:

1. Use the `hydra_service.exe` command to install, uninstall, start or stop the service.

```
> hydra_service.exe -install
```

NOTE:

This command must be run by a user with administrator privileges. After that all users will be able to launch MPI jobs using `mpiexec.smpd`.

2. Use the following command to remove the service:

```
> hydra_service.exe -remove
```

2.4.2. Job Startup Commands

mpiexec

The `mpiexec` is a more scalable alternative to the SMPD process manager.

Syntax

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

or

```
mpiexec <g-options> <l-options> <executable1> : \
<l-options> <executable2>
```

Arguments

<g-options>	Global options that apply to all MPI processes
<l-options>	Local options that apply to a single arg-set
<executable>	`.\a.exe` or `path\name` of the executable file

2.4.3. Global Options

-hostfile <*hostfile*> or -f <*hostfile*>

Use this option to specify host names on which to run the application. If a host name is repeated, this name is used only once.

See also the *I_MPI_HYDRA_HOST_FILE* environment variable for more details.

NOTE:

Use the *-perhost*, *-ppn*, *-grr*, *and* *-rr* options to change the process placement on the cluster nodes.

-machinefile *<machine file>* or -machine *<machine file>*

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

When you are pinning within a machine, the option `-binding=map` is available within the machine file for each line. For example:

```
> cat .\machinefile
node0:2 binding=map=0,3
node1:2 binding=map=[2,8]
node0:1 binding=map=8
> mpiexec.hydra -machinefile .\machinefile -n 5 -l numactl --show
[4] policy: default
[4] preferred node: current
[4] physcpubind: 8
[4] cpubind: 0
[4] nodebind: 0
[4] membind: 0 1
[0] policy: default
[0] preferred node: current
[0] physcpubind: 0
[0] cpubind: 0
[0] nodebind: 0
[0] membind: 0 1
[1] policy: default
[1] preferred node: current
[1] physcpubind: 3
[1] cpubind: 1
[1] nodebind: 1
[1] membind: 0 1
[3] policy: default
[3] preferred node: current
[3] physcpubind: 3
[3] cpubind: 1
[3] nodebind: 1
[3] membind: 0 1
[2] policy: default
[2] preferred node: current
[2] physcpubind: 1
[2] cpubind: 1
[2] nodebind: 1
[2] membind: 0 1
```

-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 environment variables to be sent to all MPI processes.

-pmi-connect *<mode>*

Use this option to choose the Process Management Interface* (PMI) message caching mode. Possible values for *<mode>* are:

- nocache - do not cache PMI messages.
- cache - cache PMI messages on the local pmi_proxy management processes to minimize PMI requests. Cached information is propagated to the child management processes.
- lazy-cache - cache mode with on-request propagation of the PMI information.

The lazy-cache mode is the default mode.

See the *I_MPI_HYDRA_PMI_CONNECT* environment variable for more details.

-perhost *<# of processes >*, -ppn *<# of processes >*, or -grr *<# of processes>*

Use this option to place the indicated number of consecutive MPI processes on every host in the group using round robin scheduling. See the *I_MPI_PERHOST* environment variable for more details.

-rr

Use this option to place consecutive MPI processes on different hosts using the round robin scheduling. This option is equivalent to -perhost 1. See the *I_MPI_PERHOST* environment variable for more details.

(SDK only) –trace-pt2pt

Use this option to collect the information about point-to-point operations.

(SDK only) –trace-collectives

Use this option to collect the information about collective operations.

NOTE:

Use the -trace-pt2pt and -trace-collectives to reduce the size of the resulting trace file or the number of message checker reports. These options work with both statically and dynamically linked applications.

-configfile *<filename>*

Use this option to specify the file *<filename>* that contains the command-line options. Blank lines and lines that start with '#' as the first character are ignored.

-branch-count *<num>*

Use this option to restrict the number of child management processes launched by the mpiexec command, or by each pmi_proxy management process.

See the *I_MPI_HYDRA_BRANCH_COUNT* environment variable for more details.

-pmi-aggregate or -pmi-noaggregate

Use this option to switch on or off, respectively, the aggregation of the PMI requests. The default value is -pmi-aggregate, which means the aggregation is enabled by default.

See the *I_MPI_HYDRA_PMI_AGGREGATE* environment variable for more details.

-hosts *<nodelist>*

Use this option to specify a particular *<nodelist>* on which to run the MPI processes. For example, the following command runs the executable a.out on hosts host1 and host2:

```
> mpiexec -n 2 -hosts host1,host2 ./a.out
```

NOTE:

If *<nodelist>* consists of only one node, this option is interpreted as a local option. See Local Options for details.

-iface *<interface>*

Use this option to choose the appropriate network interface. For example, if the IP emulation of your InfiniBand* network is configured to ib0, you can use the following command.

```
> mpiexec -n 2 -iface ib0 ./a.out
```

See the *I_MPI_HYDRA_IFACE* environment variable for more details.

-l

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

-tune [*<arg >*]

where:

<arg> = {*<dir_name>*, *<configuration_file>*}.

Use this option to optimize the Intel® MPI Library performance by using the data collected by the mpitune utility.

NOTE:

Use the `mpitune` utility to collect the performance tuning data before using this option.

If *<arg>* is not specified, the best-fit tune options are selected for the given configuration. The default location of the configuration file is *<installdir>*/*<arch>*/`etc` directory.

To specify a different location for the configuration file, set *<arg>=<dir_name>*.

To specify a different configuration file, set *<arg>=<configuration_file>*.

-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 *Configuration Files*.

-ordered-output

Use this option to avoid intermingling of data output from the MPI processes. This option affects both the standard output and the standard error streams.

NOTE:

When using this option, end the last output line of each process with the end-of-line (\n) character. Otherwise the application may stop responding.

-path *<directory>*

Use this option to specify the path to the *<executable>* file.

-tmpdir

Use this option to set a directory for temporary files.

See the *I_MPI_TMPDIR* environment variable for more details.

-version or -V

Use this option to display the version of the Intel® MPI Library.

-info

Use this option to display build information of the Intel® MPI Library. When this option is used, the other command line arguments are ignored.

-delegate

Use this option to enable the domain-based authorization with the delegation ability.

-impersonate

Use this option to enable the limited domain-based authorization. You will not be able to open files on remote machines or access mapped network drives.

-localhost

Use this option to explicitly specify the local host name for the launching node.

Example:

```
> mpiexec -localhost <localhost_ip> -machinefile <file> -n 2 test.exe
```

-localroot

Use this option to launch the root process directly from `mpiexec` if the host is local. You can use this option to launch GUI applications. The interactive process should be launched before any other process in a job.

Example:

```
> mpiexec -n 1 -host <host2> -localroot interactive.exe : -n 1 -host <host1>
background.exe
```

-localonly

Use this option to run an application on the local node only. If you use this option only for the local node, the Hydra service is not required.

-register

Use this option to encrypt the user name and password to the registry.

-remove

Use this option to delete the encrypted credentials from the registry.

-validate

Validate the encrypted credentials for the current host.

-whoami

Use this option to print the current user name.

-map <drive:\\host\share>

Use this option to create network mapped drive on nodes before starting <executable>. Network drive will be automatically removed after the job completion.

-mapall

Use this option to request creation of all user created network mapped drives on nodes before starting <executable>. Network drives will be automatically removed after the job completion.

2.4.3.1. Binding Options

-binding

Use this option to pin or bind MPI processes to a particular processor and avoid undesired process migration. In the following syntax, the quotes may be omitted for a one-member list. Each parameter corresponds to a single pinning property.

This option is supported on both Intel® and non-Intel microprocessors, but it may perform additional optimizations for Intel microprocessors than it performs for non-Intel microprocessors.

Syntax

```
-binding "<parameter>=<value>[;<parameter>=<value> ...]"
```

Parameters

pin	Pinning switch
enable | yes | on | 1	Turn on the pinning property. This is the default value
disable | no | off | 0	Turn off the pinning property

cell	Pinning resolution
unit	Basic processor unit (logical CPU)
core	Processor core in multi-core system

map	Process mapping
spread	The processes are mapped consecutively to separate processor cells. Thus, the processes do not share the common resources of the adjacent cells.

`scatter`	The processes are mapped to separate processor cells. Adjacent processes are mapped upon the cells that are the most remote in the multi-core topology.
`bunch`	The processes are mapped to separate processor cells by #processes/#sockets processes per socket. Each socket processor portion is a set of the cells that are the closest in the multi-core topology.
`p0,p1,...,pn`	The processes are mapped upon the separate processors according to the processor specification on the `p0,p1,...,pn` list: the`i`th process is mapped upon the processor pi, where pi takes one of the following values: • processor number like n • range of processor numbers like n-m • `-1` for no pinning of the corresponding process
`[m0,m1,...,mn]`	The i^{th} process is mapped upon the processor subset defined by m_i hexadecimal mask using the following rule: The j_{th} processor is included into the subset m_i if the j^{th} bit of m_i equals `1`.

`domain`	Processor domain set on a node
`cell`	Each domain of the set is a single processor cell (unit or core).
`core`	Each domain of the set consists of the processor cells that share a particular core.
`cache1`	Each domain of the set consists of the processor cells that share a particular level 1 cache.
`cache2`	Each domain of the set consists of the processor cells that share a particular level 2 cache.
`cache3`	Each domain of the set consists of the processor cells that share a particular level 3 cache.
`cache`	The set elements of which are the largest domains among cache1, cache2, and cache3
`socket`	Each domain of the set consists of the processor cells that are located on a particular socket.

`node`	All processor cells on a node are arranged into a single domain.
`<size>[:<layout>]`	Each domain of the set consists of *`<size>`* processor cells. *`<size>`* may have the following values: • `auto` - domain size = #cells/#processes • `omp` - domain size = OMP_NUM_THREADS environment variable value • positive integer - exact value of the domain size ***NOTE:*** Domain size is limited by the number of processor cores on the node. Each member location inside the domain is defined by the optional *`<layout>`* parameter value: • `compact` - as close with others as possible in the multi-core topology • `scatter` - as far away from others as possible in the multi-core topology • `range` - by BIOS numbering of the processors If *`<layout>`* parameter is omitted, `compact` is assumed as the value of *`<layout>`*

`order`	Linear ordering of the domains
`compact`	Order the domain set so that adjacent domains are the closest in the multi-core topology
`scatter`	Order the domain set so that adjacent domains are the most remote in the multi-core topology
`range`	Order the domain set according to the BIOS processor numbering

`offset`	Domain list offset
`<n>`	Integer number of the starting domain among the linear ordered domains. This domain gets number zero. The numbers of other domains will be cyclically shifted.

2.4.3.2. Bootstrap Options

-bootstrap<*bootstrap server*>

Use this option to select a built-in bootstrap server to use. A bootstrap server is the basic remote node access mechanism that is provided by the system. The default bootstrap server is service.

Arguments

<arg>	Global options that apply to all MPI processes
`service`	Use hydra service agent. This is the default value
`ssh`	Use secure shell
`fork`	Use this option to run an application on the local node only

To enable Intel® MPI Library to use the `-bootstrap ssh` option, provide the ssh connectivity between nodes. Ensure that the corresponding ssh client location is listed in your `PATH` environment variable.

-bootstrap-exec <*bootstrap server*>

Use this option to set the executable to be used as a bootstrap server. For example:

```
$ mpiexec -bootstrap-exec <bootstrap_server_executable> \
-f hosts.file -env <VAR1> <VAL1> -n 2 ./a.out
```

See the *I_MPI_HYDRA_BOOTSTRAP* environment variable for more details.

2.4.3.3. Other Options

-verbose or -v

Use this option to print debug information from `mpiexec`, such as:

- Service processes arguments
- Environment variables and arguments passed to start an application
- PMI requests/responses during a job life cycle

See the *I_MPI_HYDRA_DEBUG* environment variable for more details.

-print-rank-map

Use this option to print out the MPI rank mapping.

-print-all-exitcodes

Use this option to print the exit codes of all processes.

2.4.4. Local Options

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

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

See the *I_MPI_HYDRA_ENV* environment variable for more details.

-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 environment variables to be sent to the MPI processes.

-host <nodename>

Use this option to specify a particular `<nodename>` on which the MPI processes are to be run. For example, the following command executes `a.out` on hosts `host1` and `host2`:

```
> mpiexec -n 2 -host host1 ./a.out : -n 2 -host host2 ./a.out
```

-path <directory>

Use this option to specify the path to the `<executable>` file to be run in the current arg-set.

-wdir <directory>

Use this option to specify the working directory in which the `<executable>` file runs in the current arg-set.

-hostos<host OS>

Use this option to specify an operating system installed on a particular host. MPI processes are launched on each host in accordance with this option specified. The default value is `windows`.

Arguments

`<arg>`	String parameter

`linux`	The host with Linux* OS installed.
`windows`	The host with Windows* OS installed. This is the default value

NOTE:

The option is used in conjunction with `-host` option. For instance, the following command runs the executable `a.exe` on `host1` and `b.out` on `host2`:

```
> mpiexec -n 1 -host host1 -hostos windows a.exe : -n 1 -host host2 \ -
hostos linux ./a.out
```

2.4.5. Extended Device Control Options

-rdma

Use this option to select an RDMA-capable network fabric. The application attempts to use the 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. The application attempts to use the 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 a DAPL capable network fabric. The application attempts to use a DAPL capable network fabric. If no such fabric is available, another fabric 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 a DAPL capable network fabric. 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 an OFA capable network fabric. The application attempts to use an 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 an OFA capable network fabric. 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 a TMI capable network fabric. The application attempts to use a TMI capable network fabric. If no such fabric is available, another fabric 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 a TMI capable network fabric. The application fails 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. 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. 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 Intel® True Scale Fabric. The application attempts to use Intel True Scale 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 Intel True Scale Fabric. 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.

2.4.6. Environment Variables

I_MPI_HYDRA_HOST_FILE

Set the host 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 the host file

Description

Set this environment variable to specify the hosts file.

I_MPI_HYDRA_DEBUG

Print out the debug information.

Syntax

`I_MPI_HYDRA_DEBUG=<arg>`

Arguments

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

Description

Set this environment variable to enable the debug mode.

I_MPI_HYDRA_ENV

Control the environment propagation.

Syntax

`I_MPI_HYDRA_ENV=<arg>`

Arguments

<arg>	String parameter
`all`	Pass all environment to all MPI processes

Description

Set this environment variable to control the environment propagation to the MPI processes. By default, the entire launching node environment is passed to the MPI processes. Setting this variable also overwrites environment variables set by the remote shell.

I_MPI_JOB_TIMEOUT, I_MPI_MPIEXEC_TIMEOUT

(MPIEXEC_TIMEOUT)

Set the timeout period for `mpiexec`.

Syntax

`I_MPI_JOB_TIMEOUT=<timeout>`

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

NOTE:

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

I_MPI_JOB_TIMEOUT_SIGNAL

(MPIEXEC_TIMEOUT_SIGNAL)

Define the signal to be sent when a job is terminated because of 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 sent to stop the MPI job if the timeout period specified by the `I_MPI_JOB_TIMEOUT` environment variable expires. If you set a signal number unsupported by the system, the `mpiexec` operation prints a warning message and continues the task termination using the default signal number `9` (`SIGKILL`).

I_MPI_JOB_ABORT_SIGNAL

Define a signal to be sent to all processes when a job is terminated unexpectedly.

Syntax

`I_MPI_JOB_ABORT_SIGNAL=<number>`

Arguments

<number>	Define signal number
<n> > 0	The default value is `9 (SIGKILL)`

Description

Set this environment variable to define a signal for task termination. If you set an unsupported signal number, `mpiexec` prints a warning message and uses the default signal `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 environment variable to control propagation of the signals (`SIGINT`, `SIGALRM`, and `SIGTERM`). If you enable signal propagation, the received signal is sent to all processes of the MPI job. If you disable signal propagation, all processes of the MPI job are stopped with the default signal 9 (`SIGKILL`).

I_MPI_HYDRA_BOOTSTRAP

Set the bootstrap server.

Syntax

`I_MPI_HYDRA_BOOTSTRAP=<arg>`

Arguments

`<arg>`	String parameter
`service`	Use hydra service agent
`ssh`	Use secure shell. This is the default value
`fork`	Use fork call

Description

Set this environment variable to specify the bootstrap server.

NOTE:

Set the `I_MPI_HYDRA_BOOTSTRAP` environment variable in the shell environment before executing the `mpiexec` command. Do not use the `-env` option to set the `<arg>` value. This option is used for passing environment variables to the MPI process environment.

I_MPI_HYDRA_BOOTSTRAP_EXEC

Set the executable to be used as a bootstrap server.

Syntax

`I_MPI_HYDRA_BOOTSTRAP_EXEC=<arg>`

Arguments

`<arg>`	String parameter
`<executable>`	The name of the executable

Description

Set this environment variable to specify the executable to be used as a bootstrap server.

I_MPI_HYDRA_PMI_CONNECT

Define the processing method for `PMI` messages.

Syntax

`I_MPI_HYDRA_PMI_CONNECT=<value>`

Arguments

<value>	The algorithm to be used
nocache	Do not cache PMI messages.
cache	Cache PMI messages on the local pmi_proxy management processes to minimize the number of PMI requests. Cached information is automatically propagated to child management processes.
lazy-cache	cache mode with on-demand propagation. This is the default value.

Description

Use this environment variable to select the PMI messages processing method.

I_MPI_PERHOST

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

Syntax

I_MPI_PERHOST=*<value>*

Arguments

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

Description

Set this environment variable to define the default setting for the -perhost option. The -perhost option implied with the respective value if the I_MPI_PERHOST environment variable is defined.

I_MPI_HYDRA_BRANCH_COUNT

Set the hierarchical branch count.

Syntax

I_MPI_HYDRA_BRANCH_COUNT =*<num>*

Arguments

<num>	Number
<n> >= 0	• The default value is -1 if less than 128 nodes are used. This also means that there is no hierarchical structure

	• The default value is `32` if more than 127 nodes are used

Description

Set this environment variable to restrict the number of child management processes launched by the `mpiexec` operation or by each `pmi_proxy` management process.

I_MPI_HYDRA_PMI_AGGREGATE

Turn on/off aggregation of the `PMI` messages.

Syntax

`I_MPI_HYDRA_PMI_AGGREGATE=<arg>`

Arguments

<arg>	Binary indicator
`enable | yes | on | 1`	Enable `PMI` message aggregation. This is the default value
`disable | no | off | 0`	Disable `PMI` message aggregation

Description

Set this environment variable to enable/disable aggregation of PMI messages .

I_MPI_HYDRA_IFACE

Set the network interface.

Syntax

`I_MPI_HYDRA_IFACE=<arg>`

Arguments

<arg>	String parameter
<network interface>	The network interface configured in your system

Description

Set this environment variable to specify the network interface to use. For example, use `-iface ib0`, if the IP emulation of your `InfiniBand*` network is configured on `ib0`.

I_MPI_TMPDIR

(TMPDIR)

Set the temporary directory.

Syntax

`I_MPI_TMPDIR=<arg>`

Arguments

<arg>	String parameter
<path>	Set the temporary directory. The default value is /tmp

Description

Set this environment variable to specify the temporary directory to store the `mpicleanup` input file.

I_MPI_JOB_RESPECT_PROCESS_PLACEMENT

Specify whether to use the job scheduler provided process-per-node parameter.

Syntax

`I_MPI_JOB_RESPECT_PROCESS_PLACEMENT=<arg>`

Arguments

<value>	Binary indicator
enable | yes | on | 1	Use the process placement provided by job scheduler. This is the default value
disable | no | off | 0	Do not use the process placement provided by job scheduler

Description

If you set `I_MPI_JOB_RESPECT_PROCESS_PLACEMENT=enable`, then Hydra process manager uses PPN provided by job scheduler.

If you set `I_MPI_JOB_RESPECT_PROCESS_PLACEMENT = disable`, then Hydra process manager uses PPN provided in command line option or using `I_MPI_PERHOST` environment variable.

2.5. Integration with Microsoft* HPC Job Scheduler

The Intel® MPI Library job startup command `mpiexec` can be called out of Microsoft* HPC Pack 2012 Job Scheduler to execute MPI application. In this case the `mpiexec` command automatically inherits the host list, process count and the working directory allocated to the job.

Use the following command to submit MPI job:

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

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

2.6. Integration with PBS Pro* Job Scheduler

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

Example:

Content of job script:

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

REM PBS -l walltime=1:00:00

cd %PBS_O_WORKDIR%

mpiexec test.exe
```

Use the following command to submit the job:

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

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

2.7. Hetero Operating System Cluster Support

The Intel® MPI Library provides support for heterogeneous Windows-Linux environment. Hydra process manager available on Windows* OS and Linux* OS is used to provide possibility for Intel MPI Library on Linux* OS and Windows* OS to cooperate within one job. For more information about hydra process manager, see Scalable Process Management System (Hydra).

To run Linux-Windows operating system (OS) mixed job, do the following:

- Ensure the Intel MPI Library is installed and operable on each node of your job.
- The `-demux=select` and `I_MPI_FABRICS=shm:tcp` are supported for the operating system mixed run.
- Set the `-bootstrap` option. The default value in operating system mixed run mode is `-bootstrap service`. To enable such configuration, ensure the hydra service is launched on Windows * OS and Hydra persist server on Linux * OS on each node of an MPI job. Provide the ssh connectivity between Linux and Windows machines for the `-bootstrap ssh`.
- Use `-hostos` option to specify an operating system installed on a particular host.
- Use `I_MPI_ROOT` and `LD_LIBRARY_PATH` local environment variables to overwrite incorrect settings for adjacent operating system environment inheritance.

For example, the following command runs `IMB-MPI1` job under Windows-Linux heterogeneous environment:

```
> mpiexec -demux select -genv I_MPI_FABRICS shm:tcp -env I_MPI_ROOT \
<linux_installdir> -env LD_LIBRARY_PATH <linux_installdir>/<arch>/lib -hostos \
```

```
linux -host <lin_host> -n 2 <linux_installdir>/<arch>/bin/IMB-MPI1 : -host \
<win_host> -n 3 <windows_installdir>\\<arch>\\bin\\IMB-MPI1
```

2.8. Processor Information Utility

cpuinfo

The `cpuinfo` utility provides processor architecture information.

Syntax

```
cpuinfo [[-]<options>]]
```

Arguments

`<options>`	Sequence of one-letter options. Each option controls a specific part of the output data
`g`	General information about single cluster node shows: • the processor product name • the number of packages/sockets on the node • core and threads numbers on the node and within each package • SMT mode enabling
`i`	Logical processors identification table identifies threads, cores, and packages of each logical processor accordingly. • *Processor* - logical processor number. • *Thread Id* - unique processor identifier within a core. • *Core Id* - unique core identifier within a package. • *Package Id* - unique package identifier within a node.
`d`	Node decomposition table shows the node contents. Each entry contains the information on packages, cores, and logical processors. • *Package Id* - physical package identifier. • *Cores Id* - list of core identifiers that belong to this package. • *Processors Id* - 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.
`c`	Cache sharing by logical processors shows information of sizes and processors groups, which share particular cache level. • Size - cache size in bytes.

	• Processors - a list of processor groups enclosed in the parentheses those share this cache or no sharing otherwise.
s	Microprocessor signature hexadecimal fields (Intel platform notation) show signature values: • extended family • extended model • family • model • type • stepping
f	Microprocessor feature flags indicate what features the microprocessor supports. The Intel platform notation is used.
A	Equivalent to gidcsf
gidc	Default sequence
?	Utility usage info

Description

The cpuinfo utility prints out the processor architecture information that can be used to define suitable process pinning settings. The output consists of a number of tables. Each table corresponds to one of the single options listed in the arguments table.

NOTE:

The architecture information is available on systems based on the Intel® 64 architecture.

The cpuinfo utility is available for both Intel microprocessors and non-Intel microprocessors, but it may provide only partial information about non-Intel microprocessors.

Examples

cpuinfo output for the processor of Intel® microarchitecture code name Sandy Bridge:

```
$ cpuinfo A
Intel(R) Processor information utility, Version 4.1.0 Build 20120713
Copyright (C) 2005-2012 Intel Corporation.  All rights reserved.

=====  Processor composition  =====
Processor name    : Genuine Intel(R)
Packages(sockets) : 2
Cores             : 16
Processors(CPUs)  : 32
Cores per package : 8
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           0           4           0
5           0           5           0
6           0           6           0
7           0           7           0
8           0           0           1
9           0           1           1
10          0           2           1
11          0           3           1
12          0           4           1
13          0           5           1
14          0           6           1
15          0           7           1
16          1           0           0
17          1           1           0
18          1           2           0
19          1           3           0
20          1           4           0
21          1           5           0
22          1           6           0
23          1           7           0
24          1           0           1
25          1           1           1
26          1           2           1
27          1           3           1
28          1           4           1
29          1           5           1
30          1           6           1
31          1           7           1
=====  Placement on packages  =====
Package Id. Core Id.        Processors
0           0,1,2,3,4,5,6,7 (0,16)(1,17)(2,18)(3,19)(4,20)(5,21)(6,22)(7,23)
1           0,1,2,3,4,5,6,7 (8,24)(9,25)(10,26)(11,27)(12,28)(13,29)(14,30)(15,31)

=====  Cache sharing  =====
Cache   Size    Processors
L1      32  KB  (0,16)(1,17)(2,18)(3,19)(4,20)(5,21)(6,22)(7,23)(8,24)(9,25)(10,26)(11,27)(12,28)(13,29)(14,30)(15,31)
L2      256 KB  (0,16)(1,17)(2,18)(3,19)(4,20)(5,21)(6,22)(7,23)(8,24)(9,25)(10,26)(11,27)(12,28)(13,29)(14,30)(15,31)
L3      20  MB  (0,1,2,3,4,5,6,7,16,17,18,19,20,21,22,23)(8,9,10,11,12,13,14,15,24,25,26,27,28,29,30,31)

=====  Processor Signature  =====
| xFamily | xModel | Type | Family | Model | Stepping |
|---------|--------|------|--------|-------|----------|
| 00      | 2      | 0    | 6      | d     | 5        |

=====  Processor Feature Flags  =====
| SSE3 | PCLMULDQ | DTES64 | MONITOR | DS-CPL | VMX | SMX | EIST | TM2 | SSSE3 | CNXT-ID | FMA | CX16 | xTPR |
|------|----------|--------|---------|--------|-----|-----|------|-----|-------|---------|-----|------|------|
| 1    | 1        | 1      | 1       | 1      | 1   | 1   | 1    | 1   | 1     | 0       | 0   | 1    | 1    |

| PDCM | PCID | DCA | SSE4.1 | SSE4.2 | x2APIC | MOVBE | POPCNT | TSC-DEADLINE | AES | XSAVE | OSXSAVE | AVX | F16C | RDRAND |
|------|------|-----|--------|--------|--------|-------|--------|--------------|-----|-------|---------|-----|------|--------|
| 1    | 1    | 1   | 1      | 1      | 1      | 0     | 1      | 1            | 1   | 1     | 1       | 1   | 0    | 0      |

| FPU | VME | DE | PSE | TSC | MSR | PAE | MCE | CX8 | APIC | SEP | MTRR | PGE | MCA | CMOV | PAT | PSE-36 |
|-----|-----|----|-----|-----|-----|-----|-----|-----|------|-----|------|-----|-----|------|-----|--------|
| 1   | 1   | 1  | 1   | 1   | 1   | 1   | 1   | 1   | 1    | 1   | 1    | 1   | 1   | 1    | 1   | 1      |

| PSN | CLFSH | DS | ACPI | MMX | FXSR | SSE | SSE2 | SS | HTT | TM | PBE |
|-----|-------|----|------|-----|------|-----|------|----|-----|----|-----|
| 0   | 1     | 1  | 1    | 1   | 1    | 1   | 1    | 1  | 1   | 1  | 1   |
```

FSGBASE	BMI1	AVX2	SMEP	BMI2	ERMS	INVPCID
0	0	0	0	0	0	0

3. User Authorization

This topic describes different authentication methods under Windows* OS and how to use these methods to enable the authorization.

3.1. Overview

The Intel® MPI Library supports several authentication methods under Windows* OS:

- The password-based authorization
- The domain-based authorization with the delegation ability
- The limited domain-based authorization

The password-based authorization is the typical method of providing remote computer access using your account name and password.

The domain-based authorization methods use the Security Service Provider Interface (SSPI) provided by Microsoft* in a Windows* environment. The SSPI allows domain to authenticate the user on the remote machine in accordance to the domain policies. You do not need to enter and store your account name and password when using such methods.

NOTE:

Both domain-based authorization methods may increase MPI task launch time in comparison with the password-based authorization. This depends on the domain configuration.

NOTE:

The limited domain-based authorization restricts your access to the network. You will not be able to open files on remote machines or access mapped network drives.

3.2. Installation

This feature is supported on Windows clusters under the following Microsoft operation systems: Windows* HPC Server 2012 and Windows* HPC Pack 2012.

Microsoft's Kerberos Distribution Center* must be enabled on your domain controller. This is the default behavior.

Using the domain-based authorization method with the delegation ability requires specific installation of the domain. You can perform this installation in the following ways:

- Use the IMPI installer if you have domain administrator rights.
- Follow the actions described in the installation topic.

3.2.1. Active Directory* Setup

To enable the delegation in the Active Directory* do the following:

1. Log in on the domain controller under the admin account
2. Enable the delegation for cluster nodes:
    a. Open the **Computers** list in the **Active Directory Users and Computers** administrative utility
    b. Right click on a desired computer object and select **Properties**
    c. If the account is located:
        i. in a Windows 2000 functional level domain, check the **Trust computer for delegation** option
        ii. in a Windows 2003 functional level domain, select the **Delegation** tab and check the **Trust this computer for delegation to any service (Kerberos only)** option
3. Enable the delegation for users:
    a. Open the **Users** list in the **Active Directory Users and Computers** administrative utility
    b. Right click on a desired user object and select **Properties**
    c. Select the **Account** tab and disable the **Account is sensitive and cannot be delegated** option
4. Register Service Principal Name (SPN) for cluster nodes. Use one of the following methods for registering SPN:
    a. Use the Microsoft*-provided `setspn.exe` utility. For example, execute the following command on the domain controller:
    `setspn.exe -A impi_hydra/<host>:<port>/impi_hydra <host>`
    where
    <host> is a cluster node name
    <port> is a Hydra port. The default value is 8679. Change this number only if your hydra service uses the non-default port
    b. Log into each desired node under the admin account and execute the `hydra_service -register_spn` command

NOTE:

In case of any issues with the MPI task start, reboot the machine from which the MPI task is started. Alternatively, execute the `klist purge` command there if the Microsoft*-provided `klist.exe` utility is available.

3.3. Environment Variables

I_MPI_AUTH_METHOD

Select a user authorization method.

Syntax

`I_MPI_AUTH_METHOD=<method>`

Arguments

<method>	Define an authorization method
password	Use the password-based authorization. This is the default value
delegate	Use the domain-based authorization with delegation ability
impersonate	Use the limited domain-based authorization. You will not be able to open files on remote machines or access mapped network drives

Description

Set this environment variable to select a desired authorization method. If this environment variable is not defined, mpiexec uses the password-based authorization method by default.

NOTE:

Alternatively, you can change the default behavior by using the mpiexec -delegate or mpiexec -impersonate options.

4. 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.

4.1. Using mpitune 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 [ -a \"<application command line>\" ] [ -of <file-name> ] \
        [ -t \"<test_cmd_line>\" ] [-cm ] [ -d ] [ -D] \
        [ -dl [d1[,d2...[,dN]]] ] [ -fl [f1[,f2…[,fN]]] ] \
        [ -hf <hostsfile> ] [ -h ] [ -hr {min:max|min:|:max} ] \
        [ -i <count> ] [ -mr {min:max|min:|:max}] [ -od <outputdir> ] \
        [ -odr <outputdir> ] [ -pr {min:max|min:|:max}] \
        [ -sf [file-path] ] [ -ss ] [ -s ] [ -td <dir-path> ] \
        [ -tl <minutes> ] [ -mh ] [ -os <opt1,...,optN> ] \
        [ -oe <opt1,...,optN> ] [ -V ] [ -vi {percent} ; -vix {X factor} ]\
        [ - zb ] [ -t ] [ -so ] [ -ar \"reg-expr\" ] [ -trf <appoutfile> ]\
        [ -m {base|optimized} ] [ -avd {min|max} ] \
        [ -pm {mpd|hydra} ] \
        [ -co ] [ -sd ] [ -soc ]
```

or

```
mpitune [ --application\"<app_cmd  _ line>\" ] [ --output-file <file-name> ]\
        [ --test \"<test_cmd_line>\" ] [ --cluster-mode ] [ --debug ] \
        [ --distinct ] [ --device-list [d1[,d2,… [,dN]]] ] \
        [ --fabric-list [f1[,f2...[,fN]]] ] \
        [ --host-file <hostsfile> ] [ --help ] \
        [ --host-range {min:max|min:|:max} ] [ --iterations<count> ] \
        [ --message-range {min:max|min:|:max} ] \
        [ --output-directory <outputdir> ] \
```

```
[ --output-directory-results <outputdir> ] \

[ --ppn-range {min:max|min:|:max} ;
 --perhost-range {min:max|min:|:max} ] \

[ --session-file [file-path] ] [ --show-session ] [ --silent ] \

[--temp-directory <dir-path> ] [ --time-limit <minutes> ] \

[ --master-host ] [ --options-set <opt1,...,optN> ] \

[ --options-exclude <opt1,...,optN> ] [ --version ] \

[ --valuable-improvement ; --valuable-improvement-x   {X factor}   ]\

[ --zero-based ] [ --trace] [ --scheduler-only ] \

[ --application-regexp \"reg-expr\" ] \

[ --test-regexp-file <appoutfile> ] [ --model {base|optimized} ] \

[ --application-value-direction {min|max} ] \

[ --process-manager {mpd|hydra} ] \

[ -co ] [ -sd ] [ -soc ]
```

Arguments

`-a \"<app_ cmd_line>\"|`
 	
`--application \"<app_cmd_line>\"`
	Switch on the application-specific mode. Quote the full command line as shown, including the backslashes.

`-of <file-name> |`
 	
`--output-file <file-name>`
	Specify the name of the application configuration file to be generated in the application-specific mode. By default, use the file name `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 including the backslashes.

`-cm {exclusive|full} | --cluster-mode {exclusive|full}`	Set the cluster usage mode • `full` - maximum number of tasks are executed. This is the default mode. • `exclusive` - only one task is executed on the cluster at a time.
`-d | --debug`	Print out the debug information.
`-D | --distinct`	Tune all options separately from each other. This argument is applicable only for the cluster-specific mode.
`-dl [d1[,d2...[,dN]]] |`	Select the device(s) you want to tune. Any previously set fabrics are ignored.. By default, use all devices listed in

`--device-list [d1[,d2,…[,dN]]]`	the `<installdir>\<arch>\etc\devices.xml` file.
`-fl [f1[,f2...[,fN]]] |` `--fabric-list [f1[,f2…[,fN]]]`	Select the fabric(s) you want to tune. Any previously set devices are ignored. By default, use all fabrics listed in the `<installdir>\<arch>\etc\fabrics.xml` file.
`-hf <hostsfile> |` `--host-file <hostsfile>`	Specify an alternative host file name. By default, use the `mpd.hosts`.
`-h | --help`	Display the help message.
`-hr {min:max|min:|:max} |` `--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 uses 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`.
`-mr {min:max|min:|:max} |` `--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 uses the default values as appropriate.
`-od <outputdir> |` `--output-directory <outputdir>`	Specify the directory name for all output files: log-files, session-files, local host-files and report-files. By default, use the current directory. This directory should be accessible from all hosts.
`-odr <outputdir> |` `--output-directory-results <outputdir>`	Specify the directory name for the resulting configuration files. By default, use the current directory in the application-specific mode and the `<installdir>\<arch>\etc` in the cluster-specific mode. If `<installdir>\<arch>\etc` is unavailable, the current directory is used as the default value in the cluster-specific mode.

`-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 uses the default values as appropriate.

`-sf [file-path] |`	Continue the tuning process starting from the state saved

`--session-file [file-path]`	in the `file-path` session file.
`-ss |` `--show-session`	Show information about the session file and exit. This option works only jointly with the `-sf` option.
`-s | --silent`	Suppress all diagnostics.
`-td <dir-path> |` `--temp-directory <dir-path>`	Specify a directory name for the temporary data. The Intel MPI Library uses the `mpitunertemp` folder in the current directory by default. 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 the `mpitune`.
`-os <opt1,...,optN>|` `--options-set <opt1,...,optN>`	Use `mpitune` to tune the only required options you have set in the option values
`-oe <opt1,...,optN>|` `--options-exclude <opt1,...,optN>`	Exclude the settings of the indicated Intel® MPI Library options from the tuning process.
`-V | --version`	Print out the version information.

`-vi {percent} |>`
`--valuable-improvement {percent}`
`-vix {X factor} |`
`--valuable-improvement-x {X factor}`
	Control the threshold for performance improvement. The default threshold is `3%`.

`-zb | --zero-based`	Set zero as the base for all options before tuning. This argument is applicable only for the cluster-specific mode.
`-t | --trace`	Print out error information such as error codes and tuner trace back.
`-so | --scheduler-only`	Create the list of tasks to be executed, display the tasks, and terminate execution.

`-ar \"reg-expr\" |`
 	
`--application-regexp \"reg-expr\"`
	Use `reg-expr` to determine the performance expectations of the application. This option is applicable only for the application-specific mode. The `reg-expr` setting should contain only one group of numeric values which is used by `mpitune` for analysis. Use backslash for symbols when setting the value of this argument in accordance with the operating system requirements.

`-trf <appoutfile> |`
 	
`--test-regexp-file <appoutfile>`
	Use a test output file to check the correctness of the regular expression. This argument is applicable only for the cluster-specific mode when you use the `-ar` option.

`-m {base|optimized} |`
 	
`--model {base|optimized}`
	Specify the search model:
	• Set `base` to use the old model.
	• Set `optimized` to use the new faster search model. This is the default value.

`-avd {min|max} |`
 	
`--application-value-direction {min|max}`
	Specify the direction of the value optimization :
	• Set `min` to specify that lower is better. For example, use this value when optimizing the wall time.
	• Set `max` to specify that higher is better. For example, use this value when optimizing the solver ratio.

`-pm {mpd|hydra} |`
 	
`--process-manager {mpd|hydra}`
	Specify the process manager used to run the benchmarks. The default value is `hydra`.

`-co | --collectives-only`	Tune collective operations only.
`-sd | --save-defaults`	Use `mpitune` to save the default values of the Intel® MPI Library options.
`-soc | --skip-options-check`	Specify whether to check the command line options.

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. If configuration files from previous `mpitune` sessions exist, `mpitune` creates a copy of the existing files before starting execution.

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.

4.1.1. Cluster Specific Tuning

To find the optimal settings for tuning your cluster, run the `mpitune` utility once after the Intel® MPI Library installation and after every cluster configuration change (processor or memory upgrade, network reconfiguration, etc.). To get the list of settings, run the utility under the user account that was used for the Intel® MPI Library installation, or appropriately set the tuner data directory through the `--output-directory` option and the results directory through the `--output-directory-results` option.

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

For example:

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

```
> mpitune
```

- Use the optimal recorded values when running on the cluster

```
> mpiexec -tune -n 32 .\myprog
```

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

NOTE:

When you use the `-tune` option in the cluster specific mode (such as, without the tuning configuration file name), you need to explicitly select the communication device or fabric, the number of processes per node, and the total number of processes. For example:

```
> mpirun -tune -genv I_MPI_FABRICS shm:dapl -ppn 8 -n 32 .\myprog
```

4.1.1.1. Replacing the Default Benchmark

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

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

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
   ```

4.1.2. Application Specific Tuning

Run the tuning process for any 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 that is applicable to the configuration (fabric, rank placement, etc.).

NOTE:

In the application specific mode, you can achieve the best tuning results using a similar command line and environment.

For example:

Collect configuration settings for the given application

```
> mpitune --application \"mpiexec -n 32 .\myprog\" -of .\myprog.conf
```

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 are saved in the current working directory.

The resulting application configuration file contains the optimal Intel® MPI Library parameters for this application and configuration only. To tune the Intel® MPI Library for the same application in a different configuration (number of hosts, workload, etc.), rerun the automated tuning utility with the desired configuration.

NOTE:

By default, the automated tuning utility overwrites the existing application configuration files. If you want to keep various application and configuration files, you should use a naming convention to save the different versions and select the correct file when you need it.

4.1.3. 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 any environment variables that have 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 environment variable and preserves the default library heuristics.

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

4.2. Process Pinning

Use this feature to pin a 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.

4.2.1. Processor Identification

The following schemes are used to identify logical processors in a system:

- System-defined logical enumeration
- 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 to this CPU bit in the kernel affinity bit-mask. Use the `cpuinfo` utility, provided with your Intel MPI Library installation, to find out the logical CPU numbers.

The 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 the example for one 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.

4.2.2. 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 environment variable to turn off the process pinning feature of the Intel® MPI Library.

I_MPI_PIN_PROCESSOR_LIST

(I_MPI_PIN_PROCS)

Define a processor subset and the mapping rules for MPI processes within this subset.

Syntax

```
I_MPI_PIN_PROCESSOR_LIST=<value>
```

The environment variable value has the following syntax forms:

1. *<proclist>*

2. `[<procset>][:[grain=<grain>][,shift=<shift>][,preoffset=<preoffset>][,postoffset=<postoffset>]`

3. `[<procset>][:map=<map>]`

The paragraphs below provide detail descriptions for the values of these syntax forms.

Deprecated Syntax

`I_MPI_PIN_PROCS=<proclist>`

NOTE:

The `postoffset` keyword has `offset` alias.

NOTE:

The second form of the 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.

NOTE:

The `grain`, `shift`, `preoffset`, and `postoffset` parameters have a unified definition style.

This environment variable is available for both Intel® and non-Intel microprocessors, but it may perform additional optimizations for Intel microprocessors than it performs for non-Intel microprocessors.

Syntax

`I_MPI_PIN_PROCESSOR_LIST=<proclist>`

Arguments

<proclist>	A comma-separated list of logical processor numbers and/or ranges of processors. The process with the i-th rank is pinned to 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>*.

Syntax

```
I_MPI_PIN_PROCESSOR_LIST=[<procset>][:[grain=<grain>][,shift=<shift>]\[,preoffs
et=<preoffset>][,postoffset=<postoffset>]
```

Arguments

<procset>	Specify a processor subset based on the topological numeration. The default value is `allcores`.
`all`	All logical processors. This subset is defined to be the number of CPUs on a node.
`allcores`	All cores (physical CPUs). This subset is defined to be the number of cores on a node. This is the default value. If Intel® Hyper-Threading Technology is disabled, `allcores` equals to `all`.
`allsocks`	All packages/sockets. This subset is defined to be the number of sockets on a node.

<grain>	Specify the pinning granularity cell for a defined *<procset>*. The minimal *<grain>* is a single element of the *<procset>*. The maximal grain is the number of *<procset>* elements in a socket. The *<grain>* value must be a multiple of the *<procset>* value. Otherwise, minimal grain is assumed. The default value is the minimal *<grain>*.
<shift>	Specify the granularity of the round robin scheduling shift of the cells for the *<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, which is equal to 1 normal increment.
<preoffset>	Specify the cyclic shift of the processor subset *<procset>* defined before the round robin shifting on the *<preoffset>* value. 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 the cyclic shift of the processor subset *<procset>* 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.

The following table displays the values for *<grain>*, *<shift>*, *<preoffset*>, and *<postoffset>* options:

<n>	Specify an explicit value of the corresponding parameters. *<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 an L1 cache.
cache2	Specify the parameter value equal to the amount of the corresponding parameter units that share an L2 cache.
cache3	Specify the parameter value equal to the amount of the corresponding parameter units that share an L3 cache.
cache	The largest value among cache1, cache2, and cache3.
socket | sock	Specify the parameter value equal to the amount of the corresponding parameter units contained in one physical package/socket.
half | mid	Specify the parameter value equal to socket/2.
third	Specify the parameter value equal to socket/3.
quarter	Specify the parameter value equal to socket/4.
octavo	Specify the parameter value equal to socket/8.

Syntax

I_MPI_PIN_PROCESSOR_LIST=[*<procset>*][:map=*<map>*]

Arguments

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

Description

Set the I_MPI_PIN_PROCESSOR_LIST environment variable to define the processor placement. To avoid conflicts with differing shell versions, the environment variable value may need to be enclosed in quotes.

NOTE:

This environment variable is valid only if `I_MPI_PIN` is enabled.

The `I_MPI_PIN_PROCESSOR_LIST` environment variable has the following different syntax variants:

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

For example, process mapping for `I_MPI_PIN_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

- `grain/shift/offset` mapping. This method provides cyclic shift of a defined `grain` along the processor list with steps 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

Predefined mapping scenario. In this case popular process pinning schemes are defined as keywords selectable at runtime. There are two such scenarios: `bunch` and `scatter`.

In this case popular process pinning schemes are defined as keywords that are selectable at runtime. 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 so as not to share common resources: FSB, caches, cores.

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

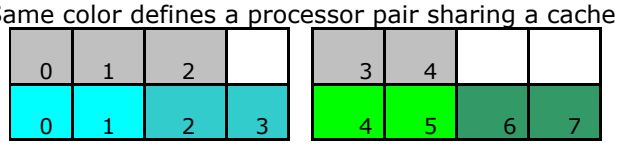
Legend:

gray - MPI processes

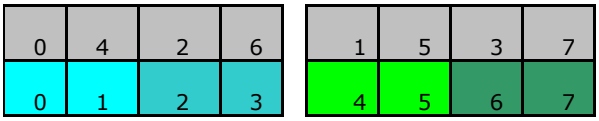
cyan - 1st socket processors

green - 2nd socket processors

Same color defines a processor pair sharing a cache



`bunch` scenario for 5 processes



`scatter` scenario for full loading

Examples

1. To pin the processes to CPU0 and CPU3 on each node globally, use the following command:

```
> mpiexec.exe -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:

```
> mpiexec.exe -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 the process pinning, use the following command:

```
> mpiexec.exe -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 the number of processes is greater than the number of CPUs used for pinning, the process list is wrapped around to the start of the processor list.

I_MPI_PIN_CELL

Set this environment variable to define the pinning resolution granularity. `I_MPI_PIN_CELL` specifies the minimal processor cell allocated when an MPI process is running.

Syntax

`I_MPI_PIN_CELL=<cell>`

Arguments

<cell>	Specify the resolution granularity
`unit`	Basic processor unit (logical CPU)
`core`	Physical processor core

Description

Set this environment variable to define the processor subset used when a process is running. You can choose from two scenarios:

- all possible CPUs in a system (`unit` value)
- all cores in a system (`core` value)

The environment variable has effect on both pinning kinds:

- one-to-one pinning through the `I_MPI_PIN_PROCESSOR_LIST` environment variable
- one-to-many pinning through the `I_MPI_PIN_DOMAIN` environment variable

The default value rules are:

- If you use `I_MPI_PIN_DOMAIN`, then the cell granularity is `unit`.
- If you use `I_MPI_PIN_PROCESSOR_LIST`, then the following rules apply:
- When the number of processes is greater than the number of cores, the cell granularity is `unit`.

- When the number of processes is equal to or less than the number of cores, the cell granularity is `core`.

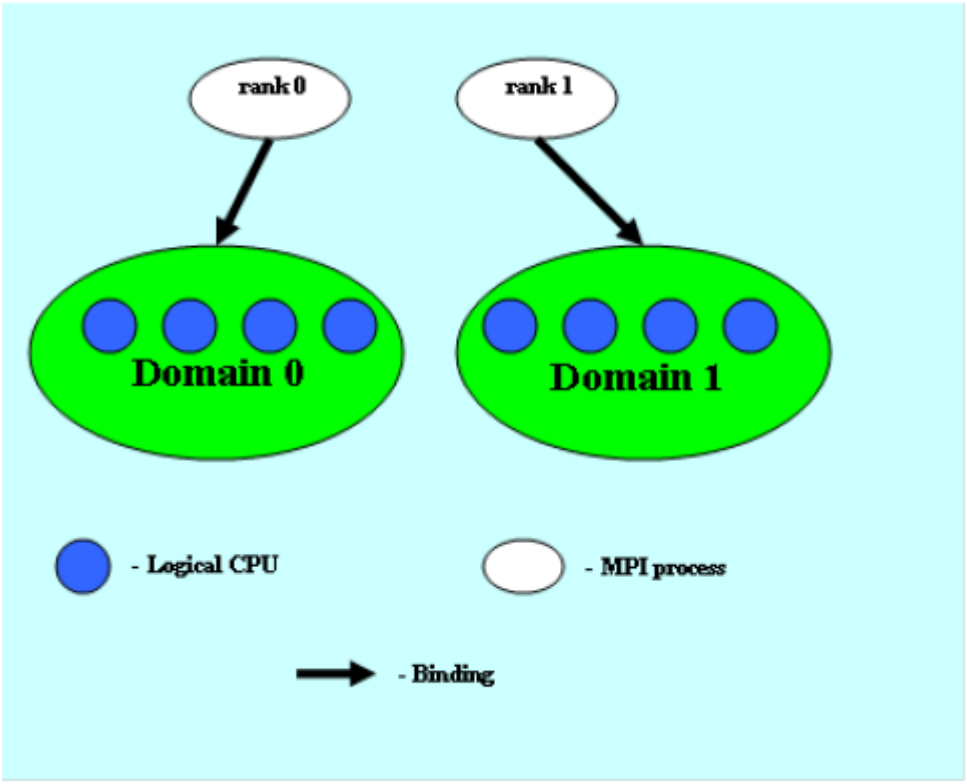
NOTE:

The `core` value is not affected by the enabling/disabling of Hyper-threading technology in a system.

4.2.3. Interoperability with OpenMP* API

I_MPI_PIN_DOMAIN

The Intel® MPI Library provides an additional environment variable to control process pinning for hybrid MPI/OpenMP* applications. This environment 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.



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.

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

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

The I_MPI_PIN_DOMAIN environment variable has the following syntax forms:

- Domain description through multi-core terms *<mc-shape>*
- Domain description through domain size and domain member layout *<size>[:<layout>]*
- Explicit domain description through bit mask *<masklist>*

The following tables describe these syntax forms.

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 the 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 the node. This is the recommended value.
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 single domain.
cache3	Logical processors that share a particular level 3 cache are arranged into a single 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 a positive decimal number *<n>*

<layout>	Ordering of domain members. The default value is compact
platform	Domain members are ordered according to their 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]	For <masklist>, each m_i is a hexadecimail bit mask defining an individual domain. The following rule is used: the i^{th} logical processor is included into the domain if the corresponding mi value is set to 1. All remaining processors are put into a separate domain. BIOS numbering is used. ***NOTE:*** To ensure that your configuration in <masklist> is parsed correctly, use square brackets to enclose the domains specified by the <masklist>. For example: I_MPI_PIN_DOMAIN=[0x55,0xaa]

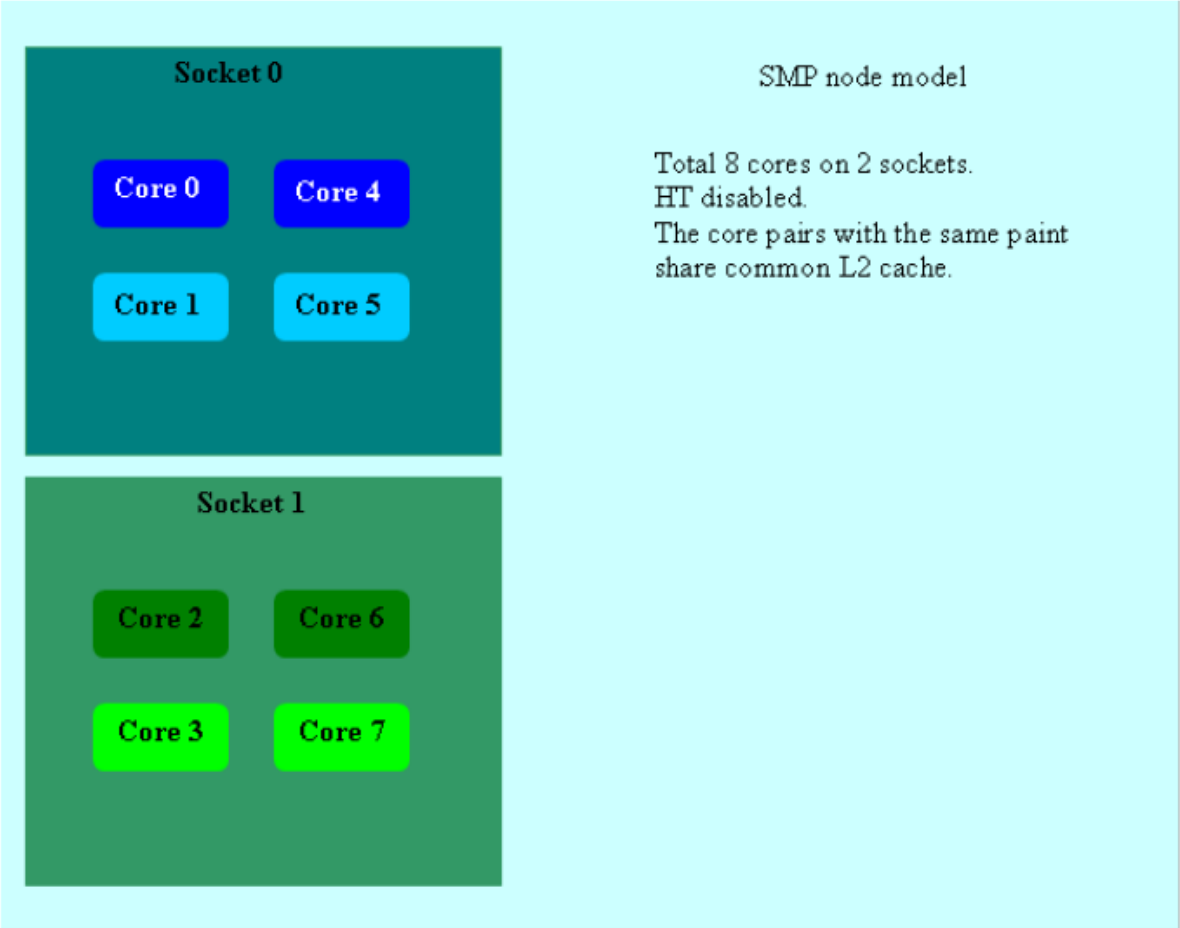
NOTE:

These options are available for both Intel® and non-Intel microprocessors, but they may perform additional optimizations for Intel microprocessors than they perform for non-Intel microprocessors.

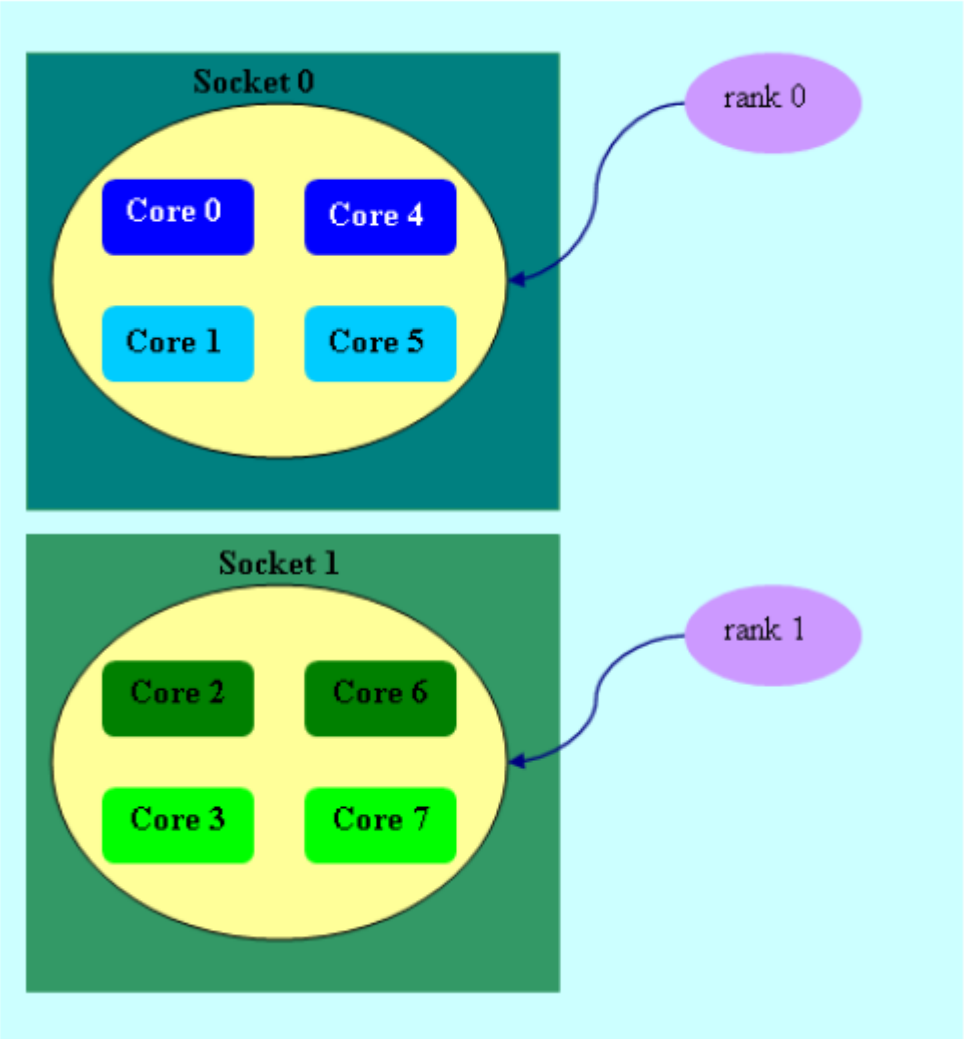
NOTE:

To pin OpenMP* processes/threads inside the domain, the corresponding OpenMP feature (for example, the KMP_AFFINITY environment variable for Intel® Composer XE) should be used.

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

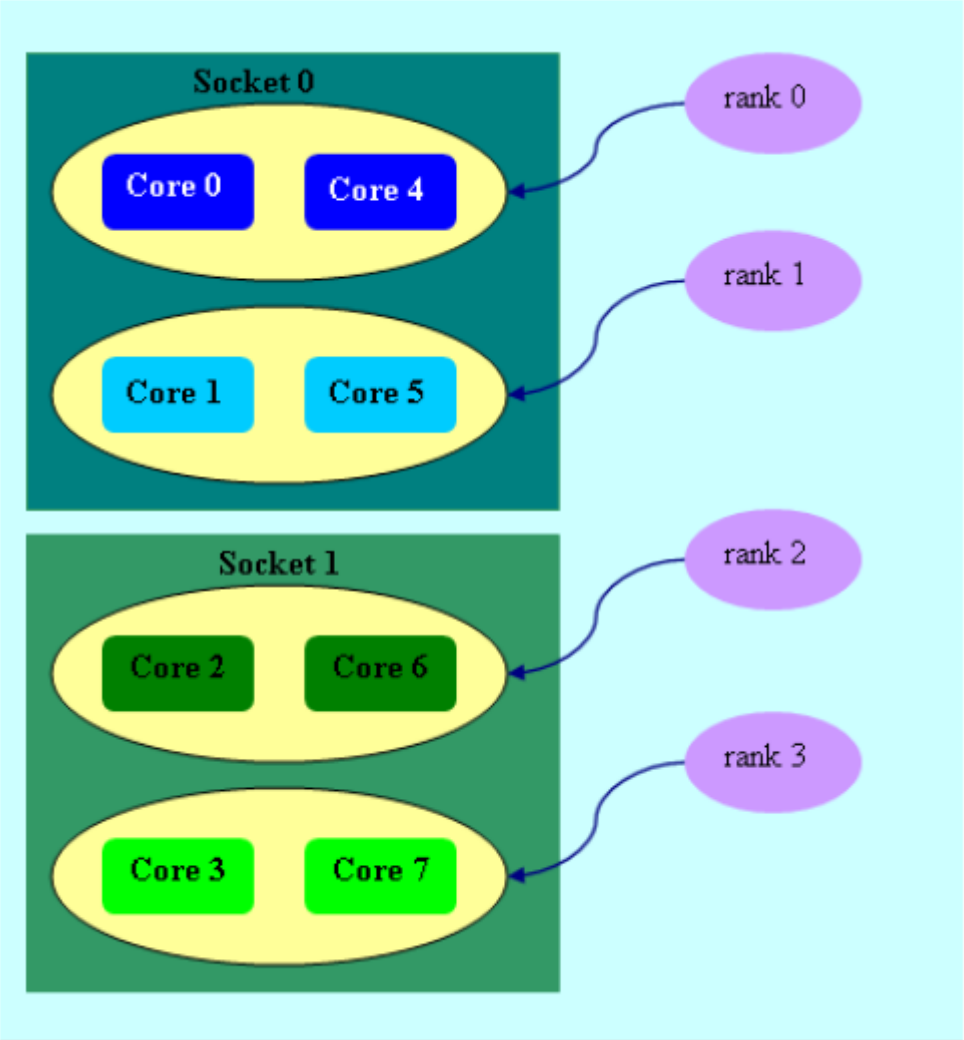


Picture 3.2-2 Model of a Node



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

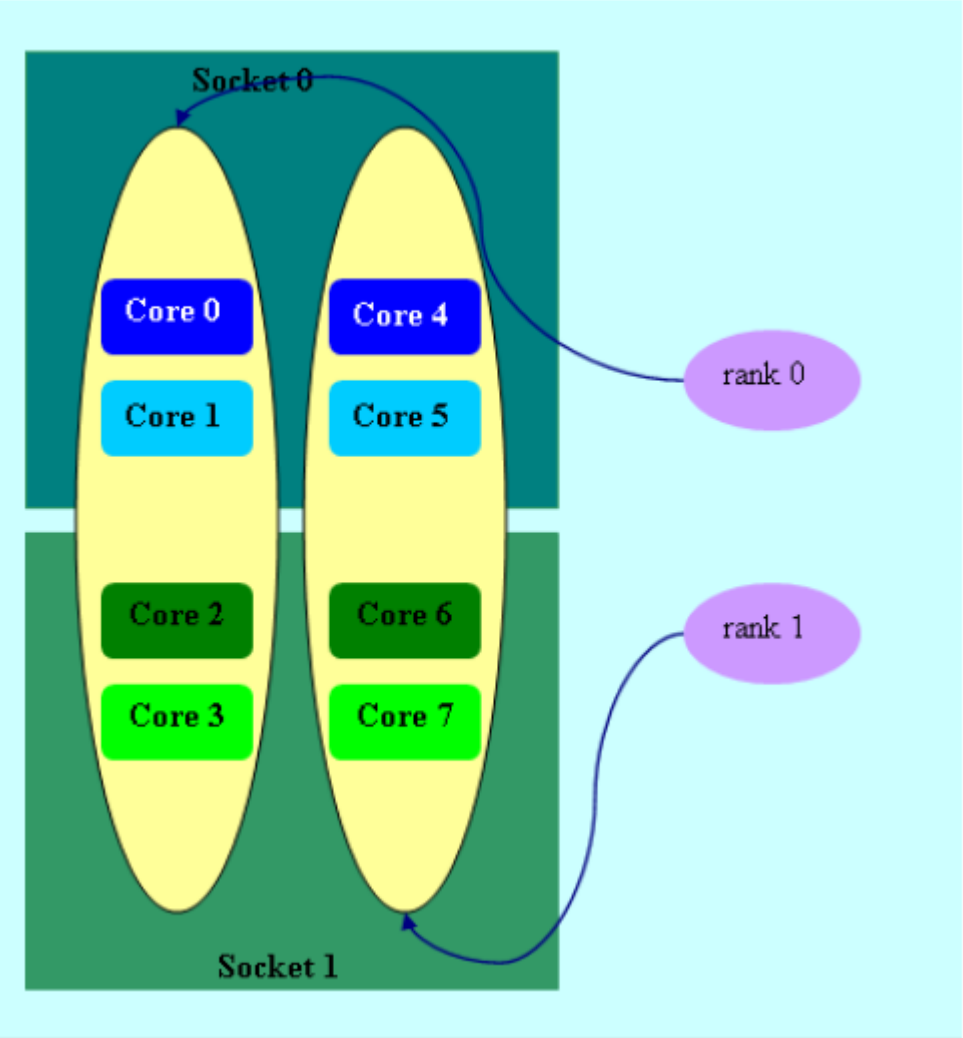
In Picture 3.2-3, two domains are defined according to the number of sockets. Process rank 0 can migrate on all cores on the 0-th socket. Process rank 1 can migrate on all cores on the first socket.



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

In Picture 3.2-4, four domains are defined according to the amount of common L2 caches. Process rank 0 runs on cores {0,4} that share an L2 cache. Process rank 1 runs on cores {1,5} that share

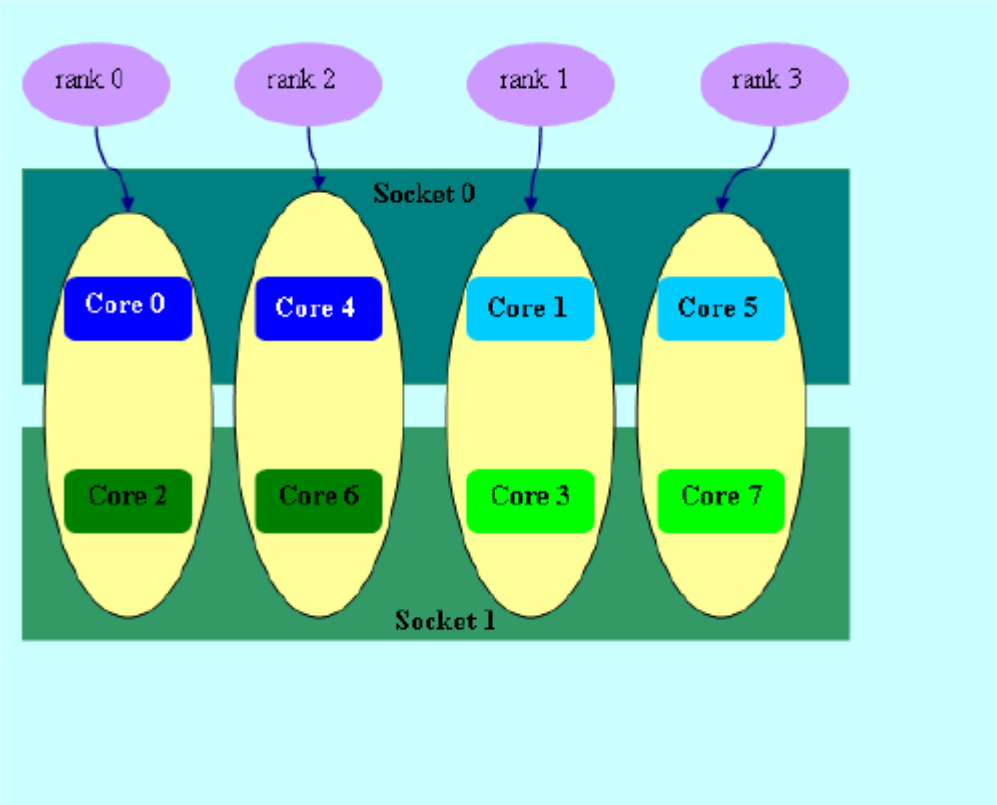
an L2 cache as well, and so on.



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

In Picture 3.2-5, two domains with size=4 are defined. The first domain contains cores {0,1,2,3}, and the second domain contains cores {4,5,6,7}. Domain members (cores) have consecutive

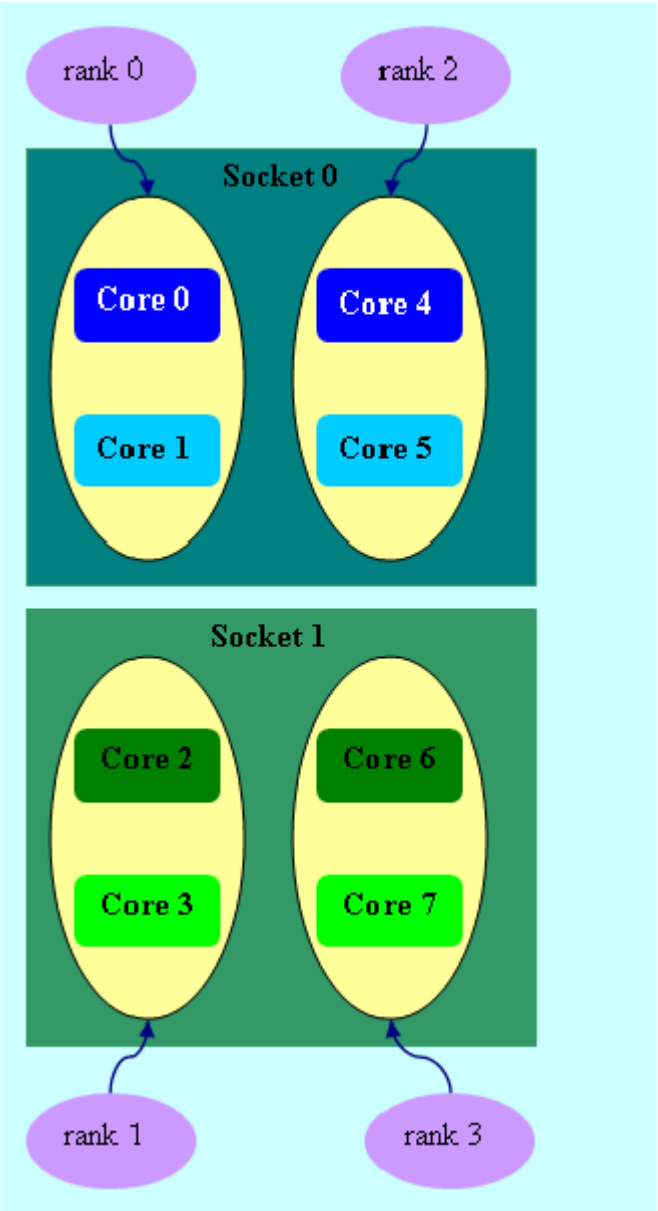
numbering as defined by the `platform` option.



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

In Picture 3.2-6, domain size=2 (defined by the number of CPUs=8 / number of processes=4), `scatter` layout. Four domains {0,2}, {1,3}, {4,6}, {5,7} are defined. Domain members do not

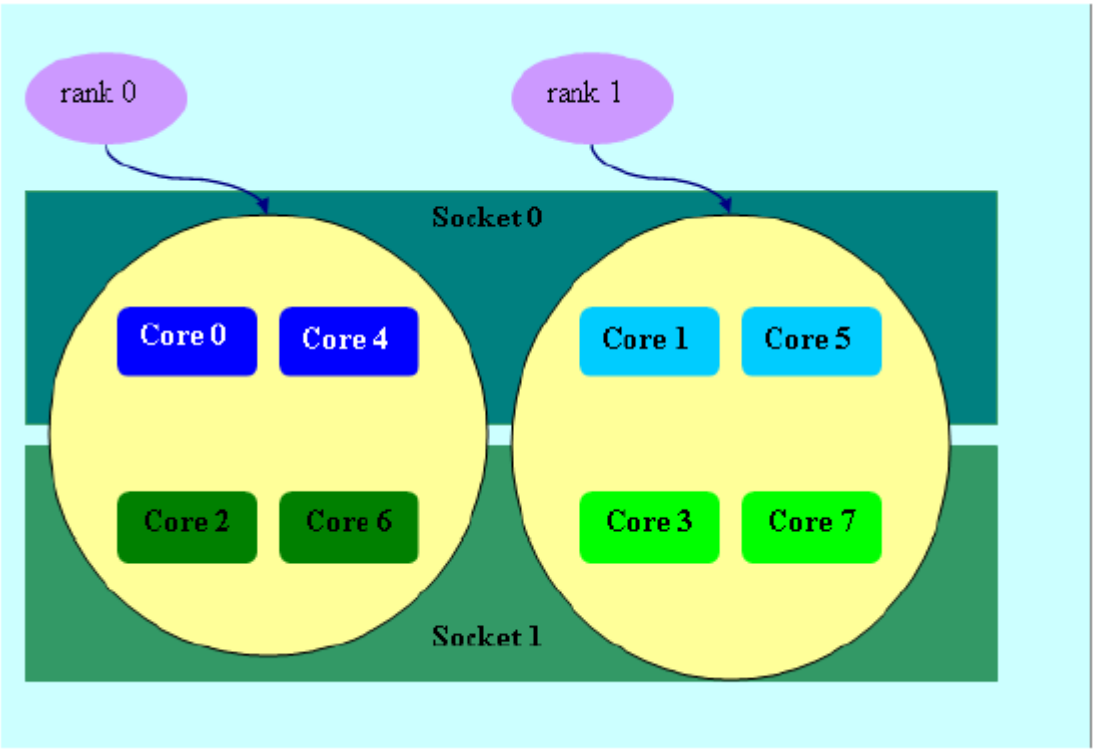
share any common resources.



Picture 3.2-7 `setenv OMP_NUM_THREADS=2`

```
mpiexec -n 4 -env I_MPI_PIN_DOMAIN omp:platform ./a.out
```

In Picture 3.2-7, 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 [0x55,0xaa] ./a.out`

In Picture 3.2-8 (the example for `I_MPI_PIN_DOMAIN=<masklist>`), the first 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}.

I_MPI_PIN_ORDER

Set this environment variable to define the mapping order for MPI processes to domains as specified by the `I_MPI_PIN_DOMAIN` environment variable.

Syntax

`I_MPI_PIN_ORDER=<order>`

Arguments

<order>	Specify the ranking order
`range`	The domains are ordered according to the processor's BIOS numbering. This is a platform-dependent numbering
`scatter`	The domains are ordered so that adjacent domains have minimal sharing of common resources
`compact`	The domains are ordered so that adjacent domains share common resources as much as possible. This is the default value

Description

The optimal setting for this environment variable is application-specific. If adjacent MPI processes prefer to share common resources, such as cores, caches, sockets, FSB, use the `compact` value. Otherwise, use the `scatter` value. Use the `range` value as needed.

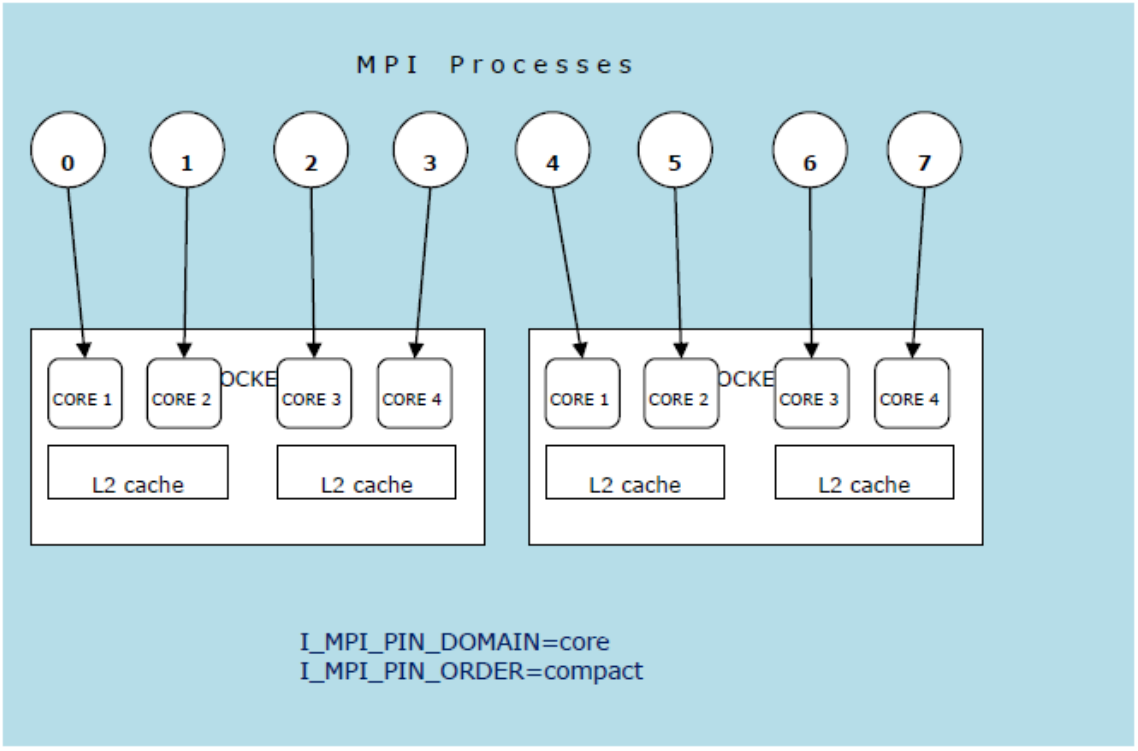
The options `scatter` and `compact` are available for both Intel® and non-Intel microprocessors, but they may perform additional optimizations for Intel microprocessors than they perform for non-Intel microprocessors.

Example

For the following configuration:

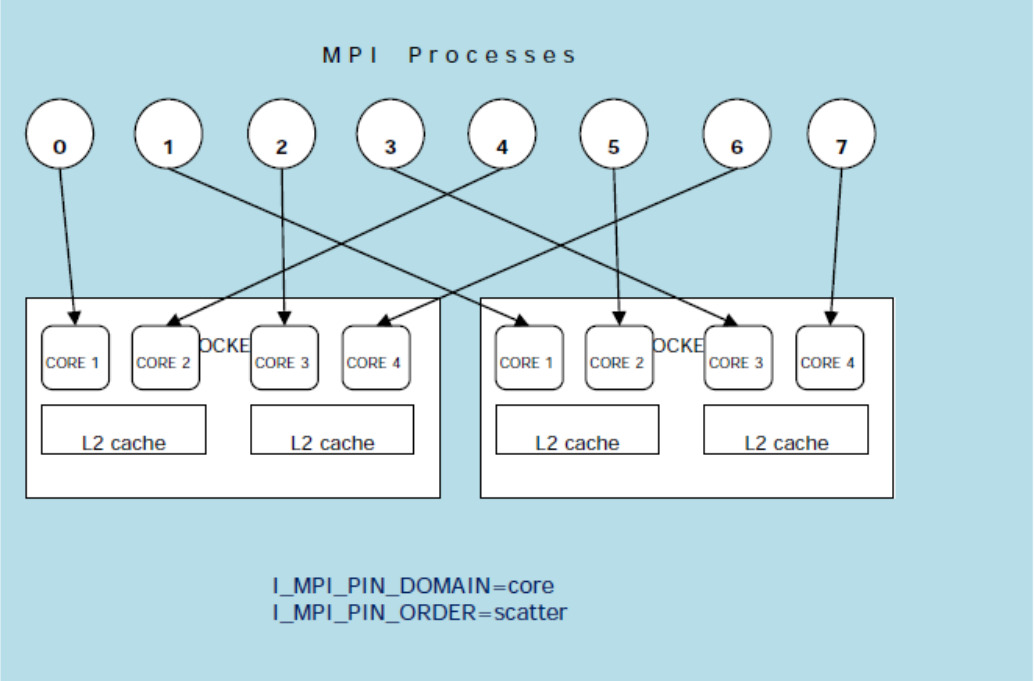
- Two socket nodes with four cores and a shared L2 cache for corresponding core pairs.
- 8 MPI processes you want to run on the node using the following settings:

  - For compact order:
    ```
    I_MPI_PIN_DOMAIN=core
    I_MPI_PIN_ORDER=compact
    ```



Picture 3.2-9 Compact Order Example

  - For scatter order:
    ```
    I_MPI_PIN_DOMAIN=core
    I_MPI_PIN_ORDER=scatter
    ```



Picture 3.2-10 Scatter Order Example

4.3. Fabrics Control

This topic provides you with the information on how to use environment variables to control the following fabrics:

- Communication fabrics
- Shared memory fabrics
- DAPL-capable network fabrics
- TCP-capable network fabrics

4.3.1. Communication Fabrics Control

I_MPI_FABRICS

Select the particular network fabrics to be used.

Syntax

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

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

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

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

Arguments

<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*)

Description

Set this environment 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 environment 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-node communication.
- For other cases, the library uses shm for intra-node communication, and the first available fabric from the fabrics list for inter-node communication. See *I_MPI_FABRICS_LIST* for details.

The shm fabric is available for both Intel® and non-Intel microprocessors, but it may perform additional optimizations for Intel microprocessors than it performs for non-Intel microprocessors.

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> -machinefile smpd.hosts <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
```

I_MPI_FABRICS_LIST

Define a fabrics list.

Syntax

```
I_MPI_FABRICS_LIST=<fabrics list>
```

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

<fabric> := {dapl, tcp}

Arguments

<fabrics list>	Specify a list of fabrics. The default value is `dapl, tcp`

Description

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

For example, if `I_MPI_FABRICS_LIST=dapl, tcp`, and `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 more information on fallback, see *I_MPI_FALLBACK*.

I_MPI_FALLBACK

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

Syntax

```
I_MPI_FALLBACK=<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` environment variable is not set.
`disable | no| off |0`	Terminate the job if MPI cannot initialize the one of the fabrics selected by the `I_MPI_FABRICS` environment variable. This is the default value if the `I_MPI_FABRICS` environment variable is set.

Description

Set this environment 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=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 environment 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*>	Set the eager/rendezvous message size threshold for 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 environment 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_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. The maximum value is equal to 2147483647

Description

Set the spin count limit. The loop for polling the fabric(s) spins <scount> times before the library releases the processes if no incoming messages are received for processing. Within every spin loop, the shm fabric (if enabled) is polled an extra I_MPI_SHM_SPIN_COUNT times. 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

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

Syntax

I_MPI_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 environment 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.

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 environment 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

Turn on/off the dynamic connection establishment.

Syntax

`I_MPI_DYNAMIC_CONNECTION=<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 environment 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 the total number of processes is less than 64.

4.3.2. Shared Memory Control

I_MPI_SHM_CACHE_BYPASS

Control the message transfer algorithm for the shared memory.

Syntax

`I_MPI_SHM_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 environment 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. This feature is enabled by default.

I_MPI_SHM_CACHE_BYPASS_THRESHOLDS

Set the message copying algorithm threshold.

Syntax

`I_MPI_SHM_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

<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
<nb_send_pk>	Set the threshold for sent messages when processes are pinned on cores located in the same physical processor package
<nb_recv_pk>	Set the threshold for received messages when processes are pinned on cores located in the same physical processor package

Description

Set this environment variable to control the thresholds for the message copying algorithm. Intel® MPI Library uses different message copying implementations which are optimized to operate with different memory hierarchy levels. Intel® MPI Library copies messages greater than or equal in size to the defined threshold value using copying algorithm optimized for far memory access.The value of `-1` disables using of those algorithms. The default values depend on architecture and may vary among the Intel® MPI Library versions. This environment variable is valid only when `I_MPI_SHM_CACHE_BYPASS` is enabled.

This environment variable is available for both Intel and non-Intel microprocessors, but it may perform additional optimizations for Intel microprocessors than it performs for non-Intel microprocessors.

I_MPI_SHM_FBOX

Control the usage of the shared memory fast-boxes.

Syntax

`I_MPI_SHM_FBOX=`*`<arg>`*

Arguments

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

Description

Set this environment variable to control the usage of fast-boxes. Each pair of MPI processes on the same computing node has two shared memory fast-boxes, for sending and receiving eager messages.

Turn off the usage of fast-boxes to avoid the overhead of message synchronization when the application uses mass transfer of short non-blocking messages.

I_MPI_SHM_FBOX_SIZE

Set the size of the shared memory fast-boxes.

Syntax

`I_MPI_SHM_FBOX_SIZE=<nbytes>`

Arguments

<nbytes>	Size of shared memory fast-boxes in bytes
> 0	The default *<nbytes>* value is equal to 65472 bytes

Description

Set this environment variable to define the size of shared memory fast-boxes. The value must be multiple of 64.

I_MPI_SHM_CELL_NUM

Change the number of cells in the shared memory receiving queue.

Syntax

`I_MPI_SHM_CELL_NUM=<num>`

Arguments

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

Description

Set this environment variable to define the number of cells in the shared memory receive queue. Each MPI process has own shared memory receive queue, where other processes put eager messages. The queue is used when shared memory fast-boxes are blocked by another MPI request.

I_MPI_SHM_CELL_SIZE

Change the size of a shared memory cell.

Syntax

`I_MPI_SHM_CELL_SIZE=<nbytes>`

Arguments

<nbytes>	Size of a shared memory cell, in bytes
> 0	The default *<nbytes>* value is equal to 65472 bytes

Description

Set this environment variable to define the size of shared memory cells. The value must be a multiple of 64.

If a value is set, I_MPI_INTRANODE_EAGER_THRESHOLD is also changed and becomes equal to the given value.

I_MPI_SHM_LMT

Control the usage of large message transfer (LMT) mechanism for the shared memory.

Syntax

I_MPI_SHM_LMT=<*arg*>

Arguments

<*arg*>	Binary indicator
direct	Turn on the direct copy LMT mechanism. This is the default value
disable | no | off | 0	Turn off LMT mechanism

Description

Set this environment variable to control the usage of the large message transfer (LMT) mechanism. To transfer rendezvous messages, you can use the LMT mechanism by employing either of the following implementations:

- Use intermediate shared memory queues to send messages.
- Use direct copy mechanism that transfers messages without intermediate buffer.

I_MPI_SHM_LMT_BUFFER_NUM

Change the number of shared memory buffers for the large message transfer (LMT) mechanism.

Syntax

I_MPI_SHM_LMT_BUFFER_NUM=<*num*>

Arguments

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

Description

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

I_MPI_SHM_LMT_BUFFER_SIZE

Change the size of shared memory buffers for the LMT mechanism.

Syntax

I_MPI_SHM_LMT_BUFFER_SIZE=<*nbytes*>

Arguments

<nbytes>	The size of shared memory buffers in bytes
> 0	The default *<nbytes>* value is equal to 32768 bytes

Description

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

I_MPI_SSHM

Control the usage of the scalable shared memory mechanism.

Syntax

I_MPI_SSHM =*<arg>*

Arguments

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

Description

Set this environment variable to control the usage of an alternative shared memory mechanism. This mechanism replaces the shared memory fast-boxes, receive queues and LMT mechanism.

If a value is set, the I_MPI_INTRANODE_EAGER_THRESHOLD environment variable is changed and becomes equal to 262,144 bytes.

I_MPI_SSHM_BUFFER_NUM

Change the number of shared memory buffers for the alternative shared memory mechanism.

Syntax

I_MPI_SSHM_BUFFER_NUM=*<num>*

Arguments

<num>	The number of shared memory buffers for each process pair
> 0	The default value is 4

Description

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

I_MPI_SSHM_BUFFER_SIZE

Change the size of shared memory buffers for the alternative shared memory mechanism.

Syntax

`I_MPI_SSHM_BUFFER_SIZE=<nbytes>`

Arguments

<nbytes>	The size of shared memory buffers in bytes
> 0	The default *<nbytes>* value is 65472 bytes

Description

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

I_MPI_SSHM_DYNAMIC_CONNECTION

Control the dynamic connection establishment for the alternative shared memory mechanism.

Syntax

`I_MPI_SSHM_DYNAMIC_CONNECTION=<arg>`

Arguments

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

Description

Set this environment variable to control the 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.

I_MPI_SHM_BYPASS

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

Syntax

`I_MPI_SHM_BYPASS=<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 environment 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` environment variable are transferred using shared memory.
- Messages larger than the threshold value of the `I_MPI_INTRANODE_EAGER_THRESHOLD` environment variable are transferred through the network fabric layer.

NOTE:

This environment variable is applicable only when shared memory and a 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.

I_MPI_SHM_SPIN_COUNT

Control the spin count value for the shared memory fabric.

Syntax

`I_MPI_SHM_SPIN_COUNT=<shm_scount>`

Arguments

`<scount>`	Define the spin count of the loop when polling the `shm` fabric
`> 0`	The default `<shm_scount>` value is equal to `100` spins

Description

Set the spin count limit of the shared memory fabric to increase the frequency of polling. This configuration allows polling of the `shm` fabric `<shm_scount>` times before the control is passed to the overall network fabric polling mechanism.

To tune application performance, use the `I_MPI_SHM_SPIN_COUNT` environment variable. The best value for `<shm_scount>` can be chosen on an experimental basis. It depends largely on the application and the particular computation environment. An increase in the `<shm_scount>` value will benefit multi-core platforms when the application uses topological algorithms for message passing.

4.3.3. 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

This environment variable is applicable only when shared memory and a network fabric are turned on either by default or by setting the `I_MPI_FABRICS` environment variable to `shm:<fabric>` or an equivalent `I_MPI_DEVICE` setting. This mode is available only for `dapl` and `tcp` fabrics.

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 `dat.dll` for DAPL* 1.2 providers and `dat2.dll` for DAPL* 2.0 providers

Description

Set this environment 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 environment variable affects only DAPL capable fabrics.

I_MPI_DAPL_TRANSLATION_CACHE

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

Syntax

`I_MPI_DAPL_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 environment 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 situations. See product *Release Notes* for further details.

I_MPI_DAPL_TRANSLATION_CACHE_AVL_TREE

Enable/disable the AVL tree* based implementation of the RDMA translation cache in the DAPL path.

Syntax

`I_MPI_DAPL_TRANSLATION_CACHE_AVL_TREE=<arg>`

Arguments

<arg>	Binary indicator
`enable | yes | on | 1`	Turn on the AVL tree based RDMA translation cache
`disable | no | off | 0`	Turn off the AVL tree based RDMA translation cache. This is the default value

Description

Set this environment variable to enable the AVL tree based implementation of RDMA translation cache in the DAPL path. When the search in RDMA translation cache handles over 10,000 elements, the AVL tree based RDMA translation cache is faster than the default implementation.

I_MPI_DAPL_DIRECT_COPY_THRESHOLD

Change the threshold of the DAPL direct-copy protocol.

Syntax

`I_MPI_DAPL_DIRECT_COPY_THRESHOLD=<nbytes>`

Arguments

<nbytes>	Define the DAPL direct-copy protocol threshold
`> 0`	The default *<nbytes>* value depends on the platform

Description

Set this environment 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. 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. This approach is faster for large messages.

This environment variable is available for both Intel® and non-Intel microprocessors, but it may perform additional optimizations for Intel microprocessors than it performs for non-Intel microprocessors.

I_MPI_DAPL_EAGER_MESSAGE_AGGREGATION

Control the use of concatenation for adjourned MPI send requests. Adjourned MPI send requests are those that cannot be sent immediately.

Syntax

`I_MPI_DAPL_EAGER_MESSAGE_AGGREGATION =<arg>`

Arguments

`<arg>`	Binary indicator
`enable | yes | on | 1`	Enable the concatenation for adjourned MPI send requests
`disable | no | off | 0`	Disable the concatenation for adjourned MPI send requests. This is the default value

Set this environment variable to control the use of concatenation for adjourned MPI send requests intended for the same MPI rank. In some cases, this mode can improve the performance of applications, especially when `MPI_Isend()` is used with short message sizes and the same destination rank, such as:

```
for( i = 0; i< NMSG; i++)
    {ret = MPI_Isend( sbuf[i], MSG_SIZE,  datatype,  dest , tag, \
comm, &req_send[i]);
    }
```

I_MPI_DAPL_DYNAMIC_CONNECTION_MODE

Choose the algorithm for establishing the DAPL* connections.

Syntax

`I_MPI_DAPL_DYNAMIC_CONNECTION_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 environment 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

Turn on/off scalable algorithm for DAPL read progress.

Syntax

`I_MPI_DAPL_SCALABLE_PROGRESS=<arg>`

Arguments

<arg>	Binary indicator
`enable | yes | on | 1`	Turn on scalable algorithm. When the number of processes is larger than `128`, this is the default value
`disable | no | off | 0`	Turn off scalable algorithm. When the number of processes is less than or equal to `128`, this is the default value

Description

Set this environment 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

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

Syntax

`I_MPI_DAPL_BUFFER_NUM=<nbuf>`

Arguments

<nbuf>	Define the number of buffers for each pair in a process group
`> 0`	The default value depends on the platform

Description

Set this environment 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

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

Syntax

`I_MPI_DAPL_BUFFER_SIZE=<nbytes>`

Arguments

<nbytes>	Define the size of pre-registered buffers
> 0	The default value depends on the platform

Description

Set this environment 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

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

Syntax

`I_MPI_DAPL_RNDV_BUFFER_ALIGNMENT=<arg>`

Arguments

<arg>	Define the alignment for the sending buffer
> 0 and a power of 2	The default value is 64

Set this environment 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

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

Syntax

`I_MPI_DAPL_RDMA_RNDV_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 environment 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

Check the value of the DAPL attribute, `max_rdma_size`.

Syntax

`I_MPI_DAPL_CHECK_MAX_RDMA_SIZE=<arg>`

Arguments

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

Description

Set this environment 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

Control message fragmentation threshold.

Syntax

`I_MPI_DAPL_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` environment 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 environment variable to control message fragmentation size according to the following scheme:

- If the `I_MPI_DAPL_CHECK_MAX_RDMA_SIZE` environment 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 environment 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

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

Syntax

I_MPI_DAPL_CONN_EVD_SIZE=<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 environment variable to define the event queue size of the DAPL event dispatcher that handles connection related events. If this environment 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 equal or larger than the calculated value.

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

I_MPI_DAPL_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>`

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 environment 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.

I_MPI_DAPL_DESIRED_STATIC_CONNECTIONS_NUM

Define the number of processes that establish DAPL static connections at the same time.

Syntax

`I_MPI_DAPL_DESIRED_STATIC_CONNECTIONS_NUM=<num_procesess>`

Arguments

<num_procesess>	Define the number of processes that establish DAPL static connections at the same time
> 0	The default *<num_procesess>* value is equal to 256

Description

Set this environment variable to control the algorithm of DAPL static connection establishment.

If the number of processes in the MPI job is less than or equal to *<num_procesess>*, all MPI processes establish the static connections simultaneously. Otherwise, the processes are distributed into several groups. The number of processes in each group is calculated to be close to *<num_procesess>*. Then static connections are established in several iterations, including intergroup connection setup.

I_MPI_CHECK_DAPL_PROVIDER_COMPATIBILITY

Enable/disable the check that the same DAPL provider is selected by all ranks.

Syntax

`I_MPI_CHECK_DAPL_PROVIDER_COMPATIBILITY=<arg>`

Arguments

<arg>	Binary indicator
`enable | yes | on | 1`	Turn on the check that the DAPL provider is the same on all ranks. This is default value
`disable | no | off | 0`	Turn off the check that the DAPL provider is the same on all ranks

Description

Set this variable to make a check if the DAPL provider is selected by all MPI ranks. If this check is enabled, Intel® MPI Library checks the name of DAPL provider and the version of DAPL. If these parameters are not the same on all ranks, Intel MPI Library does not select the RDMA path and may fall to sockets. Turning off the check reduces the execution time of `MPI_Init()`. It may be significant for MPI jobs with a large number of processes.

4.3.4. TCP-capable Network Fabrics Control

I_MPI_TCP_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`	Use IPoIB* network interface
`eth`	Use Ethernet network interface. 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. Trailing zero bits imply a 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 environment 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 is 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`

I_MPI_TCP_BUFFER_SIZE

Change the size of the TCP socket buffers.

Syntax

`I_MPI_TCP_BUFFER_SIZE=<nbytes>`

Arguments

<nbytes>	Define the size of the TCP socket buffers
`> 0`	The default *<nbytes>* value is equal to 128 Kb.

Description

Set this environment variable to define the size of the TCP socket buffers.

Use the `I_MPI_TCP_BUFFER_SIZE` environment variable for tuning your application performance for a given number of processes.

NOTE:

TCP socket buffers of a large size can require more memory for an application with large number of processes. Alternatively, TCP socket buffers of a small size can considerably decrease the bandwidth of each socket connection especially for 10 Gigabit Ethernet and IPoIB (see *I_MPI_TCP_NETMASK* for details).

4.4. 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 a way to control the algorithm selection explicitly: `I_MPI_ADJUST` environment variable family, which is described in the following section.

The environment variable is available for both Intel® and non-Intel microprocessors, but it may perform additional optimizations for Intel microprocessors than it performs for non-Intel microprocessors.

4.4.1. 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 optimized default settings

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

Description

Set this environment 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.

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 5. Knomial 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. Shumilin's ring algorithm 8. Ring algorithm 9. Knomial 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	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 algorithm 8. Knomial 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 5. Rabenseifner's algorithm 6. Topology aware Rabenseifner's algorithm 7. Knomial 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. In the following table, "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

Use the following settings to select the second algorithm for MPI_Reduce operation:

```
I_MPI_ADJUST_REDUCE=2
```

Use the following settings to define the algorithms for MPI_scatter operation:

```
I_MPI_ADJUST_REDUCE_SCATTER=4:0-100,5001-10000;1:101-3200,2:3201-5000;3
```

In this case. algorithm 4 is used for the message sizes between 0 and 100 bytes and from 5001 and 10000 bytes, algorithm 1 is used for the message sizes between 101 and 3200 bytes, algorithm 2 is used for the message sizes between 3201 and 5000 bytes, and algorithm 3 is used for all other messages.

I_MPI_ADJUST_REDUCE_SEGMENT

Syntax

```
I_MPI_ADJUST_REDUCE_SEGMENT=<block_size>|<algid>:<block_size>[,<algid>:<block_size>[...]]
```

Arguments

<algid>	Algorithm identifier
1	Shumilin’s algorithm
3	Topology aware Shumilin’s algorithm
<block_size>	Size in bytes of a message segment
> 0	The default value is 14000

Description:

Set an internal block size to control MPI_Reduce message segmentation for the specified algorithm. If the <algid> value is not set, the <block_size> value is applied for all the algorithms, where it is relevant.

NOTE:

This environment variable is relevant for Shumilin’s and topology aware Shumilin’s algorithms only (algorithm N1 and algorithm N3 correspondingly).

I_MPI_ADJUST_ALLGATHER_KN_RADIX

Syntax

`I_MPI_ADJUST_ALLGATHER_KN_RADIX=<radix>`

Arguments

`<radix>`	An integer that specifies a radix used by the Knomial `MPI_Allgather` algorithm to build a knomial communication tree
`> 1`	The default value is `2`

Description:

Set this environment together with `I_MPI_ADJUST_ALLGATHER=5` to select the knomial tree radix for the corresponding `MPI_Allgather` algorithm.

I_MPI_ADJUST_BCAST_KN_RADIX

Syntax

`I_MPI_ADJUST_BCAST_KN_RADIX=<radix>`

Arguments

`<radix>`	An integer that specifies a radix used by the Knomial `MPI_Bcast` algorithm to build a knomial communication tree
`> 1`	The default value is `4`

Description:

Set this environment together with `I_MPI_ADJUST_BCAST=8` to select the knomial tree radix for the corresponding `MPI_Bcast` algorithm.

I_MPI_ADJUST_ALLREDUCE_KN_RADIX

Syntax

`I_MPI_ADJUST_ALLREDUCE_KN_RADIX=<radix>`

Arguments

`<radix>`	An integer that specifies a radix used by the Knomial `MPI_Allreduce` algorithm to build a knomial communication tree
`> 1`	The default value is `4`

Description:

Set this environment together with `I_MPI_ADJUST_ALLREDUCE=9` to select the knomial tree radix for the corresponding `MPI_Allreduce` algorithm.

I_MPI_ADJUST_REDUCE_KN_RADIX

Syntax

`I_MPI_ADJUST_REDUCE_KN_RADIX=<radix>`

Arguments

`<radix>`	An integer that specifies a radix used by the Knomial `MPI_Reduce` algorithm to build a knomial communication tree
`> 1`	The default value is `4`

Description:

Set this environment together with `I_MPI_ADJUST_REDUCE=7` to select the knomial tree radix for the corresponding `MPI_Reduce` algorithm.

4.5. Miscellaneous

This topic provides the following information:

- Compatibility Control
- Dynamic Process Support
- Statistics Gathering Mode
- ILP64 Support
- Unified Memory Management

4.5.1. Compatibility Control

I_MPI_COMPATIBILITY

Select the runtime compatibility mode.

Syntax

`I_MPI_COMPATIBILITY=<value>`

Arguments

`<value>`	Define compatibility mode
`not defined`	Enable MPI-2.2 standard compatibility. This is the default mode
`3`	Enable the Intel® MPI Library 3.x compatible mode
`4`	Enable the Intel® MPI Library 4.0.x compatible mode

Description

Set this environment variable to choose the Intel® MPI runtime compatible mode. By default, the library complies with the MPI-2.2 standard. If your application depends on the MPI-2.1 behavior, set the value of the environment variable `I_MPI_COMPATIBILITY` to `4`. If your application depends on the pre-MPI-2.1 behavior, set the value of the environment variable `I_MPI_COMPATIBILITY` to `3`.

4.5.2. Dynamic Process Support

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

- 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 set of hosts indicated within `<machine_file>` (see *mpiexec* for details) is used for placement of spawned processes. The spawned processes are placed onto different hosts in round-robin or per-host 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 command:

```
> mpiexec -n 1 -machinefile smpd.hosts -gwdir <path_to_executable>      -genv
I_MPI_FABRICS shm:tcp <spawn_app>
```

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

```
host1
host2
host3
host4
```

the original spawning 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`.

If the `smpd.hosts` contains the following information:

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

the ordinary process is placed on `host1`, while the dynamic processes is distributed as follows: 1 – on `host1`, 2 and 3 – on `host2`, and 4 – on `host1`.

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

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

and the following command on the intended client hosts:

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

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

```
> mpiexec -n 1 -genv I_MPI_FABRICS shm:tcp <join_server_app> < <port_number>

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

4.5.3. Statistics Gathering Mode

This topic describes the Intel® MPI Library statistics gathering modes and how to use such gathering facility through environment variables. The Intel® MPI Library supports the following statistics formats:

- Native statistics format
- IPM statistics format

You can see the information about native statistic format in the topic, Native Statistic Format and the information about IPM statics format in the topic, IPM Statistics Format. There is also possibility to collect both types of statistics. See Native and IPM Statistics for more details.

4.5.3.1. Native Statistics Format

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 sent to 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=[native:][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
`20`	Output additional time information for all MPI functions

Description

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

NOTE:

`n, m` are positive integer numbers. They define the range of output information. The statistics from level `n` to level `m` inclusive are output. If an *n* value is not provided, the default value is `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 environment variable to select the target subsystem in which to collect statistics. All collective and point-to-point operations, including the point-to-point operations performed inside the collectives, are covered by default.

Examples

- The default settings are equivalent to:
  `I_MPI_STATS_SCOPE=coll;p2p`
- 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`
- 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 are 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 environment 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 environment variable is not used, then level 4 statistics is not gathered.

If a range is not specified, 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 environment 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 environment variable to define the statistics output file. By default, the stats.txt file is created in the current directory.

If this variable is not set and the statistics output file already exists, an index is appended to its name. For example, if `stats.txt` exists, the created statistics output file is named as `stats(2).txt`; if `stats(2).txt` exists, the created file is named as `stats(3).txt`, and so on.

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:

```
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.

4.5.3.2. Region Control

The Intel® MPI Library also supports an optional region feature. The region is an IPM statistics format feature. See IPM Statistics Format for more details about IPM. This feature requires the source code modification. The `MPI_Pcontrol` function can be used.

Region is a named part of the source code marked by the start/end points through the standard `MPI_Pcontrol` function calls. The `MPI_Pcontrol` function isn't used for the following special permanent regions:

- Main region contains statistics information about all MPI calls from `MPI_Init` to `MPI_Finalize`. The main region gets the `"*"` name for IPM statistics output. The default output file for this region is `stats.txt` for native statistics format.
- Complementary region contains statistics information not included into any named region. The region gets the `"ipm_noregion"` name in output for IPM statistics format. The default output file for this region is `stats_noregion.txt` for native statistics format.

If named regions are not used, the main regions and the complementary regions are identical and the complementary region is ignored.

Each region contains its own independent statistics information about MPI functions called inside the region.

The Intel® MPI Library supports the following types of regions:

- Discontiguous (several open and close).
- Intersected.
- Covering a subset of MPI processes (part of the `MPI_COMM_WORLD` environment variable).

A region is opened by the `MPI_Pcontrol(1, <name>)` call and closed by the `MPI_Pcontrol(-1, <name>)` call where `name` is a zero terminated string with the region name. The `<name>` is used in output for IPM statistics format. The default output file for the region is `stats_<name>.txt` for native statistics format.

All open regions are closed automatically inside the `MPI_Finalize` environment variable.

4.5.3.3. IPM Statistics Format

The Intel® MPI Library supports integrated performance monitoring (IPM) summary format as part of the built-in statistics gathering mechanism described above. You do not need to modify the source code or re-link your application to collect this information.

The `I_MPI_STATS_BUCKETS` environment variable is not applicable to the IPM format. The `I_MPI_STATS_ACCURACY`environment variable is available to control extra functionality.

I_MPI_STATS

Control the statistics data output format.

Syntax

`I_MPI_STATS=<level>`

Argument

<level>	Level of statistics data
ipm	Summary data throughout all regions
ipm:terse	Basic summary data

Description

Set this environment variable to `ipm` to get the statistics output that contains region summary. Set this environment variable to `ipm:terse` argument to get the brief statistics output.

I_MPI_STATS_FILE

Define the output file name.

Syntax

`I_MPI_STATS_FILE=<name>`

Argument

<name>	File name for statistics data gathering

Description

Set this environment variable to change the statistics output file name from the default name of `stats.ipm`.

If this variable is not set and the statistics output file already exists, an index is appended to its name. For example, if `stats.ipm` exists, the created statistics output file is named as `stats(2).ipm`; if `stats(2).ipm` exists, the created file is named as `stats(3).ipm`, and so on.

I_MPI_STATS_SCOPE

Define a semicolon separated list of subsets of MPI functions for statistics gathering.

Syntax

`I_MPI_STATS_SCOPE=<subset>[;<subset>[;…]]`

Argument

<subset>	Target subset
all2all	Collect statistics data for all-to-all functions types
all2one	Collect statistics data for all-to-one functions types

attr	Collect statistics data for attribute control functions
comm	Collect statistics data for communicator control functions
err	Collect statistics data for error handling functions
group	Collect statistics data for group support functions
init	Collect statistics data for initialize/finalize functions
io	Collect statistics data for input/output support function
one2all	Collect statistics data for one-to-all functions types
recv	Collect statistics data for receive functions
req	Collect statistics data for request support functions
rma	Collect statistics data for one sided communication functions
scan	Collect statistics data for scan collective functions
send	Collect statistics data for send functions
sendrecv	Collect statistics data for send/receive functions
serv	Collect statistics data for additional service functions
spawn	Collect statistics data for dynamic process functions
status	Collect statistics data for status control function
sync	Collect statistics data for barrier synchronization
time	Collect statistics data for timing support functions
topo	Collect statistics data for topology support functions
type	Collect statistics data for data type support functions

Description

Use this environment variable to define a subset or subsets of MPI functions for statistics gathering specified by the following table. A union of all subsets is used by default.

Table 5.2-1 Stats Subsets of MPI Functions

all2all	*MPI_File_get_errhandler*
MPI_Allgather	*MPI_File_set_errhandler*
MPI_Allgatherv	*MPI_Win_call_errhandler*
MPI_Allreduce	*MPI_Win_create_errhandler*
MPI_Alltoll	*MPI_Win_get_errhandler*
MPI_Alltoallv	*MPI_Win_set_errhandler*
MPI_Alltoallw	
MPI_Reduce_scatter	**group**
MPI_Iallgather	*MPI_Group_compare*
MPI_Iallgatherv	*MPI_Group_difference*
MPI_Iallreduce	*MPI_Group_excl*
MPI_Ialltoll	*MPI_Group_free*
MPI_Ialltoallv	*MPI_Group_incl*
MPI_Ialltoallw	*MPI_Group_intersection*
MPI_Ireduce_scatter	*MPI_Group_range_excl*
MPI_Ireduce_scatter_block	*MPI_Group_range_incl*
	MPI_Group_rank
all2one	*MPI_Group_size*
MPI_Gather	*MPI_Group_translate_ranks*
MPI_Gatherv	*MPI_Group_union*
MPI_Reduce	
MPI_Igather	**init**
MPI_Igatherv	*MPI_Init*
MPI_Ireduce	*MPI_Init_thread*
	MPI_Finalize
attr	
MPI_Comm_create_keyval	**io**
MPI_Comm_delete_attr	*MPI_File_close*
MPI_Comm_free_keyval	*MPI_File_delete*
MPI_Comm_get_attr	*MPI_File_get_amode*
MPI_Comm_set_attr	*MPI_File_get_atomicity*

MPI_Comm_get_name	*MPI_File_get_byte_offset*
MPI_Comm_set_name	*MPI_File_get_group*
MPI_Type_create_keyval	*MPI_File_get_info*
MPI_Type_delete_attr	*MPI_File_get_position*
MPI_Type_free_keyval	*MPI_File_get_position_shared*
MPI_Type_get_attr	*MPI_File_get_size*
MPI_Type_get_name	*MPI_File_get_type_extent*
MPI_Type_set_attr	*MPI_File_get_view*
MPI_Type_set_name	*MPI_File_iread_at*
MPI_Win_create_keyval	*MPI_File_iread*
MPI_Win_delete_attr	*MPI_File_iread_shared*
MPI_Win_free_keyval	*MPI_File_iwrite_at*
MPI_Win_get_attr	*MPI_File_iwrite*
MPI_Win_get_name	*MPI_File_iwrite_shared*
MPI_Win_set_attr	*MPI_File_open*
MPI_Win_set_name	*MPI_File_preallocate*
MPI_Get_processor_name	*MPI_File_read_all_begin*
	MPI_File_read_all_end
comm	*MPI_File_read_all*
MPI_Comm_compare	*MPI_File_read_at_all_begin*
MPI_Comm_create	*MPI_File_read_at_all_end*
MPI_Comm_dup	*MPI_File_read_at_all*
MPI_Comm_free	*MPI_File_read_at*
MPI_Comm_get_name	*MPI_File_read*
MPI_Comm_group	*MPI_File_read_ordered_begin*
MPI_Comm_rank	*MPI_File_read_ordered_end*
MPI_Comm_remote_group	*MPI_File_read_ordered*
MPI_Comm_remote_size	*MPI_File_read_shared*
MPI_Comm_set_name	*MPI_File_seek*
MPI_Comm_size	*MPI_File_seek_shared*
MPI_Comm_split	*MPI_File_set_atomicity*

MPI_Comm_test_inter	*MPI_File_set_info*
MPI_Intercomm_create	*MPI_File_set_size*
MPI_Intercomm_merge	*MPI_File_set_view*
	MPI_File_sync
err	*MPI_File_write_all_begin*
MPI_Add_error_class	*MPI_File_write_all_end*
MPI_Add_error_code	*MPI_File_write_all*
MPI_Add_error_string	*MPI_File_write_at_all_begin*
MPI_Comm_call_errhandler	*MPI_File_write_at_all_end*
MPI_Comm_create_errhandler	*MPI_Ibsend*
MPI_Comm_get_errhandler	*MPI_Irsend*
MPI_Comm_set_errhandler	*MPI_Issend*
MPI_Errhandler_free	*MPI_Send_init*
MPI_Error_class	*MPI_Bsend_init*
MPI_Error_string	*MPI_Rsend_init*
MPI_File_call_errhandler	*MPI_Ssend_init*
MPI_File_create_errhandler	
MPI_File_write_at_all	*sendrecv*
MPI_File_write_at	*MPI_Sendrecv*
MPI_File_write	*MPI_Sendrecv_replace*
MPI_File_write_ordered_begin	
MPI_File_write_ordered_end	**serv**
MPI_File_write_ordered	*MPI_Alloc_mem*
MPI_File_write_shared	*MPI_Free_mem*
MPI_Register_datarep	*MPI_Buffer_attach*
	MPI_Buffer_detach
one2all	*MPI_Op_create*
MPI_Bcast	*MPI_Op_free*
MPI_Scatter	
MPI_Scatterv	**spawn**
MPI_Ibcast	*MPI_Close_port*

MPI_Iscatter	*MPI_Comm_accept*
MPI_Iscatterv	*MPI_Comm_connect*
	MPI_Comm_disconnect
recv	*MPI_Comm_get_parent*
MPI_Recv	*MPI_Comm_join*
MPI_Irecv	*MPI_Comm_spawn*
MPI_Recv_init	*MPI_Comm_spawn_multiple*
MPI_Probe	*MPI_Lookup_name*
MPI_Iprobe	*MPI_Open_port*
	MPI_Publish_name
req	*MPI_Unpublish_name*
MPI_Start	
MPI_Startall	**status**
MPI_Wait	*MPI_Get_count*
MPI_Waitall	*MPI_Status_set_elements*
MPI_Waitany	*MPI_Status_set_cancelled*
MPI_Waitsome	*MPI_Test_cancelled*
MPI_Test	
MPI_Testall	**sync**
MPI_Testany	*MPI_Barrier*
MPI_Testsome	
MPI_Cancel	*MPI_Ibarrier*
MPI_Grequest_start	
MPI_Grequest_complete	**time**
MPI_Request_get_status	*MPI_Wtick*
MPI_Request_free	*MPI_Wtime*
rma	**topo**
MPI_Accumulate	*MPI_Cart_coords*
MPI_Get	*MPI_Cart_create*
MPI_Put	*MPI_Cart_get*

MPI_Win_complete	*MPI_Cart_map*
MPI_Win_create	*MPI_Cart_rank*
MPI_Win_fence	*MPI_Cart_shift*
MPI_Win_free	*MPI_Cart_sub*
MPI_Win_get_group	*MPI_Cartdim_get*
MPI_Win_lock	*MPI_Dims_create*
MPI_Win_post	*MPI_Graph_create*
MPI_Win_start	*MPI_Graph_get*
MPI_Win_test	*MPI_Graph_map*
MPI_Win_unlock	*MPI_Graph_neighbors*
MPI_Win_wait	*MPI_Graphdims_get*
MPI_Win_allocate	*MPI_Graph_neighbors_count*
MPI_Win_allocate_shared	*MPI_Topo_test*
MPI_Win_create_dynamic	
MPI_Win_shared_query	**type**
MPI_Win_attach	*MPI_Get_address*
MPI_Win_detach	*MPI_Get_elements*
MPI_Win_set_info	*MPI_Pack*
MPI_Win_get_info	*MPI_Pack_external*
MPI_Win_get_accumulate	*MPI_Pack_external_size*
MPI_Win_fetch_and_op	*MPI_Pack_size*
MPI_Win_compare_and_swap	*MPI_Type_commit*
MPI_Rput	*MPI_Type_contiguous*
MPI_Rget	*MPI_Type_create_darray*
MPI_Raccumulate	*MPI_Type_create_hindexed*
MPI_Rget_accumulate	*MPI_Type_create_hvector*
MPI_Win_lock_all	*MPI_Type_create_indexed_block*
MPI_Win_unlock_all	*MPI_Type_create_resized*
MPI_Win_flush	*MPI_Type_create_struct*
MPI_Win_flush_all	*MPI_Type_create_subarray*
MPI_Win_flush_local	*MPI_Type_dup*

MPI_Win_flush_local_all	*MPI_Type_free*
MPI_Win_sync	*MPI_Type_get_contents*
	MPI_Type_get_envelope
scan	*MPI_Type_get_extent*
MPI_Exscan	*MPI_Type_get_true_extent*
MPI_Scan	*MPI_Type_indexed*
MPI_Iexscan	*MPI_Type_size*
MPI_Iscan	*MPI_Type_vector*
	MPI_Unpack_external
send	*MPI_Unpack*
MPI_Send	
MPI_Bsend	
MPI_Rsend	
MPI_Ssend	
MPI_Isend	

I_MPI_STATS_ACCURACY

Use the `I_MPI_STATS_ACCURACY` environment variable to decrease statistics output.

Syntax

`I_MPI_STATS_ACCURACY=<percentage>`

Argument

<percentage>	Float threshold value

Description

Set this environment variable to collect data only on those MPI functions that take a larger portion of the elapsed time as a percentage of the total time spent inside all MPI calls.

Example

The following example represents a simple application code and IPM summary statistics format:

```
int main (int argc, char *argv[])

{

    int i, rank, size, nsend, nrecv;
```

```
MPI_Init (&argc, &argv);

MPI_Comm_rank (MPI_COMM_WORLD, &rank);
nsend = rank;

MPI_Wtime();

for (i = 0; i < 200; i++)
{
    MPI_Barrier(MPI_COMM_WORLD);
}

      /* open "reduce" region for all processes */

MPI_Pcontrol(1, "reduce");
for (i = 0; i < 1000; i++)
    MPI_Reduce(&nsend, &nrecv, 1, MPI_INT, MPI_MAX, 0, MPI_COMM_WORLD);

           /* close "reduce" region */

MPI_Pcontrol(-1, "reduce");

if (rank == 0)
{

       /* "send" region for 0-th process only */

    MPI_Pcontrol(1, "send");
    MPI_Send(&nsend, 1, MPI_INT, 1, 1, MPI_COMM_WORLD);
    MPI_Pcontrol(-1, "send");
}
if (rank == 1)
```

```
    {
        MPI_Recv(&nrecv, 1, MPI_INT, 0, 1, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    }


       /* reopen "reduce" region */
    MPI_Pcontrol(1, "reduce");
    for (i = 0; i < 1000; i++)
        MPI_Reduce(&nsend, &nrecv, 1, MPI_INT, MPI_MAX, 0, MPI_COMM_WORLD);


    MPI_Wtime();
    MPI_Finalize ();


    return 0;
}
```

```
Command:
mpiexec -n 4 -env I_MPI_STATS ipm:terse ./a.out
```

Statistics output:

```
################################################################
####################

#

# command : unknown (completed)

# host    : SVLMPICL704/Windows          mpi_tasks : 4 on 1 nodes

# start   : 06/17/11/14:10:40            wallclock : 0.037681 sec

# stop    : 06/17/11/14:10:40            %comm     : 99.17

# gbytes  : 0.00000e+000 total           gflop/sec : NA

#

################################################################
####################
```

Command:

```
mpiexec -n 4 -env I_MPI_STATS ipm ./a.out
```

Stats output:

```
################################################################
####################

#
```

```
# command : unknown (completed)
# host    : SVLMPICL704/Windows          mpi_tasks : 4 on 1 nodes
# start   : 06/17/11/14:10:40            wallclock : 0.037681 sec
# stop    : 06/17/11/14:10:40            %comm     : 99.17
# gbytes  : 0.00000e+000 total           gflop/sec : NA
#
##############################################################################
# region  : *   [ntasks] = 4
#
#                  [total]      <avg>        min          max
# entries             4           1           1           1
# wallclock           0.118763    0.0296908   0.0207312   0.0376814
# user                0.0156001   0.00390002  0           0.0156001
# system              0           0           0           0
# mpi                 0.117782    0.0294454   0.0204467   0.0374543
# %comm                           99.1735     98.6278     99.3973
# gflop/sec           NA          NA          NA          NA
# gbytes              0           0           0           0
#
#
#                  [time]       [calls]      <%mpi>       <%wall>
# MPI_Init            0.0944392   4           80.18        79.52
# MPI_Reduce          0.0183164   8000        15.55        15.42
# MPI_Recv            0.00327056  1           2.78         2.75
# MPI_Barrier         0.00174499  800         1.48         1.47
# MPI_Send            4.23448e-006 1          0.00         0.00
# MPI_Finalize        3.07963e-006 4          0.00         0.00
# MPI_Wtime           1.53982e-006 8          0.00         0.00
# MPI_Comm_rank       1.5398e-006  4          0.00         0.00
# MPI_TOTAL           0.117782    8822        100.00       99.17
```

```
##############################################################################
###################

# region  : reduce   [ntasks] = 4

#

#                    [total]      <avg>        min          max

# entries             8           2           2           2

# wallclock           0.0190786    0.00476966    0.00273201    0.00665929

# user                0           0           0           0

# system              0           0           0           0

# mpi                 0.0183199    0.00457997    0.00255377    0.00643987

# %comm                           96.0231      93.4761      97.0543

# gflop/sec           NA          NA          NA          NA

# gbytes              0           0           0           0

#

#

#                    [time]       [calls]      <%mpi>       <%wall>

# MPI_Reduce          0.0183164    8000         99.98        96.00

# MPI_Finalize        3.07963e-006  4           0.02         0.02

# MPI_Wtime           3.84956e-007  4           0.00         0.00

# MPI_TOTAL           0.0183199    8008         100.00       96.02

##############################################################################
###################

# region  : send   [ntasks] = 4

#

#                    [total]      <avg>        min          max

# entries             1           0           0           1

# wallclock           1.22389e-005  3.05971e-006  1e-006       9.23885e-006

# user                0           0           0           0

# system              0           0           0           0

# mpi                 4.23448e-006  1.05862e-006  0           4.23448e-006

# %comm                           34.5986      0           45.8334

# gflop/sec           NA          NA          NA          NA
```

```
# gbytes              0          0          0          0
#
#
#                    [time]       [calls]      <%mpi>       <%wall>
# MPI_Send             4.23448e-006  1          100.00       34.60
################################################################
###################
# region  : ipm_noregion   [ntasks] = 4
#
#                    [total]      <avg>        min          max
# entries              13          3          3          4
# wallclock            0.0996611     0.0249153     0.0140604     0.0349467
# user                 0.0156001     0.00390002    0          0.0156001
# system               0          0          0          0
# mpi                  0.0994574     0.0248644     0.0140026     0.0349006
# %comm                           99.7957       99.5893       99.8678
# gflop/sec            NA          NA          NA          NA
# gbytes               0          0          0          0
#
#
#                    [time]       [calls]      <%mpi>       <%wall>
# MPI_Init             0.0944392     4          94.95        94.76
# MPI_Recv             0.00327056    1          3.29         3.28
# MPI_Barrier          0.00174499    800        1.75         1.75
# MPI_Comm_rank          1.5398e-006   4          0.00         0.00
# MPI_Wtime            1.15486e-006  4          0.00         0.00
# MPI_TOTAL            0.0994574     813        100.00       99.80
```

4.5.3.4. Native and IPM Statistics

The statistics in each supported format can be collected separately. To collect statistics in all formats with the maximal level of details, use the `I_MPI_STATS` environment variable.

I_MPI_STATS

Syntax

```
I_MPI_STATS=all
```

NOTE: The `I_MPI_STATS_SCOPE` environment variable is not applicable when both types of statistics are collected.

To control the amount of statistics information, use the ordinary `I_MPI_STATS` values, separated by comma.

Syntax

```
I_MPI_STATS=[native:][n-]m,ipm[:terse]
```

NOTE: Currently the alias all corresponds to `I_MPI_STATS=native:20,ipm` and can be changed.

4.5.4. ILP64Support

The term *ILP64* means that integer, 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 integer entities occupy 4 bytes. 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

4.5.4.1. 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
  ```

4.5.4.2. Known Issues and Limitations

- Data type 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.

- If you want to use the Intel® Trace Collector with the Intel MPI ILP64 executable files, you must use a special ITC library. 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.

4.5.5. 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.

To use the unified memory management subsystem, link your application against the `libimalloc.dll`.

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
}
```

4.6. Secure Loading of Dynamic Link Libraries*

The Intel® MPI Library provides enhanced security options for the loading of Dynamic Link Libraries*. You can enable the enhanced security mode for the dynamic library loading, as well as define a set of directories in which the library will attempt to locate an external DLL*.

The security options are placed in the `HKEY_LOCAL_MACHINE\Software\Intel\MPI` protected Windows* registry key. The location prevents the options from being changed with non-administrative privileges.

SecureDynamicLibraryLoading

Select the secure DLL loading mode.

Syntax

`SecureDynamicLibraryLoading=<value>`

Arguments

`<value>`	Binary indicator
`enable | yes | on | 1`	Enable the secure DLL loading mode
`disable | no | off | 0`	Disable the secure DLL loading mode. This is the default value

Description

Use `HKEY_LOCAL_MACHINE\Software\Intel\MPI` registry key to define the `SecureDynamicLibraryLoading` registry entry. Set this entry to enable the secure DLL loading mode.

I_MPI_DAT_LIBRARY

Select a particular DAT library to be used in the DLL enhanced security mode.

Syntax

`I_MPI_DAT_LIBRARY=<library>`

Arguments

`<library>`	Specify the name of the library to be loaded

Description

In the secure DLL loading mode, the library changes the default-defined set of directories to locate DLLs. Therefore, the current working directory and the directories that are listed in the `PATH` environment variable may be ignored. To select a specific external DAT library to be loaded, define the `I_MPI_DAT_LIBRARY` entry of the `HKEY_LOCAL_MACHINE\Software\Intel\MPI` registry key. Specify the full path to the DAT library.

NOTE:

The `I_MPI_DAT_LIBRARY` environment variable has no effect in the secure DLL loading mode. See *I_MPI_DAT_LIBRARY* for more details.

SecurePath

Specify a set of directories to locate an external DLL.

Syntax

`SecurePath=<path>[;<path>[...]]`

Arguments

<path>	Specify the path to a directory

Description

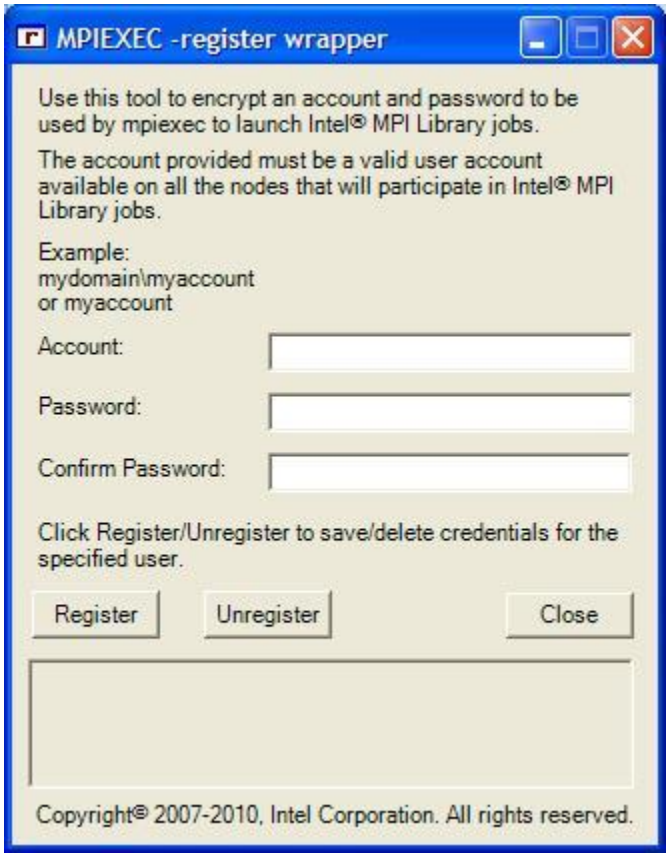
Use `HKEY_LOCAL_MACHINE\Software\Intel\MPI` registry key to define the SecurePath registry entry. Set this entry to specify a set of directories to locate an external DLL in the secure DLL loading mode. Use a safe set of directories instead of some publicly writable directories to avoid insecure library loading.

NOTE:

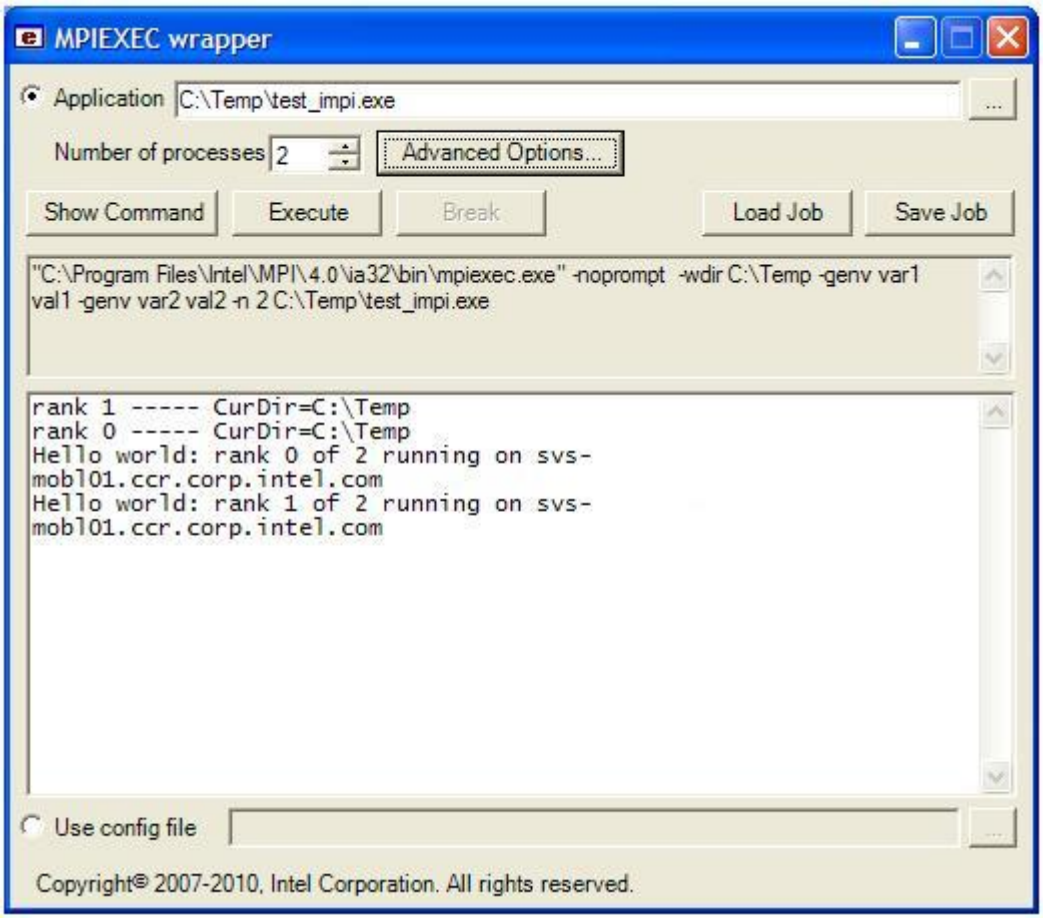
Use this option when the library is unable to load a DLL in the secure DLL loading mode. The option has no effect if the secure DLL loading mode is turned off.

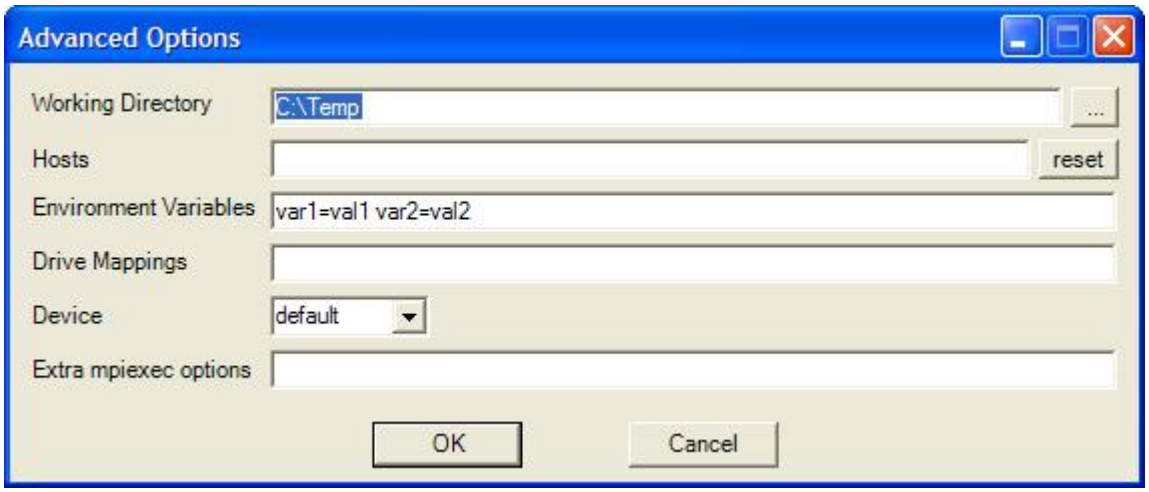
5. Graphical Utilities

The Intel® MPI Library provides three graphical utilities: `wmpiregister`, `wmpiexec`, and `wmpiconfig`. These utilities simplify using the Intel® MPI Library under Windows* OS.



Use the `wmpiregister` utility to encrypt and store your account name and a password. The specified account and the password are used for all subsequent MPI jobs you start. The first time you use this utility, you need to enter an account name and a password during the first `mpiexec` invocation.



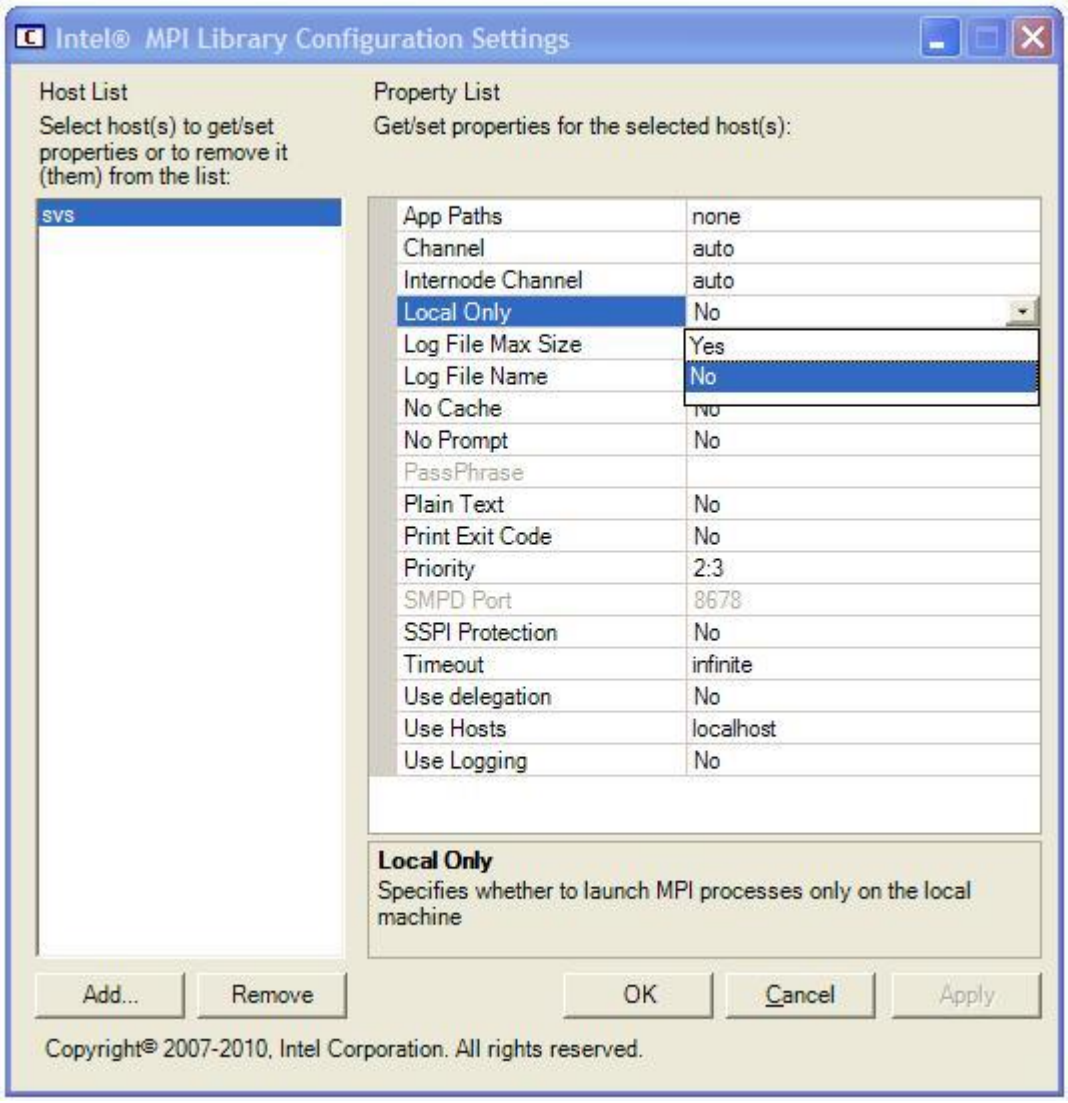


Use the `wmpiexec` utility as a graphical interface to the `mpiexec` command. This utility allows you to:

1. Describe the job by specifying the following:
    - an application to run
    - a number of instances
    - host names

- a communication device to be used
- a working directory for the MPI processes
- environment variables to be set for the MPI processes
- drive mappings to be used
- extra MPI options for `wmpiexec`

2. Save the job description using the **Save Job** button (optional).
3. Load the job description using the **Load Job** button (optional).
4. View the actual `mpiexec` command line using the **Show Command** button.
5. Launch the job using the **Execute** button.
6. Break the job execution using the **Break** button.



Use the `wmpiconfig` utility to view/change the Intel® MPI Library settings for different hosts. This affects every job run on that host. The work with the `wmpiconfig` utility can be split into three steps:

1. Select the host(s) for which you want to change the Intel® MPI Library settings, and add them to the host list using the **Add** button.
2. Select host(s) in the host list to view the properties. If more than one host name is selected, intersection of the properties is displayed.
3. Change properties for the selected host(s) and press the **Apply** button in confirmation.

6. Glossary

cell	A pinging resolution in descriptions for pinning property.
hyper-threading technology	A feature within the IA-64 and Intel® 64 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.

7. Index

{

-{cc cxx fc}=<compiler> 11

A

Active Directory* 56

--application 58

B

-binding 36

-bootstrap 39

-bootstrap-exec 39

-branch-count 33

C

-configfile 33

-configfile <filename> 15

cpuinfo 51, 52

D

-delegate 17

-dir <directory> 19

E

-echo 10

-env <ENVVAR> <value> 18, 40

-envexcl <list of env var names> 19

-envlist <list of env var names> 18, 40

-envnone 18, 40

-exitcodes 17

G

-g<l-option> 16

-genv <ENVVAR> <value> 31

-genvall 31

-genvlist 32

-genvnone 32

-grr 32

H

-h 18, 27

--help 18, 27

-host <nodename> 19

-hostfile 30

-hostos 40

-hosts 27, 33

hydra 29

hydra_service 29

I

I_MPI_{CC CXX FC F77 F90} 12

I_MPI_{CC CXX FC F77 F90}_PROFILE 12

I_MPI_ADJUST 107

I_MPI_ADJUST_<opname> 107

I_MPI_ADJUST_ALLGATHER_KN_RADIX 113

I_MPI_ADJUST_ALLREDUCE_KN_RADIX 113

I_MPI_ADJUST_BCAST_KN_RADIX 113

I_MPI_ADJUST_REDUCE_KN_RADIX 114

I_MPI_ADJUST_REDUCE_SEGMENT 112

I_MPI_AUTH_METHOD 57

I_MPI_CHECK_DAPL_PROVIDER_COMPATIBILITY 105

I_MPI_COMPATIBILITY 114

I_MPI_COMPILER_CONFIG_DIR 13

I_MPI_DAPL_BUFFER_NUM 100

I_MPI_DAPL_BUFFER_SIZE 100

I_MPI_DAPL_CHECK_MAX_RDMA_SIZE 102

I_MPI_DAPL_CONN_EVD_SIZE 103

I_MPI_DAPL_DESIRED_STATIC_CONNECTIONS_NUM 104

I_MPI_DAPL_DIRECT_COPY_THRESHOLD 98
I_MPI_DAPL_EAGER_MESSAGE_AGGREGATION 99
I_MPI_DAPL_MAX_MSG_SIZE 102
I_MPI_DAPL_RDMA_RNDV_WRITE 101
I_MPI_DAPL_RNDV_BUFFER_ALIGNMENT 101
I_MPI_DAPL_SCALABLE_PROGRESS 100
I_MPI_DAPL_SR_BUF_NUM 103
I_MPI_DAPL_SR_THRESHOLD 103
I_MPI_DAPL_TRANSLATION_CACHE 97
I_MPI_DAPL_TRANSLATION_CACHE_AVL_TREE 98
I_MPI_DAT_LIBRARY 24, 97, 139
I_MPI_DEBUG 10, 21
I_MPI_DEBUG_OUTPUT 21
I_MPI_DYNAMIC_CONNECTION 89
I_MPI_EAGER_THRESHOLD 87, 88
I_MPI_FABRICS 84, 85, 86, 87
I_MPI_FABRICS_LIST 85, 86, 87
I_MPI_FALLBACK 85, 86
I_MPI_HYDRA_BOOTSTRAP 46
I_MPI_HYDRA_BOOTSTRAP_EXEC 46
I_MPI_HYDRA_BRANCH_COUNT 47
I_MPI_HYDRA_DEBUG 43
I_MPI_HYDRA_ENV 43
I_MPI_HYDRA_HOST_FILE 42
I_MPI_HYDRA_IFACE 48
I_MPI_HYDRA_PMI_AGGREGATE 48
I_MPI_HYDRA_PMI_CONNECT 46
I_MPI_INTRANODE_EAGER_THRESHOLD 87, 88, 93, 94, 96
I_MPI_JOB_TIMEOUT 23, 24, 44
I_MPI_JOB_TIMEOUT_SIGNAL 44
I_MPI_PERHOST 47
I_MPI_PIN 66
I_MPI_PIN_CELL 72
I_MPI_PIN_DOMAIN 72, 73, 74
I_MPI_PIN_ORDER 82, 83
I_MPI_PIN_PROCESSOR_LIST 66, 69, 70, 72
I_MPI_PIN_PROCS 66
I_MPI_PRINT_VERSION 22
I_MPI_ROOT 13
I_MPI_SCALABLE_OPTIMIZATION 88
I_MPI_SHM_BYPASS 95
I_MPI_SHM_CACHE_BYPASS 90
I_MPI_SHM_CACHE_BYPASS_THRESHOLDS 90
I_MPI_SHM_CELL_NUM 92
I_MPI_SHM_CELL_SIZE 87, 92
I_MPI_SHM_FBOX 91
I_MPI_SHM_FBOX_SIZE 91
I_MPI_SHM_LMT 93
I_MPI_SHM_LMT_BUFFER_NUM 93
I_MPI_SHM_LMT_BUFFER_SIZE 93
I_MPI_SHM_SPIN_COUNT 96
I_MPI_SMPD_VERSION_CHECK 24
I_MPI_SPIN_COUNT 88
I_MPI_SSHM 94
I_MPI_SSHM_BUFFER_NUM 94
I_MPI_SSHM_BUFFER_SIZE 94
I_MPI_SSHM_DYNAMIC_CONNECTION 90
I_MPI_STATS 116
I_MPI_STATS_ACCURACY 123
I_MPI_STATS_BUCKETS 118, 119
I_MPI_STATS_FILE 119, 124

I_MPI_STATS_SCOPE 117, 124
I_MPI_TCP_BUFFER_SIZE 106
I_MPI_TCP_NETMASK 105
I_MPI_TMPDIR 48
I_MPI_TUNER_DATA_DIR 16, 25
-iface 33
-ilp64 9
ILP64 137
-impersonate 17
-info 35
IPM 123
--iterations 58

L

-l 33
large message transfer (LMT) 93
LMT 93
-localroot 17
-logon 17

M

-machinefile <machine file> 31
-map <drive\\host\share> 19
-mapall 19
MAX_INT 102
max_rdma_size 102
mpd 60
MPI_Allgather 111
MPI_Allgatherv 111
MPI_Allreduce 111, 117, 120
MPI_Alltoallv 111
MPI_Alltoallw 111
MPI_Barrier 111
MPI_Bcast 111, 117
MPI_COMM_WORLD 120
MPI_Exscan 111
MPI_Finalize 123
MPI_Gather 111
MPI_Gatherv 111
MPI_Init 123
MPI_Pcontrol 123
MPI_Reduce 112, 117
MPI_Reduce_scatter 111
MPI_Scan 112
MPI_Scatter 112, 118
MPI_Scatterv 118
MPICH_{CC CXX FC F77 F90} 12
mpiexec 30, 49
mpiexec.smpd 15, 23, 24, 28, 49, 57, 141, 142, 143
mpitune 58

N

-n <# of processes> 18, 40
-noconf 34
-nopopup_debug 17

O

-O 10
-ordered-output 34

P

-p <port> 16
-path <directory> 34
-perhost 32
-pmi-aggregate 33
-pmi-connect 32
-port <port> 16

-ppn 32
-print-all-exitcodes 39
-print-rank-map 39
-profile 11
-profile=<profile_name> 9, 12
-pwdfile <filename> 17
R
-register 35
-register [-user n] 18
-remove 35
-remove [-user n] 18
-rr 32
S
-s 34
SecureDynamicLibraryLoading 138
SecurePath 139
Security Service Provider Interface (SSPI) 55
Service Principal Name (SPN) 28, 56
-show 10
-show_env 11
SMPD 24, 26, 27, 28
T
-t 9
-timeout <seconds> 18
-tmpdir 34
–trace-collectives 32
-trace-pt2pt 32
-tune 33, 63
V
-v 11
-validate 18
-verbose 17
-version 35
W
-whoami 18