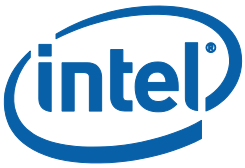


Intel® MPI Library for Linux* OS

Reference Manual

Document Number: 315399-011

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

Contents

Disclaimer and Legal Notices

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 environment variables for compiler commands, job startup commands, and MPD daemon commands as well
Section 3 Tuning Reference	Section 3 describes environment variables used to influence program behavior and performance at run time
Section 4 Glossary	Section 4 explains basic terms used in this document
Section 5 Index	Section 5 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, v2.2 (MPI-2.2) specification. It provides a standard library across Intel® platforms that enable adoption of MPI-2.2 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 Multipurpose Daemon* (MPD), Hydra* and supporting utilities, shared (.so) libraries, and documentation.
- The *Intel® MPI Library Development Kit* (SDK) includes all of the Runtime Environment components plus compilation tools, including compiler commands such as `mpiicc`, include files and modules, static (.a) libraries, debug libraries, trace libraries, and test codes.

1.2. 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 4.1 Update 3 for Linux* OS.

The following latest changes in this document were made:

- Update the `I_MPI_PIN_MODE` environment variable.
- Add the `I_MPI_JOB_RESPECT_PROCESS_PLACEMENT` environment variable.
- Add the cell description in the topic, Glossary.

1.4. Notational Conventions

The following conventions are used in this document.

Conventions and Symbols used in this Document

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

1.5. 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
- Simplified job startup command
- Scalable process management system (Hydra) commands
- Intel® Xeon Phi™ Coprocessor Support
- Multipurpose daemon commands
- 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	Default Compiler	Supported Language(s)	Supported ABI(s)
Generic Compilers			
mpicc	gcc, cc	C	32/64 bit
mpicxx	g++	C/C++	32/64 bit
mpifc	gfortran	Fortran77*/Fortran 95*	32/64 bit
GNU* Compilers Versions 3 and Higher			
mpigcc	gcc	C	32/64 bit
mpigxx	g++	C/C++	32/64 bit
mpif77	g77	Fortran 77	32/64 bit
mpif90	gfortran	Fortran 95	32/64 bit
Intel® Fortran, C++ Compilers Versions 13.1 through 14.0 and Higher			
mpiicc	icc	C	32/64 bit
mpiicpc	icpc	C++	32/64 bit
mpiifort	ifort	Fortran77/Fortran 95	32/64 bit

- Compiler commands are available only in the Intel® MPI Library Development Kit.
- Compiler commands are in the *<installdir>/<arch>/*bin directory. Where *<installdir>* refers to the Intel® MPI Library installation directory and *<arch>* is one of the following architectures:
- ia32 - IA-32 architecture
- intel64 - Intel® 64 architecture
- mic – Intel® Xeon Phi™ Coprocessor architecture
- Ensure that the corresponding underlying compilers (32-bit or 64-bit, as appropriate) are already in your PATH.
- To port existing MPI-enabled applications to the Intel® MPI Library, recompile all sources.
- To display mini-help of a compiler command, execute it without any parameters.

2.1.1. Compiler Command Options

-mt_mpi

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

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

NOTE:

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

NOTE:

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

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

-static_mpi

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

-static

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

-config=<*name*>

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

-profile=<*profile_name*>

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

- Through the configuration file `<profile_name>.conf` located in the `<installdir>/<arch>/etc`. See *Profiles* for details.
- In the absence of the respective configuration file, by linking the library `lib<profile_name>.so` or `lib<profile_name>.a` located in the same directory as the Intel® MPI Library.

-t or -trace

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

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

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

-check_mpi

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

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

-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.

-dynamic_log

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

To run the resulting programs, include `$VT_ROOT/slib` in the `LD_LIBRARY_PATH` environment variable.

-g

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

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`, `-t=log`, `-trace=log` and `-g`.

-O

Use this option to enable compiler optimization.

-fast

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

See xHost for information on implication of this option on non-Intel processors.

NOTE:

This option is supported on `mpiicc`, `mpiicpc`, and `mpiifort` Intel compiler drivers.

-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 -show -c test.c
```

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

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

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

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

Use this option to select the underlying compiler.

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

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

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

-gcc-version=<*nnn*>

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

<*nnn*> value	GNU* C++ version
320	3.2.x
330	3.3.x
340	3.4.x
400	4.0.x
410	4.1.x
420	4.2.x
430	4.3.x
440	4.4.x
450	4.5.x
460	4.6.x
470	4.7.x

By default, the library compatible with the detected version of the GNU* C++ compiler is used. Do not use this option if the GNU* C++ version is lower than 4.0.0.

-compchk

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

`-v`

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

2.1.2. Configuration Files

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

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

where:

<arch> = `{ia32,em64t,mic}` for the IA-32, the Intel® 64 architectures, and Intel® Xeon Phi™ Coprocessor architecture the respectively

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

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

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 these files, or use the `-config` option 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. The profile files are located in the *<installdir>*`/`*<arch>*`/etc` directory. The Intel® MPI Library comes with several predefined profiles for the Intel® Trace Collector:

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

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

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

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

The following 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 `/myprof.conf` with the following lines:

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

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

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

2.1.4. Environment Variables

I_MPI_{CC,CXX,FC,F77,F90}_PROFILE

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

Specify a default profiling library.

Syntax

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

Deprecated Syntax

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

Arguments

`<profile_name>`	Specify a default profiling library.

Description

Set this 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` or another Intel® MPI Library compiler driver.

I_MPI_TRACE_PROFILE

Specify a default profile for the -trace option.

Syntax

`I_MPI_TRACE_PROFILE=<profile_name>`

Arguments

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

Description

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

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

I_MPI_CHECK_PROFILE

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

Syntax

`I_MPI_CHECK_PROFILE=<profile_name>`

Arguments

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

Description

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

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

I_MPI_CHECK_COMPILER

Turn on/off compiler compatibility check.

Syntax

`I_MPI_CHECK_COMPILER=<arg>`

Arguments

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

Description

If `I_MPI_CHECK_COMPILER` is set to `enable`, the Intel MPI compiler drivers check the underlying compiler for compatibility. Normal compilation requires using a known version of the underlying compiler.

I_MPI_{CC,CXX,FC,F77,F90}

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

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

Syntax

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

Deprecated Syntax

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

Arguments

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

Description

Set this 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.

NOTE:

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

I_MPI_ROOT

Set the Intel® MPI Library installation directory path.

Syntax

`I_MPI_ROOT=<path>`

Arguments

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

Description

Set this 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
log	The logging, single threaded version of the Intel MPI Library
log_mt	The logging, multithreaded version of the Intel MPI Library

Description

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

2.2. Simplified Job Startup Command

mpirun

Syntax

mpirun *<options>*

where *<options> := <mpiexec.hydra options>* | [*<mpdboot options>*] *<mpiexec options>*

Arguments

<mpiexec.hydra options>	mpiexec.hydra options as described in the mpiexec.hydra section. This is the default operation mode.

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

Description

Use this command to launch an MPI job. The mpirun command uses Hydra* or MPD as the underlying process managers. Hydra is the default process manager. Set the I_MPI_PROCESS_MANAGER environment variable to change the default value.

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

In this case, you do not need to create the mpd.hosts file. Allocate the session using a job scheduler installed on your system, and use the mpirun command inside this session to run your MPI job.

Hydra* Specification

When running under a job manager, the mpirun command ignores the -r | --rsh option if Hydra* is used as the underlying process manager. In this case, the corresponding Hydra* bootstrap server is used. Use the bootstrap specific options or corresponding environment variables explicitly to override the auto detected bootstrap server.

The mpirun command silently ignores the MPD specific options for compatibility reasons if you select Hydra* as the active process manager. The following table provides the list of silently ignored and unsupported MPD* options. Avoid these unsupported options if the Hydra* process manager is used.

Ignored mpdboot Options	Ignored mpiexec Options	Unsupported mpdboot Options	Unsupported mpiexec Options
--loccons	-[g]envuser	--user=<user> | -u <user>	-a
--remcons	-[g]envexcl	--mpd=<mpdcmd> | -m <mpdcmd>	
--ordered | -o	-m	--shell | -s	
--maxbranch=<maxbranch> | -b <maxbranch>	-ifhn <interface/hostname>	-1	
--parallel-startup | -p	-ecfn <filename>	--ncpus=<ncpus>	

	-tvsu		

MPD* Specification

If you select MPD* as the process manager, the `mpirun` command automatically starts an independent ring of the mpd daemons, launches an MPI job, and shuts down the mpd ring upon job termination.

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

All configuration files and environment variables applicable to the `mpdboot` and `mpiexec` commands also apply to the `mpirun`command.

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

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

I_MPI_MPIRUN_CLEANUP

Control the environment cleanup after the `mpirun` command.

Syntax

`I_MPI_MPIRUN_CLEANUP=<value>`

Arguments

<value>	Define the option.
`enable | yes | on | 1`	Enable the environment cleanup.
`disable | no | off | 0`	Disable the environment cleanup. This is the default value.

Description

Use this environment variable to define whether to clean up the environment upon the `mpirun` completion. The cleanup includes the removal of the eventual stray service process, temporary files, and so on.

I_MPI_PROCESS_MANAGER

Select a process manager to be used by the `mpirun` command.

Syntax

`I_MPI_PROCESS_MANAGER=<value>`

Arguments

<value>	String value
hydra	Use Hydra* process manager. This is the default value
mpd	Use MPD* process manager

Description

Set this environment variable to select the process manager to be used by the `mpirun` command.

NOTE:

You can run each process manager directly by invoking the `mpiexec` command for MPD* and the `mpiexec.hydra` command for Hydra*.

2.3. Scalable Process Management System (Hydra) Commands

mpiexec.hydra

The `mpiexec.hydra` is a more scalable alternative to the MPD* process manager.

Syntax

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

or

```
mpiexec.hydra <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.out` or `path/name` of the executable file

Description

Use the `mpiexec.hydra` utility to run MPI applications without the MPD ring.

Use the first short command-line syntax to start all MPI processes of the *<executable>* with the single set of arguments. For example, the following command executes `a.out` over the specified processes and hosts :

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

where:

- *<# of processes>* specifies the number of processes on which to run the a.out executable
- *<hostsfile>* specifies a list of hosts on which to run the a.out executable

Use the second long command-line syntax to set different argument sets for different MPI program runs. For example, the following command executes two different binaries with different argument sets:

```
$ mpiexec.hydra -f <hostsfile> -env <VAR1> <VAL1> -n 2 ./a.out : \
-env <VAR2> <VAL2> -n 2 ./b.out
```

NOTE:

If there is no "**.**" in the PATH environment variable on all nodes of the cluster, specify *<executable>* as ./a.out instead of a.out.

NOTE:

You need to distinguish global options from local options. In a command-line syntax, place the local options after the global options.

2.3.1. 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.

-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 [*<profiling_library>*] or -t [*<profiling_library>*]

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

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

(SDK only) -check_mpi [*<checking_library>*]

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

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

-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.hydra 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.

-tv

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

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

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

NOTE:

TotalView* uses rsh by default. If you want to use ssh, set the value of the TVDSVRLAUNCHCMD environment variable to ssh.

NOTE:

The TotalView* debugger can display message queue state of your MPI program. To enable this feature, do the following steps:

1. Run your *<executable>* with the -tv option.

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

2. Answer Yes to the question about stopping the mpiexec.hydra job.

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

-tva *<pid>*

Use this option to attach the TotalView* debugger to an existing Intel® MPI job. Use the mpiexec.hydra process id as *<pid>*. For example:

```
$ mpiexec.hydra -tva <pid>
```

-gdb

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

```
$ mpiexe.hydra -gdb -n <# of processes> <executable>
```

-gdba *<pid>*

Use this option to attach the GNU* debugger to the existing Intel® MPI job. For example:

```
$ mpiexec.hydra -gdba <pid>
```

-nolocal

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

-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.hydra-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.hydra -n 2 -iface ib0 ./a.out
```

See the *I_MPI_HYDRA_IFACE* environment variable for more details.

-demux *<mode>*

Use this option to set polling mode for multiple I/O. The default is `poll`.

Arguments

<spec>	Define the polling mode for multiple I/O
`poll`	Set `poll` as the polling mode. This is the default value.
`select`	Set select as the polling mode.

See the *I_MPI_HYDRA_DEMUX* environment variable for more details.

-enable-x or -disable-x

Use this option to control the Xlib* traffic forwarding. The default value is `-disable-x`, which means the Xlib traffic will not be forwarded.

-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.hydra` 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.

-cleanup

Use this option to create a temporary file containing information about the launched processes. The file name is `mpiexec_${username}_$PPID.log`, where `PPID` is a parent process `PID`. This file is created in the temporary directory selected by the *-tmpdir* option. This file is used by the `mpicleanup` utility. If a job terminates successfully, the `mpiexec.hydra` command automatically removes this file.

See the *I_MPI_HYDRA_CLEANUP* environment variable for more details.

-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.

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. Hydra supports multiple runtime bootstrap servers such as `ssh, rsh, pdsh, fork, persist, slurm, ll, lsf, sge,` or `jmi` to launch the MPI processes. The default bootstrap server is `ssh`. By selecting `slurm`, `ll`, `lsf`, or `sge`, you use the corresponding `srun`, `llspawn.stdio`, `blaunch`, or `qrsh` internal job scheduler utility to launch service processes under the respective selected job scheduler (SLURM*, LoadLeveler*, LSF*, and SGE*).

Arguments

<arg>	String parameter
ssh	Use secure shell. This is the default value
rsh	Use remote shell
pdsh	Use parallel distributed shell
fork	Use fork call
slurm	Use SLURM* srun command
ll	Use LoadLeveler* llspawn.stdio command
lsf	Use LSF blaunch command
sge	Use Oracle Grid Engine* qrsh command
jmi	Use Job Manager Interface (tighter integration)

To enable tighter integration with the SLURM* or PBS Pro* job manager, use the jmi bootstrap server. Tighter integration includes registration of the process identifiers by the respective job managers. This configuration enables better resource accounting by the respective job manager, and better node cleanup upon job termination.

See the *-bootstrap jmi* description and the *I_MPI_HYDRA_BOOTSTRAP* environment variable for details.

-bootstrap-exec <*bootstrap server*>

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

```
$ mpiexec.hydra -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.

-bootstrap jmi

Use this option to enable tight integration with the SLURM* or PBS Pro* job schedulers. Tighter integration is implemented using a particular job scheduler application programming interface or utility. If you specify this option, the default libjmi.so library is loaded. You can overwrite the default library name through the I_MPI_HYDRA_JMI_LIBRARY environment variable.

See the *I_MPI_HYDRA_JMI_LIBRARY* environment variable for more details.

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`ith` 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.

2.3.1.2. Communication Subsystem Options

-rmk *<RMK>*

Use this option to select a resource management kernel to be used. Intel® MPI Library only supports `pbs`.

See the *I_MPI_HYDRA_RMK* environment variable for more details.

2.3.1.3. Other Options

-verbose or -v

Use this option to print debug information from `mpiexec.hydra`, 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.

2.3.2. Local Options

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

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

-umask <*umask*>

Use this option to perform the `umask <umask>` command for the remote `<executable>` file.

2.3.3. 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.

2.3.4. 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.hydra`.

Syntax

`I_MPI_JOB_TIMEOUT=<timeout>`

`I_MPI_MPIEXEC_TIMEOUT=<timeout>`

Deprecated Syntax

`MPIEXEC_TIMEOUT=<timeout>`

Arguments

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

Description

Set this environment variable to make `mpiexec.hydra` 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.hydra` 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.hydra 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.hydra 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
`ssh`	Use secure shell. This is the default value
`rsh`	Use remote shell
`pdsh`	Use parallel distributed shell
`fork`	Use fork call
`slurm`	Use SLURM* `srun` command
`ll`	Use LoadLeveler* `llspawn.stdio` command
`lsf`	Use LSF `blaunch` command
`sge`	Use Oracle Grid Engine* `qrsh` command
`jmi`	Use Job Manager Interface (tighter integration)

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.hydra` 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_RMK

Use the resource management kernel.

Syntax

`I_MPI_HYDRA_RMK=<arg>`

Arguments

<arg>	String parameter
<rmk>	Resource management kernel. The only supported value is `pbs`

Description

Set this environment variable to use the pbs resource management kernel. Intel® MPI Library only supports `pbs`.

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.hydra` 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_JOB_TRACE_LIBS

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

Syntax

`I_MPI_JOB_TRACE_LIBS=<arg>`

Deprecated Syntax

`MPIEXEC_TRACE_LIBS=<arg>`

Arguments

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

Description

Set this environment variable to choose an alternative library for preloading through the `-trace` option.

I_MPI_JOB_CHECK_LIBS

Choose the libraries to preload through the `-check_mpi` option.

Syntax

`I_MPI_JOB_CHECK_LIBS=<arg>`

Arguments

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

Description

Set this environment variable to choose an alternative library for preloading through the `-check_mpi` option.

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.hydra` 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_GDB_REMOTE_SHELL

Set the remote shell command to run GNU* debugger.

Syntax

`I_MPI_HYDRA_GDB_REMOTE_SHELL=<arg>`

Arguments

<arg>	String parameter
ssh	Secure Shell (SSH). This is the default value
rsh	Remote shell (RSH)

Description

Set this environment variable to specify the remote shell command to run the GNU* debugger on the remote machines. You can use this environment variable to specify any shell command that has the same syntax as SSH or RSH.

I_MPI_ HYDRA_JMI_LIBRARY

Define the default setting of the JMI library.

Syntax

I_MPI_HYDRA_JMI_LIBRARY=*<value>*

Arguments

<value>	Define a string value, name, or path to JMI dynamic library
libjmi_slurm.so.1.1 | libjmi_pbs.so.1.0	Set the library name or full path to library name. The default value is libjmi.so

Description

Set this environment variable to define the JMI library to be loaded by the Hydra* processor manager. Set the full path to the library if the path is not mentioned in the LD_LIBRARY_PATH environment variable. If the mpirun command is used, you do not need to set this environment variable. The JMI library is automatically detected and set.

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_HYDRA_DEMUX

Set the demultiplexer (demux) mode.

Syntax

`I_MPI_HYDRA_DEMUX=<arg>`

Arguments

<arg>	String parameter
`poll`	Set `poll` as the multiple I/O demultiplexer (`demux`) mode engine. This is the default value.
`select`	Set `select` as the multiple I/O demultiplexer (`demux`) mode engine

Description

Set this environment variable to specify the multiple I/O `demux` mode engine. The default is `Poll`.

I_MPI_HYDRA_CLEANUP

Control the creation of the default `mpicleanup` input file.

Syntax

`I_MPI_HYDRA_CLEANUP=<value>`

Arguments

<value>	Binary indicator
`enable | yes | on | 1`	Enable the `mpicleanup` input file creation
`disable | no | off | 0`	Disable the `mpicleanup` input file creation. This is the default value

Description

Set the `I_MPI_HYDRA_CLEANUP` environment variable to create the input file for the `mpicleanup` utility.

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.

2.3.5. Cleaning up Utility

mpicleanup

Clean up the environment after an abnormally terminated MPI run under the mpiexec.hydra process manager.

Syntax

```
mpicleanup [ -i <input_file> | -t -f <hostsfile> ] [ -r <rshcmd> ] \
           [ -b <branch_count> ] [-p] [-s | -d] [-h] [-V]
```

or

```
mpicleanup [ --input <input_file> | --total --file <hostsfile> ] \
           [ --rsh <rshcmd> ] [ --branch <branch_count> ] [ --parallel ] \
           [ --silent | --verbose ] [ --help ] [ --version ]
```

Arguments

`-i <input_file>	--input <input_file>`	Specify the input file generated by `mpiexec.hydra`. The default value is `mpiexec_${username}_$PPID.log` located in the temporary directory determined by the values of the `I_MPI_TMPDIR` or `TMPDIR` environment variables, or in the `/tmp` directory.
`-t	--total`	Use the total mode to stop all user processes on the specified machines. This option is not supported for the `root` user.
`-f <hostsfile>	--file <hostsfile>`	Specify the file containing the list of machines to clean up.
`-r <rshcmd>	--rsh <rshcmd>`	Specify the remote shell to use. The default shell is `ssh`.
`-b <branch_count>	--branch <branch_count>`	Define the number of the child processes. The default value is `32`.
`-p	--parallel`	Use the parallel launch mode. This option is only applicable if all hosts are available. Otherwise a part of machines may stay in an undefined state.
`-s	--silent`	Suppress extra output generation.
`-d	--verbose`	Output verbose information.
`-h	--help`	Display a help message.
`-V	--version`	Display Intel® MPI Library version information.

Description

Use this command to clean up the environment after an abnormal MPI job termination.

For example, use the following command to stop processes mentioned in the input file generated by the prior `mpiexec.hydra` invocation:

```
> mpicleanup
```

or

```
> mpicleanup --input /path/to/input.file
```

Use the following command to stop all your user processes on the machines specified in the `hostsfile` file:

```
> mpicleanup --file hostsfile --total
```

2.3.6. Checkpoint-Restart Support

The Checkpoint-Restart feature in Intel® MPI Library is designed to be application transparent. You can access to the Checkpoint-Restart functionality through the MPI process management interface. The Checkpoint-Restart options and environment variables are applicable to the Hydra process manager only.

NOTE:

The Checkpoint-Restart feature requires the OFA* network module. You can choose the OFA network module, for example, with the `I_MPI_FABRICS` environment variable by setting the value to `ofa`, or the `-ib` option.

NOTE:

To enable the Checkpoint-Restart feature, set the following:

- `1` for `I_MPI_OFA_DYNAMIC_QPS` environment variable
- `0` for `I_MPI_OFA_NUM_RDMA_CONNECTIONS` environment variable

NOTE:

Install the Berkeley Lab Checkpoint/Restart* (BLCR) Software for the Checkpoint-Restart function.

2.3.6.1. Global Options

-ckpoint<*switch*>

Arguments

<switch>	Checkpoint switch
`enable | yes | on | 1`	Enables the check point function for the application started
`disable | no | off | 0`	Disables the check point function for the application started. This is the default value

Use this option to enable/disable checkpoints capability. When this capability is disabled, other checkpoint options are ignored.

-ckpoint-interval<*sec*>

Arguments

<sec>	Interval between consecutive checkpoints in seconds

Use this option to turn on timer driven checkpoints. See also Timer Driven Checkpoint. The checkpoints are taken every *<sec>* seconds. If this option is not specified, signal driven checkpoint function may be used. See Explicit Signal Driven Checkpoint for more details.

-ckpoint-preserve<*N*>

Arguments

<N>	Maximal number of checkpoint images kept. The default value is 1

Use this option while running the checkpoint function to keep last *<N>* checkpoints to reduce checkpoint image space. By default, only the last checkpoint is kept.

-restart

Use this option to restart an application from one of the stored checkpoints. `-ckpointlib`, `-ckpoint-prefix` and `-ckpoint-num` options are meaningful for restarting. The executable name may be provided to the process manager, but is ignored. Taking checkpoints is allowed for the restarted application, so `-restart` option may be accompanied with `-ckpoint` and other applicable checkpoint options.

-ckpoint-num<*N*>

Arguments

<N>	Identifier of the checkpoint image to restart an application with. Valid values are any number equal or lower than the last checkpoint number. The default is the last checkpoint number.

Use this option while restarting an application. The checkpoint number *<N>* (counting from `0`) is taken as a restart point. To determine the best choice for this value, examine the checkpoint storage directory setting with the `-ckpoint-prefix` option.

NOTE:

The number of images determined by the `-ckpoint-preserve` option is kept at maximum.

The application will abort with an error message during startup if this checkpoint does not exist. By default, the last checkpoint is selected.

2.3.6.2. Local Options

-ckpointlib<*lib*>

Arguments

<lib>	Checkpoint-Restart system library
`blcr`	Berkeley Lab Checkpoint/Restart* (BLCR) Library. This is the default value

Use this option to select underlying Checkpoint-Restart system library. Only the Berkeley Lab Checkpoint/Restart* (BLCR) Library is supported.

NOTE:

You need to provide the same option when using the checkpoint function, or when restarting the application.

-ckpoint-prefix<*dir*>

Arguments

<dir>	The directory to store checkpoints. The default value is `/tmp`

Use this option to specify a directory to store checkpoints. By default, `/tmp` is used. The directory *<dir>* should be writable, otherwise an error will be raised during process launch, and the application will abort with an error message.

NOTE:

You need to provide the same option when using the checkpoint function, or when restarting the application.

-ckpoint-tmp-prefix<*dir*>

Arguments

<dir>	The directory to store temporary checkpoints. The default value is `/tmp`

Use this option to indicate the directory to store temporary checkpoints. Checkpoints are migrated from `-ckpoint-tmp-prefix` to the directory specified in `-ckpoint-prefix`. The directory *<dir>* should be writable, otherwise the application will abort during startup with an error message. Temporary storage is not used if the option is not set.

-ckpoint-logfile<*file*>

Use this option for monitoring checkpoint activity, the trace is dumped into `<file>`. You should be able to write in `<file>`, otherwise the application will abort during startup with an error message. This is an optional feature.

2.3.6.3. Environment Variables

I_MPI_CKPOINT

Syntax

`I_MPI_CKPOINT=<switch>`

Arguments

<switch>	Checkpoint switch
`enable | yes | on | 1`	Enables the check point function for the application started

`disable | no | off | 0`	Disables the check point function for the application started. This is the default value

Description

Use this variable to turn on taking checkpoints capability. This has the same effect as the `-ckpoint` option. If you have set the `-ckpoint` option, the Hydra process manager sets the `I_MPI_CKPOINT` even if you do not set this environment variable.

I_MPI_CKPOINTLIB

Syntax

`I_MPI_CKPOINTLIB=<lib>`

Arguments

<lib>	Checkpoint-Restart system library
`blcr`	Berkeley Lab Checkpoint/Restart* (BLCR) Library. This is the default value

Description

Use this variable to select underlying Checkpoint-Restart system library. This has the same effect as the `-ckpointlib` option.

I_MPI_CKPOINT_PREFIX

Syntax

`I_MPI_CKPOINT_PREFIX=<dir>`

Arguments

<dir>	The directory to store checkpoints. The default value is `/tmp`

Description

Use this variable to specify a directory to store checkpoints. This has the same effect as the `-ckpoint-prefix` option.

I_MPI_CKPOINT_TMP_PREFIX

Syntax

`I_MPI_CKPOINT_TMP_PREFIX=<dir>`

Arguments

<dir>	The directory to store temporary checkpoints

Description

Use this variable to indicate storage of temporary checkpoints while `-ckpoint-prefix` indicates permanent storage. This has the same effect as the `-ckpoint-tmp-prefix` option.

I_MPI_CKPOINT_INTERVAL

Syntax

`I_MPI_CKPOINT_INTERVAL=<sec>`

Arguments

<sec>	Interval between consecutive checkpoints in seconds

Description

Use this variable to turn on timer driven checkpoints. This has the same effect as the `-ckpoint-interval` option.

I_MPI_CKPOINT_PRESERVE

Syntax

`I_MPI_CKPOINT_PRESERVE=<N>`

Arguments

<N>	Maximal number of checkpoint images kept. The default value is `1`

Description

Use this option while running the checkpoint function to keep last *<N>* checkpoints to reduce checkpoint image space. This has the same effect as the `-ckpoint-preserve` option.

I_MPI_CKPOINT_LOGFILE

Syntax

`I_MPI_CKPOINT_LOGFILE=<file>`

Arguments

<file>	The file keeping the trace for checkpoint activity

Description

Use this option for checkpoint activity monitoring. The trace is dumped into *<file>*. This has the same effect as the `-ckpoint-logfile` option.

I_MPI_CKPOINT_NUM

Syntax

`I_MPI_CKPOINT_NUM=<N>`

Arguments

<N>	Number of checkpoint image to restart an application with

Description

Use this option while restarting application. This has the same effect as the `-ckpoint-num` option.

I_MPI_RESTART

Syntax

`I_MPI_RESTART=<switch>`

Arguments

<switch>	Restart switch
`enable | yes | on | 1`	Enables the restart of the application from one of the stored checkpoints.
`disable | no | off | 0`	Disables the restart of the application. This is the default value.

Description

Use this variable to restart an application from one of the stored checkpoints. Using this variable has the same effect as `-restart` option.

2.3.6.4. Running MPI Applications

The checkpoint-restart feature is available with the Hydra process launcher (`mpiexec.hydra`). The launcher provides two mutually exclusive methods of taking checkpoints:

- By timers
- By explicit signal

You can provide directory paths where checkpoints can be stored temporarily and permanently.

Timer Driven Checkpoint

In the following example, a checkpoint is taken every 3600 seconds (=1hour). The checkpoints are stored in a directory called `ckptdir`. Each node generates one checkpoint which is named by the node number and number of that checkpoint.

```
user@head $ mpiexec.hydra -ckpoint on -ckpoint-prefix /home/user/ckptdir -
ckpoint-interval 3600 -ckpointlib blcr -n 32 -f hosts /home/user/myapp
```

Explicit Signal Driven Checkpoint

In the following example, an application is started and then an explicit signal (`SIGUSR1`) is passed to the application to take a checkpoint. The checkpoints are stored in a directory called `ckptdir`.

```
user@head $ mpiexec.hydra -ckpoint on -ckpoint-prefix /home/user/ckptdir -
ckpointlib blcr –n 32 –f hosts /home/user/myapp

...

user@head $ kill –s SIGUSR1 <PID of mpiexec.hydra>
```

It is necessary and sufficient for you to signal the `mpiexec.hydra` process on node `head`.

Using Local Storage

In the following example, there are two locations for storing checkpoints.

- Temporary location: indicated in the argument to `-ckpoint-tmp-prefix`
- Permanent location: indicated in the argument to `-ckpoint--prefix`

```
user@head $ mpiexec.hydra -ckpoint on -ckpoint-tmp-prefix /ssd/user/ckptdir
-ckpoint-prefix /home/user/ckptdir -ckpointlib blcr -n 32 -f hosts
/home/user/myapp
```

2.3.6.5. Restarting MPI Applications

The following is an example of restarting an application from checkpoint number *<N>*.

```
user@head $ mpiexec.hydra -restart –ckpoint-prefix /home/user/ckptdir -
ckpointlib blcr -ckpoint-num <N> –n 32 –f hosts
```

When restarting, you need to revise the "hosts" file to eliminate any dead or unavailable nodes. Also, providing the executable name is not necessary when restarting because it is already stored in the checkpoint images.

2.3.6.6. Viewing Checkpoint Activity in Log File

The following is an example of launching an MPI job and specifying a checkpoint log file so that you can watch the checkpoint activity.

```
user@head $ mpiexec.hydra -ckpoint on –ckpoint-logfile /home/user/ckpt.log -
ckpoint-tmp-prefix /ssd/user/ckptdir –ckpoint-prefix /home/user/ckptdir -
ckpointlib blcr –n 32 –f hosts /home/user/myapp
```

The following output is a sample log:

```
[Mon Dec 19 13:31:36 2011] cst-linux Checkpoint log initialized (master mpiexec
pid 10687, 48 processes, 6 nodes)

[Mon Dec 19 13:31:36 2011] cst-linux Permanent checkpoint storage:
/mnt/lustre/user

[Mon Dec 19 13:31:36 2011] cst-linux Temporary checkpoint storage: /tmp

[Mon Dec 19 13:32:06 2011] cst-linux Started checkpoint number 0 ...

[Mon Dec 19 13:33:00 2011] cst-linux Finished checkpoint number 0.

[Mon Dec 19 13:33:00 2011] cst-linux Moving checkpoint 0 from /tmp to
/mnt/lustre/user ...

[Mon Dec 19 13:38:00 2011] cst-linux Moved checkpoint 0 from /tmp to
/mnt/lustre/user
```

2.3.6.7. Automatic Cleanup of Previous Checkpoints

Checkpoint images are large; thus, Intel® MPI Library only keeps the last useful checkpoint by default. The following is an example to keep `<N>` previous checkpoints. The flag is `-ckpoint-preserve <N>`. The default value of `-ckpoint-preserve` is `1` (keep only the last checkpoint).

```
user@head $ mpiexec.hydra -ckpoint on -ckpoint-preserve <N> -ckpoint-tmp-prefix
/ssd/user/ckptdir -ckpoint-prefix /home/user/ckptdir -ckpointlib blcr -n 32 -f
hosts /home/user/myapp
```

2.4. Intel® Xeon Phi™ Coprocessor Support

This topic concentrates on the Intel® MPI Library specifics related to the support of the Intel® Xeon Phi™ Coprocessor (codename: Knights Corner) based on Intel® Many Integrated Core Architecture (Intel® MIC Architecture).

2.4.1. Usage Model

To use the Intel MPI Library on Intel® Xeon Phi™ Coprocessor (codename: Knights Corner), ensure that:

- Each host and each Intel® Xeon Phi™ coprocessor must have a unique IP address and symbolic name, which is the same as to classic cluster.
- Password-less access between host and Intel® Xeon Phi™ Coprocessor by ssh is established.

If the connection fails, the following situations might cause the failure:

- The version of Intel® MIC Software Stack you used is out of date. Install a newer version.
- The `iptables` service is running on the host. Stop that service.
- The route is incomplete. Add the missing routes.

Refer to the system administrator and *Intel® MIC Software Stack readme* to configure the settings for the IP connectivity.

When using Intel MPI Library on an Intel Xeon Phi coprocessor, consider Intel Xeon Phi coprocessor card to be another cluster node with a different Intel® architecture. The way that MPI features work for the Intel Xeon Phi coprocessor is similar to the way they work for an Intel® Xeon processor.

For example, MPI libraries may be available on both Intel Xeon processor and Intel Xeon Phi coprocessor through an NFS share that has the same path for Intel Xeon processor host and Intel Xeon Phi coprocessor; MPI tasks may be started from Intel Xeon processor host or Intel Xeon Phi coprocessor, etc.

To build an application for running on the Intel Xeon Phi coprocessor and the host node, go through the following steps:

1. Establish environment settings for the compiler and for the Intel MPI Library:

   ```
   (host)$ . <compiler_installdir>/bin/compilervars.sh em64t

   (host)$ . <mpi_installdir>/em64t/bin/mpivars.sh
   ```

2. Build your application for Intel MIC Architecture, for example:

```
(host)$ mpiicc -mmic test.c -o test_hello.mic
```

3. Build your application for Intel 64 Architecture, for example:

```
(host)$ mpiicc test.c -o test_hello
```

To run an application on the Intel Xeon Phi coprocessor and the host node, go through the following steps:

1. Ensure that NFS is properly set-up between the hosts and the Intel Xeon Phi coprocessor, which is the recommended way for using Intel MPI Library on Intel MIC Architecture.

   For information on how to set up NFS on the Intel Xeon Phi coprocessor, see http://software.intel.com/en-us/articles/intel-mpi-library-for-linux-kb/all/ or http://software.intel.com/mic-developer .

2. Establish environment settings for the Intel MPI Library:

```
(host)$ . <mpi_installdir>/em64t/bin/mpivars.sh
```

3. Launch the executable file from host, for example:

```
(host)$ export I_MPI_MIC=1

(host)$ mpiexec.hydra -n 2 -host <host ID> ./test_hello : -n 2 -host
<coprocessor ID> ./test_hello.mic
```

NOTE:

See Intel® MPI Library for Linux* OS Reference Manual for `-configfile`, `-hostfile` and `-machinefile` options which also can be used.

To run the application on Intel Xeon Phi coprocessor only, follow the steps described above except for the step of building the application for Intel 64 Architecture. Meanwhile, ensure that the hostfile only contains the Intel Xeon Phi coprocessor name.

For more details, see http://software.intel.com/en-us/articles/intel-mpi-library-for-linux-kb/all/ and http://software.intel.com/mic-developer.

2.4.2. Environment Variables

I_MPI_MIC

Syntax

```
I_MPI_MIC=<value>
```

Arguments

<value>	Intel® Xeon Phi™ recognition
`enable | yes | on | 1`	Enable the Intel Xeon Phi coprocessor recognition

`disable | no | off | 0`	Disables the Intel Xeon Phi coprocessor recognition. This is the default value

Description

Set this environment variable to control whether the Intel Xeon processor of the Intel® MPI Library will try to detect and work with the Intel® MIC Architecture components.

If the value of environment variable `I_MPI_MIC` is `enable`, the default value of environment variable `I_MPI_SSHM` is `enable`.

If the value of environment variable `I_MPI_MIC` is `enable`, the default value of environment variable `I_MPI_DAPL_DIRECT_COPY_THRESHOLD` is `4906` and `4096,262144` in case of multiple providers availability.

If the value of environment variable `I_MPI_MIC` is enable, the default value of environment variable `I_MPI_DAPL_BUFFER_SIZE` is `4352`.

NOTE:

This is a provisional variable and is only temporarily introduced, until the architecture detection and other related matters are clarified.

I_MPI_MIC_PREFIX

Syntax

`I_MPI_MIC_PREFIX=<value>`

Arguments

<value>	Specify a string as the prefix of an Intel Xeon Phi coprocessor file name. The default value is an empty string

Description

Set this environment variable to add a prefix to a host executable name to get a corresponding Intel Xeon Phi coprocessor executable file name.

For example, set different locations as the value for the `I_MPI_MIC_PREFIX` environment variable to distinguish Intel MIC Architecture and Intel® 64 Architecture executable files:

```
(host)$ mpiicc test.c -o test_hello

(host)$ mpiicc -mmic test.c -o ./MIC/test_hello

(host)$ export I_MPI_MIC=1

(host)$ export I_MPI_MIC_PREFIX=./MIC/

(host)$ mpiexec.hydra -n 4 -hostfile <hostfile> test_hello
```

In the example, `./test_hello` binary is launched on Intel® 64 Architecture nodes and `./MIC/test_hello` binary is launched on Intel Xeon Phi coprocessor nodes.

I_MPI_MIC_POSTFIX

Syntax

`I_MPI_MIC_POSTFIX=<value>`

Arguments

<value>	Specify a string as the postfix of an Intel Xeon Phi coprocessor file name. The default value is an empty string

Description

Set this environment variable to add a postfix to a host executable name to get a corresponding Intel Xeon Phi coprocessor executable name.

For example, set different names as the value for the `I_MPI_MIC_POSTFIX` environment variable to distinguish Intel Xeon Phi coprocessor and Intel 64 Architecture executable files:

```
(host)$ mpiicc test.c -o test_hello

(host)$ mpiicc -mmic test.c -o test_hello.mic

(host)$ export I_MPI_MIC=1

(host)$ export I_MPI_MIC_POSTFIX=.mic

(host)$ mpiexec.hydra -n 4 -hostfile <hostfile> test_hello
```

In the example, `test_hello` binary is launched on Intel 64 Architecture nodes and `test_hello.mic` binary on Intel Xeon Phi coprocessor nodes.

I_MPI_DAPL_PROVIDER_LIST

Syntax

`I_MPI_DAPL_PROVIDER_LIST=<primary provider>[,<local secondary provider>[,<remote secondary provider>]]`

Arguments

<primary provider>	Provides the best latency and available on all network segments (cross box and within box)
<local secondary provider>	Provides the best bandwidth for local configurations (within box)
<remote secondary provider>	Provides best bandwidth for remote configurations (cross box)

Description

Use this variable to define the DAPL providers to load.

With Intel® Manycore Platform Software Stack (Intel® MPSS), the arguments of `I_MPI_DAPL_PROVIDER_LIST` are set as the following values:

- *<primary provider>*- CCL-direct

- *<local secondary provider>*- IBSCIF
- *<remote secondary provider>*- CCL-proxy

Thus, the setting is `I_MPI_DAPL_PROVIDER_LIST=<CCL-direct>[,<IBSCIF>[,<CCL-proxy>]]`

The following configuration is an example with the default `dat.conf` provided with Intel MPSS:

```
I_MPI_DAPL_PROVIDER_LIST=ofa-v2-mlx4_0-1u,ofa-v2-scif0,ofa-v2-mcm-1
```

You can adjust the threshold for secondary provider through the `I_MPI_DAPL_DIRECT_COPY_THRESHOLD` environment variable (`<secondary provider threshold>`):

```
I_MPI_DAPL_DIRECT_COPY_THRESHOLD=<primary provider direct copy threshold>[,<secondary provider threshold>]
```

`<primary provider direct copy threshold>` has to be lower than `<secondary provider threshold>`.

If the environment variable `I_MPI_DAPL_PROVIDER_LIST` contains a list of values, then the syntax of the following environment variables may be extended by the values related to all corresponding providers.

- `I_MPI_DAPL_DIRECT_COPY_THRESHOLD`
- `I_MPI_DAPL_TRANSLATION_CACHE`
- `I_MPI_DAPL_TRANSLATION_CACHE_AVL_TREE`
- `I_MPI_DAPL_CONN_EVD_SIZE`
- `I_MPI_DAPL_RDMA_RNDV_WRITE`

If only single value is set, this value applies to all providers. In case of mismatch or incorrect values, the default value is used for all providers.

For example:

```
export I_MPI_DAPL_PROVIDER_LIST=ofa-v2-mlx4_0-1,ofa-v2-scif0

export I_MPI_DAPL_TRANSLATION_CACHE=enable,disable
```

This `I_MPI_DAPL_TRANSLATION_CACHE` setting turns on the memory registration cache for the first provider; but turns it off for the second one.

I_MPI_ENV_PREFIX_LIST

Define the prefixes of environment variables for the intended platforms.

Syntax

```
I_MPI_ENV_PREFIX_LIST=[platform:prefix][,...]
```

Argument

`platform`	The intended platform (string).

	Options: `htn,nhm,wsm,snb,ivb` See I_MPI_PLATFORM for the detail options descriptions
`prefix`	A prefix (string) for a name of an environment variable to be used for the intended platform

Description

Set this environment variable to define the prefix of environment variables to be used for the intended platform.

If you specify a prefix in `I_MPI_ENV_PREFIX_LIST` for an environment variable, the prefixed environment variable overrides the respective non-prefixed environment variable on the intended platform.

If you do not specify `I_MPI_ENV_PREFIX_LIST`, environment variables are applied to all platforms.

NOTE:

Use the lower case when you specify the platform names.

Examples

1. `I_MPI_ENV_PREFIX_LIST=platform:prefix`

   `<NAME>=value` is applied to all systems.

   `<prefix>_<NAME>=value` defines `<NAME>=value` for all `<platform>` systems.

2. Assume that some machines are on the Intel® microarchitecture code name Sandy Bridge based platform, and the rest machines are on other architectures based platforms. The environment variable `OMP_NUM_THREADS` value is `3` on all platforms.

   To set `OMP_NUM_THREADS=5` for the ranks on the Intel® microarchitecture code name Sandy Bridge based platform, specify the prefix in `I_MPI_ENV_PREFIX_LIST` for `OMP_NUM_THREADS` with the following configurations:

   ```
   I_MPI_ENV_PREFIX_LIST=snb:<prefix>
   OMP_NUM_THREADS=3
   <prefix>_OMP_NUM_THREADS=5
   ```

2.4.3. Compiler Commands

The following table lists available MPI compiler commands and Intel® Composer XE 2013 for Linux* OS for Intel® MIC Architecture, languages, and application binary interfaces (ABIs) that they support.

Compiler Command	Default Compiler	Supported Language(s)	Supported ABI(s)
`mpiicc`	`icc`	C	64 bit

mpiicpc	icpc	C++	64 bit
mpiifort	ifort	Fortran77/Fortran 95	/64 bit

The compiler commands have the following common features:

- The compiler commands reside in the <installdir>/em64t/bin directory.
- The environment settings should be established by sourcing the <installdir>/em64t/bin/mpivars.sh script.
- To compile a heterogeneous MPI application, compile it twice: one time for Intel® 64 Architecture and another time for Intel® MIC Architecture.
- To distinguish targeted architectures, the scripts parse the underlying compiler options. If they detect the compiler options that target Intel® MIC Architecture (such as -mmic) is currently used by Intel® Composer XE 2013 for Linux* OS for Intel® MIC Architecture, they create an Intel® MIC Compiler executable file. Otherwise, they create an Intel® Xeon processor executable file.
- GNU* Compiler use requires that the compiler be specified with -cc/-cxx/-fc/-f77/-f90" options or through the environment variables described in the Reference Manual. For example:

```
(host)$ mpicc -cc=/usr/linux-k1om-4.7/bin/x86_64-k1om-linux-gcc

-mmic  test.c -o test_hello.mic
```

NOTE:

Use different file names and/or locations to distinguish Intel® MIC Architecture and Intel® 64 Architecture executable files.

2.5. Multipurpose Daemon Commands

mpd

Start Multipurpose daemon* (MPD).

Syntax

```
mpd  [ --help ] [ -V ] [ --version ] [ --host=<host> --port=<portnum> ] \

     [ --noconsole ] [ --trace ] [ --echo ] [ --daemon ] [ --bulletproof ]\

     [ --i fhn <interface/hostname> ] [ --listenport <listenport> ]
```

Arguments

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

`-h <host> -p <portnum>` | `--host=<host> --port=<portnum>`	Specify the host and port to be used for entering an existing ring. The `--host` and `--port` options must be specified together
`-n | --noconsole`	Do not create a console at startup
`-t | --trace`	Print internal MPD trace information
`-e | --echo`	Print a port number at startup to which other `mpds` may connect
`-d | --daemon`	Start `mpd` in daemon mode. By default, the interactive mode is enabled
`--bulletproof`	Turn MPD bulletproofing on
`--ifhn=<interface/hostname>`	Specify `<interface/hostname>` to use for MPD communications
`-l <listenport> | --listenport=<listenport>`	Specify the `mpd` listening port

Description

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

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

is equivalent to

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

If a file named `.mpd.conf` is available in the user's home directory, only the user can have read and write privileges. The file must minimally contain a line with `secretword=<secretword>`. If you want to run MPD as root, create the `mpd.conf` file in the `/etc` directory instead of `.mpd.conf` in the root's home directory to run `mpd` as root. Avoid starting the MPD ring under the root account.

mpdboot

Start mpd ring.

Syntax

```
mpdboot  [ -h ] [ -V ] [ -n <#nodes> ] [ -f <hostsfile> ] [ -r <rshcmd> ] \
         [ -u <user> ] [ -m <mpdcmd> ] [ --loccons ] [ --remcons ] \
```

```
          [ -s ] [ -d ] [ -v ] [ -1 ] [ --ncpus=<ncpus> ] [ -o ] \
          [ -b <maxbranch> ] [ -p ]
```

or

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

Arguments

`-h | --help`	Display a help message
`-V | --version`	Display Intel® MPI Library version information
`-d | --debug`	Print debug information
`-v | --verbose`	Print more information. Show the `<rshcmd>` attempts
`-n <#nodes> | --totalnum=<#nodes>`	Number of nodes in `mpd.hosts` on which daemons are started
`-r <rshcmd> | --rsh=<rshcmd>`	Specify remote shell to start daemons and jobs. The default value is ssh
`-f <hostsfile> | --file=<hostsfile>`	Path/name of the file that has the list of machine names on which the daemons are started
`-1`	Enable starting multiple `mpd` per machine
`-m <mpdcmd> | --mpd=<mpdcms>`	Specify the full path name of the `mpd` on the remote hosts
`-s | --shell`	Specify the shell
`-u <user> | --user=<user>`	Specify the user
`--loccons`	Do not create local MPD consoles
`--remcons`	Do not create remote MPD consoles
`--ncpus=<ncpus>`	Indicate how many processors to use on the local machine (other nodes are listed in the hosts file)

`-o	--ordered`	Start all the `mpd` daemons in the order as specified in the `mpd.hosts` file
`-b <maxbranch> |` `--maxbranch=<maxbranch>`	Use this option to indicate the maximum number of the `mpd` daemons to enter the `mpd` ring under another. This helps to control the parallelism of the `mpd` ring start. The default value is four
`-p |--parallel-startup`	Use this option to allow parallel fast starting of `mpd` daemons under one local root. No daemon checking is performed. This option also supports shells which do not transfer the output from the remote commands

Description

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

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

NOTE:

The `mpdboot` command spawns an MPD daemon on the host machine, even if the machine name is not listed in the `mpd.hosts` file.

mpdexit

Shut down a single `mpd` daemon.

Syntax

```
mpdexit [ --help ] [ -V ] [--version ] <mpdid>
```

Arguments

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

Description

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

mpdallexit

Shut down all `mpd` daemons on all nodes.

Syntax

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

Arguments

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

Description

Use this command to shut down all MPD rings you own.

mpdcleanup

Clean up the environment after an `mpd` crash.

Syntax

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

or

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

Arguments

`-h | --help`	Display a help message
`-V | --version`	Display Intel® MPI Library version information
`-f <hostsfile> | --file=<hostsfile>`	Specify the file containing a list of machines to clean up
`-r <rshcmd> | --rsh=<rshcmd>`	Specify the remote shell to use
`-u <user> | --user=<user>`	Specify the user
`-c <cleancmd> | --clean=<cleancmd>`	Specify the command to use for removing the UNIX* socket. The default command is `/bin/rm -f`

`-a	--all`	Kill all `mpd` daemons related to the current settings of the `I_MPI_JOB_CONTEXT` environment variable on all hosts specified in *<hostsfile>*

Description

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

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

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

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

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

mpdtrace

Determine whether `mpd` is running.

Syntax

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

Arguments

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

Description

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

mpdcheck

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

Syntax

```
mpdcheck [ -v ] [ -l ] [ -h ] [ --help ] [ -V ] [ --version ]

mpdcheck -pc [ -v ] [ -l]

mpdcheck -f <host_file> [ -ssh ] [ -v ] [ -l]

mpdcheck -s [ -v ] [ -l]

mpdcheck -c < server_host> <server_port> [ -v ] [ -l]
```

Arguments

`-h | --help`	Display a help message
`-V | --version`	Display Intel® MPI Library version information
`-pc`	Print configuration information about a local host
`-f <host_file>`	Print information about the hosts listed in `<host_file>`
`-ssh`	Invoke testing of `ssh` on each remote host. Use in conjunction with the `-f` option
`-s`	Run `mpdcheck` as a server on one host
`-c <server_host> <server_port>`	Run `mpdcheck` as a client on the current or different host. Connect to the `<server_host> <server_port>`
`-l`	Print diagnostic messages in extended format
`-v`	Print the actions that `mpdcheck` is performing

Description

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

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

mpdringtest

Test the MPD ring.

Syntax

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

Arguments

`--help`	Display a help message
`-V | --version`	Display Intel® MPI Library version information
`<number of loops>`	Number of loops

Description

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

mpdlistjobs

Syntax

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

or

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

Arguments

`-h | --help`	Display a help message
`-V | --version`	Display Intel® MPI Library version information
`-u <username>|` `--user=<username>`	List jobs of a particular user
`-a <jobalias> | --alias=<jobalias>`	List information about the particular job specified by `<jobalias>`
`-j <jobid> |` `--jobid=<jobid>`	List information about the particular job specified by `<jobid>`

Description

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

mpdsigjob

Apply a signal to a process running an application.

Syntax

```
mpdsigjob [ --help ] [ -V ] [ --version ] <sigtype> \
[-j <jobid> | -a <jobalias> ] [-s | -g ]
```

Arguments

`--help`	Display a help message
`-V | --version`	Display Intel® MPI Library version information
`<sigtype>`	Specify the signal type to send. Valid options are `-j` or `-a`.
`-a <jobalias>`	Send a signal to the job specified by `<jobalias>`
`-j <jobid>`	Send a signal to the job specified by `<jobid>`
`-s`	Deliver a signal to a single user process

`-g`	Deliver a signal to a group of processes. This is the default behavior.

Description

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

mpdkilljob

Terminate a job.

Syntax

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

Arguments

`--help`	Display a help message
`-V | --version`	Display Intel® MPI Library version information
`<jobnum>`	Kill the job specified by *`<jobnum>`*
`-a` *`<jobalias>`*	Kill the job specified by *`<jobalias>`*

Description

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

mpdhelp

Print brief help concerning MPD commands.

Syntax

```
mpdhelp [ -V ] [ --version ]
```

Arguments

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

Description

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

2.5.1. Job Startup Commands

mpiexec

Syntax

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

or

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

or

```
mpiexec -configfile <file>
```

Arguments

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

Description

Use the first command-line syntax to 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 -n <# of processes> ./a.out
```

Use the second command-line syntax to start several MPI programs or the same MPI program with different argument sets. For example, the following command runs each given executable on a different host:

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

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

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

NOTE:

If there is no "**.**" in the PATH environment variable on all nodes in the cluster, specify *<executable>* as ./a.out rather than a.out.

2.5.1.1. Extended Device Control Options

Use these options to select a specific fabric combination.

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

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

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

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

-rdma

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

-RDMA

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

-dapl

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

-ib

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

-IB

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

-tmi

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

-mx

Use this option to select Myrinet MX* network fabric for inter-node communication. The application attempts to use Myrinet MX* 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_TMI_PROVIDER mx -genv I_MPI_DAPL_PROVIDER mx -genv I_MPI_FALLBACK 1` setting.

-MX

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

-psm

Use this option to select Intel® True Scale Fabric for inter-node communication. The application attempts to use Intel True Scale 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_TMI_PROVIDER psm -genv I_MPI_FALLBACK 1` setting.

-PSM

Use this option to select Intel True Scale Fabric for inter-node communication. The application fails if no such fabric is found. This option is equivalent to the `-genv I_MPI_FABRICS_LIST tmi -genv I_MPI_TMI_PROVIDER psm -genv I_MPI_FALLBACK 0` setting.

2.5.1.2. Global Options

-version or -V

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

-h or -help or --help

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

-tune [<*arg* >]

where:

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

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

If *<arg>* is not specified, the best-fit tuning options are selected for the given configuration. The default location of the configuration file is *<installdir>*/*<arch>*/`etc` directory. You can override this default location by explicitly stating: *<arg>*=*<dir_name>*. The provided configuration file is used if you set *<arg>*=*<configuration_file>*.

See *Automatic Tuning Utility* for more details.

-nolocal

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

-perhost *<# of processes>*

Use this option to place the indicated number of consecutive MPI processes on every host in a group using round robin scheduling. The total number of processes to start is controlled by the `-n` option.

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

To change this default behavior, set the number of processes per host by using the `-perhost` option, and set the total number of processes by using the `-n` option. See *Local Options* for details. The first *<# of processes>* indicated by the `-perhost` option is executed on the first host; the next *<# of processes>* is executed on the next host, and so on.

See also the *I_MPI_PERHOST* environment variable.

-rr

Use this option to execute consecutive MPI processes on different hosts using round robin scheduling. This option is equivalent to `-perhost 1`.

-grr *<# of processes>*

Use this option to place the indicated number of consecutive MPI processes on every host using round robin scheduling. This option is equivalent to `-perhost` *<# of processes>*.

-ppn *<# of processes>*

Use this option to place the indicated number of consecutive MPI processes on every host using round robin scheduling. This option is equivalent to `-perhost` *<# of processes>*.

-machinefile *<machine file>*

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

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>*].

NOTE:

The -machinefile, -ppn, -rr, and -perhost options are intended for process distribution. If used simultaneously, -machinefile takes precedence.

-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.out and b.out using the shm:dapl fabric over host1 and host2 respectively:

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

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

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

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

NOTE:

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

-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,.

-genv <ENVVAR> <value>

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

-genvuser

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

-genvall

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

-genvnone

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

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

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

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

NOTE:

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

(SDK only) -check_mpi [<checking_library>]

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

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

NOTE:

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

-tv

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

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

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

NOTE:

Ensure the environment variable `TVDSVRLAUNCHCMD=ssh` because the TotalView* uses `rsh` by default.

NOTE:

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

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

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

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

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

-tva <*jobid*>

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

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

-tvsu

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

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

NOTE:

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

-gdb

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

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

-gdba *<jobid>*

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

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

-a *<alias>*

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

-ordered-output

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

NOTE:

For this option to work, the last line output by each process must end with the end-of-line (\n) character. Otherwise the application may stop responding.

-m

Use this option to merge output lines.

-l

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

-s *<spec>*

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

Arguments

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

-noconf

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

-ifhn *<interface/hostname>*

Use this option to specify the network interface for communication with the local MPD daemon; where *<interface/hostname>* is an IP address or a hostname associated with the alternative network interface.

-ecfn *<filename>*

Use this option to list XML exit codes to the file *<filename>*.

2.5.1.3. 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.

-envuser

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

-envall

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

-envnone

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

-envlist *<list of env var names>*

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

-envexcl *<list of env var names>*

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

-host <*nodename*>

Use this option to specify a particular <*nodename*> on which to run MPI processes in the current argument set. For example, the following command runs the executable `a.out` on host `host1` only:

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

-path <*directory*>

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

-wdir <*directory*>

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

-umask <*umask*>

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

2.5.1.4. Configuration Files

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

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

System-wide <*installdir*>`/etc/mpiexec.conf`. The default location of the configuration file is the <*installdir*>/<*arch*>`/etc`.

User-specific `$HOME/.mpiexec.conf`

Session-specific `$PWD/mpiexec.conf`

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

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

```
-genv I_MPI_FABRICS <fabric>
```

2.5.1.5. 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 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.

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

Each printed line has the following format:

`[<identifier>] <message>`

where

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

For example, the following command:

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

may produce the following output:

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

while the command

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

or

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

may produce the following output:

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

NOTE:

Compiling with `mpiicc -g` adds a considerable amount of printed debug information.

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_PERHOST

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

Syntax

I_MPI_PERHOST=<*value*>

Arguments

<*value*>	Define the default process layout
<*n*> > 0	<*n*> processes per node
all	All logical CPUs on a node
allcores	All cores (physical CPUs) on a node

Description

Set this environment variable to define the default setting for the -perhost option. If -perhost is explicitly called in the command line, the I_MPI_PERHOST environment variable has no effect. The -perhost option assumes the value of the I_MPI_PERHOST environment variable if this environment variable is defined.

NOTE:

When I_MPI_PERHOST is defined together with mpiexec -host option, I_MPI_PERHOST is ignored.

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.

(SDK only) I_MPI_JOB_TRACE_LIBS

(MPIEXEC_TRACE_LIBS)

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

Syntax

`I_MPI_JOB_TRACE_LIBS=<arg>`

Deprecated Syntax

`MPIEXEC_TRACE_LIBS=<arg>`

Arguments

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

Description

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

(SDK only) I_MPI_JOB_CHECK_LIBS

Choose the libraries to preload through the `-check_mpi` option.

Syntax

`I_MPI_JOB_CHECK_LIBS=<arg>`

Arguments

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

Description

Set this environment variable to choose an alternative library for preloading by the `-check_mpi` option.

I_MPI_JOB_STARTUP_TIMEOUT

Set the `mpiexec` job startup timeout.

Syntax

`I_MPI_JOB_STARTUP_TIMEOUT=<timeout>`

Arguments

`<timeout>`	Define `mpiexec` job startup timeout period in seconds
`<n>` >= 0	The default timeout value is 20 seconds

Description

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

NOTE:

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

I_MPI_JOB_TIMEOUT

(MPIEXEC_TIMEOUT)

Set the `mpiexec` timeout.

Syntax

`I_MPI_JOB_TIMEOUT=<timeout>`

Deprecated Syntax

`MPIEXEC_TIMEOUT=<timeout>`

Arguments

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

Description

Set this 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 for setting the *<timeout>* value. Those options are used only for passing environment variables to the MPI process environment.

I_MPI_JOB_TIMEOUT_SIGNAL

(MPIEXEC_TIMEOUT_SIGNAL)

Define a signal to be used when a job is terminated 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 for task termination upon the timeout period specified by the environment variable `I_MPI_JOB_TIMEOUT`. If you set a signal number unsupported by the system,, `mpiexec` prints a warning message and continues 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`) that may be received by the MPD daemons. If signal propagation is enabled, the received signal is sent to all processes of the MPI job. If signal propagation is disabled, all processes of the MPI job are stopped with the default signal `9` (`SIGKILL`).

I_MPI_OUTPUT_CHUNK_SIZE

Set the size of the `stdout/stderr` output buffer.

Syntax

`I_MPI_OUTPUT_CHUNK_SIZE=<size>`

Arguments

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

Description

Set this environment variable to increase the size of the buffer used to intercept the standard output and standard error streams from the processes. If the `<size>` value is not greater than zero, the environment variable setting is ignored and a warning message is displayed.

Use this setting for applications that create a significant amount of output from different processes. With the `-ordered-output` `mpiexec` option, this setting helps to prevent the output from garbling.

NOTE:

Set the `I_MPI_OUTPUT_CHUNK_SIZE` environment variable in the shell environment before executing the `mpiexec` command. Do not use the `-genv` or `-env` options for setting the `<size>`

value. Those options are used only for passing environment variables to the MPI process environment.

I_MPI_PMI_EXTENSIONS

Turn on/off the use of the Intel® MPI Library Process Management Interface (PMI) extensions.

Syntax

`I_MPI_PMI_EXTENSIONS=<arg>`

Arguments

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

Description

The Intel® MPI Library automatically detects if your process manager supports the PMI extensions. If supported, the extensions substantially decrease task startup time. Set `I_MPI_PMI_EXTENSIONS` to `disable` if your process manager does not support these extensions.

I_MPI_JOB_FAST_STARTUP

(I_MPI_PMI_FAST_STARTUP)

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

Syntax

`I_MPI_JOB_FAST_STARTUP=<arg>`

Deprecated Syntax

`I_MPI_PMI_FAST_STARTUP=<arg>`

Arguments

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

Description

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

TOTALVIEW*

Select a particular TotalView* executable file to use.

Syntax

`TOTALVIEW=<path>`

Arguments

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

Description

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

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.

2.5.2. Configuration Files

$HOME/.mpd.conf

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

Syntax

The file has a single line:

`secretword=<mpd password>`

or

`MPD_SECRETWORD=<mpd password>`

Description

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

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

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

NOTE:

`MPD_SECRETWORD` is a synonym for `secretword`.

---

mpd.hosts

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

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

Syntax

The format of the `mpd.hosts` file is a list of node names, one name per line. Blank lines and the portions of any lines that follow a `#` character are ignored.

2.5.3. Environment Variables

I_MPI_JOB_CONFIG_FILE

(I_MPI_MPD_CONF)

Set the path/name of the `mpd` configuration file.

Syntax

`I_MPI_JOB_CONFIG_FILE=<path/name>`

Deprecated Syntax

`I_MPI_MPD_CONF=<path/name>`

Arguments

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

Description

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

I_MPI_JOB_CONTEXT

(MPD_CON_EXT)

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

Syntax

`I_MPI_JOB_CONTEXT=`*`<tag>`*

Deprecated Syntax

`MPD_CON_EXT=`*`<tag>`*

Arguments

<tag>	Unique MPD identifier

Description

Set this environment variable to different unique values to allow several MPD rings to co-exist. Each MPD ring is associated with a separate `I_MPI_JOB_CONTEXT` value. Once this environment variable is set, you can start one MPD ring and work with it without affecting other available MPD rings. Set the appropriate `I_MPI_JOB_CONTEXT` value to work with a particular MPD ring. See *Simplified Job Startup Command* to learn about an easier way to run several Intel® MPI Library jobs at once.

I_MPI_JOB_TAGGED_PORT_OUTPUT

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

Syntax

`I_MPI_JOB_TAGGED_PORT_OUTPUT=`*`<arg>`*

Arguments

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

Description

The tagged output format works at the `mpdboot` stage and prevents a failure during startup due to unexpected output from a remote shell like `ssh`. `mpdboot` sets this environment variable to `1`

automatically. Set `I_MPI_JOB_TAGGED_PORT_OUTPUT` to `disable` if you do not want to use the new format.

I_MPI_MPD_CHECK_PYTHON

Toggle the Python* versions check at the MPD ring startup stage.

Syntax

`I_MPI_MPD_CHECK_PYTHON=<arg>`

Arguments

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

Description

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

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

I_MPI_MPD_RSH

Set the remote shell to start `mpd` daemons.

Syntax

`I_MPI_MPD_RSH =<arg>`

Arguments

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

Description

Set this environment variable to define the default setting for the `--rsh` *mpdboot* option. If `--rsh` is explicitly called in the command line, the `I_MPI_MPD_RSH` environment variable has no effect. If the `--rsh` option is not explicitly defined, it assumes the value of the `I_MPI_MPD_RSH` environment variable.

I_MPI_MPD_TMPDIR

TMPDIR

Set a temporary directory for the MPD subsystem.

Syntax

`I_MPI_MPD_TMPDIR=<arg>`

`TMPDIR=<arg>`

Arguments

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

Description

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

NOTE:

The `mpd2.console_*` file path length is limited in some operating systems. If you get the following diagnostic message: `socket.error: AF_UNIX path too long`. you need to decrease the length of the *<directory name>* string to avoid this issue.

NOTE:

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

I_MPI_MPD_CLEAN_LOG

Control the removal of the log file upon MPD termination.

Syntax

`I_MPI_MPD_CLEAN_LOG=<value>`

Arguments

<value>	Define the value
`enable | yes | on | 1`	Remove the log file
`disable | no | off | 0`	Keep the log file. This is the default value

Description

Set this environment variable to define the `mpdallexit` behavior. If you enable this environment variable, the `mpdallexit` removes the log file created during its execution. If you disable this environment variable, the `mpdallexit` keeps the log file.

2.6. 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 IA-32 and Intel® 64 architectures.

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 family 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. Tuning Reference

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

3.1. 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]]] ] [ -er ] \
        [ -hf <hostsfile> ] [ -h ] [ -hr {min:max|min:|:max} ] \
        [ -i <count> ] [ -mr {min:max|min:|:max}] [ -od <outputdir> ] \
        [ -odr <outputdir> ] [ -r <rshcmd>] [ -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]]] ] [ --existing-ring ] \
        [ --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> ] [ --rsh <rshcmd> ] \
```

```
[ --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 `$PWD/app.conf`.

`-t \"<test_cmd_line>\" |`
 	
`--test \"<test_cmd_line>\"`
	Replace the default Intel® MPI Benchmarks by the indicated benchmarking program in the cluster-specific mode. Quote the full command line as shown 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]]] |`
 	
`--device-list [d1[,d2,…[,dN]]]`
	Select the device(s) you want to tune. Any previously set fabrics are ignored.. By default, use all devices listed in the `<installdir>/<arch>/etc/devices.xml` file.

`-fl [f1[,f2...[,fN]]] |`	Select the fabric(s) you want to tune. Any previously set

--fabric-list *[f1[,f2…[,fN]]]*	devices are ignored. By default, use all fabrics listed in the *<installdir>/<arch>*/etc/fabrics.xml file.
-er | --existing-ring	Use an existing MPD ring. By default, a new MPD ring is created. This argument is applicable only if I_MPI_PROCESS_MANAGER is set to mpd.
-hf *<hostsfile>* | --host-file *<hostsfile>*	Specify an alternative host file name. By default, use the $PWD/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, $PWD is used as the default value in the cluster-specific mode.
-r *<rshcmd>* | --rsh *<rshcmd>*	Specify the remote shell used to start daemons (as applicable) and jobs. The default value is ssh.
-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]* | `--session-file` *[file-path]*	Continue the tuning process starting from the state saved 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. Use `$PWD/mpitunertemp` 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 `mpitine` 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 `mpirun` 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.

3.1.1. 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 `mpirun` 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

```
$ mpirun -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
```

3.1.1.1. Replacing the Default Benchmark

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

3.1.2. 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 \"mpirun -n 32 ./myprog\" -of ./myprog.conf
```

Use the optimal recorded values for your application

```
$ mpirun -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.

3.1.3. Tuning Utility Output

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

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

The Intel MPI Library tuning utility ignores 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`.

3.2. 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.

3.2.1. 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 of this CPU bit in the kernel affinity bit-mask. Use the `cpuinfo` utility, provided with your Intel MPI Library installation, or the `cat /proc/cpuinfo` command 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 below for one possible processor numbering scenario with two sockets, four cores (two cores per socket), and eight logical processors (two processors per core).

NOTE:

Logical and topological enumerations are not the same.

Table 3.2-1 Logical Enumeration

0	4	1	5	2	6	3	7

Table 3.2-2 Hierarchical Levels

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

Table 3.2-3 Topological Enumeration

0	1	2	3	4	5	6	7

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

3.2.2. Environment Variables

I_MPI_PIN

Turn on/off process pinning.

Syntax

```
I_MPI_PIN=<arg>
```

Arguments

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

Description

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

I_MPI_PIN_MODE

Choose the pinning method.

Syntax

```
I_MPI_PIN_MODE=<pinmode>
```

Arguments

<pinmode>	Choose the CPU pinning mode
`mpd|pm`	Pin processes inside the process manager involved (Multipurpose Daemon*-MPD or Hydra*). This is the

	default value
`lib`	Pin processes inside the Intel MPI Library

Description

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

Set the `I_MPI_PIN_MODE` environment variable to `mpd|pm` to make the `mpd` daemon or the Hydra process launcher pin processes through system specific means, if they are available. The pinning is done before the MPI process launch. Therefore, it is possible to co-locate the process CPU and memory in this case. This pinning method has an advantage over a system with Non-Uniform Memory Architecture (NUMA) like SGI* Altix*. Under NUMA, a processor can access its own local memory faster than non-local memory.

Set the `I_MPI_PIN_MODE` environment variable to `lib` to make the Intel® MPI Library pin the processes. This mode does not offer the capability to co-locate the CPU and memory for a process.

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>]`

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.

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

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.

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>*.

<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.

<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.

<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.
<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.

<n>	Specify an explicit value of the corresponding parameters previously mentioned. *<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.

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

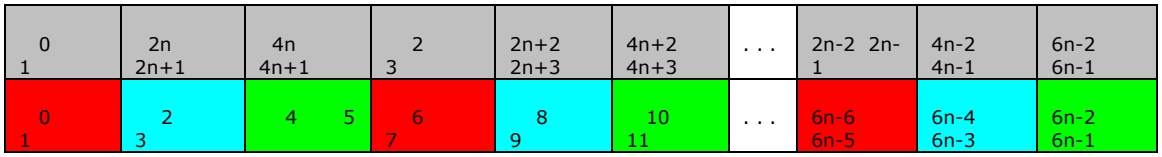
- A)

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

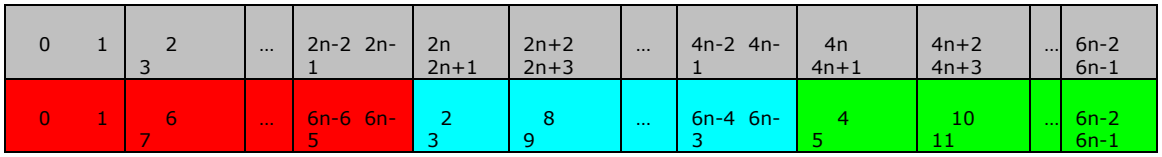
- B)

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

- C)



- D)



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

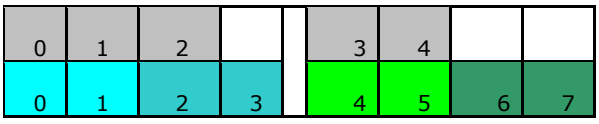
Legend:

gray - MPI processes

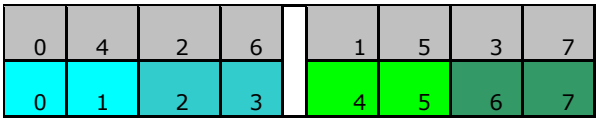
cyan - 1st socket processors

green - 2nd socket processors

Same color defines a processor pair sharing a cache



`bunch` scenario for 5 processes



`scatter` scenario for full loading

Examples

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

```
$ mpirun -genv I_MPI_PIN_PROCESSOR_LIST 0,3 \

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

To pin the processes to different CPUs on each node individually (CPU0 and CPU3 on host1 and CPU0, CPU1 and CPU3 on host2), use the following command:

```
$ mpirun -host host1 -env I_MPI_PIN_PROCESSOR_LIST 0,3 \

-n <# of processes> <executable> : \

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

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

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

```
$ mpirun -genv I_MPI_DEBUG 4 -m -host host1 \

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

-host host2 -env I_MPI_PIN_PROCESSOR_LIST 1,2,3 \ -n <# of processes>
<executable>
```

NOTE:

If 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_PROCESSOR_EXCLUDE_LIST

Define a subset of logical processors.

Syntax

```
I_MPI_PIN_PROCESSOR_EXCLUDE_LIST=<proclist>
```

Arguments

<proclist>	A comma-separated list of logical processor numbers and/or ranges of processors.
<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>*.

Description

Set this environment variable to define the logical processors Intel® MPI Library do not use for pining capability on the intended hosts. Logical processors are numbered as in `/proc/cpuinfo`.

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.

I_MPI_PIN_RESPECT_CPUSET

Respect the process affinity mask.

Syntax

I_MPI_PIN_RESPECT_CPUSET=*<value>*

Arguments

<value>	Binary indicator
enable | yes | on | 1	Respect the process affinity mask. This is the default value

`disable | no | off | 0`	Do not respect the process affinity mask

Description

If `I_MPI_PIN_RESPECT_CPUSET=enable`, the Hydra process launcher uses its process affinity mask on each intended host to determine logical processors for applying Intel MPI Library pinning capability.

If `I_MPI_PIN_RESPECT_CPUSET=disable`, the Hydra process launcher does not use its process affinity mask to determine logical processors for applying Intel MPI Library pinning capability.

I_MPI_PIN_RESPECT_HCA

In the presence of Infiniband architecture* host channel adapter (IBA* HCA), adjust the pinning according to the location of IBA HCA.

Syntax

`I_MPI_PIN_RESPECT_HCA=<value>`

Arguments

<value>	Binary indicator
`enable | yes | on | 1`	Use the location of IBA HCA (if available). This is the default value
`disable | no | off | 0`	Do not use the location of IBA HCA

Description

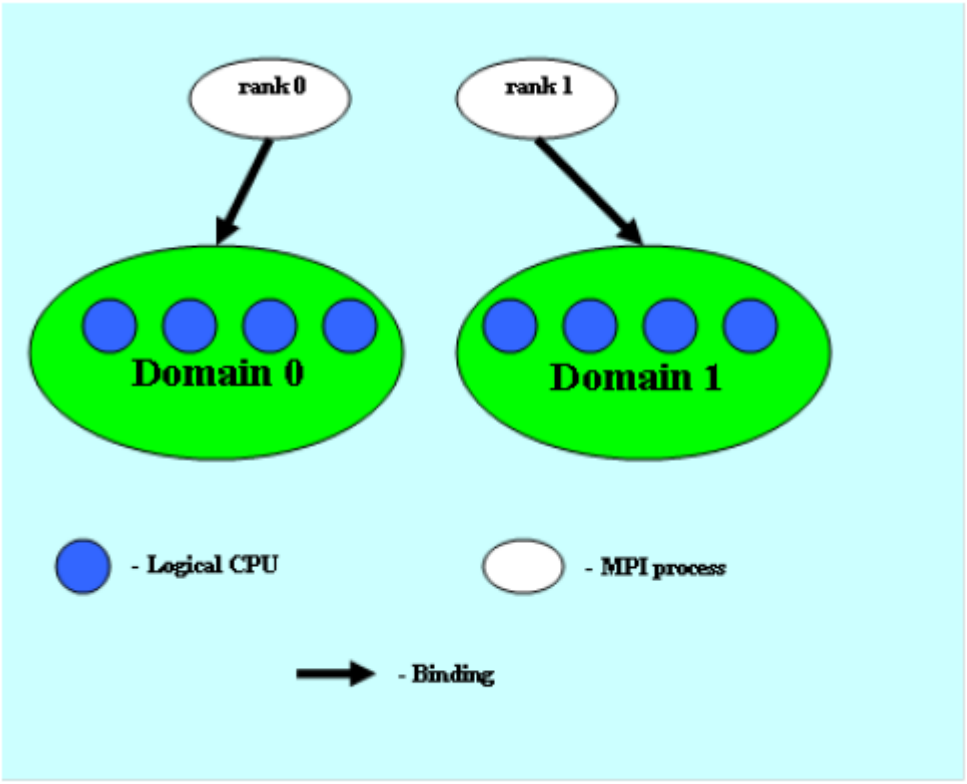
If `I_MPI_PIN_RESPECT_HCA=enable`, the Hydra process launcher uses the location of IBA HCA on each intended host for applying Intel MPI Library pinning capability.

If `I_MPI_PIN_RESPECT_CPUSET=disable`, the Hydra process launcher does not use the location of IBA HCA on each intended host for applying Intel MPI Library pinning capability.

3.2.3. Interoperability with OpenMP* API

I_MPI_PIN_DOMAIN

The Intel® MPI Library provides an additional environment variable to control process pinning for hybrid Intel MPI Library 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 below.



Picture 3.2-1 Domain Example

Each MPI process can create a number of children threads for running within the corresponding domain. The process threads can freely migrate from one logical processor to another within the particular domain.

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

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)
`[m1,...,mn]`	Each `mi` number defines one separate 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:

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.

NOTE:

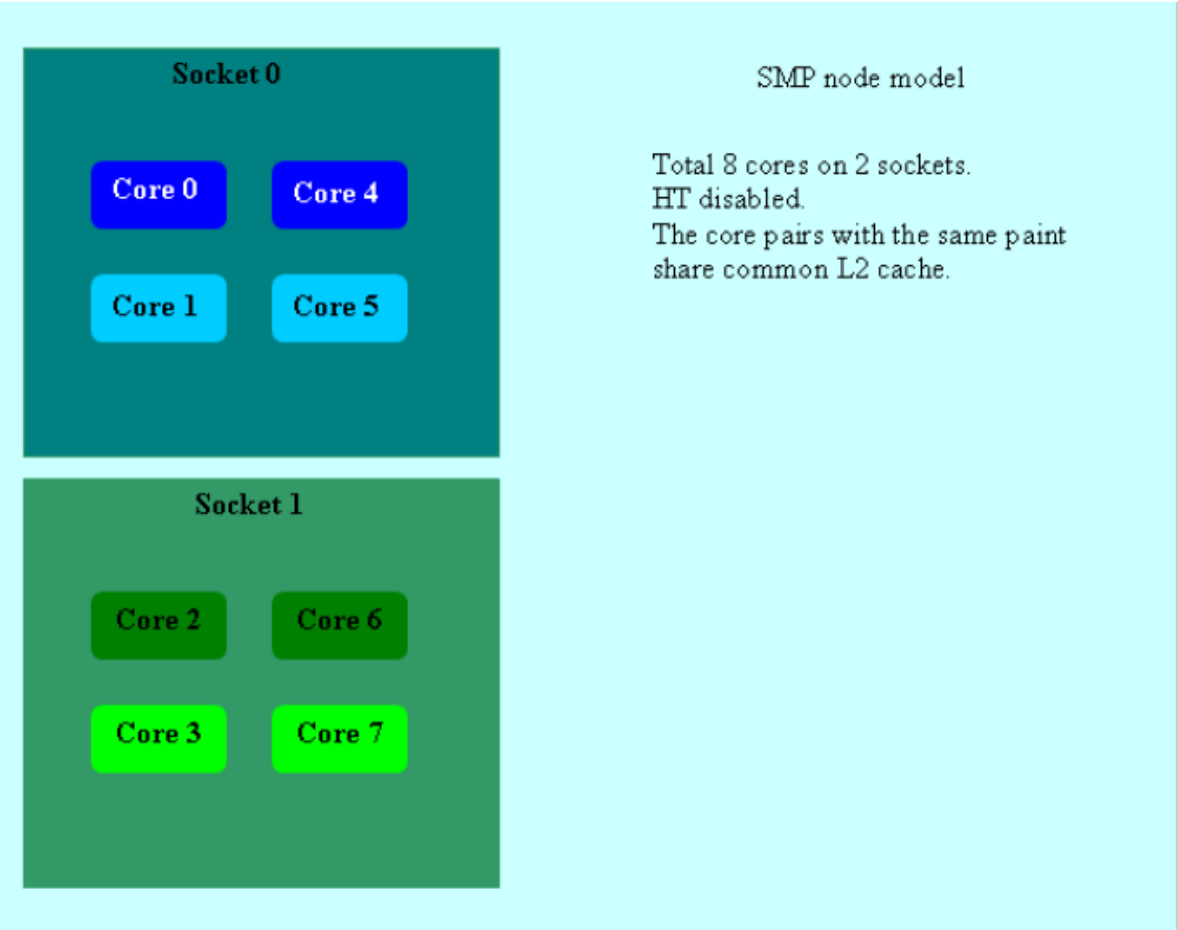
The following configurations are effectively the same as if pinning is not applied:

- If `I_MPI_PIN_DOMAIN=auto` and a single process is running on a node (for example, due to `I_MPI_PERHOST=1`)
- `I_MPI_PIN_DOMAIN=node`

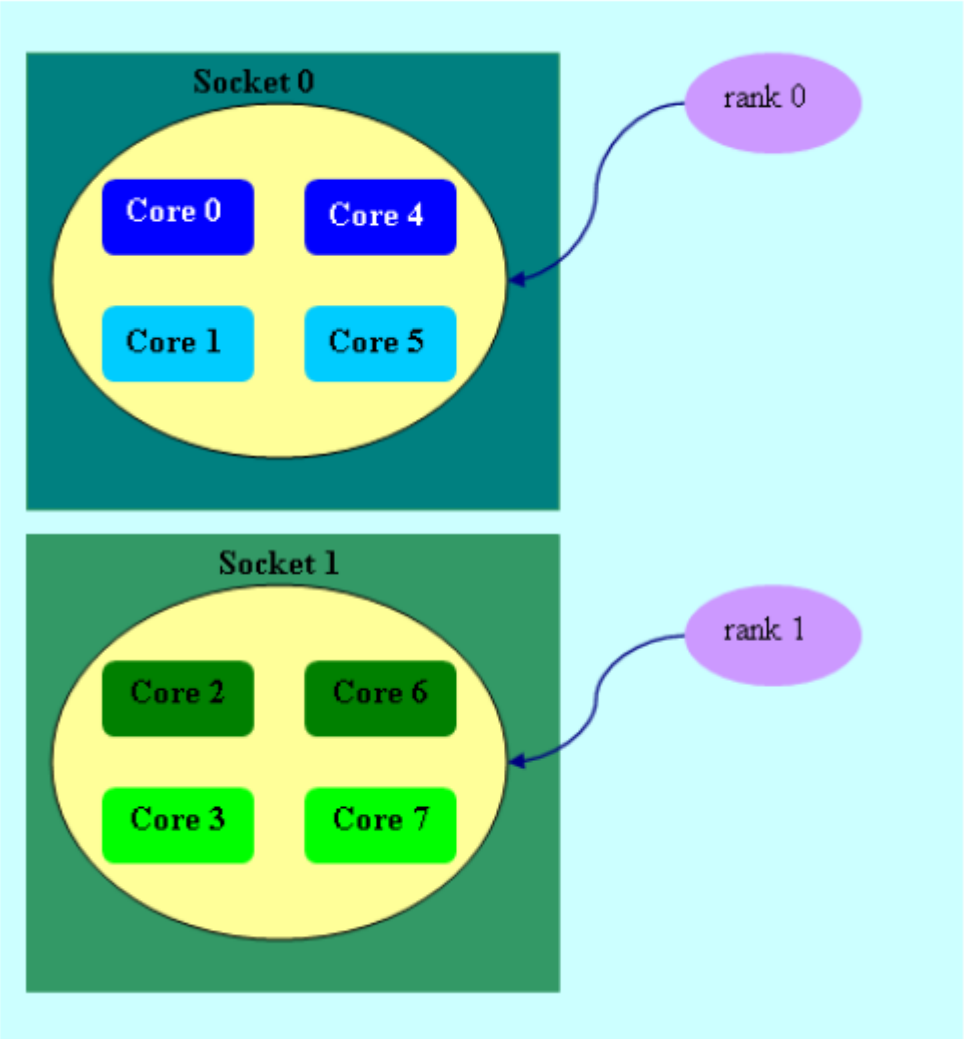
If you do not want the process to be migrated between sockets on a multi-socket platform, specify the domain size as `I_MPI_PIN_DOMAIN=socket` or smaller.

You can also use `I_MPI_PIN_PROCESSOR_LIST`, which produces a single-cpu process affinity mask for each rank (the affinity mask is supposed to be automatically adjusted in presence of IBA* HCA).

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

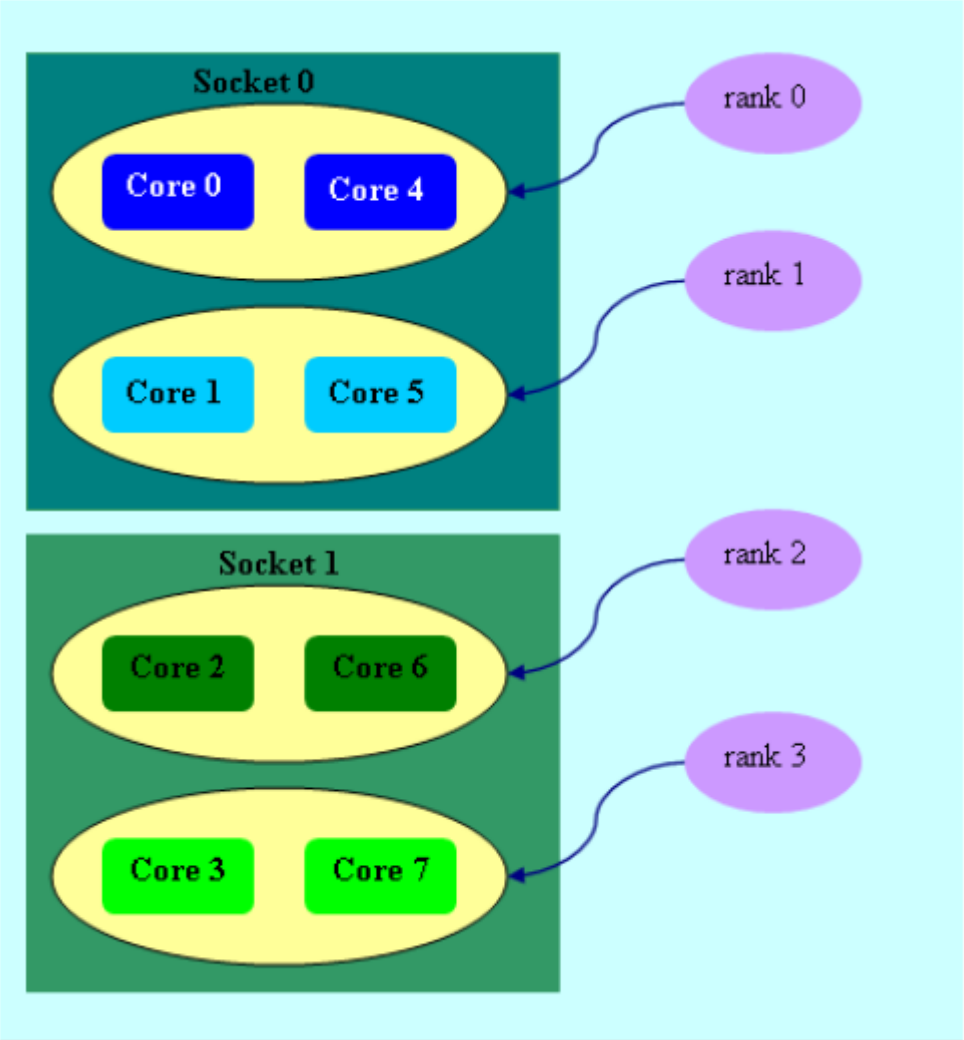


Picture 3.2-2 Model of a Node



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

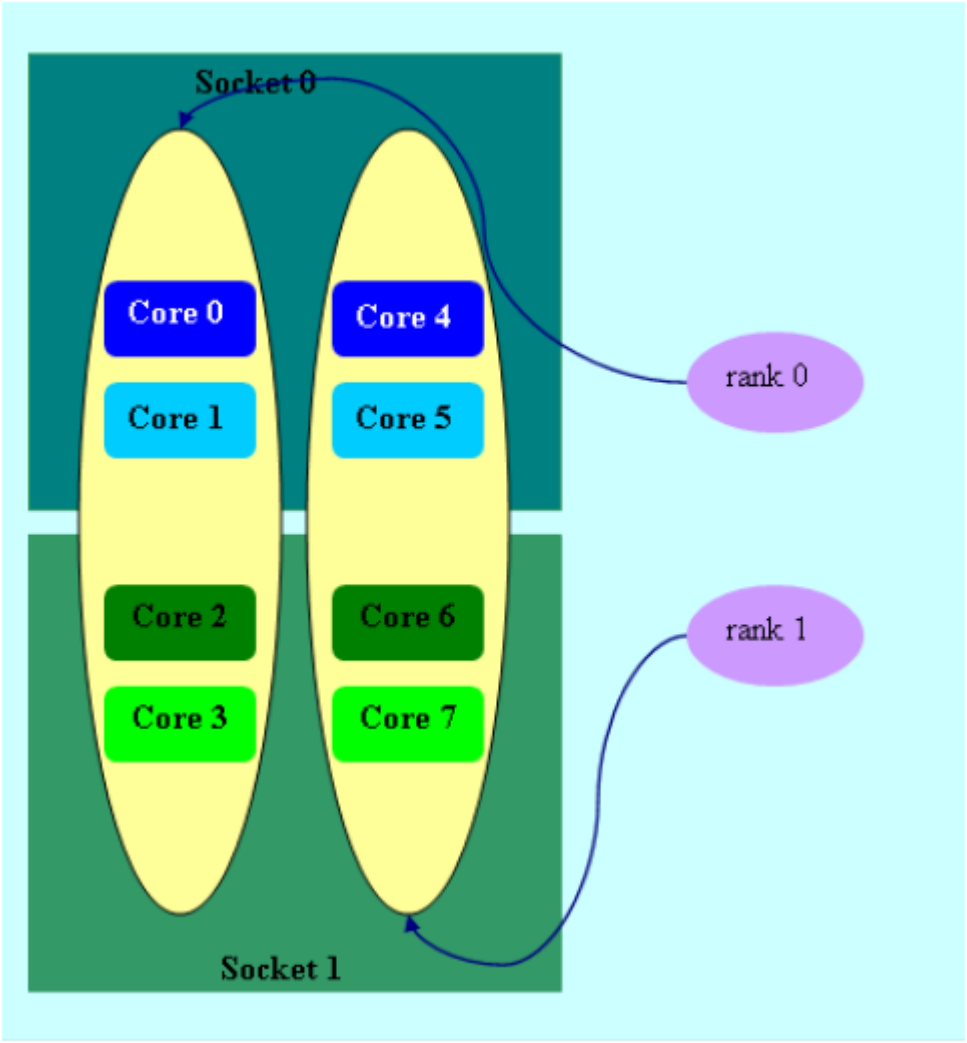
Two domains are defined according to the number of sockets. Process rank 0 can migrate on all cores on the 0-th socket. Process rank 1 can migrate on all cores on the first socket.



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

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

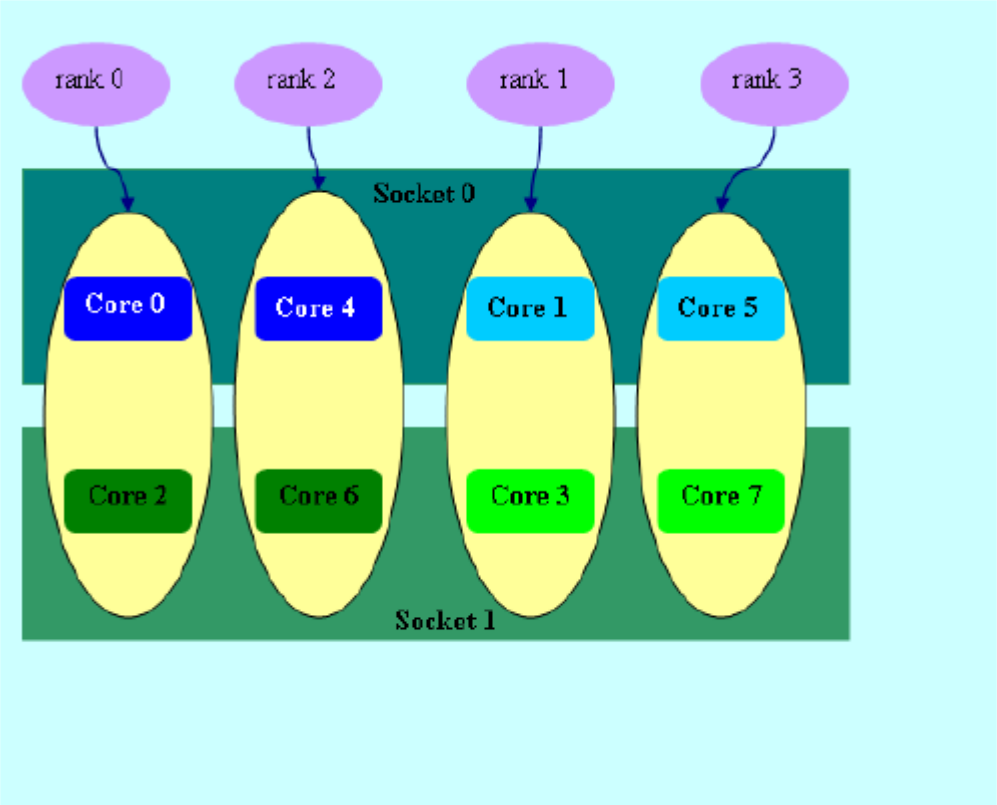
well, and so on.



Picture3.2-5 `mpiexec -n 2 -env I_MPI_PIN_DOMAIN 4:platform ./a.out`

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

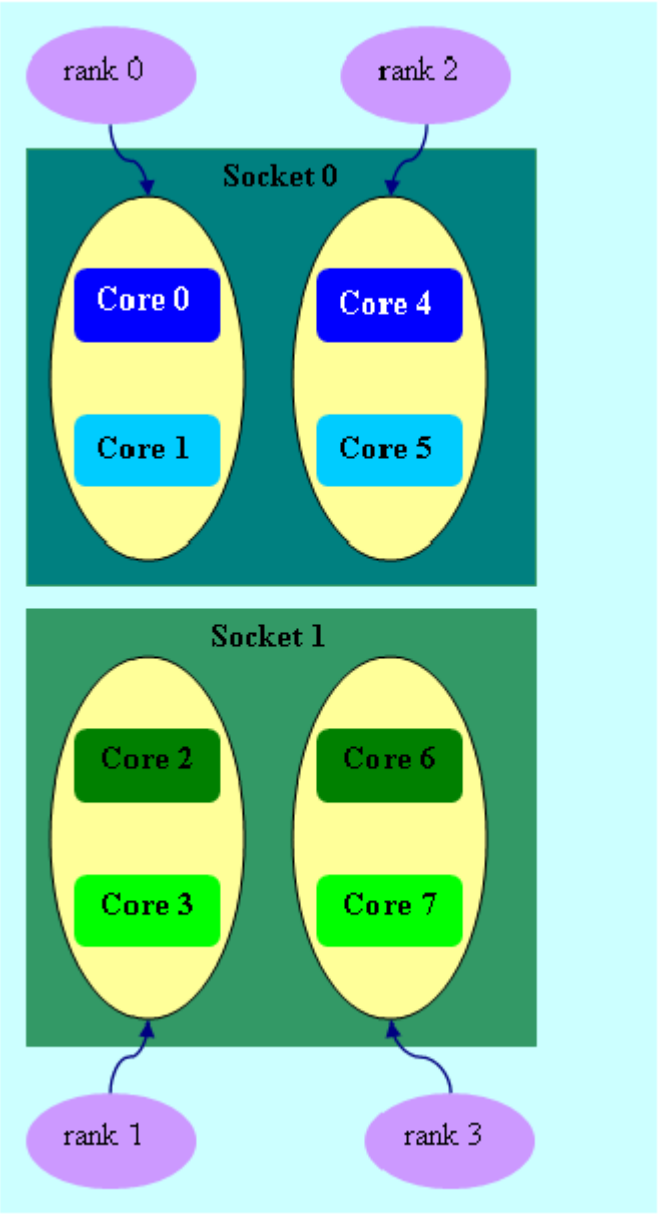
defined by the `platform` option.



Picture3.2-6 `mpiexec -n 4 -env I_MPI_PIN_DOMAIN auto:scatter ./a.out`

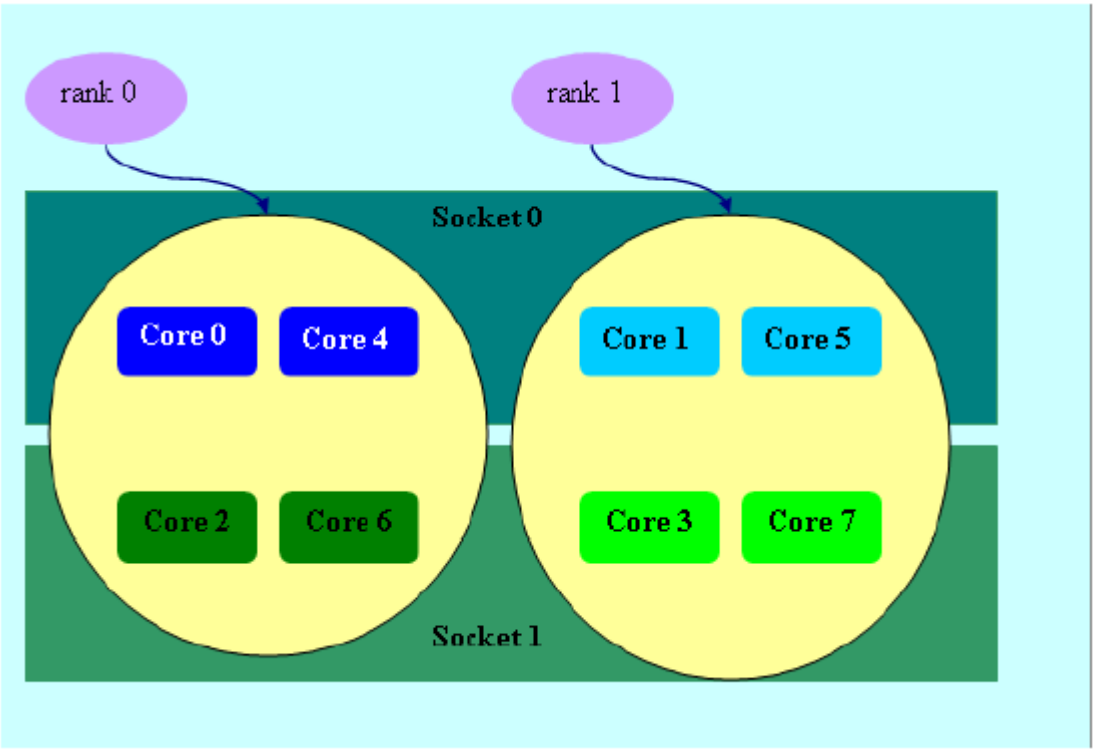
Domain size=2 (defined by the number of CPUs=8 / number of processes=4), `scatter` layout. Four domains {0,2}, {1,3}, {4,6}, {5,7} are defined. Domain members do not share any common

resources.



Picture3.2-7 `mpiexec -n 4 -env I_MPI_PIN_DOMAIN omp:platform ./a.out`
`setenv OMP_NUM_THREADS=2`

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



Picture3.2-8 `mpiexec -n 2 -env I_MPI_PIN_DOMAIN [55,aa] ./a.out`

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.

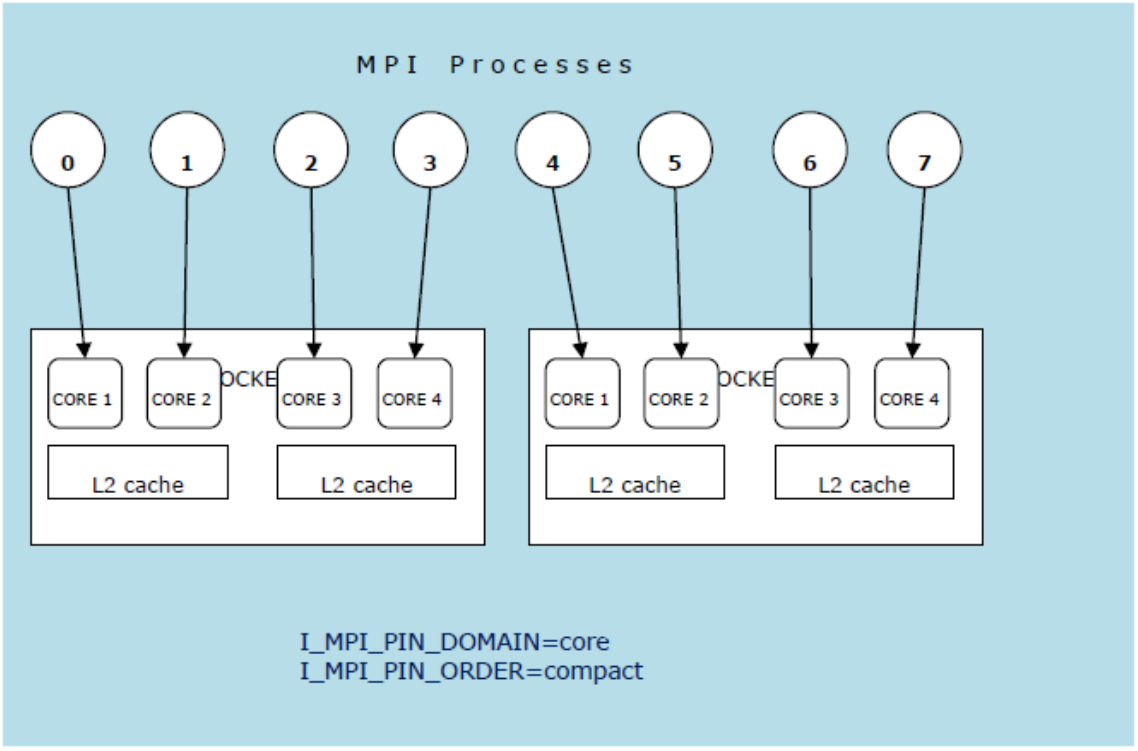
The options `scatter` and `compact` are available for both Intel® and non-Intel microprocessors, but they may perform additional optimizations for Intel microprocessors than they perform for non-Intel microprocessors.

Example

For the following configuration:

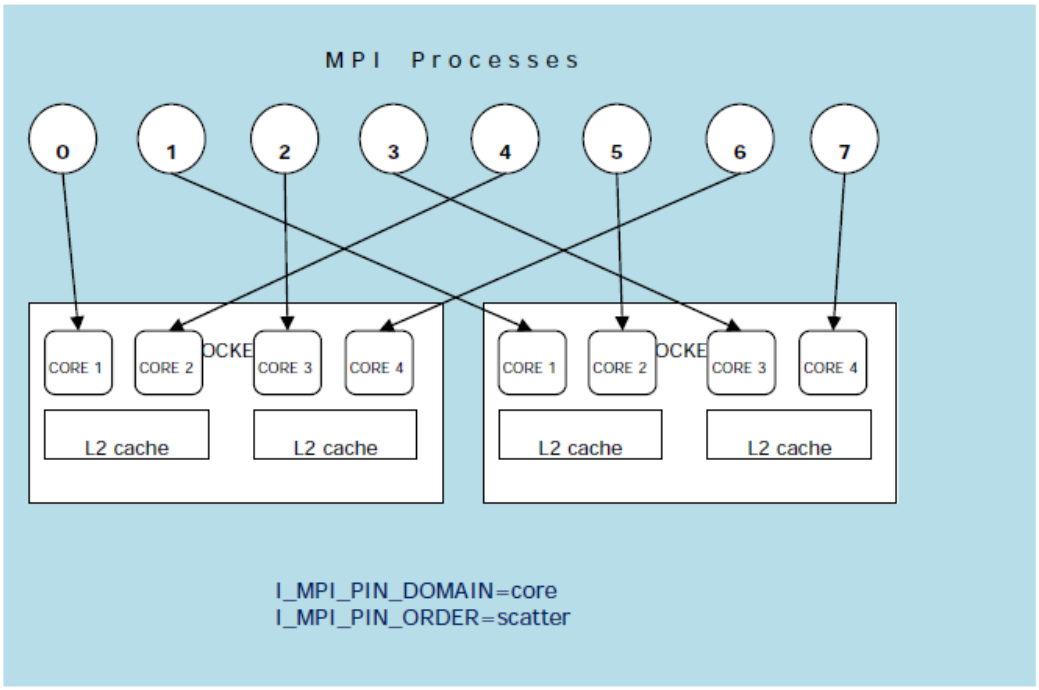
- Two socket nodes with four cores and a shared L2 cache for corresponding core pairs.
- 8 MPI processes you want to run on the node using the following settings:

  - 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

3.3. 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
- DAPL UD-capable network fabrics
- TCP-capable network fabrics
- TMI-capable network fabrics
- OFA*-capable network fabrics

3.3.1. Communication Fabrics Control

I_MPI_FABRICS

(I_MPI_DEVICE)

Select the particular network fabrics to be used.

Syntax

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

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

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

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

Deprecated Syntax

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

Arguments

<fabric>	Define a network fabric
shm	Shared-memory
dapl	DAPL-capable network fabrics, such as InfiniBand*, iWarp*, Dolphin*, and XPMEM* (through DAPL*)
tcp	TCP/IP-capable network fabrics, such as Ethernet and InfiniBand* (through IPoIB*)
tmi	TMI-capable network fabrics including Intel® True Scale Fabric, Myrinet*, (through Tag Matching Interface)
ofa	OFA-capable network fabric including InfiniBand* (through OFED* verbs)

Correspondence with I_MPI_DEVICE

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

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

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

```
$ mpiexec -n <# of processes> \
```

```
-env I_MPI_DEVICE rdssm:OpenIB-cma <executable>
```

For these devices, if *<provider>* is not specified, the first DAPL* provider in the `/etc/dat.conf` file is used.

Description

Set this 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> -perhost <# of procs per host> <executable>
```

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

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

or

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

NOTE:

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

I_MPI_FABRICS_LIST

Define a fabrics list.

Syntax

I_MPI_FABRICS_LIST=*<fabrics list>*

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

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

Arguments

<fabrics list>	Specify a list of fabrics
dapl,ofa,tcp,tmi	This is the default value
dapl,tcp,ofa,tmi	If you specify I_MPI_WAIT_MODE=enable, this is the default value

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

(I_MPI_FALLBACK_DEVICE)

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

Syntax

I_MPI_FALLBACK=*<arg>*

Deprecated Syntax

I_MPI_FALLBACK_DEVICE=*<arg>*

Arguments

<arg>	Binary indicator

enable | yes | on | 1	Fall back to the first available fabric. This is the default value if I_MPI_FABRICS(I_MPI_DEVICE) environment variable is not set.
disable | no| off |0	Terminate the job if MPI 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(I_MPI_DEVICE) 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=ofa,tmi,dapl,tcp, I_MPI_FALLBACK=enable and the initialization of DAPL-capable network fabrics fails, the library falls back to TCP-capable network fabric.

I_MPI_EAGER_THRESHOLD

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

Syntax

I_MPI_EAGER_THRESHOLD=*<nbytes>*

Arguments

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

Description

Set this 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_INTRANODE_DIRECT_COPY

Turn on/off the intranode direct copy communication mode.

Syntax

`I_MPI_INTRANODE_DIRECT_COPY=<arg>`

Arguments

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

Description

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

- Messages shorter than or equal to the threshold value of the I_MPI_INTRANODE_EAGER_THRESHOLD environment variable are transferred using the shared memory.

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_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

(I_MPI_SOCK_SCALABLE_OPTIMIZATION)

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

Syntax

`I_MPI_SCALABLE_OPTIMIZATION=<arg>`

Deprecated Syntax

`I_MPI_SOCK_SCALABLE_OPTIMIZATION=<arg>`

Arguments

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

Description

Set this 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.

NOTE:

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

I_MPI_WAIT_MODE

Turn on/off wait mode.

Syntax

`I_MPI_WAIT_MODE=<arg>`

Arguments

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

Description

Set this 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

(I_MPI_USE_DYNAMIC_CONNECTIONS)

Turn on/off the dynamic connection establishment.

Syntax

`I_MPI_DYNAMIC_CONNECTION=<arg>`

Deprecated Syntax

`I_MPI_USE_DYNAMIC_CONNECTIONS=<arg>`

Arguments

<arg>	Binary indicator
`enable | yes | on | 1`	Turn on the dynamic connection establishment. This is the

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

Description

Set this 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.

3.3.2. Shared Memory Control

I_MPI_SHM_CACHE_BYPASS

(I_MPI_CACHE_BYPASS)

Control the message transfer algorithm for the shared memory.

Syntax

```
I_MPI_SHM_CACHE_BYPASS=<arg>
```

Deprecated Syntax

```
I_MPI_CACHE_BYPASS=<arg>
```

Arguments

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

Description

Set this 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

(I_MPI_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>]`

Deprecated Syntax

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

Arguments

<nb_send>	Set the threshold for sent messages in the following situations: • Processes are pinned on cores that are not located in the same physical processor package • Processes are not pinned
>= 0	• For machines optimized with Intel® Streaming SIMD Extensions 4.2 (Intel® SSE4.2) or Intel® AES New Instructions (Intel® AES-NI), the default *<nb_send>* value is -1. This value disables the copying bypass cache • For other architectures, the default *<nb_send>* value is 16,384 bytes
<nb_recv>	Set the threshold for received messages in the following situations: • Processes are pinned on cores that are not located in the same physical processor package • Processes are not pinned
>= 0	• For machines optimized with Intel® SSE4.2, the default *<nb_send>* value is -1. This value disables the copying bypass cache • For machines optimized with Intel® AES-NI, the default *<nb_send>* value is MAX(1Mb, L3/NP), where L3 indicates the size of Level 3 cache and NP indicates the number of processes on the node • For other architectures, the default *<nb_recv_pk>* value is 2,097,152 bytes
<nb_send_pk>	Set the threshold for sent messages when processes are pinned on cores located in the same physical processor package
>= 0	The default *<nb_send_pk>* value is -1 (copying bypass cache is disabled)
<nb_recv_pk>	Set the threshold for received messages when processes are pinned on cores located in the same physical processor package
>= 0	• For machines optimized with Intel® SSE4.2, the default *<nb_send>* value is -1. This value disables the copying bypass

	cache • For machines optimized with Intel® AES-NI, the default <nb_send> value is MAX(1Mb, L3/NP), where L3 indicates the size of Level 3 cache and NP indicates the number of processes on the node • For other architectures, the default <nb_recv_pk> value is 2,097,152 bytes

Description

Set this environment variable to control the thresholds for the message copying algorithm. MPI copies messages greater than or equal in size to the defined threshold value so that the messages bypass the cache. The value of -1 disables cache bypass. 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>`

Deprecated Syntax

`I_MPI_INTRANODE_DIRECT_COPY=<arg>`

Arguments

<arg>	Binary indicator
`shm`	Turn on the shared memory 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.

NOTE:

Two arguments of the `I_MPI_SHM_LMT` environment variable are related to the `I_MPI_INTRANODE_DIRECT_COPY` environment variable:

- `I_MPI_SHM_LMT=direct` is equal to the deprecated setting `I_MPI_INTRANODE_DIRECT_COPY=enable`.
- `I_MPI_SHM_LMT=shm` is equal to the deprecated setting `I_MPI_INTRANODE_DIRECT_COPY=disable`.

I_MPI_SHM_LMT_BUFFER_NUM

(I_MPI_SHM_NUM_BUFFERS)

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

Syntax

`I_MPI_SHM_LMT_BUFFER_NUM=<num>`

Deprecated Syntax

`I_MPI_SHM_NUM_BUFFERS=<num>`

Arguments

<num>	The number of shared memory buffers for each process pair

> 0	The default value is 8

Description

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

I_MPI_SHM_LMT_BUFFER_SIZE

(I_MPI_SHM_BUFFER_SIZE)

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

Syntax

`I_MPI_SHM_LMT_BUFFER_SIZE=<nbytes>`

Deprecated Syntax

`I_MPI_SHM_BUFFER_SIZE=<nbytes>`

Arguments

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

(I_MPI_INTRANODE_SHMEM_BYPASS, I_MPI_USE_DAPL_INTRANODE)

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

Syntax

`I_MPI_SHM_BYPASS=<arg>`

Deprecated Syntaxes

`I_MPI_INTRANODE_SHMEM_BYPASS=<arg>`

`I_MPI_USE_DAPL_INTRANODE=<arg>`

Arguments

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

Description

Set this 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>` or an equivalent `I_MPI_DEVICE` setting. 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`	When internode communication uses the `dapl` or `tcp` fabric, the default `<shm_scount>` value is equal to `100` spins When internode communication uses the `ofa, tmi` or `dapl` (DAPL UD-enabled only) fabric, the default `<shm_scount>` value is equal to `10` spins. The maximum value is equal to `2147483647`

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.

3.3.3. DAPL-capable Network Fabrics Control

I_MPI_DAPL_PROVIDER

Define the DAPL provider to load.

Syntax

`I_MPI_DAPL_PROVIDER=<name>`

Arguments

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

Description

Set this environment variable to define the name of DAPL provider to load. This name is also defined in the `dat.conf` configuration file.

I_MPI_DAT_LIBRARY

Select the DAT library to be used for DAPL* provider.

Syntax

`I_MPI_DAT_LIBRARY=<library>`

Arguments

<library>	Specify the DAT library for DAPL provider to be used. Default values are `libdat.so or libdat.so.1` for DAPL* 1.2 providers and `libdat2.so or libdat2.so.2` 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 on DAPL and DAPL UD capable fabrics.

I_MPI_DAPL_TRANSLATION_CACHE

(I_MPI_RDMA_TRANSLATION_CACHE)

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

Syntax

`I_MPI_DAPL_TRANSLATION_CACHE=<arg>`

Deprecated Syntax

`I_MPI_RDMA_TRANSLATION_CACHE=<arg>`

Arguments

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

Description

Set this 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

(I_MPI_RDMA_EAGER_THRESHOLD, RDMA_IBA_EAGER_THRESHOLD)

Change the threshold of the DAPL direct-copy protocol.

Syntax

I_MPI_DAPL_DIRECT_COPY_THRESHOLD=*<nbytes>*

Deprecated Syntaxes

I_MPI_RDMA_EAGER_THRESHOLD=*<nbytes>*

RDMA_IBA_EAGER_THRESHOLD=*<nbytes>*

Arguments

<nbytes>	Define the DAPL direct-copy protocol threshold
> 0	The default *<nbytes>* value is equal to 23728 bytes

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

(I_MPI_DYNAMIC_CONNECTION_MODE, I_MPI_DYNAMIC_CONNECTIONS_MODE)

Choose the algorithm for establishing the DAPL* connections.

Syntax

`I_MPI_DAPL_DYNAMIC_CONNECTION_MODE=<arg>`

Deprecated Syntax

`I_MPI_DYNAMIC_CONNECTION_MODE=<arg>`

`I_MPI_DYNAMIC_CONNECTIONS_MODE=<arg>`

Arguments

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

Description

Set this 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

(I_MPI_RDMA_SCALABLE_PROGRESS)

Turn on/off scalable algorithm for DAPL read progress.

Syntax

`I_MPI_DAPL_SCALABLE_PROGRESS=<arg>`

Deprecated Syntax

`I_MPI_RDMA_SCALABLE_PROGRESS=<arg>`

Arguments

<arg>	Binary indicator
`enable | yes | on | 1`	Turn on scalable algorithm. 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

(I_MPI_RDMA_BUFFER_NUM, NUM_RDMA_BUFFER)

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

Syntax

`I_MPI_DAPL_BUFFER_NUM=<nbuf>`

Deprecated Syntaxes

`I_MPI_RDMA_BUFFER_NUM=<nbuf>`

`NUM_RDMA_BUFFER=<nbuf>`

Arguments

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

Description

Set this 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

(I_MPI_RDMA_BUFFER_SIZE, I_MPI_RDMA_VBUF_TOTAL_SIZE)

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

Syntax

I_MPI_DAPL_BUFFER_SIZE=*<nbytes>*

Deprecated Syntaxes

I_MPI_RDMA_BUFFER_SIZE=*<nbytes>*

I_MPI_RDMA_VBUF_TOTAL_SIZE=*<nbytes>*

Arguments

<nbytes>	Define the size of pre-registered buffers
> 0	The default *<nbytes>* value is equal to 23808 bytes

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

(I_MPI_RDMA_RNDV_BUFFER_ALIGNMENT, I_MPI_RDMA_RNDV_BUF_ALIGN)

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

Syntax

`I_MPI_DAPL_RNDV_BUFFER_ALIGNMENT=<arg>`

Deprecated Syntaxes

`I_MPI_RDMA_RNDV_BUFFER_ALIGNMENT=<arg>`

`I_MPI_RDMA_RNDV_BUF_ALIGN=<arg>`

Arguments

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

(I_MPI_RDMA_RNDV_WRITE, I_MPI_USE_RENDEZVOUS_RDMA_WRITE)

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

Syntax

`I_MPI_DAPL_RDMA_RNDV_WRITE=<arg>`

Deprecated Syntaxes

`I_MPI_RDMA_RNDV_WRITE=<arg>`

`I_MPI_USE_RENDEZVOUS_RDMA_WRITE=<arg>`

Arguments

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

Description

Set this 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

(I_MPI_RDMA_CHECK_MAX_RDMA_SIZE)

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

Syntax

`I_MPI_DAPL_CHECK_MAX_RDMA_SIZE=<arg>`

Deprecated Syntax

`I_MPI_RDMA_CHECK_MAX_RDMA_SIZE=<arg>`

Arguments

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

Description

Set this 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

(I_MPI_RDMA_MAX_MSG_SIZE)

Control message fragmentation threshold.

Syntax

`I_MPI_DAPL_MAX_MSG_SIZE=<nbytes>`

Deprecated Syntax

`I_MPI_RDMA_MAX_MSG_SIZE=<nbytes>`

Arguments

<nbytes>	Define the maximum message size that can be sent through DAPL without fragmentation
`> 0`	If the `I_MPI_DAPL_CHECK_MAX_RDMA_SIZE` 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

(I_MPI_RDMA_CONN_EVD_SIZE, I_MPI_CONN_EVD_QLEN)

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

Syntax

I_MPI_DAPL_CONN_EVD_SIZE=<size>

Deprecated Syntaxes

I_MPI_RDMA_CONN_EVD_SIZE=<size>

I_MPI_CONN_EVD_QLEN=<size>

Arguments

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

Description

Set this 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

(I_MPI_RDMA_WRITE_IMM)

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

Syntax

I_MPI_DAPL_RDMA_WRITE_IMM=<arg>

Deprecated syntax

I_MPI_RDMA_WRITE_IMM=<arg>

Arguments

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

Description

Set this 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.

3.3.4. DAPL UD-capable Network Fabrics Control

I_MPI_DAPL_UD

Enable/disable using DAPL UD path.

Syntax

`I_MPI_DAPL_UD=<arg>`

Arguments

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

Description

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

I_MPI_DAPL_UD_PROVIDER

Define the DAPL provider to work with IB UD transport.

Syntax

`I_MPI_DAPL_UD_PROVIDER=<name>`

Arguments

<name>	Define the name of DAPL provider to load

Description

Set this environment variable to define the name of DAPL provider to load. This name is also defined in the `dat.conf` configuration file. Make sure that specified DAPL provider supports UD IB extension.

I_MPI_DAPL_UD_DIRECT_COPY_THRESHOLD

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

Syntax

`I_MPI_DAPL_UD_DIRECT_COPY_THRESHOLD=<nbytes>`

Arguments

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

Description

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

- Messages shorter than or equal to *<nbytes>* are sent using the eager protocol through the internal pre-registered buffers. 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_UD_RECV_BUFFER_NUM

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

Syntax

`I_MPI_DAPL_UD_RECV_BUFFER_NUM=<nbuf>`

Arguments

<nbuf>	Define the number of buffers for receiving messages

> 0	The default value is 16 + n*4 where n is a total number of process in MPI job

Description

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

NOTE:

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

I_MPI_DAPL_UD_SEND_BUFFER_NUM

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

Syntax

`I_MPI_DAPL_UD_SEND_BUFFER_NUM=<nbuf>`

Arguments

<nbuf>	Define the number of buffers for sending messages
> 0	The default value is 16 + n*4 where n is a total number of process in MPI job

Description

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

NOTE:

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

I_MPI_DAPL_UD_ACK_SEND_POOL_SIZE

Change the number of ACK UD buffers for sending messages.

Syntax

`I_MPI_DAPL_UD_ACK_SEND_POOL_SIZE=<nbuf>`

Arguments

<nbuf>	Define the number of ACK UD buffers for sending messages
> 0	The default value is 256

Description

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

I_MPI_DAPL_UD_ACK_RECV_POOL_SIZE

Change the number of ACK UD buffers for receiving messages.

Syntax

`I_MPI_DAPL_UD_ACK_RECV_POOL_SIZE=<nbuf>`

Arguments

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

Description

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

I_MPI_DAPL_UD_TRANSLATION_CACHE

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

Syntax

`I_MPI_DAPL_UD_TRANSLATION_CACHE=<arg>`

Arguments

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

Description

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

Using the cache substantially improves performance. See product *Release Notes* for further details.

I_MPI_DAPL_UD_TRANSLATION_CACHE_AVL_TREE

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

Syntax

`I_MPI_DAPL_UD_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 UD 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_UD_REQ_EVD_SIZE

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

Syntax

I_MPI_DAPL_UD_REQ_EVD_SIZE=*<size>*

Arguments

<size>	Define the length of the event queue
> 0	The default value is 2,000

Description

Set this environment variable to define the event queue size of the DAPL event dispatcher that handles completions of sending DAPL UD data transfer operations (DTO). If this 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 is at least equal to the calculated value, but it can also provide a larger queue size.

I_MPI_DAPL_UD_CONN_EVD_SIZE

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

Syntax

I_MPI_DAPL_UD_CONN_EVD_SIZE=*<size>*

Arguments

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

Description

Set this 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 is at least equal to the calculated value, but it can also provide a larger queue size.

I_MPI_DAPL_UD_RECV_EVD_SIZE

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

Syntax

`I_MPI_DAPL_UD_RECV_EVD_SIZE=<size>`

Arguments

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

Description

Set this environment variable to define the event queue size of the DAPL event dispatcher that handles completions of receiving DAPL UD data transfer operations (DTO). If this 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 is at least equal to the calculated value, but it can also provide a larger queue size.

I_MPI_DAPL_UD_RNDV_MAX_BLOCK_LEN

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

Syntax

`I_MPI_DAPL_UD_RNDV_MAX_BLOCK_LEN=<nbytes>`

Arguments

<arg>	Define maximum size of block that is passed at one iteration of DAPL UD direct-copy protocol
> 0	The default value is 1,048,576

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

I_MPI_DAPL_UD_RNDV_BUFFER_ALIGNMENT

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

Syntax

`I_MPI_DAPL_UD_RNDV_BUFFER_ALIGNMENT=<arg>`

Arguments

<arg>	Define the alignment of the sending buffer

`> 0 and a power of 2`	The default value is `16`

Set this 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, this may increase data transfer bandwidth.

I_MPI_DAPL_UD_RNDV_COPY_ALIGNMENT_THRESHOLD

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

Syntax

`I_MPI_DAPL_UD_RNDV_COPY_ALIGNMENT_THRESHOLD=<nbytes>`

Arguments

<nbytes>	Define send buffer alignment threshold
`> 0 and a power of 2`	The default value is `32,768`

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

I_MPI_DAPL_UD_RNDV_DYNAMIC_CONNECTION

Control the algorithm of dynamic connection establishment for DAPL UD endpoints used in the direct copy protocol.

Syntax

`I_MPI_DAPL_UD_RNDV_DYNAMIC_CONNECTION=<arg>`

Arguments

<arg>	Binary indicator
`enable | yes | on | 1`	Turns on the dynamic connection mode. This is the default value
`disable | no | off | 0`	Turns off the dynamic connections mode

Set this variable to control the dynamic connection establishment of DAPL UD endpoints used in the direct copy protocol.

If you disable the dynamic connection mode, all possible connections are established during the MPI startup phase.

If you enable the mode, the connection is established when an application calls the MPI function to pass the data from one process to another and invokes the communication between the two processes.

NOTE:

For the RNDV dynamic connection mode, the size of the messages passed in the data is larger than the value you set in the `I_MPI_DAPL_UD_DIRECT_COPY_THRESHOLD` environment variable.

I_MPI_DAPL_UD_EAGER_DYNAMIC_CONNECTION

Control the algorithm of the dynamic connection establishment for DAPL UD endpoints used in eager protocol.

Syntax

`I_MPI_DAPL_UD_EAGER_DYNAMIC_CONNECTION=<arg>`

Arguments

`<arg>`	Binary indicator
`enable | yes | on | 1`	Turn on the dynamic connection mode. If the number of processes is over `64`, this is the default value
`disable | no | off | 0`	Turn off the dynamic connections mode

Set this variable to control the dynamic connection establishment of DAPL UD endpoints involved in eager protocol. Eager protocol is used to transfer messages through internal pre-registered buffers.

If you disable this mode, all possible connections are established during MPI startup phase.

If you enable this mode, the connection is established when an application calls the MPI function to pass the data from one process to another and invokes the communication between the two processes.

NOTE:

For the eager dynamic connection mode, the size of the messages passed in the data is shorter than or equal to the value you set in the `I_MPI_DAPL_UD_DIRECT_COPY_THRESHOLD` environment variable.

I_MPI_DAPL_UD_DESIRED_STATIC_CONNECTIONS_NUM

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

Syntax

`I_MPI_DAPL_UD_DESIRED_STATIC_CONNECTIONS_NUM=<num_procesess>`

Arguments

`<num_procesess>`	Define the number of processes that establish DAPL UD static connections at the same time
`> 0`	The default value is equal to `200`

Description

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

If the number of processes in an 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_DAPL_UD_RDMA_MIXED

Control the use of the DAPL UD/RDMA mixed communication.

Syntax

`I_MPI_DAPL_UD_RDMA_MIXED =<arg>`

Arguments

<arg>	Binary indicator
`enable | yes | on | 1`	Turn on the use of DAPL UD/RDMA mixed communication
`disable | no | off | 0`	Turn off the use of DAPL UD/RDMA mixed communication. This is the default value

Description

Set this environment variable to enable the DAPL UD/RDMA mixed mode for transferring data. In the DAPL UD/RDMA mixed mode, small messages are passed through the UD transport and large messages are passed through the RDMA transport. If you set the `I_MPI_DAPL_UD_RDMA_MIXED` environment variable and a certain DAPL provider attribute indicates that UD IB extension is supported, the DAPL UD/RDMA mixed mode is enabled.

The following set of I_MPI_DAPL_UD* environment variables also controls the DAPL UD/RDMA mixed mode:

- `I_MPI_DAPL_UD_PROVIDER`
- `I_MPI_DAPL_UD_EAGER_DYNAMIC_CONNECTION`
- `I_MPI_DAPL_UD_RNDV_DYNAMIC_CONNECTION`
- `I_MPI_DAPL_UD_DIRECT_COPY_THRESHOLD`
- `I_MPI_DAPL_UD_RECV_BUFFER_NUM`
- `I_MPI_DAPL_UD_SEND_BUFFER_NUM`
- `I_MPI_DAPL_UD_NUMBER_CREDIT_UPDATE`
- `I_MPI_DAPL_UD_ACK_SEND_POOL_SIZE`
- `I_MPI_DAPL_UD_ACK_RECV_POOL_SIZE`
- `I_MPI_DAPL_UD_RESENT_TIMEOUT`

- I_MPI_DAPL_UD_MAX_MSG_SIZE
- I_MPI_DAPL_UD_SEND_BUFFER_SIZE
- I_MPI_DAPL_UD_REQ_EVD_SIZE
- I_MPI_DAPL_UD_REQUEST_QUEUE_SIZE
- I_MPI_DAPL_UD_MULTIPLE_EAGER_SEND
- I_MPI_DAPL_UD_NA_SBUF_LIMIT
- I_MPI_DAPL_UD_RECV_EVD_SIZE
- I_MPI_DAPL_UD_CONNECTION_TIMEOUT
- I_MPI_DAPL_UD_PORT
- I_MPI_DAPL_UD_CREATE_CONN_QUAL,
- I_MPI_DAPL_UD_FINALIZE_RETRY_COUNT
- I_MPI_DAPL_UD_FINALIZE_TIMEOUT
- I_MPI_DAPL_UD_TRANSLATION_CACHE
- I_MPI_DAPL_UD_TRANSLATION_CACHE_AVL_TREE
- I_MPI_DAPL_UD_TRANSLATION_CACHE_MAX_ENTRY_NUM
- I_MPI_DAPL_UD_TRANSLATION_CACHE_MAX_MEMORY_SIZE
- I_MPI_DAPL_UD_PKT_LOSS_OPTIMIZATION
- I_MPI_DAPL_UD_DFACTOR
- I_MPI_DAPL_UD_DESIRED_STATIC_CONNECTIONS_NUM
- I_MPI_DAPL_UD_CONN_EVD_SIZE
- I_MPI_DAPL_UD_RNDV_BUFFER_ALIGNMENT
- I_MPI_DAPL_UD_RNDV_COPY_ALIGNMENT_THRESHOLD

The following set of environment variables is specific for DAPL UD/RDMA mixed mode:

- I_MPI_DAPL_UD_MAX_RDMA_SIZE
- I_MPI_DAPL_UD_MAX_RDMA_DTOS

I_MPI_DAPL_UD_MAX_RDMA_SIZE

Control the maximum message size that can be sent though the RDMA for DAPL UD/RDMA mixed mode.

Syntax

`I_MPI_DAPL_UD_MAX_RDMA_SIZE =<nbytes>`

Arguments

<nbytes>	Define the maximum message size that can be sent through RDMA without fragmentation
> 0	The default *<nbytes>* value is 4 MB

Description

Set this environment variable to define the maximum message size that can be sent though RDMA protocol for the DAPL UD/RDMA mixed mode. If the message size is greater than this value, this message is divided into several fragments and is sent by several RDMA operations.

I_MPI_DAPL_UD_MAX_RDMA_DTOS

Control the maximum number of uncompleted RDMA operations per connection for the DAPL UD/RDMA mixed mode.

Syntax

`I_MPI_DAPL_UD_MAX_RDMA_DTOS=<arg>`

Arguments

<arg>	Define the maximum number of RDMA operations per connection
> 0	The default *<arg>* value is 8

Description

Set this environment variable to define the maximum number of RDMA operations per connection for the DAPL UD/RDMA mixed mode.

3.3.5. TCP-capable Network Fabrics Control

I_MPI_TCP_NETMASK

(I_MPI_NETMASK)

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

Syntax

`I_MPI_TCP_NETMASK=<arg>`

Arguments

<arg>	Define the network interface (string parameter)
<interface_mnemonic>	Mnemonic of the network interface: ib or eth
ib	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 default value of the TCP socket buffer size on your Linux system.

Description

Set this environment variable to manually define the size of the TCP socket buffers. The TCP socket buffer size is restricted by the existing TCP settings on your Linux system.

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).

I_MPI_TCP_POLLING_MODE

Set this environment variable to define a polling mode.

Syntax

I_MPI_TCP_POLLING_MODE=*<mode>*

Arguments

<mode>	Specify the polling mode
poll	The polling mode based on the poll() function. This is *the* default value
epoll[:edge]	The polling mode based on the epoll() function as an edge-triggered interface
epoll:level	The polling mode based on the epoll() function as a level-triggered interface

Set this environment variable to select the polling mode for the tcp fabric.

Use the I_MPI_TCP_POLLING_MODE environment variable for tuning application performance. You can choose the best polling mode on an experimental basis. The best mode depends on the specific application and on the number of processes. The epoll polling mode is a preferable mode in the following situations:

- for large number of processes
- for APP client-server type
- for MPI_ANY_SOURCE tag matching

3.3.6. TMI-capable Network Fabrics Control

I_MPI_TMI_LIBRARY

Select the TMI library to be used.

Syntax

`I_MPI_TMI_LIBRARY=<library>`

Arguments

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

Description

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

I_MPI_TMI_PROVIDER

Define the name of the TMI provider to load.

Syntax

`I_MPI_TMI_PROVIDER=<name>`

Arguments

<name>	The name of the TMI provider to load

Description

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

3.3.7. OFA*-capable Network Fabrics Control

I_MPI_OFA_NUM_ADAPTERS

Set the number of connection adapters.

Syntax

`I_MPI_OFA_NUM_ADAPTERS=<arg>`

Arguments

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

Description

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

I_MPI_OFA_ADAPTER_NAME

Set the name of adapter that is used.

Syntax

`I_MPI_OFA_ADAPTER_NAME=<arg>`

Arguments

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

Description

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

I_MPI_OFA_NUM_PORTS

Set the number of used ports on each adapter.

Syntax

`I_MPI_OFA_NUM_PORTS=<arg>`

Arguments

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

Description

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

I_MPI_OFA_NUM_RDMA_CONNECTIONS

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

Syntax

`I_MPI_OFA_NUM_RDMA_CONNECTIONS=<num_conn>`

Arguments

<num_conn>	Define the maximum number of connections that can use the `rdma` exchange protocol

>= 0	Create the specified number of connections that use the rdma exchange protocol. All other processes use the send/ receive exchange protocol
-1	Create $\log_2$(number of processes) rdma connections
>= number of processes	Create rdma connections for all processes. This is the default value

Description

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

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

I_MPI_OFA_SWITCHING_TO_RDMA

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

Syntax

I_MPI_OFA_SWITCHING_TO_RDMA=*<number>*

Arguments

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

Description

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

I_MPI_OFA_RAIL_SCHEDULER

Set the method of choosing rails for short messages.

Syntax

I_MPI_OFA_RAIL_SCHEDULER=*<arg>*

Arguments

<arg>	Mode selector
ROUND_ROBIN	Next time use next rail

FIRST_RAIL	Always use the first rail for short messages
PROCESS_BIND	Always use the rail specific for process

Description

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

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

I_MPI_OFA_TRANSLATION_CACHE

Turn on/off the memory registration cache.

Syntax

I_MPI_OFA_TRANSLATION_CACHE=*<arg>*

Arguments

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

Description

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

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

I_MPI_OFA_TRANSLATION_CACHE_AVL_TREE

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

Syntax

I_MPI_OFA_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 OFA 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_OFA_USE_XRC

Control the use of extensible reliable connection (XRC) capability.

Syntax

`I_MPI_OFA_USE_XRC=<arg>`

Arguments

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

Description

Set this environment variable to control the use of XRC when you are using a large cluster with several thousands of nodes.

I_MPI_OFA_DYNAMIC_QPS

Control the library to create queue pairs (QPs) dynamically.

Syntax

`I_MPI_OFA_DYNAMIC_QPS=<arg>`

Arguments

`<arg>`	Binary indicator
`enable | yes | on | 1`	Create QPs dynamically. This is the default value if the number of processes is larger than or equal `2,000`
`disable | no | off | 0`	Create all QPs during the initial stage. This is the default value if the number of processes is less than `2,000`

Description

Set this environment variable to turn on dynamic creation of QPs.

I_MPI_OFA_PACKET_SIZE

Set the size of the packet used for sending.

Syntax

`I_MPI_OFA_PACKET_SIZE=<arg>`

Arguments

`<arg>`	Define the size of packet in bytes
`> 0`	Use the specified packet size. The default value is `8192`

Description

Set the packet size in bytes. If the number is negative, the size is set to `8`.

I_MPI_OFA_LIBRARY

Set the name of the used OFA library.

Syntax

`I_MPI_OFA_LIBRARY=<arg>`

Arguments

`<arg>`	Define the name of the OFA library
`Name`	Use the specified library. By default, the name is `libibverbs.so`

Description

Set the name of the InfiniBand* (IB*) library. If the library with the specified name does not exist, an error is returned.

I_MPI_OFA_NONSWITCH_CONF

Define the nonstandard template for port connections.

Syntax

`I_MPI_OFA_NONSWITCH_CONF=<arg>`

Arguments

`<arg>`	Define the template for port connections
`Name`	Use the specified template

Description

The nodes in clusters are normally connected so that $port_i$ of a node can access $port_i$ of all other nodes. Use this environment variable if ports are not connected in this way. The following example is the template format:

`host1@port11#port12#...#host2@port21#port22....`

$Port_i^j$ defines the port used to send from $host_i$ to $host_j$

For example:

```
node1@1#1#2#node2@2#1#1#node3@1#2#1#
```

This sample specifies the following configuration:

- Port1 of node1 connected to port2 of node2
- Port2 of node1 connected to port1 of node3
- Port1 of node2 connected to port2 of node3
- Port2 of node2 connected to port1 of node2
- Port1 of node3 connected to port2 of node1
- Port2 of node3 connected to port1 of node2

Port1 is always used to communicate with itself (loopback).

3.3.8. Failover Support in the OFA* Device

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

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

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

Intel® MPI Library also recognizes the following event

- `IBV_EVENT_PORT_ACTIVE` Link became active on a port

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

3.4. Collective Operation Control

Each collective operation in the Intel® MPI Library supports a number of communication algorithms. In addition to highly optimized default settings, the library provides two ways to

control the algorithm selection explicitly: the novel I_MPI_ADJUST environment variable family and the deprecated I_MPI_MSG environment variable family. They are described in the following sections.

These environment variables 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.

3.4.1. 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 reasonable settings

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

Description

Set this 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. See below.

Table 3.5-1 Environment Variables, Collective Operations, and Algorithms

Environment Variable	Collective Operation	Algorithms
I_MPI_ADJUST_ALLGATHER	MPI_Allgather	1. Recursive doubling algorithm 2. Bruck's algorithm 3. Ring algorithm 4. Topology aware Gatherv +

		Bcast algorithm
I_MPI_ADJUST_ALLGATHERV	MPI_Allgatherv	1. Recursive doubling algorithm 2. Bruck's algorithm 3. Ring algorithm 4. Topology aware Gatherv + Bcast algorithm
I_MPI_ADJUST_ALLREDUCE	MPI_Allreduce	1. Recursive doubling algorithm 2. Rabenseifner's algorithm 3. Reduce + Bcast algorithm 4. Topology aware Reduce + Bcast algorithm 5. Binomial gather + scatter algorithm 6. Topology aware binominal gather + scatter algorithm 7. Shumilin's ring algorithm 8. Ring algorithm
I_MPI_ADJUST_ALLTOALL	MPI_Alltoall	1. Bruck's algorithm 2. Isend/Irecv + waitall algorithm 3. Pair wise exchange algorithm 4. Plum's algorithm
I_MPI_ADJUST_ALLTOALLV	MPI_Alltoallv	1. Isend/Irecv + waitall algorithm 2. Plum's algorithm
I_MPI_ADJUST_ALLTOALLW	MPI_Alltoallw	Isend/Irecv + waitall algorithm
I_MPI_ADJUST_BARRIER	MPI_Barrier	1. Dissemination algorithm 2. Recursive doubling algorithm 3. Topology aware dissemination algorithm

		4. Topology aware recursive doubling algorithm 5. Binominal gather + scatter algorithm 6. Topology aware binominal gather + scatter algorithm
I_MPI_ADJUST_BCAST	MPI_Bcast	1. Binomial algorithm 2. Recursive doubling algorithm 3. Ring algorithm 4. Topology aware binomial algorithm 5. Topology aware recursive doubling algorithm 6. Topology aware ring algorithm 7. Shumilin's bcast algorithm
I_MPI_ADJUST_EXSCAN	MPI_Exscan	1. Partial results gathering algorithm 2. Partial results gathering regarding algorithm layout of processes
I_MPI_ADJUST_GATHER	MPI_Gather	1. Binomial algorithm 2. Topology aware binomial algorithm 3. Shumilin's algorithm
I_MPI_ADJUST_GATHERV	MPI_Gatherv	1. Linear algorithm 2. Topology aware linear algorithm
I_MPI_ADJUST_REDUCE_SCATTER	MPI_Reduce_scatter	1. Recursive having algorithm 2. Pair wise exchange algorithm 3. Recursive doubling algorithm

		4. Reduce + Scatterv algorithm 5. Topology aware Reduce + Scatterv algorithm
I_MPI_ADJUST_REDUCE	MPI_Reduce	1. Shumilin's algorithm 2. Binomial algorithm 3. Topology aware Shumilin's algorithm 4. Topology aware binomial algorithm 5. Rabenseifner's algorithm 6. Topology aware Rabenseifner's algorithm
I_MPI_ADJUST_SCAN	MPI_Scan	1. Partial results gathering algorithm 2. Topology aware partial results gathering algorithm
I_MPI_ADJUST_SCATTER	MPI_Scatter	1. Binomial algorithm 2. Topology aware binomial algorithm 3. Shumilin's algorithm
I_MPI_ADJUST_SCATTERV	MPI_Scatterv	1. Linear algorithm 2. Topology aware linear algorithm

The message size calculation rules for the collective operations are described in the table below. 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_Reduce_scatter operation:

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

In this case. algorithm 4 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).

3.4.2. I_MPI_MSG Family

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

I_MPI_FAST_COLLECTIVES

Control the default library behavior during selection of the most appropriate collective algorithm.

Syntax

I_MPI_FAST_COLLECTIVES=*<arg>*

Arguments

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

Description

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

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

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

NOTE:

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

I_MPI_BCAST_NUM_PROCS

Control `MPI_Bcast` algorithm thresholds.

Syntax

`I_MPI_BCAST_NUM_PROCS=<nproc>`

Arguments

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

I_MPI_BCAST_MSG

Control `MPI_Bcast` algorithm thresholds.

Syntax

`I_MPI_BCAST_MSG=<nbytes1,nbytes2>`

Arguments

<nbytes1,nbytes2>	Define the message size threshold range (in bytes) for choosing the `MPI_Bcast` algorithm

`> 0`
 	
`nbytes2 >= nbytes1`
	The default pair of values is `12288,524288`

Description

Set these environment variables to control the selection of the three possible `MPI_Bcast` algorithms according to the following scheme (See Table 3.5-1 for algorithm descriptions):

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

The second algorithm is selected if the message size is greater than or equal to *<nbytes1>* and less than *<nbytes2>*, and the number of processes in the operation is a power of two.

If none of the above conditions is satisfied, the third algorithm is selected.

I_MPI_ALLTOALL_NUM_PROCS

Control `MPI_Alltoall` algorithm thresholds.

Syntax

`I_MPI_ALLTOALL_NUM_PROCS=<nproc>`

Arguments

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

I_MPI_ALLTOALL_MSG

Control `MPI_Alltoall` algorithm thresholds.

Syntax

`I_MPI_ALLTOALL_MSG=<nbytes1,nbytes2>`

Arguments

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

Description

Set these environment variables to control the selection of the three possible `MPI_Alltoall` algorithms according to the following scheme (See Table 3.5-1 for algorithm descriptions):

The first algorithm is selected if the message size is greater than or equal to *<nbytes1>*, and the number of processes in the operation is not less than *<nproc>*.

The second algorithm is selected if the message size is greater than *<nbytes1>* and less than or equal to *<nbytes2>*, or if the message size is less than *<nbytes2>* and the number of processes in the operation is less than *<nproc>*.

If none of the above conditions is satisfied, the third algorithm is selected.

I_MPI_ALLGATHER_MSG

Control `MPI_Allgather` algorithm thresholds.

Syntax

`I_MPI_ALLGATHER_MSG=<nbytes1,nbytes2>`

Arguments

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

nbytes2 >= nbytes1	

Description

Set this environment variable to control the selection of the three possible MPI_Allgather algorithms according to the following scheme (See Table 3.5-1 for algorithm descriptions):

The first algorithm is selected if the message size is less than *<nbytes2>* and the number of processes in the operation is a power of two.

The second algorithm is selected if the message size is less than *<nbytes1>* and number of processes in the operation is not a power of two.

If none of the above conditions is satisfied, the third algorithm is selected.

I_MPI_ALLREDUCE_MSG

Control MPI_Allreduce algorithm thresholds.

Syntax

I_MPI_ALLREDUCE_MSG=*<nbytes>*

Arguments

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

Description

Set this environment variable to control the selection of the two possible MPI_Allreduce algorithms according to the following scheme (See Table 3.5-1 for algorithm descriptions):

The first algorithm is selected if the message size is less than or equal *<nbytes>*, or the reduction operation is user-defined, or the count argument is less than the nearest power of two less than or equal to the number of processes.

If the above condition is not satisfied, the second algorithm is selected.

I_MPI_REDSCAT_MSG

Control the MPI_Reduce_scatter algorithm thresholds.

Syntax

I_MPI_REDSCAT_MSG=*<nbytes1,nbytes2>*

Arguments

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

Description

Set this environment variable to control the selection of the three possible `MPI_Reduce_scatter` algorithms according to the following scheme (See Table 3.5-1 for algorithm descriptions):

The first algorithm is selected if the reduction operation is commutative and the message size is less than *<nbytes2>*.

The second algorithm is selected if the reduction operation is commutative and the message size is greater than or equal to *<nbytes2>*, or if the reduction operation is not commutative and the message size is greater than or equal to *<nbytes1>*.

If none of the above conditions is satisfied, the third algorithm is selected.

I_MPI_SCATTER_MSG

Control `MPI_Scatter` algorithm thresholds.

Syntax

`I_MPI_SCATTER_MSG=`*<nbytes>*

Arguments

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

Description

Set this environment variable to control the selection of the two possible `MPI_Scatter` algorithms according to the following scheme (See Table 3.5-1 for algorithm descriptions):

The first algorithm is selected on the intercommunicators if the message size is greater than *<nbytes>*.

If the above condition is not satisfied, the second algorithm is selected.

I_MPI_GATHER_MSG

Control `MPI_Gather` algorithm thresholds.

Syntax

`I_MPI_GATHER_MSG=`*<nbytes>*

Arguments

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

Description

Set this environment variable to control the selection of the two possible `MPI_Gather` algorithms according to the following scheme (See Table 3.5-1 for algorithm descriptions):

The first algorithm is selected on the intercommunicators if the message size is greater than *<nbytes>*.

If the above condition is not satisfied, the second algorithm is selected.

3.5. Miscellaneous

This topic provides the following information:

- Timer Control
- Compatibility Control
- Dynamic Process Support
- Fault Tolerance
- Statistics Gathering Mode
- ILP64 Support
- Unified Memory Management
- File System Support
- Multi-threaded memcpy Support

3.5.1. Timer Control

I_MPI_TIMER_KIND

Select the timer used by the `MPI_Wtime` and `MPI_Wtick` calls.

Syntax

`I_MPI_TIMER_KIND=<timername>`

Arguments

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

Description

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

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

3.5.2. 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`.

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

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

```
$ mpdboot -n 4 -r ssh
```

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

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

```
host1
host2
host3
host4
```

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

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

```
$ mpdboot -n 1

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

and use the following commands on the intended client hosts:

```
$ mpdboot -n 1

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

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

```
$ mpdboot -n 1 -r ssh

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

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

3.5.4. Fault Tolerance

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

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

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

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

- An application sets error handler `MPI_ERRORS_RETURN` to communicator `MPI_COMM_WORLD` (all new communicators inherit error handler from it)
- An application uses master-slave model. In this case, the application is stopped only if the master is finished or does not respond
- An application uses only point-to-point communication between a master and a number of slaves. It does not use inter slave communication or MPI collective operations.

- Handle a certain MPI error code on a point-to-point operation with a particular failed slave rank for application to avoid further communication with this rank. The slave rank can be blocking/non-blocking send, receive, probe and test,
- Any communication operation can be used on subset communicator system. If an error appears in a collective operation, any communication inside this communicator will be prohibited.
- Master failure means the job stops.
- Fault Tolerance functionality is not available for spawned processes.

3.5.4.1. Environment Variables

I_MPI_FAULT_CONTINUE

Turn on/off support for fault tolerant applications.

Syntax

`I_MPI_FAULT_CONTINUE=<arg>`

Arguments

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

Description

Set this environment variable to provide support for fault tolerant applications.

3.5.4.2. Usage Model

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

```
if(live_ranks[rank]) {

      mpi_err = MPI_Send(buf, count, dtype, rank, tag, MPI_COMM_WORLD);

if(mpi_err != MPI_SUCCESS) {

            live_ranks[rank] = 0;

}

}
```

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

3.5.5. Statistics Gathering Mode

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

- Native statistics format
- IPM statistics format

3.5.5.1. 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. Expand values of `I_MPI_STATS` environment variable additionally to existing values.

Syntax

```
I_MPI_STATS=[n-] m
```

Arguments

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

Description

Set this 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.

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.

3.5.5.2. 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.

The Intel® MPI Library also supports an optional `ipm` region feature. This feature requires the source code modification. The `MPI_Pcontrol` function can be used.

Region Control

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 in output.
- Complementary region contains statistics information not included into any named region. The region gets the `"ipm_noregion"` name in output.

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.

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

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`.

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 4.2-1 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_Group_compare
all2one	*MPI_Group_difference*
MPI_Gather	*MPI_Group_excl*
MPI_Gatherv	*MPI_Group_free*
MPI_Reduce	*MPI_Group_incl*
	MPI_Group_intersection
attr	*MPI_Group_range_excl*
MPI_Comm_create_keyval	*MPI_Group_range_incl*
MPI_Comm_delete_attr	*MPI_Group_rank*
MPI_Comm_free_keyval	*MPI_Group_size*
MPI_Comm_get_attr	*MPI_Group_translate_ranks*
MPI_Comm_set_attr	*MPI_Group_union*
MPI_Comm_get_name	
MPI_Comm_set_name	**init**
MPI_Type_create_keyval	*MPI_Init*

MPI_Type_delete_attr	*MPI_Init_thread*
MPI_Type_free_keyval	*MPI_Finalize*
MPI_Type_get_attr	
MPI_Type_get_name	**io**
MPI_Type_set_attr	*MPI_File_close*
MPI_Type_set_name	*MPI_File_delete*
MPI_Win_create_keyval	*MPI_File_get_amode*
MPI_Win_delete_attr	*MPI_File_get_atomicity*
MPI_Win_free_keyval	*MPI_File_get_byte_offset*
MPI_Win_get_attr	*MPI_File_get_group*
MPI_Win_get_name	*MPI_File_get_info*
MPI_Win_set_attr	*MPI_File_get_position*
MPI_Win_set_name	*MPI_File_get_position_shared*
MPI_Get_processor_name	*MPI_File_get_size*
	MPI_File_get_type_extent
comm	*MPI_File_get_view*
MPI_Comm_compare	*MPI_File_iread_at*
MPI_Comm_create	*MPI_File_iread*
MPI_Comm_dup	*MPI_File_iread_shared*
MPI_Comm_free	*MPI_File_iwrite_at*
MPI_Comm_get_name	*MPI_File_iwrite*
MPI_Comm_group	*MPI_File_iwrite_shared*
MPI_Comm_rank	*MPI_File_open*
MPI_Comm_remote_group	*MPI_File_preallocate*
MPI_Comm_remote_size	*MPI_File_read_all_begin*
MPI_Comm_set_name	*MPI_File_read_all_end*
MPI_Comm_size	*MPI_File_read_all*
MPI_Comm_split	*MPI_File_read_at_all_begin*
MPI_Comm_test_inter	*MPI_File_read_at_all_end*
MPI_Intercomm_create	*MPI_File_read_at_all*
MPI_Intercomm_merge	*MPI_File_read_at*

	MPI_File_read
err	*MPI_File_read_ordered_begin*
MPI_Add_error_class	*MPI_File_read_ordered_end*
MPI_Add_error_code	*MPI_File_read_ordered*
MPI_Add_error_string	*MPI_File_read_shared*
MPI_Comm_call_errhandler	*MPI_File_seek*
MPI_Comm_create_errhandler	*MPI_File_seek_shared*
MPI_Comm_get_errhandler	*MPI_File_set_atomicity*
MPI_Comm_set_errhandler	*MPI_File_set_info*
MPI_Errhandler_free	*MPI_File_set_size*
MPI_Error_class	*MPI_File_set_view*
MPI_Error_string	*MPI_File_sync*
MPI_File_call_errhandler	*MPI_File_write_all_begin*
MPI_File_create_errhandler	*MPI_File_write_all_end*
MPI_File_write_at_all	*MPI_File_write_all*
MPI_File_write_at	*MPI_File_write_at_all_begin*
MPI_File_write	*MPI_File_write_at_all_end*
MPI_File_write_ordered_begin	*MPI_Ibsend*
MPI_File_write_ordered_end	*MPI_Irsend*
MPI_File_write_ordered	*MPI_Issend*
MPI_File_write_shared	*MPI_Send_init*
MPI_Register_datarep	*MPI_Bsend_init*
	MPI_Rsend_init
one2all	*MPI_Ssend_init*
MPI_Bcast	
MPI_Scatter	*sendrecv*
MPI_Scatterv	*MPI_Sendrecv*
	MPI_Sendrecv_replace
recv	
MPI_Recv	**serv**
MPI_Irecv	*MPI_Alloc_mem*

MPI_Recv_init	*MPI_Free_mem*
MPI_Probe	*MPI_Buffer_attach*
MPI_Iprobe	*MPI_Buffer_detach*
	MPI_Op_create
req	*MPI_Op_free*
MPI_Start	
MPI_Startall	**spawn**
MPI_Wait	*MPI_Close_port*
MPI_Waitall	*MPI_Comm_accept*
MPI_Waitany	*MPI_Comm_connect*
MPI_Waitsome	*MPI_Comm_disconnect*
MPI_Test	*MPI_Comm_get_parent*
MPI_Testall	*MPI_Comm_join*
MPI_Testany	*MPI_Comm_spawn*
MPI_Testsome	*MPI_Comm_spawn_multiple*
MPI_Cancel	*MPI_Lookup_name*
MPI_Grequest_start	*MPI_Open_port*
MPI_Grequest_complete	*MPI_Publish_name*
MPI_Request_get_status	*MPI_Unpublish_name*
MPI_Request_free	
	status
rma	*MPI_Get_count*
MPI_Accumulate	*MPI_Status_set_elements*
MPI_Get	*MPI_Status_set_cancelled*
MPI_Put	*MPI_Test_cancelled*
MPI_Win_complete	
MPI_Win_create	**sync**
MPI_Win_fence	*MPI_Barrier*
MPI_Win_free	
MPI_Win_get_group	
MPI_Win_lock	**time**

MPI_Win_post	*MPI_Wtick*
MPI_Win_start	*MPI_Wtime*
MPI_Win_test	
MPI_Win_unlock	**topo**
MPI_Win_wait	*MPI_Cart_coords*
	MPI_Cart_create
scan	*MPI_Cart_get*
MPI_Exscan	*MPI_Cart_map*
MPI_Scan	*MPI_Cart_rank*
	MPI_Cart_shift
send	*MPI_Cart_sub*
MPI_Send	*MPI_Cartdim_get*
MPI_Bsend	*MPI_Dims_create*
MPI_Rsend	*MPI_Graph_create*
MPI_Ssend	*MPI_Graph_get*
MPI_Isend	*MPI_Graph_map*
	MPI_Graph_neighbors
	MPI_Graphdims_get
	MPI_Graph_neighbors_count
	MPI_Topo_test
	type
	MPI_Get_address
	MPI_Get_elements
	MPI_Pack
	MPI_Pack_external
	MPI_Pack_external_size
	MPI_Pack_size
	MPI_Type_commit
	MPI_Type_contiguous
	MPI_Type_create_darray

	MPI_Type_create_hindexed *MPI_Type_create_hvector* *MPI_Type_create_indexed_block* *MPI_Type_create_resized* *MPI_Type_create_struct* *MPI_Type_create_subarray* *MPI_Type_dup* *MPI_Type_free* *MPI_Type_get_contents* *MPI_Type_get_envelope* *MPI_Type_get_extent* *MPI_Type_get_true_extent* *MPI_Type_indexed* *MPI_Type_size* *MPI_Type_vector* *MPI_Unpack_external* *MPI_Unpack*

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
```

Stats output:

```
################################################################################
#
# command : ./a.out (completed)
# host    : svlmpihead01/x86_64_Linux       mpi_tasks : 4 on 1 nodes
# start   : 05/25/11/05:44:13               wallclock : 0.092012 sec
# stop    : 05/25/11/05:44:13               %comm     : 98.94
# gbytes  : 0.00000e+00 total               gflop/sec : NA
#
################################################################################
```

```
Command:
mpiexec -n 4 -env I_MPI_STATS ipm ./a.out
Stats output:
################################################################################
#
# command : ./a.out (completed)
```

```
# host    : svlmpihead01/x86_64_Linux       mpi_tasks : 4 on 1 nodes
# start   : 05/25/11/05:44:13               wallclock : 0.092012 sec
# stop    : 05/25/11/05:44:13               %comm     : 98.94
# gbytes  : 0.00000e+00 total               gflop/sec : NA
#
################################################################################
# region  : *   [ntasks] = 4
#
#                         [total]       <avg>         min           max
# entries                 4             1             1             1
# wallclock               0.332877      0.0832192     0.0732641     0.0920119
# user                    0.047992      0.011998      0.006999      0.019996
# system                  0.013997      0.00349925    0.002999      0.004
# mpi                     0.329348      0.082337      0.0723064     0.0912335
# %comm                                 98.9398       98.6928       99.154
# gflop/sec               NA            NA            NA            NA
# gbytes                  0             0             0             0
#
#
#                         [time]        [calls]       <%mpi>        <%wall>
# MPI_Init                0.236192      4             71.71         70.95
# MPI_Reduce              0.0608737     8000          18.48         18.29
# MPI_Barrier             0.027415      800           8.32          8.24
# MPI_Recv                0.00483489    1             1.47          1.45
# MPI_Send                1.50204e-05   1             0.00          0.00
# MPI_Wtime               1.21593e-05   8             0.00          0.00
# MPI_Finalize            3.33786e-06   4             0.00          0.00
# MPI_Comm_rank           1.90735e-06   4             0.00          0.00
# MPI_TOTAL               0.329348      8822          100.00        98.94
################################################################################
# region  : reduce   [ntasks] = 4
#
#                         [total]       <avg>         min           max
```

```
# entries                 8            2            2            2
# wallclock               0.0638561    0.015964     0.00714302   0.0238571
# user                    0.034994     0.0087485    0.003999     0.015997
# system                  0.003999     0.00099975   0            0.002999
# mpi                     0.0608799    0.01522      0.00633883   0.0231845
# %comm                                95.3392      88.7417      97.1808
# gflop/sec               NA           NA           NA           NA
# gbytes                  0            0            0            0
#
#
#                         [time]       [calls]      <%mpi>       <%wall>
# MPI_Reduce              0.0608737    8000         99.99        95.33
# MPI_Finalize            3.33786e-06  4            0.01         0.01
# MPI_Wtime               2.86102e-06  4            0.00         0.00
# MPI_TOTAL               0.0608799    8008         100.00       95.34
################################################################################
# region  : send   [ntasks] = 4
#
#                         [total]      <avg>        min          max
# entries                 1            0            0            1
# wallclock               2.89876e-05  7.24691e-06  1e-06        2.59876e-05
# user                    0            0            0            0
# system                  0            0            0            0
# mpi                     1.50204e-05  3.75509e-06  0            1.50204e-05
# %comm                                51.8165      0            57.7982
# gflop/sec               NA           NA           NA           NA
# gbytes                  0            0            0            0
#
#
#                         [time]       [calls]      <%mpi>       <%wall>
# MPI_Send                1.50204e-05  1            100.00       51.82
################################################################################
# region  : ipm_noregion   [ntasks] = 4
```

```
#
#                           [total]         <avg>           min             max
# entries                   13              3               3               4
# wallclock                 0.26898         0.0672451       0.0661182       0.068152
# user                      0.012998        0.0032495       0.001           0.004999
# system                    0.009998        0.0024995       0               0.004
# mpi                       0.268453        0.0671132       0.0659676       0.068049
# %comm                                     99.8039         99.7721         99.8489
# gflop/sec                 NA              NA              NA              NA
# gbytes                    0               0               0               0
#
#
#                           [time]          [calls]         <%mpi>          <%wall>
# MPI_Init                  0.236192        4               87.98           87.81
# MPI_Barrier               0.027415        800             10.21           10.19
# MPI_Recv                  0.00483489      1               1.80            1.80
# MPI_Wtime                 9.29832e-06     4               0.00            0.00
# MPI_Comm_rank             1.90735e-06     4               0.00            0.00
# MPI_TOTAL                 0.268453        813             100.00          99.80

################################################################################
###ILP64 Support
```

3.5.6. ILP64 Support

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

3.5.6.1. Using ILP64

Use the following options to enable the ILP64 interface

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

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

$ mpiifort -ilp64 -o test test.o
```

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

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

- Use the C compiler driver option `-ilp64` for both compilation (to choose an appropriate `mpi.h` header) and linkage (to link against ILP64 binary). For example,

  ```
  $ mpiicc -ilp64 -c test.c
  $ mpiicc -ilp64 -o test test.o
  ```

  If you use ILP64 data model in the C program, be cautious about mutual correspondence between C datatypes and MPI datatypes, namely, use `MPI_LONG` datatype while operating on data of type long int (or any other 64-bit integer type).

  If you want your C program to be LP64/ILP64 portable by specifying a flexible size integer datatype, don't forget to introduce that flexibility for the MPI integer datatype. This example program is supposed to work correctly both in LP64 and ILP64 data models:

  ```
  #include <stdlib.h>
  #include <stdio.h>
  #include "mpi.h"
  #ifdef ILP64
  #define my_int long
  #define MY_MPI_INT MPI_LONG
  #else
  #define my_int int
  #define MY_MPI_INT MPI_INT
  #endif
  int main() {
  my_int i, size, rank, buf[5] = {-1, -1, -1, -1, -1};
  MPI_Init( NULL, NULL );
  MPI_Comm_size( MPI_COMM_WORLD, &size );
  MPI_Comm_rank( MPI_COMM_WORLD, &rank );
  if( size < 2 ) return 1;
  if( rank == 0 ) {
  for( i = 0; i < 5; i++ ) buf[i] = i;
  MPI_Send( buf, 5, MY_MPI_INT, 1, 123, MPI_COMM_WORLD );
  } else if( rank == 1 ) {
  MPI_Recv( buf, 5, MY_MPI_INT, 0, 123, MPI_COMM_WORLD, MPI_STATUS_IGNORE );
  ```

```
for( i = 0; i < 5; i++ ) printf( "Received %d in buf[%d]\n", (int)buf[i],
(int)i );

}

MPI_Finalize();

return 0;

}
```

The program will work correctly in both variants:

```
$ mpiicc -o test_lp64 test.c ; mpirun -n 2 ./test_lp64

Received 0 in buf[0]

Received 1 in buf[1]

Received 2 in buf[2]

Received 3 in buf[3]

Received 4 in buf[4]

$ mpiicc -o test_ilp64 -DILP64 -ilp64 test.c ; mpirun -n 2 ./test_ilp64

Received 0 in buf[0]

Received 1 in buf[1]

Received 2 in buf[2]

Received 3 in buf[3]

Received 4 in buf[4]
```

3.5.6.2. 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.

3.5.7. Unified Memory Management

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

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

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

The respective standard C library functions are used by default.

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

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

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

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

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

3.5.8. File System Support

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

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

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

```
$ mpiexec -env I_MPI_EXTRA_FILESYSTEM on \

-env I_MPI_EXTRA_FILESYSTEM_LIST panfs -n 2 ./test
```

3.5.8.1. Environment Variables

I_MPI_EXTRA_FILESYSTEM

Turn on/off native parallel file systems support.

Syntax

```
I_MPI_EXTRA_FILESYSTEM=<arg>
```

Arguments

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

Description

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

I_MPI_EXTRA_FILESYSTEM_LIST

Select specific file systems support.

Syntax

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

Arguments

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

Description

Set this environment variable to request support for the specific parallel file system. This environment variable is handled only if the I_MPI_EXTRA_FYLESYSTEM is enabled. The Intel® MPI

Library will try to load shared modules to support the file systems specified by `I_MPI_EXTRA_FILESYSTEM_LIST`.

3.5.9. Multi-threaded memcpy Support

This topic provides information on how to use a multi-threaded version of *memcpy* implemented in the Intel® MPI Library for Intel® Xeon Phi™ Coprocessors. You can use this experimental feature to reach higher memory bandwidth between the ranks communicated through shared memory for some applications.

I_MPI_MT_MEMCPY

Controls usage of the multi-threaded *memcpy*.

Syntax

`I_MPI_MT_MEMCPY=<value>`

Arguments

<value>	Controls the usage of the multi-threaded *memcpy*
`enable | yes | on | 1`	Enable the multi-threaded *memcpy* in the single threaded version of the Intel® MPI Library (`MPI_THREAD_SINGLE`). This configuration is ignored for the thread safe version of Intel® MPI Library
`disable | no | off | 0`	Disable the usage of the multi-threaded *memcpy*. This is the default value

Description

Set this environment variable to control whether to use multi-threaded version of *memcpy* for intra-node communication.

I_MPI_MT_MEMCPY_NUM_THREADS

Change the number of threads involved in performing multi-threaded *memcpy*.

Syntax

`I_MPI_MT_MEMCPY_NUM_THREADS=<num>`

Arguments

<num>	The number of threads involved in performing multi-threaded *memcpy*
`>0`	The default value is the lesser of 8 and the number of physical cores within the MPI process pinning domain

Description

Use this environment variable to set the number of threads which perform *memcpy* operations per each MPI rank. The value `1` is equivalent to the setting `I_MPI_MT_MEMCPY=disable`.

I_MPI_MT_MEMCPY_THRESHOLD

Change the threshold for using multi-threaded memcpy.

Syntax

`I_MPI_MT_MEMCPY_THRESHOLD=<nbytes>`

Arguments

<nbytes>	Define the multi-threaded *memcpy* threshold in bytes
>0	The default value is 32768

Description

Set this environment variable to control the threshold for using multi-threaded *memcpy*. If the threshold is larger than the shared memory buffer size (for example, see I_MPI_SHM_LMT_BUFFER_SIZE or I_MPI_SSHM_BUFFER_SIZE), multi-threaded *memcpy* will never be used. The usage of multi-threaded *memcpy* is selected according to the following scheme:

- Buffers shorter than or equal to <nbytes> are sent using the serial version of *memcpy*. This approach is faster for short and medium buffers.
- Buffers larger than <nbytes> are sent using the multi-threaded *memcpy*. This approach is faster for large buffers.

I_MPI_MT_MEMCPY_SPIN_COUNT

Control the spin count value.

Syntax

`I_MPI_MT_MEMCPY_SPIN_COUNT=<scount>`

Arguments

<scount>	Define the loop spin count when a thread waits for data to copy before sleeping
>0	The default value is equal to 100000. The maximum value is equal to 2147483647

Description

Set the spin count limit for the loop for waiting for data to be copied by the thread. When the limit is exceeded and there is no data to copy, the thread goes to sleep.

Use the I_MPI_MT_MEMCPY_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.

4. Glossary

cell	A pinging resolution in descriptions for pinning property.
hyper-threading technology	A feature within the IA-32, 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.

5. Index