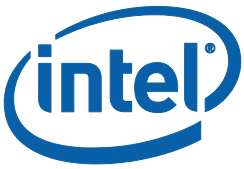


Intel® MPI Library for Windows* OS

Developer Reference

Contents

Legal Information

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

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

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

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Intel technologies features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at Intel.com, or from the OEM or retailer.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel, the Intel logo, Xeon, and Xeon Phi are trademarks of Intel Corporation in the U.S. and/or other countries.

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

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

1. Introduction

This *Developer Reference* provides you with the complete reference for the Intel® MPI Library. It is intended to help an experienced user fully utilize the Intel MPI Library functionality. You can freely redistribute this document in any desired form.

1.1. Introducing Intel® MPI Library

Intel® MPI Library is a multi-fabric message passing library that implements the Message Passing Interface, v3.1 (MPI-3.1) specification. It provides a standard library across Intel® platforms that enable adoption of MPI-3.1 functions as their needs dictate.

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.

Intel® MPI Library comprises the following main components:

- The *Intel® MPI Library Runtime Environment* (RTO) has the tools you need to run programs, including scalable process management system (Hydra*) and supporting utilities, dynamic (`.dll`) libraries, and documentation.
- The *Intel® MPI Library Software Development Kit* (SDK) includes all of the Runtime Environment components plus compilation tools, including compiler drivers such as `mpiicc`, include files and modules, debug libraries, program database (`.pdb`) files, and test codes.

You can get the latest information of Intel® MPI Library at https://software.intel.com/intel-mpi-library.

1.2. What's New

This document reflects the updates for Intel® MPI Library 2018 release for Windows* OS:

The following latest changes in this document were made:

Intel MPI Library 2018

- Removed support of the Intel® Xeon Phi™ coprocessors (formerly code named Knights Corner).
- Changes in environment variables:
    - `I_MPI_DAPL_TRANSLATION_CACHE` is now disabled by default

Intel MPI Library 2017 Update 2

- Added the environment variable `I_MPI_HARD_FINALIZE` in Other Environment Variables.

Intel MPI Library 2017 Update 1

- Topology-aware collective communication algorithms support (I_MPI_ADJUST Family).
- Added a new algorithm for `I_MPI_ADJUST_GATHER` and related environment variable `I_MPI_ADJUST_GATHER_SEGMENT` (I_MPI_ADJUST Family).
- Added the environment variable `I_MPI_PORT_RANGE` in Hydra Environment Variables.

Intel MPI Library 2017

- Document layout changes.

1.3. Notational Conventions

The following conventions are used in this document.

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

1.4. Related Information

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

- Product Web Site
- Intel® MPI Library Support
- Intel® Cluster Tools Products
- Intel® Software Development Products

2. Command Reference

2.1. Compiler Commands

(SDK only)

The following table lists the available Intel® MPI Library compiler commands with their underlying compilers and programming languages.

Table 2.1-1 Intel® MPI Library Compiler Wrappers

Compiler Command	Underlying Compiler	Supported Language(s)
Common Compilers		
`mpicc.bat`	`cl.exe`	C
`mpicxx.bat`	`cl.exe`	C++
`mpifc.bat`	`ifort.exe`	Fortran 77/Fortran 95
Microsoft* Visual C++* Compilers		
`mpicl.bat`	`cl.exe`	C/C++
Intel® Fortran, C++ Compilers		
`mpiicc.bat`	`icl.exe`	C
`mpiicpc.bat`	`icl.exe`	C++
`mpiifort.bat`	`ifort.exe`	Fortran 77/Fortran 95

NOTES:

- Compiler commands are available only in the Intel® MPI Library Software Development Kit (SDK).
- For the supported versions of the listed compilers, refer to the *Release Notes*.
- Compiler wrapper scripts are located in the `<installdir>\intel64\bin` directory.
- The environment settings can be established by running the `<installdir>\intel64\bin\mpivars.bat` script. If you need to use a specific library configuration, you can pass one of the following arguments to the `mpivars.bat` script to switch to the corresponding configuration: `debug`, `release`, `debug_mt`, or `release_mt`. The multi-threaded optimized library is chosen by default.
- Ensure that the corresponding underlying compiler is already in your `PATH`. If you use the Intel® Compilers, run the `compilervars.bat` script from the installation directory to set up the compiler environment.
- To display mini-help of a compiler command, execute it without any parameters.

2.1.1. Compiler Command Options

-profile=<*profile_name*>

Use this option to specify an MPI profiling library. <*profile_name*> is the name of the configuration file (profile) that loads the corresponding profiling library. The profiles are taken from <*installdir*>\<*arch*>\etc.

You can create your own profile as <*installdir*>\<*arch*>\etc\<*profile_name*>.conf. You can define the following environment variables in a configuration file:

- PROFILE_PRELIB – libraries (and paths) to load before the Intel® MPI Library
- PROFILE_POSTLIB – libraries to load after the Intel® MPI Library
- PROFILE_INCPATHS – C preprocessor arguments for any include files

For example, create a file <*installdir*>\<*arch*>\etc\myprof.conf with the following lines:

```
SET PROFILE_PRELIB=<path_to_myprof>\lib\myprof.lib
SET PROFILE_INCPATHS=-I"<paths_to_myprof>\include"
```

Use the -profile=myprof option for the relevant compiler wrapper to select this new profile.

-t or -trace

Use the -t or -trace option to link the resulting executable file against the Intel® Trace Collector library.

To use this option, include the installation path of the Intel® Trace Collector in the VT_ROOT environment variable. Source the itacvars.bat script provided in the Intel® Trace Analyzer and Collector installation folder.

-check_mpi

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

To use this option, include the installation path of the Intel® Trace Collector in the VT_ROOT environment variable. Source the itacvars.bat script provided in the Intel® Trace Analyzer and Collector installation folder.

-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.

-no_ilp64

Use this option to disable the ILP64 support explicitly. This option must be used in conjunction with -i8 option of Intel® Fortran Compiler.

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.

-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 -Zi.

/Zi, /Z7 or /ZI

Use these options to compile a program in debug mode and link the resulting executable against the debugging version of the Intel® MPI Library. See `I_MPI_DEBUG` for information on how to use additional debugging features with the `/Zi`, `/Z7`, `/ZI` or `debug` builds.

NOTE

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

-O

Use this option to enable compiler optimization.

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

-echo

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

-show

Use this option to learn how the underlying compiler is invoked, without actually running it. 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:

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

-show_env

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

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

Use this option to select the underlying compiler.

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

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

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

NOTE

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

-v

Use this option to print the compiler wrapper script version.

2.1.2. Compilation Environment Variables

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

Specify the default profiling library.

Syntax

I_MPI_CC_PROFILE=<profile_name>

I_MPI_CXX_PROFILE=<profile_name>

I_MPI_FC_PROFILE=<profile_name>

I_MPI_F77_PROFILE=<profile_name>

I_MPI_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 for mpiicc or another Intel® MPI Library compiler wrapper.

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=<compiler>

I_MPI_CXX=<compiler>

I_MPI_FC=<compiler>

I_MPI_F77=<compiler>

I_MPI_F90=<compiler>

Arguments

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

Description

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

NOTE

Some compilers may require additional command line options.

I_MPI_ROOT

Set the Intel® MPI Library installation directory path.

Syntax

I_MPI_ROOT=<path>

Arguments

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

Description

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

VT_ROOT

Set Intel® Trace Collector installation directory path.

Syntax

`VT_ROOT=<path>`

Arguments

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

Description

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

I_MPI_COMPILER_CONFIG_DIR

Set the location of the compiler configuration files.

Syntax

`I_MPI_COMPILER_CONFIG_DIR=<path>`

Arguments

<path>	Specify the location of the compiler configuration files. The default value is *<installdir>**<arch>*\etc

Description

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

I_MPI_LINK

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

Syntax

`I_MPI_LINK=<arg>`

Arguments

<arg>	Version of library
`opt`	The optimized, single threaded version of the Intel® MPI Library
`opt_mt`	The optimized, multithreaded version of the Intel MPI Library
`dbg`	The debugging, single threaded version of the Intel MPI Library
`dbg_mt`	The debugging, multithreaded version of the Intel MPI Library
`opt_mt_compat`	The optimized, multithreaded version of the Intel MPI Library (backward compatibility mode)

`dbg_compat`	The debugging, single threaded version of the Intel MPI Library (backward compatibility mode)
`dbg_mt_compat`	The debugging, multithreaded version of the Intel MPI Library (backward compatibility mode)

Description

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

NOTE

The backward compatibility mode is used for linking with old Intel MPI Library names (`impimt.dll`, `impid.dll`, and `impidmt.dll`).

2.2. Hydra Process Manager Commands

2.2.1. Hydra Service

hydra_service

Hydra Service agent.

Syntax

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

Arguments

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

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

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

`-remove_spn`	Remove SPN from the Windows* domain for the cluster node on which this command is executed

Description

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

Examples

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

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

NOTE

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

2. Use the following command to remove the service:

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

2.2.2. Job Startup Command

mpiexec

The `mpiexec` utility is a scalable MPI process manager for running MPI applications.

Syntax

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

or

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

Arguments

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

Description

Use the `mpiexec` utility to run MPI applications.

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 `test.exe` over the specified processes and hosts:

```
> mpiexec -f <hostfile> -n <# of processes> test.exe
```

where:

- *`<# of processes>`* specifies the number of processes on which to run the `test.exe` executable
- *`<hostfile>`* specifies a list of hosts on which to run the `test.exe` 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 -f <hostfile> -env <VAR1> <VAL1> -n 2 prog1.exe : ^
-env <VAR2> <VAL2> -n 2 prog2.exe
```

NOTE

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

2.2.3. Global Options

This section describes the global options of the Intel® MPI Library's Hydra process manager. Global options are applied to all arguments sets in the launch command. Argument sets are separated by a colon ':'.

-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.

- Use the `-perhost`, `-ppn`, and `-grr` options to place consecutive MPI processes on every host using the round robin scheduling.
- Use the `-rr` option to place consecutive MPI processes on different hosts using the round robin scheduling.

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

Use this option to control process placement through a machine file. To define the total number of processes to start, use the `-n` option. To pin processes within a machine, use the option `binding=map` in the machine file. For example:

```
> type machinefile
node0:2 binding=map=0,3
node1:2 binding=map=[2,8]
node0:1 binding=map=8
```

For details on using the `binding` option, see Binding 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.

-genvexcl *<list of env var names>*

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

-genvlist *<list>*

Use this option to pass a list of environment variables with their current values. *<list>* is a comma separated list of environment variables to be sent to all MPI processes.

-pmi-connect *<mode>*

Use this option to choose the caching mode of process management interface (PMI) message. Possible values for *<mode>* are:

<mode>	The caching mode 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-request propagation of the PMI information.
`alltoall`	Information is automatically exchanged between all `pmi_proxy` before any get request can be done. This 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 specified 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.

NOTE

When running under a job scheduler, these options are ignored by default. To be able to control process placement with these options, disable the `I_MPI_JOB_RESPECT_PROCESS_PLACEMENT` variable.

-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.

-trace-pt2pt

Use this option to collect the information about point-to-point operations using Intel® Trace Analyzer and Collector. The option requires that your application be linked against the Intel® Trace Collector profiling library.

-trace-collectives

Use this option to collect the information about collective operations using Intel® Trace Analyzer and Collector. The option requires that your application be linked against the Intel® Trace Collector profiling library.

NOTE

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

-configfile *<filename>*

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

-branch-count *<num>*

Use this option to restrict the number of child management processes launched by the Hydra process manager, 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.

-nolocal

Use this option to avoid running the *<executable>* on the host where `mpiexec` 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 the MPI processes should be run. For example, the following command runs the executable `a.out` on the hosts `host1` and `host2`:

```
> mpiexec -n 2 -ppn 1 -hosts host1,host2 test.exe
```

NOTE

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

-iface *<interface>*

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

```
> mpiexec -n 2 -iface ib0 test.exe
```

See the `I_MPI_HYDRA_IFACE` environment variable for more details.

-l, -prepend-rank

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

-tune [*<arg >*]

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.

<arg> is the directory containing tuned settings or a configuration file that applies these settings. If *<arg>* is not specified, the most optimal settings are selected for the given configuration. The default location of the configuration file is *<installdir>*`/`*<arch>*`/etc` directory.

-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.

-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.

-version or -V

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

-info

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

-delegate

Use this option to enable the domain-based authorization with the delegation ability. See User Authorization for details.

-impersonate

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

-localhost

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

-localroot

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

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

-localonly

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

-register

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

-remove

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

-validate

Validate the encrypted credentials for the current host.

-whoami

Use this option to print the current user name.

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

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

-mapall

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

-logon

Use this option to force the prompt for user credentials.

-noprompt

Use this option to suppress the prompt for user credentials.

-port/-p

Use this option to specify the port that the service is listening on.

-verbose or -v

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

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

See the `I_MPI_HYDRA_DEBUG` environment variable for more details.

-print-rank-map

Use this option to print out the MPI rank mapping.

-print-all-exitcodes

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

Binding Option

-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.

NOTE

This option is related to the family of `I_MPI_PIN` environment variables, which have higher priority than the `-binding` option. Hence, if any of these variables are set, the option is ignored.

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

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

Parameter	
`cell`	Pinning resolution
Values	

`unit`	Basic processor unit (logical CPU)
`core`	Processor core in multi-core system

Parameter	
`map`	Process mapping
Values	
`spread`	The processes are mapped consecutively to separate processor cells. Thus, the processes do not share the common resources of the adjacent cells.
`scatter`	The processes are mapped to separate processor cells. Adjacent processes are mapped upon the cells that are the most remote in the multi-core topology.
`bunch`	The processes are mapped to separate processor cells by #processes/#sockets processes per socket. Each socket processor portion is a set of the cells that are the closest in the multi-core topology.
`p0,p1,...,pn`	The processes are mapped upon the separate processors according to the processor specification on the `p0,p1,...,pn` list: the `i`th process is mapped upon the processor `pi`, where `pi` takes one of the following values: • processor number like n • range of processor numbers like n-m • `-1` for no pinning of the corresponding process
`[m0,m1,...,mn]`	The `i`th process is mapped upon the processor subset defined by m_i hexadecimal mask using the following rule: The `j`th processor is included into the subset m_i if the `j`th bit of m_i equals `1`.

Parameter	
`domain`	Processor domain set on a node
Values	
`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>`*

Parameter	
`order`	Linear ordering of the domains
Values	
`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

Parameter	
`offset`	Domain list offset
Values	
`<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.

Bootstrap Options

-bootstrap *<bootstrap server>*

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

Arguments

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

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

-bootstrap-exec *<bootstrap server>*

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

```
> mpiexec -bootstrap-exec <bootstrap_server_executable> -f hostfile -env <VAR1>
<VAL1> -n 2 test.exe
```

See the `I_MPI_HYDRA_BOOTSTRAP` environment variable for more details.

2.2.4. Local Options

This section describes the local options of the Intel® MPI Library's Hydra process manager. Local options are applied only to the argument set they are specified in. Argument sets are separated by a colon '`:`'.

-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 argument set.

-envall

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

-envexcl *<list of env var names>*

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

-envlist *<list>*

Use this option to pass a list of environment variables with their current values. *<list>* 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 `test.exe` on hosts `host1` and `host2`:

```
> mpiexec -n 2 -host host1 test.exe : -n 2 -host host2 test.exe
```

-path *<directory>*

Use this option to specify the path to the *<executable>* file to be run in the current argument set.

-wdir *<directory>*

Use this option to specify the working directory in which the *<executable>* file runs in the current argument set.

-umask *<umask>*

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

-hostos *<host OS>*

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

Arguments

<arg>	String parameter
`linux`	The host with Linux* OS installed.
`windows`	The host with Windows* OS installed. This is the default value.

NOTE

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

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

2.2.5. Extended Fabric Control Options

This section lists the options used for quick selection of the communication fabrics. Using these options overrides the related environment variables, described in Fabrics Control.

-dapl, -rdma

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, the `tcp` fabric is used. This option is equivalent to the setting: `-genv I_MPI_FABRICS_LIST dapl,tcp -genv I_MPI_FALLBACK 1`.

-DAPL, -RDMA

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 setting: `-genv I_MPI_FABRICS_LIST dapl`.

2.2.6. Hydra 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>	The full or relative path to the host file

Description

Set this environment variable to specify the hosts file.

I_MPI_HYDRA_DEBUG

Print out the debug information.

Syntax

`I_MPI_HYDRA_DEBUG=<arg>`

Arguments

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

Description

Set this environment variable to enable the debug mode.

I_MPI_HYDRA_ENV

Control the environment propagation.

Syntax

`I_MPI_HYDRA_ENV=<arg>`

Arguments

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

Description

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

I_MPI_JOB_TIMEOUT, I_MPI_MPIEXEC_TIMEOUT (MPIEXEC_TIMEOUT)

Set the timeout period for `mpiexec`.

Syntax

`I_MPI_JOB_TIMEOUT=<timeout>`

`I_MPI_MPIEXEC_TIMEOUT=<timeout>`

Deprecated Syntax

`MPIEXEC_TIMEOUT=<timeout>`

Arguments

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

Description

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

NOTE

Set this environment variable in the shell environment before executing the `mpiexec` command. Setting the variable through the `-genv` and `-env` options has no effect.

I_MPI_HYDRA_BOOTSTRAP

Set the bootstrap server.

Syntax

`I_MPI_HYDRA_BOOTSTRAP=<arg>`

Arguments

`<arg>`	String parameter

`service`	Use hydra service agent
`ssh`	Use secure shell. This is the default value
`fork`	Use fork call

Description

Set this environment variable to specify the bootstrap server.

NOTE

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

I_MPI_HYDRA_BOOTSTRAP_EXEC

Set the executable file to be used as a bootstrap server.

Syntax

`I_MPI_HYDRA_BOOTSTRAP_EXEC=`*<arg>*

Arguments

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

Description

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

I_MPI_HYDRA_PMI_CONNECT

Define the processing method for PMI messages.

Syntax

`I_MPI_HYDRA_PMI_CONNECT=`*<value>*

Arguments

<value>	The algorithm to be used
`nocache`	Do not cache PMI messages
`cache`	Cache `PMI` messages on the local `pmi_proxy` management processes to minimize the number of `PMI` requests. Cached information is automatically propagated to child management processes.
`lazy-cache`	`cache` mode with on-demand propagation.
`alltoall`	Information is automatically exchanged between all `pmi_proxy` before any get request can be done. This is the default value.

Description

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

I_MPI_PMI2

Control the use of PMI-2 protocol.

Syntax

`I_MPI_PMI2=<arg>`

Arguments

<arg>	Binary indicator
`enable | yes | on | 1`	Enable PMI-2 protocol
`disable | no | off | 0`	Disable PMI-2 protocol. This is the default value

Description

Set this environment variable to control the use of PMI-2 protocol.

I_MPI_PERHOST

Define the default behavior for the `-perhost` option of the `mpiexec` command.

Syntax

`I_MPI_PERHOST=<value>`

Arguments

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

Description

Set this environment variable to define the default behavior for the `-perhost` option. Unless specified explicitly, the `-perhost` option is implied with the value set in `I_MPI_PERHOST`.

NOTE

When running under a job scheduler, this environment variable is ignored by default. To be able to control process placement with `I_MPI_PERHOST`, disable the `I_MPI_JOB_RESPECT_PROCESS_PLACEMENT` variable.

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 value also means that there is no hierarchical structure • The default value is 32 if more than 127 nodes are used

Description

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

I_MPI_HYDRA_PMI_AGGREGATE

Turn on/off aggregation of the PMI messages.

Syntax

I_MPI_HYDRA_PMI_AGGREGATE=<arg>

Arguments

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

Description

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

I_MPI_HYDRA_IFACE

Set the network interface.

Syntax

I_MPI_HYDRA_IFACE=*<arg>*

Arguments

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

Description

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

I_MPI_TMPDIR (TMPDIR)

Set the temporary directory.

Syntax

`I_MPI_TMPDIR=<arg>`

Arguments

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

Description

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

I_MPI_JOB_RESPECT_PROCESS_PLACEMENT

Specify whether to use the process-per-node placement provided by the job scheduler, or set explicitly.

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 the variable is set, the Hydra process manager uses the process placement provided by job scheduler (default). In this case the `-ppn` option and its equivalents are ignored. If you disable the variable, the Hydra process manager uses the process placement set with `-ppn` or its equivalents.

I_MPI_PORT_RANGE

Set allowed port range.

Syntax

`I_MPI_PORT_RANGE=<range>`

Arguments

<range>	String parameter
<min>:<max>	Allowed port range

Description

Set this environment variable to specify the allowed port range for the Intel® MPI Library.

2.3. Processor Information Utility

cpuinfo

The `cpuinfo` utility provides processor architecture information.

Syntax

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

Arguments

`<options>`	Sequence of one-letter options. Each option controls a specific part of the output data.
`g`	General information about single cluster node shows: • the processor product name • the number of packages/sockets on the node • core and threads numbers on the node and within each package • SMT mode enabling
`i`	Logical processors identification table identifies threads, cores, and packages of each logical processor accordingly. • *Processor* - logical processor number. • *Thread Id* - unique processor identifier within a core. • *Core Id* - unique core identifier within a package. • *Package Id* - unique package identifier within a node.
`d`	Node decomposition table shows the node contents. Each entry contains the information on packages, cores, and logical processors. • *Package Id* - physical package identifier. • *Cores Id* - list of core identifiers that belong to this package. • *Processors Id* - list of processors that belong to this package. This list order directly corresponds to the core list. A group of processors enclosed in brackets belongs to one core.
`c`	Cache sharing by logical processors shows information of sizes and processors groups, which share particular cache level. • Size - cache size in bytes. • Processors - a list of processor groups enclosed in the parentheses those share this cache or no sharing otherwise.
`s`	Microprocessor signature hexadecimal fields (Intel platform notation) show signature values: • extended family • extended model • family

	• model • type • stepping
`f`	Microprocessor feature flags indicate what features the microprocessor supports. The Intel platform notation is used.
`A`	Equivalent to `gidcsf`
`gidc`	Default sequence
`?`	Utility usage info

Description

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

NOTE

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

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

An example of the `cpuinfo` output:

```
> cpuinfo -gdcs
=====  Processor composition  =====
Processor name    : Intel(R) Xeon(R)  X5570
Packages(sockets) : 2
Cores             : 8
Processors(CPUs)  : 8
Cores per package : 4
Threads per core  : 1
=====  Processor identification  =====
Processor       Thread Id.      Core Id.        Package Id.
0               0               0               0
1               0               0               1
2               0               1               0
3               0               1               1
4               0               2               0
5               0               2               1
6               0               3               0
7               0               3               1
=====  Placement on packages  =====
Package Id.     Core Id.        Processors
0               0,1,2,3         0,2,4,6
1               0,1,2,3         1,3,5,7
=====  Cache sharing  =====
Cache   Size            Processors
L1      32  KB          no sharing
L2      256 KB          no sharing
L3      8   MB          (0,2,4,6)(1,3,5,7)
=====  Processor Signature  =====
```

```
 _________ _________ ______ _________ _______ ___________
| xFamily | xModel | Type | Family | Model | Stepping |
|_________|________|______|________|_______|__________|
| 00      | 1      | 0    | 6      | a     | 5        |
|_________|________|______|________|_______|__________|
```

3. Tuning Reference

3.1. 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 [options]
```

Options

`-a \"<app_cmd_line>\"` `--application \"<app_cmd_line>\"`	Enable the application-specific mode. Quote the full command line as shown, including the backslashes.
`-of <file-name>` `--output-file <file-name>`	Specify the name of the application configuration file to be generated in the application-specific mode. By default, use the file name `app.conf`.
`-t \"<test_cmd_line>\"` `--test \"<test_cmd_line>\"`	Replace the default Intel® MPI Benchmarks by the indicated benchmarking program in the cluster-specific mode. Quote the full command line as shown including the backslashes.
`-cm {exclusive|full}` `--cluster-mode {exclusive|full}`	Set the cluster usage mode: • `full` - maximum number of tasks are executed. This is the default mode. • `exclusive` - only one task is executed on the cluster at a time.
`-d | --debug`	Print out the debug information.
`-D | --distinct`	Tune all options separately from each other. This argument is applicable only for the cluster-specific mode.
`-dl [d1[,d2...[,dN]]]` `--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]]]` `--fabric-list [f1[,f2…[,fN]]]`	Select the fabric(s) you want to tune. Any previously set devices are ignored. By default, use all fabrics listed in the `<installdir>\<arch>\etc\fabrics.xml` file.
`-hf <hostsfile>` `--host-file <hostsfile>`	Specify an alternative host file name. By default, use the `mpd.hosts`.
`-h | --help`	Display the help message.

`-hr` *{min:max|min:|:max}* `--host-range` *{min:max|min:|:max}*	Set the range of hosts used for testing. The default minimum value is `1`. The default maximum value is the number of hosts defined by the `mpd.hosts`. The `min:` or `:max` format uses the default values as appropriate.
`-i` *<count>* `--iterations` *<count>*	Define how many times to run each tuning step. Higher iteration counts increase the tuning time, but may also increase the accuracy of the results. The default value is `3`.
`-mr` *{min:max|min:|:max}* `--message-range` *{min:max|min:|:max}*	Set the message size range. The default minimum value is `0`. The default maximum value is `4194304` (`4mb`). By default, the values are given in bytes. They can also be given in the following format: `16kb`, `8mb` or `2gb`. The `min:` or `:max` format uses the default values as appropriate.
`-od` *<outputdir>* `--output-directory` *<outputdir>*	Specify the directory name for all output files: log-files, session-files, local host-files and report-files. By default, use the current directory. This directory should be accessible from all hosts.
`-odr` *<outputdir>* `--output-directory-results` *<outputdir>*	Specify the directory name for the resulting configuration files. By default, use the current directory in the application-specific mode and the `<installdir>\<arch>\etc` in the cluster-specific mode. If `<installdir>\<arch>\etc` is unavailable, the current directory is used as the default value in the cluster-specific mode.
`-pr` *{min:max|min:|:max}* `--ppn-range` *{min:max|min:|:max}* `--perhost-range` *{min:max|min:|:max}*	Set the maximum number of processes per host. The default minimum value is `1`. The default maximum value is the number of cores of the processor. The `min:` or `:max` format uses the default values as appropriate.
`-sf` *[file-path]* `--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. Intel MPI Library uses the `mpitunertemp` folder in the current directory by default. This directory should be accessible from all hosts.
`-tl` *<minutes>* `--time-limit` *<minutes>*	Set `mpitune` execution time limit in minutes. The default value is `0`, which means no limitations.
`-mh | --master-host`	Dedicate a single host to run the `mpitune`.
`-os` *<opt1,...,optN>* `--options-set`	Use `mpitune` to tune the only required options you have set in the option values

<opt1,...,optN>	
`-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.
`-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`
`--session-file (-sf)`	None
`--show-session (-ss)`	None

Description

Use the `mpitune` utility to create a set of Intel® MPI Library configuration files that contain optimal settings for a particular cluster or application. You can reuse these configuration files in the `mpiexec` job launcher by using the `-tune` option. If configuration files from previous `mpitune` sessions exist, `mpitune` creates a copy of the existing files before starting execution.

The MPI tuner utility operates in two modes:

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

3.2. Process Pinning

Use this feature to pin a particular MPI process to a corresponding CPU within a node 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 to find out the logical CPU numbers.

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

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

NOTE

Logical and topological enumerations are not the same.

Table 3.2-1 Logical Enumeration

0	4	1	5	2	6	3	7

Table 3.2-2 Hierarchical Levels

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

Table 3.2-3 Topological Enumeration

0	1	2	3	4	5	6	7

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

3.2.2. Default Settings

If you do not specify values for any process pinning environment variables, the default settings below are used. For details about these settings, see Environment Variables and Interoperability with OpenMP API.

- `I_MPI_PIN=on`
- `I_MPI_PIN_MODE=pm`
- `I_MPI_PIN_RESPECT_CPUSET=on`
- `I_MPI_PIN_RESPECT_HCA=on`
- `I_MPI_PIN_CELL=unit`
- `I_MPI_PIN_DOMAIN=auto:compact`
- `I_MPI_PIN_ORDER=compact`

3.2.3. Environment Variables for Process Pinning

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 control the process pinning feature of the Intel® MPI Library.

I_MPI_PIN_PROCESSOR_LIST (I_MPI_PIN_PROCS)

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

Syntax

I_MPI_PIN_PROCESSOR_LIST=<value>

The environment variable value has the following syntax forms:

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

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

Deprecated Syntax

I_MPI_PIN_PROCS=<proclist>

NOTE

The postoffset keyword has offset alias.

NOTE

The second form of the pinning procedure has three steps:

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

NOTE

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

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

Syntax

`I_MPI_PIN_PROCESSOR_LIST=<proclist>`

Arguments

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

Syntax

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

Arguments

<procset>	Specify a processor subset based on the topological numeration. The default value is `allcores`.
`all`	All logical processors. Specify this subset to define the number of CPUs on a node.
`allcores`	All cores (physical CPUs). Specify this subset to define 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. Specify this subset to define the number of sockets on a node.

<grain>	Specify the pinning granularity cell for a defined *<procset>*. The minimal *<grain>* value is a single element of the *<procset>*. The maximal *<grain>* value is the number of *<procset>* elements in a socket. The *<grain>* value must be a multiple of the *<procset>* value. Otherwise, the minimal *<grain>* value is assumed. The default value is the minimal *<grain>* value.
<shift>	Specify the granularity of the round robin scheduling shift of the cells for the *<procset>*. *<shift>* is measured in the defined *<grain>* units. The *<shift>* value must be positive integer. Otherwise, no shift is performed. The default value is no shift, which is equal to `1` normal increment
<preoffset>	Specify the cyclic shift of the processor subset *<procset>* defined before the round robin shifting on the *<preoffset>* value. The value is measured in the defined *<grain>* units. The *<preoffset>* value must be non-negative integer. Otherwise, no shift is performed. The default value is no shift.
<postoffset>	Specify the cyclic shift of the processor subset *<procset>* derived after round robin shifting on the *<postoffset>* value. The value is measured in the defined *<grain>* units. The *<postoffset>* value must be non-negative integer. Otherwise no shift is

	performed. The default value is no shift.

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

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

Syntax

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

Arguments

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

Description

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

NOTE

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

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

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

  For example, process mapping for `I_MPI_PIN_PROCESSOR_LIST=p0,p1,p2,...,pn` is as follows:

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

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

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

Legend:

gray - MPI process grains

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

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

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

D) Final map table ordered by MPI ranks

A)

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

B)

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

C)

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

D)

0 1	2 3	...	2n-2 2n-	2n 2n+1	2n+2	...	4n-2 4n-	4n 4n+1	4n+2	...	6n-2

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

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

In the `bunch` scenario the processes are mapped proportionally to sockets as closely as possible. This mapping 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, and cores.

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

Legend:

gray - MPI processes

cyan - 1st socket processors

green - 2nd socket processors

Same color defines a processor pair sharing a cache

0	1	2			3	4		
0	1	2	3		4	5	6	7

`bunch` scenario for 5 processes

0	4	2	6		1	5	3	7
0	1	2	3		4	5	6	7

`scatter` scenario for full loading

Examples

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

```
> mpiexec -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:

```
> mpiexec -host host1 -env I_MPI_PIN_PROCESSOR_LIST=0,3 -n <# of processes>
<executable> :^
-host host2 -env I_MPI_PIN_PROCESSOR_LIST=1,2,3 -n <# of processes> <executable>
```

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

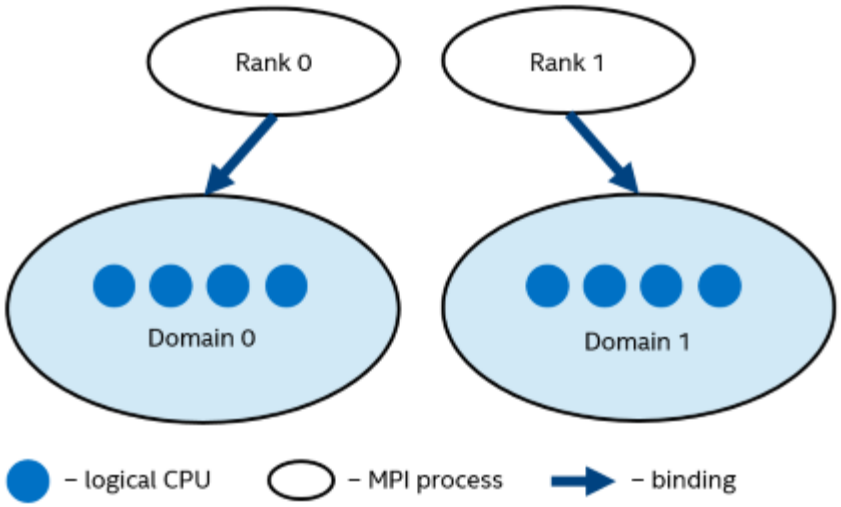
```
> mpiexec -genv I_MPI_DEBUG=4 -m -host host1 -env I_MPI_PIN_PROCESSOR_LIST=0,3 -n
<# of processes> <executable> :^
-host host2 -env I_MPI_PIN_PROCESSOR_LIST=1,2,3 -n <# of processes> <executable>
```

3.2.4. Interoperability with OpenMP* API

I_MPI_PIN_DOMAIN

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

Figure 3.2-1 Domain Example



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

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

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

The `I_MPI_PIN_DOMAIN` environment variable has the following syntax forms:

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

The following tables describe these syntax forms.

Multi-core Shape

`I_MPI_PIN_DOMAIN=`*<mc-shape>*

<mc-shape>	Define domains through multi-core terms.
`core`	Each domain consists of the logical processors that share a particular core. The number of domains on a node is equal to the number of cores on the node.
`socket | sock`	Each domain consists of the logical processors that share a particular socket. The number of domains on a node is equal to the number of sockets on the node. This is the recommended value.

`numa`	Each domain consists of the logical processors that share a particular NUMA node. The number of domains on a machine is equal to the number of NUMA nodes on the machine.
`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.

NOTE

If `Cluster on Die` is disabled on a machine, the number of NUMA nodes equals to the number of sockets. In this case, pinning for `I_MPI_PIN_DOMAIN = numa` is equivalent to pinning for `I_MPI_PIN_DOMAIN = socket`.

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, and so on). 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, and so on)

Explicit Domain Mask

`I_MPI_PIN_DOMAIN=`*`<masklist>`*

`<masklist>`	Define domains through the comma separated list of hexadecimal numbers (domain masks)

[m_1,...,m_n]	For <masklist>, each m_i is a hexadecimail bit mask defining an individual domain. The following rule is used: the i^{th} logical processor is included into the domain if the corresponding mi value is set to 1. All remaining processors are put into a separate domain. BIOS numbering is used. **NOTE** To ensure that your configuration in <masklist> is parsed correctly, use square brackets to enclose the domains specified by the <masklist>. For example: I_MPI_PIN_DOMAIN=[0x55,0xaa]

NOTE

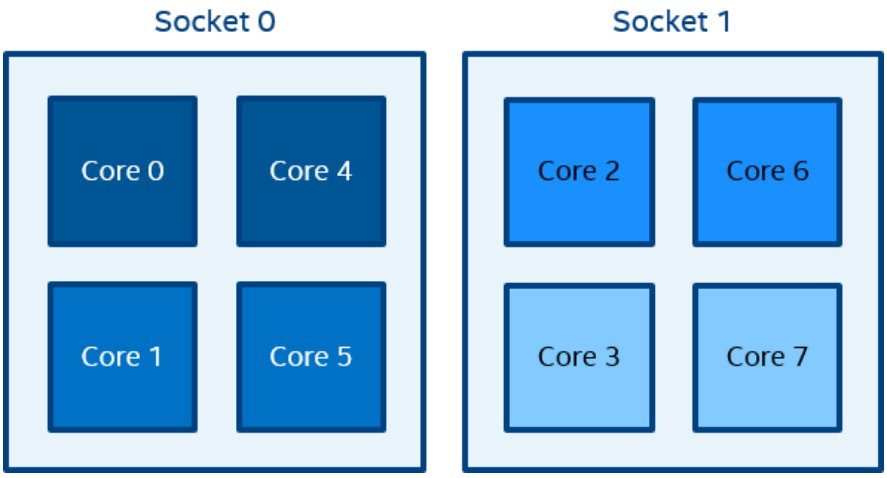
These options are available for both Intel® and non-Intel microprocessors, but they may perform additional optimizations for Intel microprocessors than they perform for non-Intel microprocessors.

NOTE

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

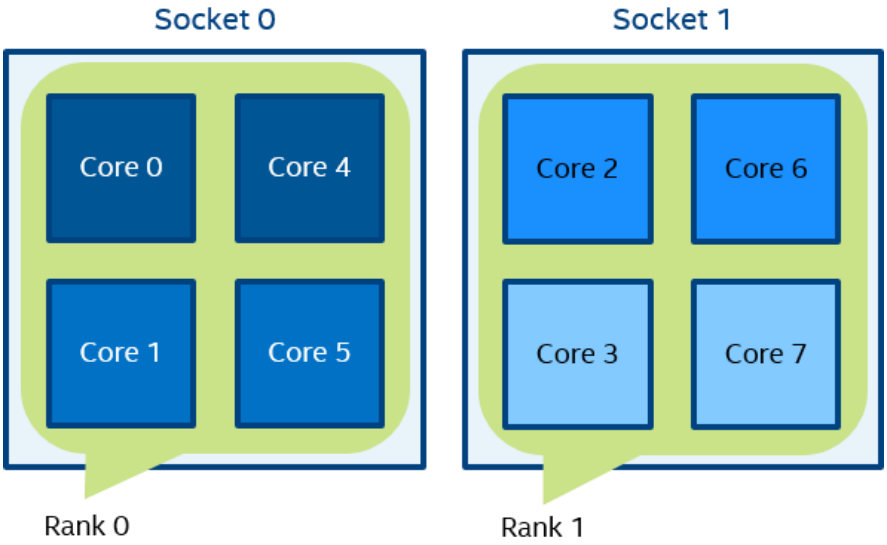
See the following model of a symmetric multiprocessing (SMP) node in the examples:

Figure 3.2-2 Model of a Node



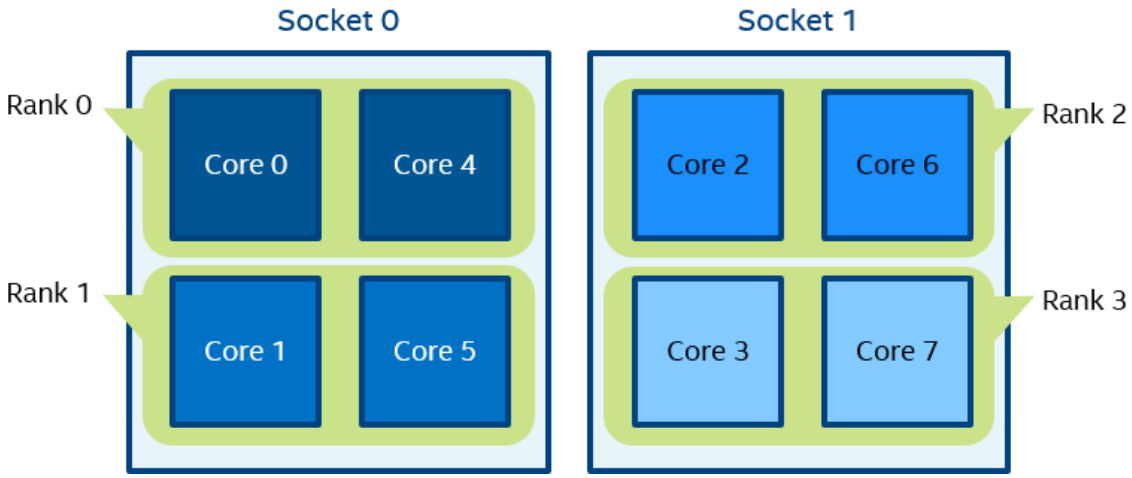
The figure above represents the SMP node model with a total of 8 cores on 2 sockets. Intel® Hyper-Threading Technology is disabled. Core pairs of the same color share the L2 cache.

Figure 3.2-3 `mpiexec -n 2 -env I_MPI_PIN_DOMAIN socket test.exe`



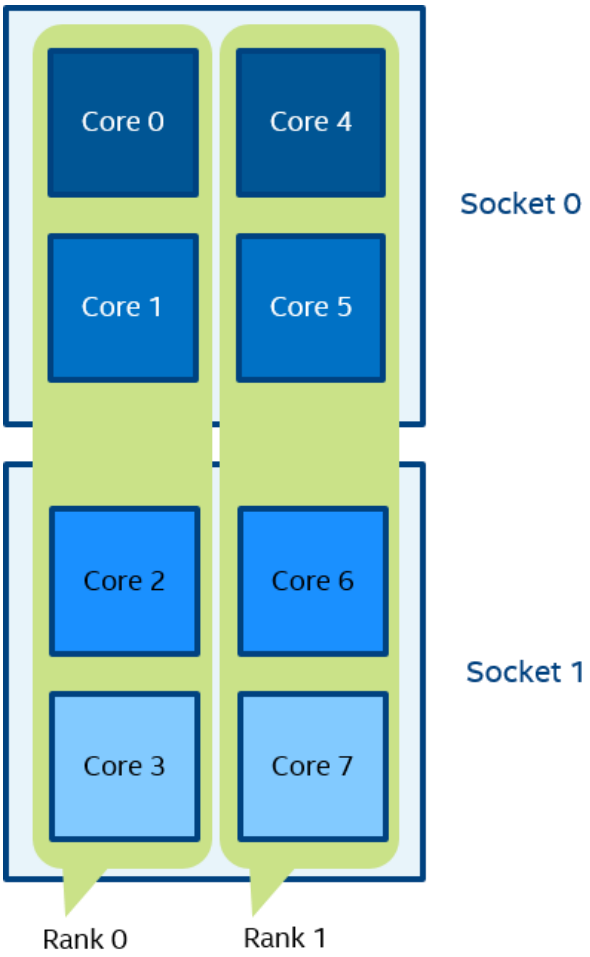
In Figure 3.2-3, two domains are defined according to the number of sockets. Process rank 0 can migrate on all cores on the 0-th socket. Process rank 1 can migrate on all cores on the first socket.

Figure 3.2-4 `mpiexec -n 4 -env I_MPI_PIN_DOMAIN cache2 test.exe`



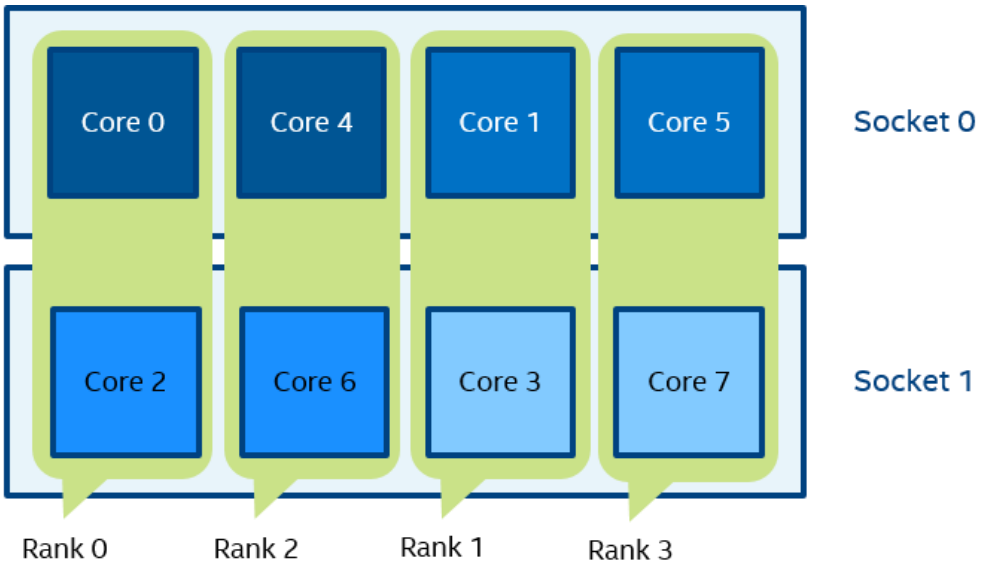
In Figure 3.2-4, four domains are defined according to the amount of common L2 caches. Process rank 0 runs on cores {0,4} that share an L2 cache. Process rank 1 runs on cores {1,5} that share an L2 cache as well, and so on.

Figure 3.2-5 `mpiexec -n 2 -env I_MPI_PIN_DOMAIN 4:platform test.exe`



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

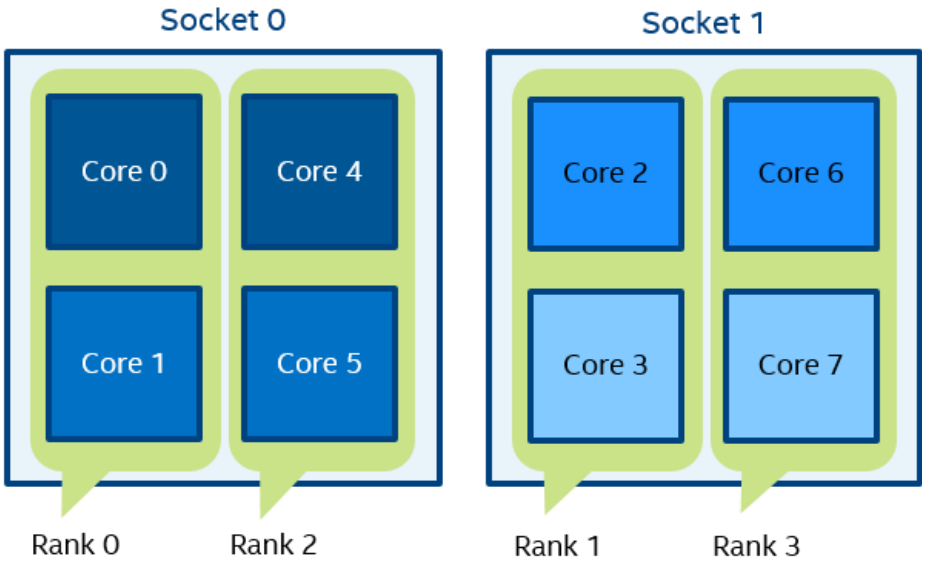
Figure 3.2-6 `mpiexec -n 4 -env I_MPI_PIN_DOMAIN auto:scatter test.exe`



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

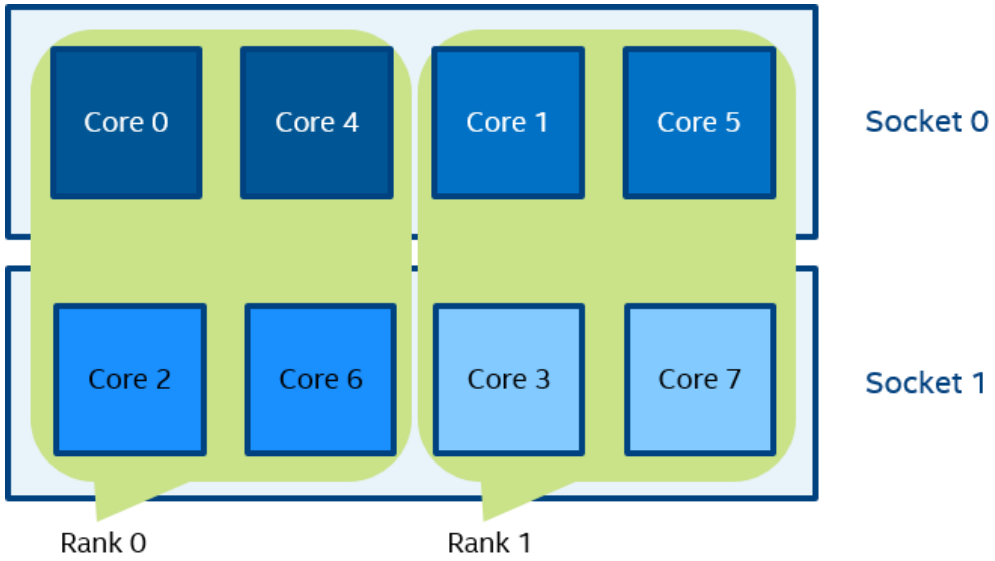
Figure 3.2-7 `set OMP_NUM_THREADS=2`

```
mpiexec -n 4 -env I_MPI_PIN_DOMAIN omp:platform test.exe
```



In Figure 3.2-7, domain size=2 (defined by `OMP_NUM_THREADS=2`), `platform` layout. Four domains {0,1}, {2,3}, {4,5}, {6,7} are defined. Domain members (cores) have consecutive numbering.

Figure 3.2-8 `mpiexec -n 2 -env I_MPI_PIN_DOMAIN [0x55,0xaa] test.exe`



In Figure 3.2-8 (the example for `I_MPI_PIN_DOMAIN=<masklist>`), the first domain is defined by the 0x55 mask. It contains all cores with even numbers {0,2,4,6}. The second domain is defined by the 0xAA mask. It contains all cores with odd numbers {1,3,5,7}.

I_MPI_PIN_ORDER

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

Syntax

`I_MPI_PIN_ORDER=<order>`

Arguments

<order>	Specify the ranking order
`range`	The domains are ordered according to the processor's BIOS numbering. This is a platform-dependent numbering
`scatter`	The domains are ordered so that adjacent domains have minimal sharing of common resources
`compact`	The domains are ordered so that adjacent domains share common resources as much as possible. This is the default value
`spread`	The domains are ordered consecutively with the possibility not to share common resources
`bunch`	The processes are mapped proportionally to sockets and the domains are ordered as close as possible on the sockets

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` or `bunch` values. Otherwise, use the `scatter` or `spread` values. Use the `range` value as needed. For detail information and examples about these values, see the Arguments table and the Example section of `I_MPI_PIN_ORDER` in this topic.

The options `scatter`, `compact`, `spread` and `bunch` 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.

Examples

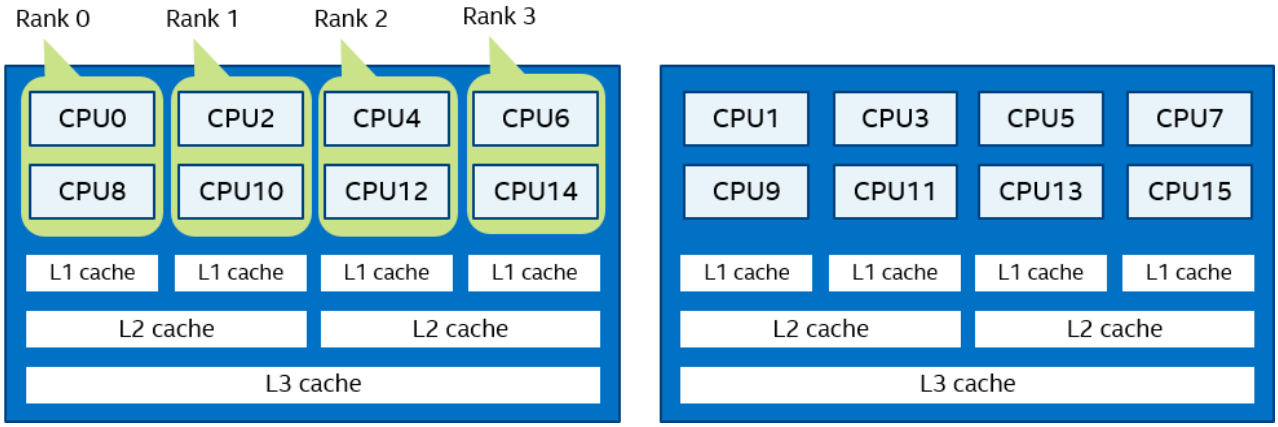
For the following configuration:

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

Compact order:

```
I_MPI_PIN_DOMAIN=2
I_MPI_PIN_ORDER=compact
```

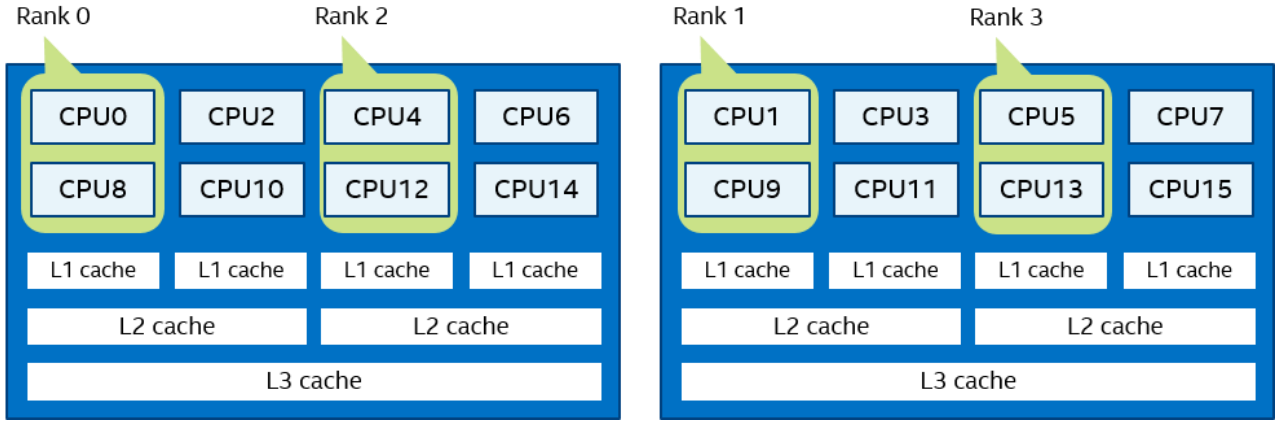
Figure 3.2-9 Compact Order Example



Scatter order:

```
I_MPI_PIN_DOMAIN=2
I_MPI_PIN_ORDER=scatter
```

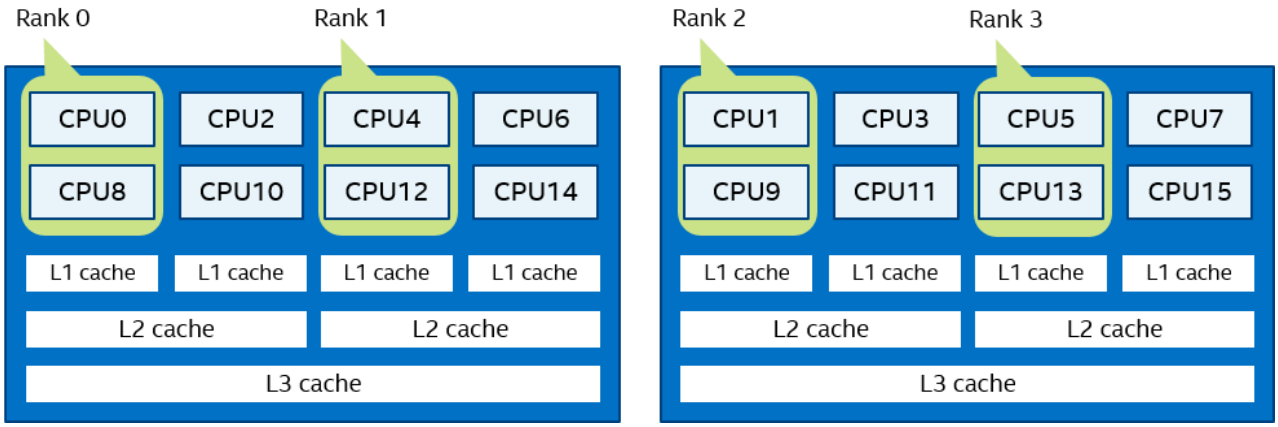
Figure 3.2-10 Scatter Order Example



Spread order:

```
I_MPI_PIN_DOMAIN=2
I_MPI_PIN_ORDER=spread
```

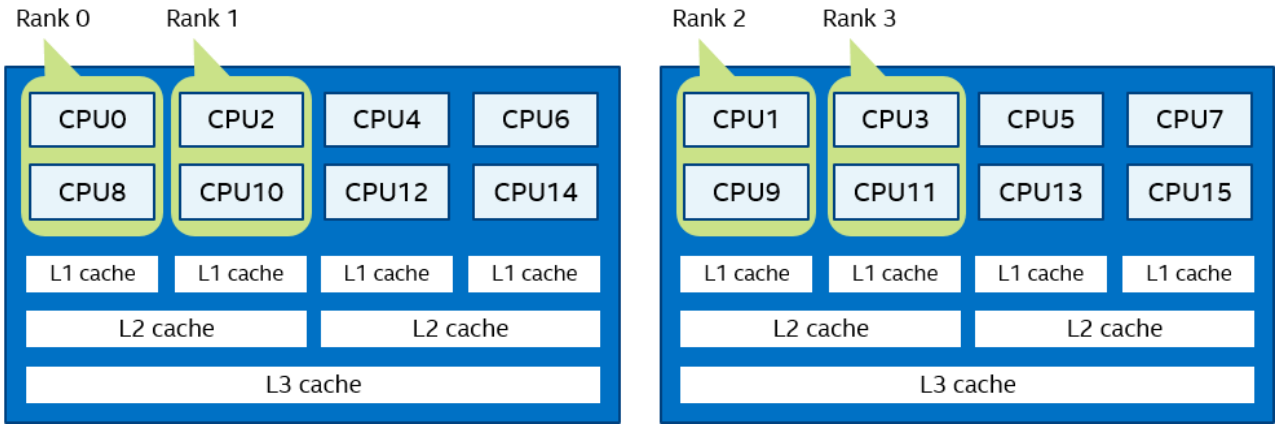
Figure 3.2-11 Spread Order Example



Bunch order:

```
I_MPI_PIN_DOMAIN=2
I_MPI_PIN_ORDER=bunch
```

Figure 3.2-12 Bunch Order Example



3.3. Fabrics Control

3.3.1. Communication Fabrics Control

I_MPI_FABRICS

Select the particular network fabrics to be used.

Syntax

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

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

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

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

Arguments

`<fabric>`	Define a network fabric.
`shm`	Shared memory (for intra-node communication only).
`dapl`	Direct Access Programming Library* (DAPL)-capable network fabrics, such as InfiniBand* and iWarp* (through DAPL).
`tcp`	TCP/IP-capable network fabrics, such as Ethernet and InfiniBand* (through IPoIB*).

Description

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

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

- If all processes start on one node, the library uses `shm` for 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 and DAPL-capable network fabric as the chosen fabric combination, use the following command:

```
> mpiexec -n <# of processes> -genv I_MPI_FABRICS=shm:dapl <executable>
```

To enable Intel® MPI Library to select most appropriate fabric combination automatically, run the application as usual, without setting the `I_MPI_FABRICS` variable:

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

I_MPI_FABRICS_LIST

Define a fabric list.

Syntax

`I_MPI_FABRICS_LIST=<fabrics list>`

where `<fabrics list> := <fabric>,...,<fabric>`

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

Arguments

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

Description

Use this environment variable to define a list of inter-node fabrics. Intel® MPI Library uses the fabric 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 a DAPL-capable network fabrics fails, Intel® MPI Library falls back to the TCP-capable network fabric. For more information on fallback, see `I_MPI_FALLBACK`.

I_MPI_FALLBACK

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

Syntax

`I_MPI_FALLBACK=<arg>`

Arguments

<arg>	Binary indicator
`enable | yes | on | 1`	Fall back to the first available fabric. This is the default value unless you set the `I_MPI_FABRICS` environment variable.
`disable | no | off | 0`	Terminate the job if MPI cannot initialize the currently set fabric. This is the default value if you set the `I_MPI_FABRICS` environment variable.

Description

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

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

NOTE

If you set `I_MPI_FABRICS` and `I_MPI_FALLBACK=enable`, the library falls back to the next fabric in the fabrics list. For example, if `I_MPI_FABRICS=dapl`, `I_MPI_FABRICS_LIST=dapl,tcp`, `I_MPI_FALLBACK=enable` and the initialization of DAPL-capable network fabrics fails, the library falls back to TCP-capable network fabric.

I_MPI_EAGER_THRESHOLD

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

Syntax

`I_MPI_EAGER_THRESHOLD=`*<nbytes>*

Arguments

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

Description

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

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

I_MPI_INTRANODE_EAGER_THRESHOLD

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

Syntax

`I_MPI_INTRANODE_EAGER_THRESHOLD=`*<nbytes>*

Arguments

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

Description

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

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

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

I_MPI_SPIN_COUNT

Control the spin count value.

Syntax

I_MPI_SPIN_COUNT=*<scount>*

Arguments

<scount>	Define the loop spin count when polling fabric(s)
> 0	The default *<scount>* value is equal to 1 when more than one process runs per processor/core. Otherwise the value equals 250 . The maximum value is equal to 2147483647

Description

Set the spin count limit. The loop for polling the fabric(s) spins *<scount>* times before the library releases the processes if no incoming messages are received for processing. Within every spin loop, the shm fabric (if enabled) is polled an extra I_MPI_SHM_SPIN_COUNT times. Smaller values for *<scount>* cause the Intel® MPI Library to release the processor more frequently.

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

I_MPI_SCALABLE_OPTIMIZATION

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

Syntax

I_MPI_SCALABLE_OPTIMIZATION=*<arg>*

Arguments

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

`| 0`	default value for less than 16 processes

Description

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

I_MPI_WAIT_MODE

Turn on/off wait mode.

Syntax

`I_MPI_WAIT_MODE=<arg>`

Arguments

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

Description

Set this environment variable to control the wait mode. If you enable this mode, 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)

Control the dynamic connection establishment.

Syntax

`I_MPI_DYNAMIC_CONNECTION=<arg>`

Arguments

<arg>	Binary indicator
`enable | yes | on | 1`	Turn on the dynamic connection establishment. This is the default for 64 or more processes
`disable | no | off | 0`	Turn off the dynamic connection establishment. This is the default for less than 64 processes

Description

Set this environment variable to control dynamic connection establishment.

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

The default value depends on the 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

Control the message transfer algorithm for the shared memory.

Syntax

`I_MPI_SHM_CACHE_BYPASS=<arg>`

Arguments

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

Description

Set this environment variable to enable/disable message transfer bypass cache for the shared memory. When you enable this feature, the MPI sends the messages greater than or equal in size to the value specified by the `I_MPI_SHM_CACHE_BYPASS_THRESHOLD` environment variable through the bypass cache. This feature is enabled by default.

I_MPI_SHM_CACHE_BYPASS_THRESHOLDS

Set the message copying algorithm threshold.

Syntax

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

Arguments

<nb_send>	Set the threshold for sent messages in the following situations: • Processes are pinned on cores that are not located in the same physical processor package • Processes are not pinned
<nb_recv>	Set the threshold for received messages in the following situations: • Processes are pinned on cores that are not located in the same physical processor package • Processes are not pinned
<nb_send_pk>	Set the threshold for sent messages when processes are pinned on cores located in the same physical processor package

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

Description

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

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

I_MPI_SHM_FBOX

Control the usage of the shared memory fast-boxes.

Syntax

`I_MPI_SHM_FBOX=`*<arg>*

Arguments

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

Description

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

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

I_MPI_SHM_FBOX_SIZE

Set the size of the shared memory fast-boxes.

Syntax

`I_MPI_SHM_FBOX_SIZE=`*<nbytes>*

Arguments

<nbytes>	The size of shared memory fast-boxes in bytes
`> 0`	The default *<nbytes>* value depends on the specific platform you use. The value range is from 8K to 64K typically.

Description

Set this environment variable to define the size of shared memory fast-boxes.

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>	The size of a shared memory cell in bytes
> 0	The default *<nbytes>* value depends on the specific platform you use. The value range is from 8K to 64K typically.

Description

Set this environment variable to define the size of shared memory cells.

If you set this environment variable, `I_MPI_INTRANODE_EAGER_THRESHOLD` is also changed and becomes equal to the given value.

I_MPI_SHM_LMT

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

Syntax

`I_MPI_SHM_LMT=<arg>`

Arguments

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

Description

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

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

I_MPI_SHM_LMT_BUFFER_NUM

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

Syntax

`I_MPI_SHM_LMT_BUFFER_NUM=<num>`

Arguments

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

Description

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

I_MPI_SHM_LMT_BUFFER_SIZE

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

Syntax

`I_MPI_SHM_LMT_BUFFER_SIZE=<nbytes>`

Arguments

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

Description

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

I_MPI_SHM_BYPASS

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

Syntax

`I_MPI_SHM_BYPASS=<arg>`

Arguments

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

Description

Set this environment variable to specify the communication mode within the node. If the intra-node communication mode through network fabric is enabled, data transfer algorithms are selected according to the following scheme:

- Messages shorter than or equal in size to the threshold value of the `I_MPI_INTRANODE_EAGER_THRESHOLD` environment variable are transferred using shared memory.
- Messages larger than the threshold value of the `I_MPI_INTRANODE_EAGER_THRESHOLD` environment variable are transferred through the network fabric layer.

NOTE

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

I_MPI_SHM_SPIN_COUNT

Control the spin count value for the shared memory fabric.

Syntax

`I_MPI_SHM_SPIN_COUNT=<shm_scount>`

Arguments

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

Description

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

To tune application performance, use the `I_MPI_SHM_SPIN_COUNT` environment variable. The best value for `<shm_scount>` can be chosen on an experimental basis. It depends largely on the application and the particular computation environment. An increase in the `<shm_scount>` value benefits 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

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

I_MPI_DAT_LIBRARY

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

Syntax

`I_MPI_DAT_LIBRARY=<library>`

Arguments

<library>	Specify the DAT library for DAPL provider to be used. Default values are `dat.dll` for DAPL* 1.2 providers and `dat2.dll` for DAPL* 2.0 providers

Description

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

I_MPI_DAPL_TRANSLATION_CACHE

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

Syntax

`I_MPI_DAPL_TRANSLATION_CACHE=<arg>`

Arguments

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

Description

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

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

I_MPI_DAPL_TRANSLATION_CACHE_AVL_TREE

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

Syntax

`I_MPI_DAPL_TRANSLATION_CACHE_AVL_TREE=<arg>`

Arguments

<arg>	Binary indicator
`enable | yes | on | 1`	Turn on the AVL tree based RDMA translation cache

`disable | no | off | 0`	Turn off the AVL tree based RDMA translation cache. This is the default value

Description

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

I_MPI_DAPL_DIRECT_COPY_THRESHOLD

Change the threshold of the DAPL direct-copy protocol.

Syntax

`I_MPI_DAPL_DIRECT_COPY_THRESHOLD=<nbytes>`

Arguments

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

Description

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

- Messages shorter than or equal to *<nbytes>* are sent using the eager protocol through the internal pre-registered buffers. This approach is faster for short messages.
- Messages larger than *<nbytes>* are sent using the direct-copy protocol. It does not use any buffering but involves registration of memory on sender and receiver sides. This approach is faster for large messages.

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

NOTE

The equivalent of this variable for Intel® Xeon Phi™ Coprocessor is `I_MIC_MPI_DAPL_DIRECT_COPY_THRESHOLD`

I_MPI_DAPL_EAGER_MESSAGE_AGGREGATION

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

Syntax

`I_MPI_DAPL_EAGER_MESSAGE_AGGREGATION=<arg>`

Arguments

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

	value

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

```
for( i = 0; i < NMSG; i++)
{
    ret = MPI Isend( sbuf[i], MSG SIZE, datatype, dest, tag, comm, &req send[i]);
}
```

I_MPI_DAPL_DYNAMIC_CONNECTION_MODE

Choose the algorithm for establishing the DAPL* connections.

Syntax

`I_MPI_DAPL_DYNAMIC_CONNECTION_MODE=<arg>`

Arguments

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

Description

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

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

I_MPI_DAPL_SCALABLE_PROGRESS

Turn on/off scalable algorithm for DAPL read progress.

Syntax

`I_MPI_DAPL_SCALABLE_PROGRESS=<arg>`

Arguments

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

Description

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

I_MPI_DAPL_BUFFER_NUM

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

Syntax

`I_MPI_DAPL_BUFFER_NUM=<nbuf>`

Arguments

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

Description

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

NOTE

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

I_MPI_DAPL_BUFFER_SIZE

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

Syntax

`I_MPI_DAPL_BUFFER_SIZE=<nbytes>`

Arguments

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

Description

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

I_MPI_DAPL_RNDV_BUFFER_ALIGNMENT

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

Syntax

`I_MPI_DAPL_RNDV_BUFFER_ALIGNMENT=<arg>`

Arguments

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

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

I_MPI_DAPL_RDMA_RNDV_WRITE

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

Syntax

`I_MPI_DAPL_RDMA_RNDV_WRITE=<arg>`

Arguments

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

Description

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

I_MPI_DAPL_CHECK_MAX_RDMA_SIZE

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

Syntax

`I_MPI_DAPL_CHECK_MAX_RDMA_SIZE=<arg>`

Arguments

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

Description

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

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

I_MPI_DAPL_MAX_MSG_SIZE

Control message fragmentation threshold.

Syntax

I_MPI_DAPL_MAX_MSG_SIZE=<*nbytes*>

Arguments

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

Description

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

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

I_MPI_DAPL_CONN_EVD_SIZE

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

Syntax

I_MPI_DAPL_CONN_EVD_SIZE=<*size*>

Arguments

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

Description

Set this environment variable to define the event queue size of the DAPL event dispatcher that handles connection related events. If this environment variable is set, the minimum value between <*size*> and the value obtained from the provider is used as the size of the event queue. The provider is required to supply a queue size that equal or larger than the calculated value.

I_MPI_DAPL_SR_THRESHOLD

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

Syntax

I_MPI_DAPL_SR_THRESHOLD=<*arg*>

Arguments

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

Description

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

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

I_MPI_DAPL_SR_BUF_NUM

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

Syntax

`I_MPI_DAPL_SR_BUF_NUM=<nbuf>`

Arguments

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

Description

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

I_MPI_DAPL_RDMA_WRITE_IMM

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

Syntax

`I_MPI_DAPL_RDMA_WRITE_IMM=<arg>`

Arguments

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

Description

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

I_MPI_DAPL_DESIRED_STATIC_CONNECTIONS_NUM

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

Syntax

`I_MPI_DAPL_DESIRED_STATIC_CONNECTIONS_NUM=<num_processess>`

Arguments

<num_processess>	Define the number of processes that establish DAPL static connections at the same

	time
> 0	The default *<num_processess>* 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_processess>*, 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_processess>*. Then static connections are established in several iterations, including intergroup connection setup.

I_MPI_CHECK_DAPL_PROVIDER_COMPATIBILITY

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

Syntax

`I_MPI_CHECK_DAPL_PROVIDER_COMPATIBILITY=<arg>`

Arguments

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

Description

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

3.3.4. TCP-capable Network Fabrics Control

I_MPI_TCP_NETMASK

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

Syntax

`I_MPI_TCP_NETMASK=<arg>`

Arguments

<arg>	Define the network interface (string parameter)
<interface_mnemonic>	Mnemonic of the network interface: `ib` or `eth`
`ib`	Use IPoIB* network interface

eth	Use Ethernet network interface. This is the default value
<interface_name>	Name of the network interface Usually the UNIX* driver name followed by the unit number
<network_address>	Network address. Trailing zero bits imply a netmask
<network_address/ <netmask>	Network address. The *<netmask>* value specifies the netmask length
<list of interfaces>	A colon separated list of network addresses and interface names

Description

Set this environment variable to choose the network interface for MPI communication over TCP-capable network fabrics. If you specify a list of interfaces, the first available interface on the node is used for communication.

Examples

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

  `I_MPI_TCP_NETMASK=ib`
- Use the following setting to select the specified network interface for socket communications:

  `I_MPI_TCP_NETMASK=ib0`
- Use the following setting to select the specified network for socket communications. This setting implies the `255.255.0.0` netmask:
  `I_MPI_TCP_NETMASK=192.169.0.0`
- Use the following setting to select the specified network for socket communications with netmask set explicitly:
  `I_MPI_TCP_NETMASK=192.169.0.0/24`
- Use the following setting to select the specified network interfaces for socket communications:
  `I_MPI_TCP_NETMASK=192.169.0.5/24:ib0:192.169.0.0`

I_MPI_TCP_BUFFER_SIZE

Change the size of the TCP socket buffers.

Syntax

`I_MPI_TCP_BUFFER_SIZE=`*<nbytes>*

Arguments

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

Description

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

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

NOTE

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

3.4. Collective Operations Control

Each collective operation in the Intel® MPI Library supports a number of communication algorithms. In addition to highly optimized default settings, the library provides a way to control the algorithm selection explicitly. You can do this by using the `I_MPI_ADJUST` environment variable family, which is described in the following section.

These environment variables are available for both Intel® and non-Intel microprocessors, but they may perform additional optimizations for Intel microprocessors than they performs 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 optimized default settings

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

Description

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

Table 3.4-1 Environment Variables, Collective Operations, and Algorithms

Environment Variable	Collective Operation	Algorithms

I_MPI_ADJUST_ALLGATHER	MPI_Allgather	1. Recursive doubling 2. Bruck's 3. Ring 4. Topology aware Gatherv + Bcast 5. Knomial
I_MPI_ADJUST_ALLGATHERV	MPI_Allgatherv	1. Recursive doubling 2. Bruck's 3. Ring 4. Topology aware Gatherv + Bcast
I_MPI_ADJUST_ALLREDUCE	MPI_Allreduce	1. Recursive doubling 2. Rabenseifner's 3. Reduce + Bcast 4. Topology aware Reduce + Bcast 5. Binomial gather + scatter 6. Topology aware binominal gather + scatter 7. Shumilin's ring 8. Ring 9. Knomial 10. Topology aware SHM-based flat 11. Topology aware SHM-based Knomial 12. Topology aware SHM-based Knary
I_MPI_ADJUST_ALLTOALL	MPI_Alltoall	1. Bruck's 2. Isend/Irecv + waitall 3. Pair wise exchange 4. Plum's
I_MPI_ADJUST_ALLTOALLV	MPI_Alltoallv	1. Isend/Irecv + waitall 2. Plum's
I_MPI_ADJUST_ALLTOALLW	MPI_Alltoallw	Isend/Irecv + waitall
I_MPI_ADJUST_BARRIER	MPI_Barrier	1. Dissemination 2. Recursive doubling 3. Topology aware dissemination

		4. Topology aware recursive doubling 5. Binominal gather + scatter 6. Topology aware binominal gather + scatter 7. Topology aware SHM-based flat 8. Topology aware SHM-based Knomial 9. Topology aware SHM-based Knary
`I_MPI_ADJUST_BCAST`	`MPI_Bcast`	1. Binomial 2. Recursive doubling 3. Ring 4. Topology aware binomial 5. Topology aware recursive doubling 6. Topology aware ring 7. Shumilin's 8. Knomial 9. Topology aware SHM-based flat 10. Topology aware SHM-based Knomial 11. Topology aware SHM-based Knary
`I_MPI_ADJUST_EXSCAN`	`MPI_Exscan`	1. Partial results gathering 2. Partial results gathering regarding layout of processes
`I_MPI_ADJUST_GATHER`	`MPI_Gather`	1. Binomial 2. Topology aware binomial 3. Shumilin's 4. Binomial with segmentation
`I_MPI_ADJUST_GATHERV`	`MPI_Gatherv`	1. Linear 2. Topology aware linear 3. Knomial
`I_MPI_ADJUST_REDUCE_SCATTER`	`MPI_Reduce_scatter`	1. Recursive halving 2. Pair wise exchange 3. Recursive doubling 4. Reduce + Scatterv

		5. Topology aware Reduce + Scatterv
I_MPI_ADJUST_REDUCE	MPI_Reduce	1. Shumilin's 2. Binomial 3. Topology aware Shumilin's 4. Topology aware binomial 5. Rabenseifner's 6. Topology aware Rabenseifner's 7. Knomial 8. Topology aware SHM-based flat 9. Topology aware SHM-based Knomial 10. Topology aware SHM-based Knary 11. Topology aware SHM-based binomial
I_MPI_ADJUST_SCAN	MPI_Scan	1. Partial results gathering 2. Topology aware partial results gathering
I_MPI_ADJUST_SCATTER	MPI_Scatter	1. Binomial 2. Topology aware binomial 3. Shumilin's
I_MPI_ADJUST_SCATTERV	MPI_Scatterv	1. Linear 2. Topology aware linear
I_MPI_ADJUST_IALLGATHER	MPI_Iallgather	1. Recursive doubling 2. Bruck’s 3. Ring
I_MPI_ADJUST_IALLGATHERV	MPI_Iallgatherv	1. Recursive doubling 2. Bruck’s 3. Ring
I_MPI_ADJUST_IALLREDUCE	MPI_Iallreduce	1. Recursive doubling 2. Rabenseifner’s 3. Reduce + Bcast 4. Ring (patarasuk)

		5. Knomial 6. Binomial
I_MPI_ADJUST_IALLTOALL	MPI_Ialltoall	1. Bruck's 2. Isend/Irecv + Waitall 3. Pairwise exchange
I_MPI_ADJUST_IALLTOALLV	MPI_Ialltoallv	Isend/Irecv + Waitall
I_MPI_ADJUST_IALLTOALLW	MPI_Ialltoallw	Isend/Irecv + Waitall
I_MPI_ADJUST_IBARRIER	MPI_Ibarrier	Dissemination
I_MPI_ADJUST_IBCAST	MPI_Ibcast	1. Binomial 2. Recursive doubling 3. Ring 4. Knomial
I_MPI_ADJUST_IEXSCAN	MPI_Iexscan	Recursive doubling
I_MPI_ADJUST_IGATHER	MPI_Igather	1. Binomial 2. Knomial
I_MPI_ADJUST_IGATHERV	MPI_Igatherv	Linear
I_MPI_ADJUST_IREDUCE_SCATTER	MPI_Ireduce_scatter	1. Recursive halving 2. Pairwise 3. Recursive doubling
I_MPI_ADJUST_IREDUCE	MPI_Ireduce	1. Rabenseifner's 2. Binomial 3. Knomial
I_MPI_ADJUST_ISCAN	MPI_Iscan	Recursive Doubling
I_MPI_ADJUST_ISCATTER	MPI_Iscatter	1. Binomial 2. Knomial
I_MPI_ADJUST_ISCATTERV	MPI_Iscatterv	Linear

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

Table 3.4-2 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 of a message segment in bytes
> 0	The default value is 14000

Description

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

NOTE

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

I_MPI_ADJUST_BCAST_SEGMENT

Syntax

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

Arguments

<algid>	Algorithm identifier
1	Binomial
4	Topology aware binomial
7	Shumilin's
8	Knomial
<block_size>	Size of a message segment in bytes
> 0	The default value is 12288

Description

Set an internal block size to control MPI_Bcast 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 only for Binomial, Topology-aware binomial, Shumilin's and Knomial algorithms.

I_MPI_ADJUST_ALLGATHER_KN_RADIX

Syntax

`I_MPI_ADJUST_ALLGATHER_KN_RADIX=<radix>`

Arguments

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

Description

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

I_MPI_ADJUST_BCAST_KN_RADIX

Syntax

`I_MPI_ADJUST_BCAST_KN_RADIX=<radix>`

Arguments

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

Description

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

I_MPI_ADJUST_ALLREDUCE_KN_RADIX

Syntax

`I_MPI_ADJUST_ALLREDUCE_KN_RADIX=<radix>`

Arguments

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

Description

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

I_MPI_ADJUST_REDUCE_KN_RADIX

Syntax

`I_MPI_ADJUST_REDUCE_KN_RADIX=<radix>`

Arguments

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

Description

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

I_MPI_ADJUST_GATHERV_KN_RADIX

Syntax

`I_MPI_ADJUST_GATHERV_KN_RADIX=<radix>`

Arguments

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

Description

Set this environment variable together with `I_MPI_ADJUST_GATHERV=3` to select the knomial tree radix for the corresponding `MPI_Gatherv` algorithm.

I_MPI_ADJUST_IALLREDUCE_KN_RADIX

Syntax

`I_MPI_ADJUST_IALLREDUCE_KN_RADIX=<radix>`

Arguments

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

Description

Set this environment variable together with `I_MPI_ADJUST_IALLREDUCE=5` to select the knomial tree radix for the corresponding `MPI_Iallreduce` algorithm.

I_MPI_ADJUST_IBCAST_KN_RADIX

Syntax

`I_MPI_ADJUST_IBCAST_KN_RADIX=<radix>`

Arguments

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

Description

Set this environment variable together with `I_MPI_ADJUST_IBCAST=4` to select the knomial tree radix for the corresponding `MPI_Ibcast` algorithm.

I_MPI_ADJUST_IREDUCE_KN_RADIX

Syntax

`I_MPI_ADJUST_IREDUCE_KN_RADIX=<radix>`

Arguments

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

Description

Set this environment variable together with `I_MPI_ADJUST_IREDUCE=3` to select the knomial tree radix for the corresponding `MPI_Ireduce` algorithm.

I_MPI_ADJUST_IGATHER_KN_RADIX

Syntax

`I_MPI_ADJUST_IGATHER_KN_RADIX=<radix>`

Arguments

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

Description

Set this environment variable together with `I_MPI_ADJUST_IGATHER=2` to select the knomial tree radix for the corresponding `MPI_Igather` algorithm.

I_MPI_ADJUST_ISCATTER_KN_RADIX

Syntax

`I_MPI_ADJUST_ISCATTER_KN_RADIX=<radix>`

Arguments

`<radix>`	An integer that specifies a radix used by the Knomial `MPI_Iscatter` algorithm to build a knomial

	communication tree
> 1	The default value is 4

Description

Set this environment variable together with `I_MPI_ADJUST_ISCATTER=2` to select the knomial tree radix for the corresponding `MPI_Iscatter` algorithm.

I_MPI_ADJUST_<*COLLECTIVE*>_SHM_KN_RADIX

Syntax

`I_MPI_ADJUST_<COLLECTIVE>_SHM_KN_RADIX=<radix>`

Arguments

`<radix>`	An integer that specifies a radix used by the Knomial or Knary SHM-based algorithm to build a knomial or knary communication tree
> 0	• If you specify the environment variables `I_MPI_ADJUST_BCAST_SHM_KN_RADIX` and `I_MPI_ADJUST_BARRIER_SHM_KN_RADIX`, the default value is 3 • If you specify the environment variables `I_MPI_ADJUST_REDUCE_SHM_KN_RADIX` and `I_MPI_ADJUST_ALLREDUCE_SHM_KN_RADIX`, the default value is 4

Description

This environment variable includes the following variables:

- `I_MPI_ADJUST_BCAST_SHM_KN_RADIX`
- `I_MPI_ADJUST_BARRIER_SHM_KN_RADIX`
- `I_MPI_ADJUST_REDUCE_SHM_KN_RADIX`
- `I_MPI_ADJUST_ALLREDUCE_SHM_KN_RADIX`

Set this environment variable to select the knomial or knary tree radix for the corresponding tree SHM-based algorithms. When you build a knomial communication tree, the specified value is used as the power for 2 to generate resulting radix (2^`<radix>`). When you build a knary communication tree, the specified value is used for the radix.

I_MPI_COLL_INTRANODE

Syntax

`I_MPI_COLL_INTRANODE=<mode>`

Arguments

`<mode>`	Intranode collectives type
`pt2pt`	Use only point-to-point communication-based collectives
`shm`	Enables shared memory collectives. This is the default value

Description

Set this environment variable to switch intranode communication type for collective operations. If there is large set of communicators, you can switch off the SHM-collectives to avoid memory overconsumption.

I_MPI_COLL_INTRANODE_SHM_THRESHOLD

Syntax

`I_MPI_COLL_INTRANODE_SHM_THRESHOLD=<nbytes>`

Arguments

`<nbytes>`	Define the maximal data block size processed by shared memory collectives.
`> 0`	Use the specified size. The default value is 16384 bytes.

Description

Set this environment variable to define the size of shared memory area available for each rank for data placement. Messages greater than this value will *not* be processed by SHM-based collective operation, but will be processed by point-to-point based collective operation. The value must be a multiple of 4096.

I_MPI_ADJUST_GATHER_SEGMENT

Syntax

`I_MPI_ADJUST_GATHER_SEGMENT=<block_size>`

Arguments

`<block_size>`	Size of a message segment in bytes.
`> 0`	Use the specified size. The default value is 16384 bytes.

Description

Set an internal block size to control the `MPI_Gather` message segmentation for the binomial algorithm with segmentation.

4. Miscellaneous

4.1. Compatibility Control

I_MPI_COMPATIBILITY

Select the runtime compatibility mode.

Syntax

`I_MPI_COMPATIBILITY=<value>`

Arguments

`<value>`	Define compatibility mode
`not defined`	The MPI-3.1 standard compatibility. This is the default mode
`3`	The Intel® MPI Library 3.x compatible mode
`4`	The Intel® MPI Library 4.0.x compatible mode

Description

Set this environment variable to choose the Intel® MPI Library runtime compatible mode. By default, the library complies with the MPI-3.1 standard. If your application depends on the MPI-2.1 behavior, set the value of the environment variable `I_MPI_COMPATIBILITY` to `4`. If your application depends on the pre-MPI-2.1 behavior, set the value of the environment variable `I_MPI_COMPATIBILITY` to `3`.

4.2. Dynamic Process Support

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

A set of hosts indicated within a machine file (see Hydra Global Options for details) is used for placement of spawned processes. The spawned processes are placed onto different hosts in round-robin or per-host fashion. The first spawned process is placed after the last process of the parent group. A specific communication fabric combination is selected using the usual fabrics selection algorithm (see `I_MPI_FABRICS` and `I_MPI_FABRICS_LIST` for details).

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

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

In the example, `<spawn_app>` spawns 4 dynamic processes. If the `hosts` file 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`.

If the `hosts` file contains the following information:

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

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

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

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

and use the following commands on the intended client hosts:

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

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

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

4.3. Statistics Gathering Mode

Intel® MPI Library provides the built-in statistics gathering facility that provides essential information about MPI program execution. You can use the native or IPM statistics formats or both at once. See description of the environment variables controlling statistics collection below.

4.3.1. Native Statistics

To enable the native statistics collection, set `I_MPI_STATS` to `native` and specify the level of detail.

I_MPI_STATS

Control statistics collection.

Syntax

```
I_MPI_STATS=[native:][n-]m
```

Arguments

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

`20`	Output additional time information for all MPI functions

Description

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

`n`, `m` are positive integer numbers and define the range of output information. The statistics from level `n` to level `m` inclusive are printed. If `n` is not provided, the default lower bound is `1`.

I_MPI_STATS_SCOPE

Select the subsystem(s) for which statistics should be collected.

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`
`Iallgather`	`MPI_Iallgather`
`Allgatherv`	`MPI_Allgatherv`
`Iallgatherv`	`MPI_Iallgatherv`
`Allreduce`	`MPI_Allreduce`
`Iallreduce`	`MPI_Iallreduce`
`Alltoall`	`MPI_Alltoall`
`Ialltoall`	`MPI_Ialltoall`
`Alltoallv`	`MPI_Alltoallv`
`Ialltoallv`	`MPI_Ialltoallv`
`Alltoallw`	`MPI_Alltoallw`
`Ialltoallw`	`MPI_Ialltoallw`

`Barrier`	`MPI_Barrier`
`Ibarrier`	`MPI_Ibarrier`
`Bcast`	`MPI_Bcast`
`Ibcast`	`MPI_Ibcast`
`Exscan`	`MPI_Exscan`
`Iexscan`	`MPI_Iexscan`
`Gather`	`MPI_Gather`
`Igather`	`MPI_Igather`
`Gatherv`	`MPI_Gatherv`
`Igatherv`	`MPI_Igatherv`
`Reduce_scatter`	`MPI_Reduce_scatter`
`Ireduce_scatter`	`MPI_Ireduce_scatter`
`Reduce`	`MPI_Reduce`
`Ireduce`	`MPI_Ireduce`
`Scan`	`MPI_Scan`
`Iscan`	`MPI_Iscan`
`Scatter`	`MPI_Scatter`
`Iscatter`	`MPI_Iscatter`
`Scatterv`	`MPI_Scatterv`
`Iscatterv`	`MPI_Iscatterv`
`Send`	Standard transfers (`MPI_Send`, `MPI_Isend`, `MPI_Send_init`)
`Sendrecv`	Send-receive transfers (`MPI_Sendrecv`, `MPI_Sendrecv_replace`)
`Bsend`	Buffered transfers (`MPI_Bsend`, `MPI_Ibsend`, `MPI_Bsend_init`)
`Csend`	Point-to-point operations inside the collectives. This internal operation serves all collectives
`Csendrecv`	Point-to-point send-receive 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 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

Set the list of ranges for message sizes and communicator sizes that are used for collecting statistics.

Syntax

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

Arguments

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

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

Description

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

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

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

If a range is not specified, the maximum possible range is assumed.

Examples

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

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

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

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

NOTE

When the '@' symbol is present, the environment variable value must be enclosed in quotes.

I_MPI_STATS_FILE

Define the statistics output file name.

Syntax

`I_MPI_STATS_FILE=<name>`

Arguments

<name>	Define the statistics output file name

Description

Set this environment variable to define the statistics output file. By default, the `stats.txt` file is created in the current directory.

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

Statistics Format

The statistics data is grouped 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:

```
> set I_MPI_STATS=4> set 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:

```
____ 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
Csendrecv 0.000000e+00 0
Send 0.000000e+00 0
Sendrecv 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
Csendrecv 0.000000e+00 0
Send 0.000000e+00 0
Sendrecv 0.000000e+00 0
Bsend 0.000000e+00 0
Rsend 0.000000e+00 0
Ssend 0.000000e+00 0
Collectives
Allreduce 7.629395e-06 2
=======================================
Communication Activity by actual args
P2P
Operation Dst Message size Calls
-----------------------------------------
Csend
1 0 4 2
Collectives
Operation Context Comm size Message size Calls Cost(%)
-------------------------------------------------------------------------
Allreduce
1 0 2 4 2 37.93
=========================================================================
____ End of stats.txt file ____
```

In the example above:

- All times are measured in microseconds.
- The message sizes are counted in bytes. **MB** means megabyte equal to 2^{20} or 1 048 576 bytes.
- The process life time is calculated as a stretch of time between `MPI_Init` and `MPI_Finalize`.
- The **Algo** field indicates the number of algorithm used by this operation with listed arguments.

- The **Cost** field represents a particular collective operation execution time as a percentage of the process life time.

4.3.2. IPM Statistics

To enable the integrated performance monitoring (IPM) statistics collection, set `I_MPI_STATS` to `ipm` or `ipm:terse`.

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

I_MPI_STATS

Control the statistics data output format.

Syntax

`I_MPI_STATS=<level>`

Arguments

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

Description

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

I_MPI_STATS_FILE

Define the output file name.

Syntax

`I_MPI_STATS_FILE=<name>`

Argument

<name>	File name for statistics data gathering

Description

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

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

I_MPI_STATS_SCOPE

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

Syntax

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

Argument

<subset>	Target subset
`all2all`	Collect statistics data for all-to-all functions types
`all2one`	Collect statistics data for all-to-one functions types
`attr`	Collect statistics data for attribute control functions
`comm`	Collect statistics data for communicator control functions
`err`	Collect statistics data for error handling functions
`group`	Collect statistics data for group support functions
`init`	Collect statistics data for initialize/finalize functions
`io`	Collect statistics data for input/output support function
`one2all`	Collect statistics data for one-to-all functions types
`recv`	Collect statistics data for receive functions
`req`	Collect statistics data for request support functions
`rma`	Collect statistics data for one sided communication functions
`scan`	Collect statistics data for scan collective functions
`send`	Collect statistics data for send functions
`sendrecv`	Collect statistics data for send/receive functions
`serv`	Collect statistics data for additional service functions
`spawn`	Collect statistics data for dynamic process functions
`status`	Collect statistics data for status control function
`sync`	Collect statistics data for barrier synchronization
`time`	Collect statistics data for timing support functions
`topo`	Collect statistics data for topology support functions
`type`	Collect statistics data for data type support functions

Description

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

Table 4.3-1 Stats Subsets of MPI Functions

all2all	**recv**
MPI_Allgather	MPI_Recv
MPI_Allgatherv	MPI_Irecv
MPI_Allreduce	MPI_Recv_init
MPI_Alltoll	MPI_Probe
MPI_Alltoallv	MPI_Iprobe
MPI_Alltoallw	**req**
MPI_Reduce_scatter	MPI_Start
MPI_Iallgather	MPI_Startall
MPI_Iallgatherv	MPI_Wait
MPI_Iallreduce	MPI_Waitall
MPI_Ialltoll	MPI_Waitany
MPI_Ialltoallv	MPI_Waitsome
MPI_Ialltoallw	MPI_Test
MPI_Ireduce_scatter	MPI_Testall
MPI_Ireduce_scatter_block	MPI_Testany
all2one	MPI_Testsome
MPI_Gather	MPI_Cancel
MPI_Gatherv	MPI_Grequest_start
MPI_Reduce	MPI_Grequest_complete
MPI_Igather	MPI_Request_get_status
MPI_Igatherv	MPI_Request_free
MPI_Ireduce	**rma**
attr	MPI_Accumulate
MPI_Comm_create_keyval	MPI_Get
MPI_Comm_delete_attr	MPI_Put
MPI_Comm_free_keyval	MPI_Win_complete
MPI_Comm_get_attr	MPI_Win_create
MPI_Comm_set_attr	MPI_Win_fence
MPI_Comm_get_name	MPI_Win_free
MPI_Comm_set_name	MPI_Win_get_group
MPI_Type_create_keyval	MPI_Win_lock
MPI_Type_delete_attr	MPI_Win_post
MPI_Type_free_keyval	MPI_Win_start

MPI_Type_get_attr	MPI_Win_test
MPI_Type_get_name	MPI_Win_unlock
MPI_Type_set_attr	MPI_Win_wait
MPI_Type_set_name	MPI_Win_allocate
MPI_Win_create_keyval	MPI_Win_allocate_shared
MPI_Win_delete_attr	MPI_Win_create_dynamic
MPI_Win_free_keyval	MPI_Win_shared_query
MPI_Win_get_attr	MPI_Win_attach
MPI_Win_get_name	MPI_Win_detach
MPI_Win_set_attr	MPI_Win_set_info
MPI_Win_set_name	MPI_Win_get_info
MPI_Get_processor_name	MPI_Win_get_accumulate
comm	MPI_Win_fetch_and_op
MPI_Comm_compare	MPI_Win_compare_and_swap
MPI_Comm_create	MPI_Rput
MPI_Comm_dup	MPI_Rget
MPI_Comm_free	MPI_Raccumulate
MPI_Comm_get_name	MPI_Rget_accumulate
MPI_Comm_group	MPI_Win_lock_all
MPI_Comm_rank	MPI_Win_unlock_all
MPI_Comm_remote_group	MPI_Win_flush
MPI_Comm_remote_size	MPI_Win_flush_all
MPI_Comm_set_name	MPI_Win_flush_local
MPI_Comm_size	MPI_Win_flush_local_all
MPI_Comm_split	MPI_Win_sync
MPI_Comm_test_inter	**scan**
MPI_Intercomm_create	MPI_Exscan
MPI_Intercomm_merge	MPI_Scan
err	MPI_Iexscan
MPI_Add_error_class	MPI_Iscan
MPI_Add_error_code	**send**
MPI_Add_error_string	MPI_Send
MPI_Comm_call_errhandler	MPI_Bsend
MPI_Comm_create_errhandler	MPI_Rsend
MPI_Comm_get_errhandler	MPI_Ssend
MPI_Comm_set_errhandler	MPI_Isend
MPI_Errhandler_free	MPI_Ibsend
MPI_Error_class	MPI_Irsend

MPI_Error_string	MPI_Issend
MPI_File_call_errhandler	MPI_Send_init
MPI_File_create_errhandler	MPI_Bsend_init
MPI_File_get_errhandler	MPI_Rsend_init
MPI_File_set_errhandler	MPI_Ssend_init
MPI_Win_call_errhandler	**sendrecv**
MPI_Win_create_errhandler	MPI_Sendrecv
MPI_Win_get_errhandler	MPI_Sendrecv_replace
MPI_Win_set_errhandler	**serv**
group	MPI_Alloc_mem
MPI_Group_compare	MPI_Free_mem
MPI_Group_difference	MPI_Buffer_attach
MPI_Group_excl	MPI_Buffer_detach
MPI_Group_free	MPI_Op_create
MPI_Group_incl	MPI_Op_free
MPI_Group_intersection	**spawn**
MPI_Group_range_excl	MPI_Close_port
MPI_Group_range_incl	MPI_Comm_accept
MPI_Group_rank	MPI_Comm_connect
MPI_Group_size	MPI_Comm_disconnect
MPI_Group_translate_ranks	MPI_Comm_get_parent
MPI_Group_union	MPI_Comm_join
init	MPI_Comm_spawn
MPI_Init	MPI_Comm_spawn_multiple
MPI_Init_thread	MPI_Lookup_name
MPI_Finalize	MPI_Open_port
io	MPI_Publish_name
MPI_File_close	MPI_Unpublish_name
MPI_File_delete	**status**
MPI_File_get_amode	MPI_Get_count
MPI_File_get_atomicity	MPI_Status_set_elements
MPI_File_get_byte_offset	MPI_Status_set_cancelled
MPI_File_get_group	MPI_Test_cancelled
MPI_File_get_info	**sync**
MPI_File_get_position	MPI_Barrier
MPI_File_get_position_shared	MPI_Ibarrier
MPI_File_get_size	**time**
MPI_File_get_type_extent	MPI_Wtick

MPI_File_get_view	MPI_Wtime
MPI_File_iread_at	**topo**
MPI_File_iread	MPI_Cart_coords
MPI_File_iread_shared	MPI_Cart_create
MPI_File_iwrite_at	MPI_Cart_get
MPI_File_iwrite	MPI_Cart_map
MPI_File_iwrite_shared	MPI_Cart_rank
MPI_File_open	MPI_Cart_shift
MPI_File_preallocate	MPI_Cart_sub
MPI_File_read_all_begin	MPI_Cartdim_get
MPI_File_read_all_end	MPI_Dims_create
MPI_File_read_all	MPI_Graph_create
MPI_File_read_at_all_begin	MPI_Graph_get
MPI_File_read_at_all_end	MPI_Graph_map
MPI_File_read_at_all	MPI_Graph_neighbors
MPI_File_read_at	MPI_Graphdims_get
MPI_File_read	MPI_Graph_neighbors_count
MPI_File_read_ordered_begin	MPI_Topo_test
MPI_File_read_ordered_end	**type**
MPI_File_read_ordered	MPI_Get_address
MPI_File_read_shared	MPI_Get_elements
MPI_File_seek	MPI_Pack
MPI_File_seek_shared	MPI_Pack_external
MPI_File_set_atomicity	MPI_Pack_external_size
MPI_File_set_info	MPI_Pack_size
MPI_File_set_size	MPI_Type_commit
MPI_File_set_view	MPI_Type_contiguous
MPI_File_sync	MPI_Type_create_darray
MPI_File_write_all_begin	MPI_Type_create_hindexed
MPI_File_write_all_end	MPI_Type_create_hvector
MPI_File_write_all	MPI_Type_create_indexed_block
MPI_File_write_at_all_begin	MPI_Type_create_resized
MPI_File_write_at_all_end	MPI_Type_create_struct
MPI_File_write_at_all	MPI_Type_create_subarray
MPI_File_write_at	MPI_Type_dup
MPI_File_write	MPI_Type_free
MPI_File_write_ordered_begin	MPI_Type_get_contents
MPI_File_write_ordered_end	MPI_Type_get_envelope

MPI_File_write_ordered	MPI_Type_get_extent
MPI_File_write_shared	MPI_Type_get_true_extent
MPI_Register_datarep	MPI_Type_indexed
one2all	MPI_Type_size
MPI_Bcast	MPI_Type_vector
MPI_Scatter	MPI_Unpack_external
MPI_Scatterv	MPI_Unpack
MPI_Ibcast	
MPI_Iscatter	
MPI_Iscatterv	

I_MPI_STATS_ACCURACY

Use the `I_MPI_STATS_ACCURACY` environment variable to reduce 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 the specified portion of the total time spent inside all MPI calls (in percent).

Examples

The following code example represents a simple application with IPM statistics collection enabled:

```
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 the 0th 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 test.exe
```

Statistics output:

```
################################################################################
#
# command : unknown (completed)
# host : NODE01/Windows mpi_tasks : 4 on 1 nodes
# start : 06/17/11/14:10:40 wallclock : 0.037681 sec
# stop : 06/17/11/14:10:40 %comm : 99.17
# gbytes : 0.00000e+000 total gflop/sec : NA
#
################################################################################
```

Command:

```
> mpiexec –n 4 –env I_MPI_STATS=ipm test.exe
```

Stats output:

```
################################################################################
#
# command : unknown (completed)
# host : NODE01/Windows mpi_tasks : 4 on 1 nodes
# start : 06/17/11/14:10:40 wallclock : 0.037681 sec
# stop : 06/17/11/14:10:40 %comm : 99.17
# gbytes : 0.00000e+000 total gflop/sec : NA
#
################################################################################
# region : * [ntasks] = 4
#
# [total] <avg> min max
# entries 4 1 1 1
# wallclock 0.118763 0.0296908 0.0207312 0.0376814
# user 0.0156001 0.00390002 0 0.0156001
# system 0 0 0 0
# mpi 0.117782 0.0294454 0.0204467 0.0374543
# %comm 99.1735 98.6278 99.3973
# gflop/sec NA NA NA NA
# gbytes 0 0 0 0
#
#
# [time] [calls] <%mpi> <%wall>
# MPI_Init 0.0944392 4 80.18 79.52
# MPI_Reduce 0.0183164 8000 15.55 15.42
# MPI_Recv 0.00327056 1 2.78 2.75
```

```
# MPI_Barrier 0.00174499 800 1.48 1.47
# MPI_Send 4.23448e-006 1 0.00 0.00
# MPI_Finalize 3.07963e-006 4 0.00 0.00
# MPI_Wtime 1.53982e-006 8 0.00 0.00
# MPI_Comm_rank 1.5398e-006 4 0.00 0.00
# MPI_TOTAL 0.117782 8822 100.00 99.17
################################################################################
# region : reduce [ntasks] = 4
#
# [total] <avg> min max
# entries 8 2 2 2
# wallclock 0.0190786 0.00476966 0.00273201 0.00665929
# user 0 0 0 0
# system 0 0 0 0
# mpi 0.0183199 0.00457997 0.00255377 0.00643987
# %comm 96.0231 93.4761 97.0543
# gflop/sec NA NA NA NA
# gbytes 0 0 0 0
#
#
# [time] [calls] <%mpi> <%wall>
# MPI_Reduce 0.0183164 8000 99.98 96.00
# MPI_Finalize 3.07963e-006 4 0.02 0.02
# MPI_Wtime 3.84956e-007 4 0.00 0.00
# MPI_TOTAL 0.0183199 8008 100.00 96.02
################################################################################
# region : send [ntasks] = 4
#
# [total] <avg> min max
# entries 1 0 0 1
# wallclock 1.22389e-005 3.05971e-006 1e-006 9.23885e-006
# user 0 0 0 0
# system 0 0 0 0
# mpi 4.23448e-006 1.05862e-006 0 4.23448e-006
# %comm 34.5986 0 45.8334
# gflop/sec NA NA NA NA
# gbytes 0 0 0 0
#
#
# [time] [calls] <%mpi> <%wall>
# MPI_Send 4.23448e-006 1 100.00 34.60
################################################################################
# region : ipm_noregion [ntasks] = 4
#
# [total] <avg> min max
# entries 13 3 3 4
# wallclock 0.0996611 0.0249153 0.0140604 0.0349467
# user 0.0156001 0.00390002 0 0.0156001
# system 0 0 0 0
# mpi 0.0994574 0.0248644 0.0140026 0.0349006
# %comm 99.7957 99.5893 99.8678
# gflop/sec NA NA NA NA
# gbytes 0 0 0 0
#
#
# [time] [calls] <%mpi> <%wall>
# MPI_Init 0.0944392 4 94.95 94.76
# MPI_Recv 0.00327056 1 3.29 3.28
# MPI_Barrier 0.00174499 800 1.75 1.75
```

```
# MPI_Comm_rank 1.5398e-006 4 0.00 0.00
# MPI_Wtime 1.15486e-006 4 0.00 0.00
# MPI_TOTAL 0.0994574 813 100.00 99.80
```

4.3.3. Native and IPM Statistics

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

```
I_MPI_STATS=all
```

NOTE

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

The value `all` corresponds to `I_MPI_STATS=native:20,ipm`. To control the amount of statistics information, use the ordinary `I_MPI_STATS` values, separated by commas:

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

4.4. 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

Intel® MPI Library provides support for the ILP64 model for Fortran applications. To enable the ILP64 mode, do the following:

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

  ```
  > mpiifort -i8 -c test.f
  > mpiifort -ilp64 test.obj
  ```

- For simple programs, use the Fortran compiler wrapper option `-i8` for compilation and linkage. Specifying `-i8` will automatically assume the ILP64 library. For example:

  ```
  > mpiifort -i8 test.f
  ```

4.4.1. 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, or user-defined reduction operations.
- Do not use the `-i8` option with the deprecated functions that store or retrieve the 4-byte integer attribute (for example, `MPI_ATTR_GET`, `MPI_ATTR_PUT`, etc.). Use their recommended alternatives instead (`MPI_COMM_GET_ATTR`, `MPI_COMM_SET_ATTR`, etc).

- If you want to use the Intel® Trace Collector with the Intel MPI Library ILP64 executable files, you must use a special Intel Trace Collector library. If necessary, the `mpiifort` compiler wrapper will select the correct Intel Trace Collector library automatically.
- There is currently no support for C and C++ applications.

4.5. Unified Memory Management

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

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

These pointers also affect the C++ `new` and `delete` operators. The respective standard C library functions are used by default.

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

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

```
#include <i_malloc.h>
#include <my_malloc.h>
int main( int argc, int argv )
{
    // override normal pointers
    i_malloc = my_malloc;
    i_calloc = my_calloc;
    i_realloc = my_realloc;
    i_free = my_free;
    #ifdef _WIN32
    // also override pointers used by DLLs
    i_malloc_dll = my_malloc;
    i_calloc_dll = my_calloc;
    i_realloc_dll = my_realloc;
    i_free_dll = my_free;
    #endif
   // now start using Intel(R) libraries
}
```

4.6. Other 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 and which Intel® MPI Library configuration was used.
`3`	Output effective MPI rank, `pid` and node mapping table.
`4`	Output process pinning information.
`5`	Output Intel MPI-specific environment variables.
`6`	Output collective operation algorithms settings.
`> 6`	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 print debugging information about the application.

NOTE

Set the same <level> value for all ranks.

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

Each printed line has the following format:

```
[<identifier>] <message>
```

where:

- <identifier> is the MPI process rank, by default. If you add the '+' sign in front of the <level> number, the <identifier> assumes the following format: rank#pid@hostname. Here, rank is the MPI process rank, pid is the process ID, and hostname is the host name. If you add the '–' sign, <identifier> is not printed at all.
- *<message>* contains the debugging output.

The following examples demonstrate possible command lines with the corresponding output:

```
> mpiexec -n 1 -env I_MPI_DEBUG=2 test.exe
...
[0] MPI startup(): shared memory data transfer mode
```

The following commands are equal and produce the same output:

```
> mpiexec -n 1 -env I_MPI_DEBUG=+2 test.exe
> mpiexec -n 1 -env I_MPI_DEBUG=2,pid,host test.exe
...
[0#1986@mpicluster001] MPI startup(): shared memory data transfer mode
```

NOTE

Compiling with the /Zi, /ZI or /Z7 option 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. This is the default value.
stderr	Output to stderr.
<file_name>	Specify the output file name for debug information (the maximum file name length is 256 symbols).

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, process ID or host name is added to the file name accordingly.

I_MPI_PRINT_VERSION

Print library version information.

Syntax

`I_MPI_PRINT_VERSION=<arg>`

Arguments

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

Description

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

I_MPI_NETMASK

Choose the network interface for MPI communication over sockets.

Syntax

`I_MPI_NETMASK=<arg>`

Arguments

<arg>	Define the network interface (string parameter)
<interface_mnemonic>	Mnemonic of the network interface: `ib` or `eth`
`ib`	Select IPoIB*
`eth`	Select Ethernet. This is the default value
<network_address>	Network address. The trailing zero bits imply netmask
<network_address/netmask>	Network address. The *<netmask>* value specifies the netmask length
<list of interfaces>	A colon separated list of network addresses or interface mnemonics

Description

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

Examples

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

   ```
   I_MPI_NETMASK=ib
   ```

   ```
   I_MPI_NETMASK=eth
   ```

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

   `I_MPI_NETMASK=192.169.0.0`

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

   `I_MPI_NETMASK=192.169.0.0/24`

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

   `I_MPI_NETMASK=192.169.0.5/24:ib0:192.169.0.0`

NOTE

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

I_MPI_HARD_FINALIZE

Turn on/off the hard (ungraceful) process finalization algorithm.

Syntax

`I_MPI_HARD_FINALIZE=<arg>`

Argument

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

Description

The hard (ungraceful) finalization algorithm may significantly reduce the application finalization time.

I_MPI_TUNER_DATA_DIR

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

Syntax

`I_MPI_TUNER_DATA_DIR=<path>`

Arguments

<path>	Specify the automatic tuning utility output directory. The default value is *<installdir>*`\intel64\etc`

Description

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

I_MPI_PLATFORM

Select the intended optimization platform.

Syntax

`I_MPI_PLATFORM=<platform>`

Arguments

<platform>	Intended optimization platform (string value)
`auto[:min]`	Optimize for the oldest supported Intel® Architecture Processor across all nodes. This is the default value
`auto:max`	Optimize for the newest supported Intel® Architecture Processor across all nodes
`auto:most`	Optimize for the most numerous Intel® Architecture Processor across all nodes. In case of a tie, choose the newer platform
`uniform`	Optimize locally. The behavior is unpredictable if the resulting selection differs from node to node
`none`	Select no specific optimization
`htn | generic`	Optimize for the Intel® Xeon® Processors 5400 series and other Intel® Architecture processors formerly code named Harpertown
`nhm`	Optimize for the Intel® Xeon® Processors 5500, 6500, 7500 series and other Intel® Architecture processors formerly code named Nehalem
`wsm`	Optimize for the Intel® Xeon® Processors 5600, 3600 series and other Intel® Architecture processors formerly code named Westmere
`snb`	Optimize for the Intel® Xeon® Processors E3, E5, and E7 series and other Intel® Architecture processors formerly code named Sandy Bridge
`ivb`	Optimize for the Intel® Xeon® Processors E3, E5, and E7 V2 series and other Intel® Architecture processors formerly code named Ivy Bridge
`hsw`	Optimize for the Intel® Xeon® Processors E3, E5, and E7 V3 series and other Intel® Architecture processors formerly code named Haswell
`bdw`	Optimize for the Intel® Xeon® Processors E3, E5, and E7 V4 series and other Intel® Architecture processors formerly code named Broadwell
`knl`	Optimize for the Intel® Xeon Phi™ processor and coprocessor formerly code named Knights Landing

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.

I_MPI_THREAD_LEVEL_DEFAULT

Set this environment variable to initialize the MPI thread environment for the multi-threaded library if `MPI_Init()` call is used for initialization.

Syntax

`I_MPI_THREAD_LEVEL_DEFAULT=<threadlevel>`

Arguments

<threadlevel>	Define the default level of thread support
`SINGLE | single`	Set the default level of thread support to `MPI_THREAD_SINGLE`
`FUNNELED | funneled`	Set the default level of thread support to `MPI_THREAD_FUNNELED`. This is the default value if `MPI_Init()` call is used for initialization
`SERIALIZED | serialized`	Set the default level of thread support to `MPI_THREAD_SERIALIZED`
`MULTIPLE | multiple`	Set the default level of thread support to `MPI_THREAD_MULTIPLE`

Description

Set `I_MPI_THREAD_LEVEL_DEFAULT` to define the default level of thread support for the multi-threaded library if `MPI_Init()` call is used for initialization.

4.7. Secure Loading of Dynamic Link Libraries*

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

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

SecureDynamicLibraryLoading

Select the secure DLL loading mode.

Syntax

`SecureDynamicLibraryLoading=<value>`

Arguments

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

Description

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

I_MPI_DAT_LIBRARY

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

Syntax

`I_MPI_DAT_LIBRARY=<library>`

Arguments

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

Description

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

NOTE

The `I_MPI_DAT_LIBRARY` environment variable has no effect in the secure DLL loading mode.

SecurePath

Specify a set of directories to locate an external DLL.

Syntax

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

Arguments

`<path>`	Specify the path to a directory

Description

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

NOTE

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

4.8. User Authorization

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

- Password-based authorization
- Domain-based authorization with the delegation ability
- Limited domain-based authorization

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

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

NOTE

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

NOTE

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

This feature is supported on clusters under Windows* HPC Server 2008 R2 or 2012. Microsoft's Kerberos Distribution Center* must be enabled on your domain controller (this is the default behavior).

Using the domain-based authorization method with the delegation ability requires specific installation of the domain. You can perform this installation by using the Intel® MPI Library installer if you have domain administrator rights or by following the instructions below.

4.8.1. Active Directory* Setup

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

1. Log in on the domain controller under the administrator account.
2. Enable the delegation for cluster nodes:
    a. Go to **Administrative Tools**.
    b. In the **Active Directory Users and Computers** administrative utility open the **Computers** list.
    c. Right click on a desired computer object and select **Properties**.
    d. If the account is located:
        - in a Windows 2000 functional level domain, check the **Trust computer for delegation** option;
        - in a Windows 2003 or newer functional level domain, select the **Delegation** tab and check the **Trust this computer for delegation to any service (Kerberos only)** option.
3. Enable the delegation for users:
    a. In the **Active Directory Users and Computers** administrative utility open the **Users** list.
    b. Right click on a desired user object and select **Properties**.
    c. Select the **Account** tab and disable the **Account is sensitive and cannot be delegated** option.
4. Register service principal name (SPN) for cluster nodes. Use one of the following methods for registering SPN:
    a. Use the Microsoft*-provided `setspn.exe` utility. For example, execute the following command on the domain controller:

    ```
    > setspn.exe -A impi_hydra/<host>:<port>/impi_hydra <host>
    ```

    where:
        - *<host>* is the cluster node name.
        - *<port>* is the Hydra port. The default value is `8679`. Change this number only if your hydra service uses the non-default port.
    b. Log into each desired node under the administrator account and execute the command:

    ```
    > hydra_service -register_spn
    ```

NOTE

In case of any issues with the MPI task start, reboot the machine from which the MPI task is started. Alternatively, execute the command:

```
> klist purge
```

I_MPI_AUTH_METHOD

Select a user authorization method.

Syntax

`I_MPI_AUTH_METHOD=`*<method>*

Arguments

<method>	Define the authorization method

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

Description

Set this environment variable to select a desired authorization method. If this environment variable is not defined, `mpiexec` uses the password-based authorization method by default. Alternatively, you can change the default behavior by using the `-delegate` or `-impersonate` options.

5. Glossary

cell	A pinning resolution in descriptions for pinning property.
hyper-threading technology	A feature within the IA-64 and Intel® 64 family of processors, where each processor core provides the functionality of more than one logical processor.
logical processor	The basic modularity of processor hardware resource that allows a software executive (OS) to dispatch task or execute a thread context. Each logical processor can execute only one thread context at a time.
multi-core processor	A physical processor that contains more than one processor core.
multi-processor platform	A computer system made of two or more physical packages.
processor core	The circuitry that provides dedicated functionalities to decode, execute instructions, and transfer data between certain sub-systems in a physical package. A processor core may contain one or more logical processors.
physical package	The physical package of a microprocessor capable of executing one or more threads of software at the same time. Each physical package plugs into a physical socket. Each physical package may contain one or more processor cores.
processor topology	Hierarchical relationships of "shared vs. dedicated" hardware resources within a computing platform using physical package capable of one or more forms of hardware multi-threading.

6. Index