Intel® MPI Library Developer Reference for Windows* OS

Contents

Intel® MPI Library Developer Reference for Windows OS*

1

This Developer Reference provides you with the complete reference for the Intel MPI Library. It is intended to help a user fully utilize the Intel MPI Library functionality.

For examples and detailed functionality descriptions, refer to the Intel® MPI Library Developer Guide for Windows* OS.

What's New

The following are some popular topics in the Intel MPI Library Developer Reference:

Command Reference

Command Reference provides reference information on compilation and runtime commands (cpuinfo, impi_info) and describes how to use these commands.

Environment Variable Reference

Environment Variable Reference provides syntax, arguments, and descriptions for Fabrics Control, Tuning, Autotuning, Main Thread Pinning, and I_MPI_ADJUST Family environment variables.

Hydra Global Options and Environment Variables

Describes the Global Options and provides Environment Variables used with the Hydra process manager.

Package Layout

You can find MPI benchmark sources, compiler configs, and MPI binding at `<I_MPI_install_dir>\opt\mpi`.

Documentation for Older Versions

Documentation for some older versions of the Intel® MPI Library is available as downloads only.

To download specific versions of prior Intel MPI Library documentation, refer to one of the following locations:

- Downloadable Documentation: Intel® oneAPI Toolkits and Components
- Intel® MPI Library - Legacy Documentation
- Download Documentation: Intel® Parallel Studio XE

Developer Reference Sections

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.

Document Organization

Section	Description
Section 1. Introduction	Introduces this document and the Intel MPI Library.
Section 2. Command Reference	Describes compilation and job startup commands and their options.
Section 3. Environment Variable Reference	Describes environment variables .

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.

You can get the latest information for the Intel® MPI Library at https://www.intel.com/content/www/us/en/developer/tools/oneapi/mpi-library.html.

What's New

This page lists some important changes to the product that are reflected in the documentation. For a list of all changes, refer to the Intel® MPI Library Release Notes or Intel® MPI Library Release Notes for Windows* OS .

Intel® MPI Library 2021.11 (Intel® oneAPI 2024.0)

- Added new environment variables to Other Environment Variables page: `I_MPI_FILESYSTEM`, `I_MPI_FILESYSTEM_FORCE`, `I_MPI_FILESYSTEM_CB_NODES`, `I_MPI_FILESYSTEM_CB_CONFIG_LIST`, `I_MPI_FILESYSTEM_NFS_DIRECT`, and `I_MPI_FILESYSTEM_GPFS_DIRECT`.
- Deprecated environment variables: `I_MPI_EXTRA_FILESYSTEM` and `I_MPI_EXTRA_FILESYSTEM_FORCE`.

Intel® MPI Library 2021.10 (Intel® oneAPI 2023.2)

- Added new environment variable `_MPI_OFI_MATCH_COMPLETE` to the OFI*-capable Network Fabrics Control page.
- Added the compiler-specific MPI wrappers: `mpiicpx` and `mpiicx` to the Compiler Commands page.

Additionally, minor updates have been made to fix inaccuracies in the document and improve user experience.

Intel® MPI Library 2021.9 (Intel® oneAPI 2023.1)

- Added `I_MPI_SHM`, `I_MPI_SHM_CELL_FWD_SIZE`, `I_MPI_SHM_CELL_BWD_SIZE`, `I_MPI_SHM_CELL_EXT_SIZE`, `I_MPI_SHM_CELL_FWD_NUM`, `I_MPI_SHM_CELL_BWD_NUM`, `I_MPI_SHM_CELL_EXT_NUM_TOTAL`, `I_MPI_SHM_CELL_FWD_HOLD_NUM`, `I_MPI_SHM_SEND_SPIN_COUNT`, `I_MPI_SHM_RECV_SPIN_COUNT` variables to Shared Memory Control page.
- Added new value for `I_MPI_HYDRA_BOOTSTRAP` variable.

Intel® MPI Library 2021.8 (Intel® oneAPI 2023.0)

- Added `I_MPI_SPAWN` function.
- Added the following CVARs: `I_MPI_OFFLOAD_SYMMETRIC`, `I_MPI_OFFLOAD_RDMA`, `I_MPI_OFFLOAD_FAST_MEMCPY`, `I_MPI_OFFLOAD_IPC`.
- Changed default value of `I_MPI_OFFLOAD_PIPELINE_THRESHOLD` variable.

Intel® MPI Library 2021.7.1 (Intel® oneAPI 2022.4)

- No documentation changes.

Intel® MPI Library 2021.7 (Intel® oneAPI 2022.3)

- No documentation changes.

Intel® MPI Library 2021.6 (Intel® oneAPI 2022.2)

- No documentation changes.

Intel® oneAPI 2021.5

- Added `hcoll` argument for `I_MPI_COLL_EXTERNAL`.
- Added option to Global Options topic.
- Added `I_MPI_COLL_DIRECT` variable to I_MPI_ADJUST Family Environment Variables topic.

Intel® oneAPI 2021.4

- No documentation changes.

Intel® oneAPI 2021.3

- Changed the default pinning order to `bunch`.
- Added new value for the I_MPI_SHM variable (icx).
- Added new variable I_MPI_HYDRA_BSTRAP_KEEP_ALIVE.

Additionally, minor updates have been made to fix inaccuracies in the document and improve user experience.

Intel® oneAPI 2021.2

- No documentation changes.

Intel® oneAPI Gold

- Removed all content specific to Intel® Parallel Studio XE (see notice on title page).
- Added Intel® Ethernet 800 Series support.
- Added MPI + OpenMP offload examples.
- Added new algorithm for MPI_Sendrecv_replace (I_MPI_ADJUST_SENDRECV_REPLACE=2).
- Reworked directory layout:

  - Removed intel64/.
  - Mpivars.[c]sh and mpi modulefile moved to env/.
  - Mpivars.[c]sh renamed to vars.[c]sh.
- Removed deprecated symbolic links.
- Removed static libraries for debug configurations.

Notational Conventions

The following conventions are used in this document.

This type style	Document names
This type style	Hyperlinks
`This type style`	Commands, arguments, options, file names
`THIS_TYPE_STYLE`	Environment variables
<this type style>	Variables or placeholders for actual values
[items]	Optional items
{ item | item }	Selectable items separated by vertical bar(s)

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

User Authorization

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

I_MPI_AUTH_METHOD

Select a user authorization method.

Syntax

I_MPI_AUTH_METHOD=*<method>*

Arguments

<method>	Define the authorization method.
<not set>	Use the default bootstrap mechanism, not relies on hydra_service. 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. The setting of I_MPI_AUTH_METHOD to any value above leads to the usage of service-based bootstrap. Alternatively, you can change the default behavior by using the -delegate or -impersonate options.

See the User Authorization section of the Intel® MPI Library Developer Guide for more information.

Command Reference

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

- Compilation Commands lists the available Intel® MPI Library compiler commands, related options, and environment variables.
- mpiexec gives full information on commands, options and environment variables for the Hydra process manager.
- cpuinfo provides the syntax, arguments, description and output examples for the cpuinfo utility.
- impi_info provides information on available environment variables.

Compiler Commands

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

Intel MPI Library Compiler Wrappers

Compiler Command	Underlying Compiler	Supported Languages
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
mpiicx.bat	icl.exe	C
mpiicpc.bat	icl.exe	C++
mpiicpx.bat	icl.exe	C++
mpiifort.bat	ifort.exe	Fortran 77/Fortran 95
mpiifx	ifort.exe	Fortran 77/Fortran 95

Notes on Compiler Commands

- 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 Intel® MPI Library System Requirements.
- To display mini-help of a compiler command, execute it without any parameters.
- Compiler wrapper scripts are located in the `<install-dir>\bin` directory.
- The environment settings can be established by running the `<install-dir>\env\vars.bat` file. To use a specific library configuration, pass the `release` or `debug` arguments to the script to switch to the corresponding configuration. The ordinary multi-threaded optimized library is chosen by default. Alternatively, you can use the `I_MPI_LIBRARY_KIND` environment variable to specify a configuration and source the script without arguments.
- Ensure that the corresponding underlying compiler is already in your `PATH`. If you use Intel® compilers, run the `vars.bat` file from the installation directory to set up the compiler environment.

Compilation 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 `<install-dir>\opt\mpi\etc`.

You can create your own profile as `<install-dir>\opt\mpi\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 `<install-dir>\opt\mpi\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 `vars. 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. The default value is `libVTmc.so`.

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

If you specify the `-i8` option for the Intel Fortran Compiler, you still have to use the `ilp64` option for linkage.

-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.

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:

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

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.

The table below lists the recommended product default LLVM and IL0 compiler options and commands used to invoke them.

LLVM Compiler Options for Intel® oneAPI

Language/Model	Product Name	Compiler Driver	Compiler Wrapper	Command	Example
C	Intel® oneAPI DPC++/C++ Compiler	icx	mpiicc	-cc=icx	> mpicc -cc=icl.exe -c test.c
C++	Intel® oneAPI DPC++/C++ C	icpx	mpiicpc	-cxx=icpx	> mpiicpc -cxx=icpx -c test.cpp
SYCL*	Intel® oneAPI DPC++/C++ Compiler	> icx-cl -fsycl	mpiicpc	> -cxx=icx -fsycl	> mpiicpc -cxx=icpx -fsycl -c test.cpp
Fortran	Intel® oneAPI Fortran Compiler	ifx	mpiifort	-fc=ifx	> mpiifort -fc=ifx -c test.

> **NOTE** Make sure that the wrapper name is 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.

-norpath

Use this option to disable `rpath` for the compiler wrapper for the Intel® MPI Library.

mpiexec.hydra

Launches an MPI job using the Hydra process manager.

Syntax

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

or

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

Arguments

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

Description

Use the `mpiexec` utility to run MPI applications using the Hydra process manager.

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.

Global Hydra 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 ':'.

-tune *<filename>*

Use this option to specify the file name that contains the tuning data in a binary format.

-usize *<usize>*

Use this option to set `MPI_UNIVERSE_SIZE`, which is available as an attribute of the `MPI_COMM_WORLD`.

<size>	Define the universe size
SYSTEM	Set the size equal to the number of cores passed to `mpiexec` through the hostfile or the resource manager.
INFINITE	Do not limit the size. This is the default value.
<value>	Set the size to a numeric value ≥ 0.

-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 following 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. For example:

```
> type machinefile
node0:2
node1:2
node0:1
```

–hosts-group

Use this option to set node ranges using brackets, commas, and dashes (like in Slurm* Workload Manager).

For more details, see the `I_MPI_HYDRA_HOST_FILE` environment variable in Hydra Environment Variables.

–silent-abort

Use this option to disable abort warning messages.

For more details, see the `I_MPI_SILENT_ABORT` environment variable in Hydra Environment Variables.

–nameserver

Use this option to specify the nameserver in the `hostname:port` format.

For more details, see the `I_MPI_HYDRA_NAMESERVER` environment variable in Hydra Environment Variables.

-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.

> **NOTE** The option does not work for localhost.

-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 with one executable per line. Blank lines and lines that start with '#' are ignored. Other options specified in the command line are treated as global.

You can specify global options in configuration files loaded by default (`mpiexec.conf` in `<installdir>/etc`, `~/.mpiexec.conf`, and `mpiexec.conf` in the working directory). The remaining options can be specified in the command line.

-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 main 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.

Arguments

-l, -prepend-rank

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

-s *<spec>*

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

Arguments

`<spec>`	Define MPI process ranks
`all`	Use all processes.
`none`	Do not direct standard output to any 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 outut 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.

–validate [–host <hostname>]

Validate the encrypted credentials for the current user.

–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.

-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. See the `I_MPI_HYDRA_SERVICE_PORT` environment variable for more details.

-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.

Arguments

-v6

Use this option to force using the IPv6 protocol.

Local Hydra 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 *<number-of-processes>* or -np *<number-of-processes>*

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

-env *<envar> <value>*

Use this option to set the *<envar>* 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.

> **NOTE** The option does not work for localhost.

-envexcl *<list-of-envvar-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.

cpuinfo

Provides information on processors used in the system.

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. • *ThreadId* - unique processor identifier within a core.

	• *CoreId* - unique core identifier within a package. • *PackageId* - 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.
n	Table shows the following information about NUMA nodes: • NUMA Id - NUMA node identifier. • Processors - a list of processors in this node. If the node has no processors, the node is not shown.
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        |
|_____|_____|_____|_____|_____|_____|
```

impi_info

Provides information on available Intel® MPI Library environment variables.

Syntax

`impi_info <options>`

Arguments

`<options>`	List of options.
`-a | -all`	Show all IMPI variables.
`-h | -help`	Show a help message.
`-v | -variable`	Show all available variables or description of the specified variable.
`-c | -category`	Show all available categories or variables of the specified category.
`-e | -expert`	Show all expert variables.

Description

The `impi_info` utility provides information on environment variables available in the Intel MPI Library. For each variable, it prints out the name, the default value, and the value data type. By default, a reduced list of variables is displayed. Use the `-all` option to display all available variables with their descriptions.

The example of the `impi_info` output:

```
> impi_info

| NAME                               | DEFAULT VALUE | DATA TYPE  |
=====================================================================
| I_MPI_THREAD_SPLIT                 | 0             | MPI_INT    |
| I_MPI_THREAD_RUNTIME               | none          | MPI_CHAR   |
| I_MPI_THREAD_MAX                   | -1            | MPI_INT    |
| I_MPI_THREAD_ID_KEY                | thread_id     | MPI_CHAR   |
```

Environment Variable Reference

This section provides information on different variables:

- Compilation Environment Variables
- Hydra Environment Variables
- I_MPI_ADJUST Family Environment Variables
- Tuning Environment Variables
- Environment Variables for Main Thread Pinning
- Environment Variables for Fabrics Control
- Other Environment Variables

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>*

Argument

<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}

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.
	
	  • `I_MPI_CC=<compiler>` affects `mpiicx` and `mpicc`
	  • `I_MPI_CXX=<compiler>` affects `mpiicpx` and `mpicxx`
	  • `I_MPI_FC=<compiler>` affects `mpifc`
	  • `I_MPI_F77=<compiler>` affects `mpif77`
	  • `I_MPI_F90=<compiler>` affects `mpiifx` and `mpif90`

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.

> **NOTE** If you are using the Visual Studio integration, you may need to use `I_MPI_ONEAPI_ROOT`.

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 *<install-dir>*\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

Argument	Library Version
opt	Multi-threaded optimized library. This is the default value
dbg	Multi-threaded debug library

Description

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

Hydra Environment Variables

I_MPI_HYDRA_HOST_FILE

Set the host file to run the application.

Syntax

```
I_MPI_HYDRA_HOST_FILE=<arg>
```

Argument

<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_HOSTS_GROUP

Set node ranges using brackets, commas, and dashes.

Syntax

```
I_MPI_HYDRA_HOSTS_GROUP=<arg>
```

Argument

<arg>	Set a node range.

Description

Set this variable to be able to set node ranges using brackets, commas, and dashes (like in Slurm* Workload Manager). For example:

```
I_MPI_HYDRA_HOSTS_GROUP="hostA[01-05],hostB,hostC[01-05,07,09-11]"
```

You can set node ranges with the `-hosts-group` option.

I_MPI_HYDRA_DEBUG

Print out the debug information.

Syntax

`I_MPI_HYDRA_DEBUG=<arg>`

Argument

`<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>`

Argument

`<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

Set the timeout period for `mpiexec` .

Syntax

`I_MPI_JOB_TIMEOUT=<timeout>`

`I_MPI_MPIEXEC_TIMEOUT=<timeout>`

Argument

`<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.

I_MPI_JOB_STARTUP_TIMEOUT

Set the `mpiexec` job startup timeout.

Syntax

I_MPI_JOB_STARTUP_TIMEOUT=*<timeout>*

Argument

<timeout>	Define `mpiexec` startup 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 if some processes are not launched. The *<timeout>* value should be greater than zero.

I_MPI_HYDRA_BOOTSTRAP

Set the bootstrap server.

Syntax

I_MPI_HYDRA_BOOTSTRAP=*<arg>*

Argument

<arg>	String parameter
service	Use hydra service agent
lsf	Use the LSF blaunch command
powershell	Use the powershell based bootstrap. This is the default values.

Description

Set this environment variable to specify the bootstrap server.

> **NOTE** LSF bootstrap is chosen automatically if LSF environment variables are found. If the -hosts option is specified, LSF bootstrap will not be chosen by default. Set I_MPI_HYDRA_BOOTSTRAP=lsf for this case.

I_MPI_HYDRA_BOOTSTRAP_EXEC

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

Syntax

I_MPI_HYDRA_BOOTSTRAP_EXEC=*<arg>*

Argument

<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>*

Argument

<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_PERHOST

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

Syntax

I_MPI_PERHOST=*<value>*

Argument

<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 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>*

Argument

<num>	Number

`<n> >= 0`	The default value is 16. This value means that hierarchical structure is enabled if the number of nodes is more than 16.
	If I_MPI_HYDRA_BRANCH_COUNT=0, then there is no hierarchical structure.
	If I_MPI_HYDRA_BRANCH_COUNT=-1, then branch count is equal to default value.

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>`

Argument

`<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>`

Argument

`<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

Specify a temporary directory.

Syntax

`I_MPI_TMPDIR=<arg>`

Argument

`<arg>`	String parameter
`<path>`	Temporary directory. The default value is `/tmp`

Description

Set this environment variable to specify a directory for temporary files.

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>*

Argument

<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

Specify a range of allowed port numbers.

Syntax

I_MPI_PORT_RANGE=*<range>*

Argument

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

Description

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

I_MPI_HYDRA_SERVICE_PORT

Set the port on which the hydra service is installed.

Syntax

I_MPI_HYDRA_SERVICE_PORT=*<int>*

Argument

<int>	Define the port number

Description

Set this environment variable to inform mpiexec.hydra, on which port the hydra service is installed. Use this variable if you want to run a number of services on different ports.

To be able to run a number of hydra services, follow these steps:

1. Start cmd and run hydra services:

```
> start hydra_service -p <port1> -d> start hydra_service -p <port2> -d
```
2. Set the environment variable to choose the service to be used:

```
set I_MPI_HYDRA_SERVICE_PORT="port2"
```

3. Run `mpiexec` as usual

I_MPI_SILENT_ABORT

Control abort warning messages.

Syntax

`I_MPI_SILENT_ABORT=<arg>`

Argument

`<arg>`	Binary indicator
`enable	yes	on	1`	Do not print abort warning message
`disable	no	off	0`	Print abort warning message. This is the default value

Description

Set this variable to disable printing of abort warning messages. The messages are printed in case of the MPI_Abort call.

You can also disable printing of these messages with the `-silent-abort` option.

I_MPI_HYDRA_NAMESERVER

Specify the nameserver.

Syntax

`I_MPI_HYDRA_NAMESERVER=<arg>`

Argument

`<arg>`	String parameter
`<hostname>:<port>`	Set the hostname and the port.

Description

Set this variable to specify the nameserver for your MPI application in the following format:

```
I_MPI_HYDRA_NAMESERVER = hostname:port
```

You can set the nameserver with the `-nameserver` option.

I_MPI_HYDRA_BSTRAP_KEEP_ALIVE

Set this variable to keep `hydra_bstrap_proxy` alive after launching `hydra_pmi_proxy`.

Syntax

`I_MPI_HYDRA_BSTRAP_KEEP_ALIVE=<arg>`

Argument

`<arg>`	Binary indicator
`enable	yes	on	1`	Do not close `hydra_bstrap_proxy`.
`disable	no	off	0`	Close `hydra_bstrap_proxy`. This is the default value.

Description

Set this variable to keep `hydra_bstrap_proxy` alive after launching `hydra_pmi_proxy`. It allows you to keep full process tree (`mpiexec` —> `hydra_bstrap_proxy` —> `hydra_pmi_proxy` —> app) connected on localhost. The default value closes `hydra_bstrap_proxy` to reduce the number of running processes.

I_MPI_ADJUST Family Environment Variables

I_MPI_ADJUST_<opname>

Control collective operation algorithm selection.

Syntax

```
I_MPI_ADJUST_<opname>="<presetid>[:<conditions>][;<presetid>:<conditions>[...]]"
```

Arguments

`<presetid>`	Preset identifier
`>= 0`	Set a number to select the desired algorithm. The value 0 uses basic logic of the collective algorithm selection.

`<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.

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

Environment Variable	Collective Operation	Algorithms
		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
		12. NUMA aware SHM-based (SSE4.2)
		13. NUMA aware SHM-based (AVX2)

Environment Variable	Collective Operation	Algorithms
		14. NUMA aware SHM-based (AVX512)
`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_SCATTE R`	`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_SENDRECV_REPL ACE`	`MPI_Sendrecv_replace`	1. Generic 2. Uniform (with restrictions)

Environment Variable	Collective Operation	Algorithms
`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 7. Reduce scatter allgather 8. SMP 9. Nreduce
`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 5. SMP 6. Tree knominal 7. Tree kary
`I_MPI_ADJUST_IEXSCAN`	`MPI_Iexscan`	1. Recursive doubling 2. SMP
`I_MPI_ADJUST_IGATHER`	`MPI_Igather`	1. Binomial 2. Knomial
`I_MPI_ADJUST_IGATHERV`	`MPI_Igatherv`	1. Linear 2. Linear ssend
`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`	1. Recursive Doubling 2. SMP

Environment Variable	Collective Operation	Algorithms
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.

> **NOTE** The I_MPI_ADJUST_SENDRECV_REPLACE=2 ("Uniform") algorithm can be used only in the case when datatype and objects count are the same across all ranks.

To get the maximum number (range) of presets available for each collective operation, use the `impi_info` command:

```
> impi_info -v I_MPI_ADJUST_ALLREDUCE
I_MPI_ADJUST_ALLREDUCE
  MPI Datatype:
    MPI_CHAR
  Description:
    Control selection of MPI_Allreduce algorithm presets.
    Arguments
    <presetid> - Preset identifier
    range: 0-27
```

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_<opname>_LIST

Syntax

`I_MPI_ADJUST_<opname>_LIST=<presetid1>[-<presetid2>][,<presetid3>][,<presetid4>-`
`<presetid5>]`

Description

Set this environment variable to specify the set of algorithms to be considered by the Intel MPI runtime for a specified `<opname>`. This variable is useful in autotuning scenarios, as well as tuning scenarios where users would like to select a certain subset of algorithms.

NOTE Setting an empty string disables autotuning for the `<opname>` collective.

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_EXTERNAL

Syntax

`I_MPI_COLL_EXTERNAL=<arg>`

Arguments

<arg>	Description
enable | yes | on | 1	Enable the external collective operations functionality using available collectives libraries.
disable | no | off | 0	Disable the external collective operations functionality. This is the default value.
hcoll	Enable the external collective operations functionality using HCOLL library.

Description

Set this environment variable to enable external collective operations. For reaching better performance, use an autotuner after enabling `I_MPI_COLL_EXTERNAL`. This process gets the optimal collectives settings.

To force external collective operations usage, use the following `I_MPI_ADJUST_<opname>` values: I_MPI_ADJUST_ALLREDUCE=24, I_MPI_ADJUST_BARRIER=11, I_MPI_ADJUST_BCAST=16, I_MPI_ADJUST_REDUCE=13, I_MPI_ADJUST_ALLGATHER=6, I_MPI_ADJUST_ALLTOALL=5, I_MPI_ADJUST_ALLTOALLV=5, I_MPI_ADJUST_SCAN=3, I_MPI_ADJUST_EXSCAN=3, I_MPI_ADJUST_GATHER=5, I_MPI_ADJUST_GATHERV=4, I_MPI_ADJUST_SCATTER=5, I_MPI_ADJUST_SCATTERV=4, I_MPI_ADJUST_ALLGATHERV=5, I_MPI_ADJUST_ALLTOALLW=2, I_MPI_ADJUST_REDUCE_SCATTER=6, I_MPI_ADJUST_REDUCE_SCATTER_BLOCK=4, I_MPI_ADJUST_IALLGATHER=5, I_MPI_ADJUST_IALLGATHERV=5, I_MPI_ADJUST_IGATHERV=3, I_MPI_ADJUST_IALLREDUCE=9, I_MPI_ADJUST_IALLTOALLV=2, I_MPI_ADJUST_IBARRIER=2, I_MPI_ADJUST_IBCAST=5, I_MPI_ADJUST_IREDUCE=4.

For more information on HCOLL tuning, refer to NVIDIA* documentation.

I_MPI_COLL_DIRECT

Syntax

`I_MPI_COLL_DIRECT=<arg>`

Arguments

<arg>	Description
on	Enable direct collectives. This is the default value.
off	Disable direct collectives.

Description

Set this environment variable to control direct collectives usage. Disable this variable to eliminate OFI* usage for intra-node communications in case of shm:ofi fabric.

I_MPI_CBWR

Control reproducibility of floating-point operations results across different platforms, networks, and topologies in case of the same number of processes.

Syntax

`I_MPI_CBWR=<arg>`

Arguments

<arg>	CBWR compatibility mode	Description
0	None	Do not use CBWR in a library-wide mode. CNR-safe communicators may be created with `MPI_Comm_dup_with_info` explicitly. This is the default value.
1	Weak mode	Disable topology aware collectives. The result of a collective operation does not depend on the rank placement. The mode guarantees results reproducibility across different runs on the same cluster (independent of the rank placement).
2	Strict mode	Disable topology aware collectives, ignore CPU architecture, and interconnect during algorithm selection. The mode guarantees results reproducibility across different

	runs on different clusters (independent of the rank placement, CPU architecture, and interconnection)

Description

Conditional Numerical Reproducibility (CNR) provides controls for obtaining reproducible floating-point results on collectives operations. With this feature, Intel MPI collective operations are designed to return the same floating-point results from run to run in case of the same number of MPI ranks.

Control this feature with the `I_MPI_CBWR` environment variable in a library-wide manner, where all collectives on all communicators are guaranteed to have reproducible results. To control the floating-point operations reproducibility in a more precise and per-communicator way, pass the `{"I_MPI_CBWR", "yes"}` key-value pair to the `MPI_Comm_dup_with_info` call.

NOTE
Setting the `I_MPI_CBWR` in a library-wide mode using the environment variable leads to performance penalty.

CNR-safe communicators created using `MPI_Comm_dup_with_info` always work in the strict mode. For example:

```
MPI_Info hint;
MPI_Comm cbwr_safe_world, cbwr_safe_copy;
MPI_Info_create(&hint);
MPI_Info_set(hint, "I_MPI_CBW", "yes");
MPI_Comm_dup_with_info(MPI_COMM_WORLD, hint, & cbwr_safe_world);
MPI_Comm_dup(cbwr_safe_world, & cbwr_safe_copy);
```

In the example above, both cbwr_safe_world and cbwr_safe_copy are CNR-safe. Use cbwr_safe_world and its duplicates to get reproducible results for critical operations.

Note that `MPI_COMM_WORLD` itself may be used for performance-critical operations without reproducibility limitations.

Tuning Environment Variables

I_MPI_TUNING_MODE

Select the tuning method.

Syntax

`I_MPI_TUNING_MODE=<arg>`

Arguments

<arg >	Description
none	Disable tuning modes. This is the default value.
`auto`	Enable autotuner.
`auto:application`	Enable autotuner with application focused strategy (alias for auto).
`auto:cluster`	Enable autotuner without application specific logic. This is typically performed with the help of benchmarks (for example, IMB-MPI1) and proxy applications.

Description

Set this environment variable to enable the autotuner functionality and set the autotuner strategy.

I_MPI_TUNING_BIN

Specify the path to tuning settings in a binary format.

Syntax

I_MPI_TUNING_BIN=<*path*>

Argument

<path >	A path to a binary file with tuning settings. By default, Intel® MPI Library uses the binary tuning file located at <$I_MPI_ONEAPI_ROOT/etc>.

Description

Set this environment variable to load tuning settings in a binary format.

I_MPI_TUNING_BIN_DUMP

Specify the file for storing tuning settings in a binary format.

Syntax

I_MPI_TUNING_BIN_DUMP=<*filename*>

Argument

<filename>	A file name of a binary that stores tuning settings. By default, the path is not specified.

Description

Set this environment variable to store tuning settings in binary format.

I_MPI_TUNING

Load tuning settings in a JSON format.

Syntax

I_MPI_TUNING=<*path*>

Argument

<path>	A path to a JSON file with tuning settings.

Description

Set this environment variable to load tuning settings in a JSON format.

By default, the Intel® MPI Library loads tuning settings in a binary format. If it is not possible, the Intel MPI Library loads the tuning file in a JSON format specified through the I_MPI_TUNING environment variable. Thus, to enable JSON tuning, turn off the default binary tuning: I_MPI_TUNING_BIN="". If it is not possible to load tuning settings from a JSON file and in a binary format, the default tuning values are used.

You do not need to turn off binary or JSON tuning settings if you use I_MPI_ADJUST family environment variables. The algorithms specified with I_MPI_ADJUST environment variables always have priority over binary and JSON tuning settings.

See Also

- Autotuning
- Environment Variables for Autotuning

Autotuning

If an application spends significant time in MPI collective operations, tuning might improve its performance.

Tuning is very dependent on the specifications of the particular platform. Autotuner searches for the best possible implementation of a collective operation during application runtime. Each collective operation has its own presets, which consist of the algorithm and its parameters, that the autotuning function goes through and then evaluates the performance of each one. Once autotuning has evaluated the search space, it chooses the fastest implementation and uses it for the rest of the application runtime, and this improves application performance. The autotuner search space can be modified by the `I_MPI_ADJUST_<opname>_LIST` variable (see I_MPI_ADJUST Family Environment Variables).

Autotuner determines the tuning parameters and makes them available for autotuning using `I_MPI_TUNING_MODE` and the `I_MPI_TUNING_AUTO` family environment variables to find the best settings (see Tuning Environment Variables and I_MPI_TUNING_AUTO Family Environment Variables).

> **NOTE** `I_MPI_TUNING_MODE` and the `I_MPI_TUNING_AUTO` family environment variables support only Intel processors, and cannot be used on other platforms.

The collectives currently available for autotuning are: MPI_Allreduce, MPI_Bcast, MPI_Barrier, MPI_Reduce, MPI_Gather, MPI_Scatter, MPI_Alltoall, MPI_Allgatherv, MPI_Reduce_scatter, MPI_Reduce_scatter_block, MPI_Scan, MPI_Exscan, MPI_Iallreduce, MPI_Ibcast, MPI_Ibarrier, MPI_Ireduce, MPI_Igather, MPI_Iscatter, MPI_Ialltoall, MPI_Iallgatherv, MPI_Ireduce_scatter, MPI_Ireduce_scatter_block, MPI_Iscan, and MPI_Iexscan.

Using autotuner involves these steps:

1. Launch the application with autotuner enabled and specify the dump file that stores results:

   `I_MPI_TUNING_MODE=auto`

   `I_MPI_TUNING_BIN_DUMP=tuning-results.dat`

2. Launch the application with the tuning results generated at the previous step:

   `I_MPI_TUNING_BIN=./tuning-results.dat`

   Or use the `-tune` Hydra option.

If you experience performance issues, see I_MPI_TUNING_AUTO Family Environment Variables.

Examples

```
• > export I_MPI_TUNING_MODE=auto
  > export I_MPI_TUNING_AUTO_SYNC=1
  > export I_MPI_TUNING_AUTO_ITER_NUM=5
  > export I_MPI_TUNING_BIN_DUMP=tuning_results.dat
  > mpirun -n 128 -ppn 64 IMB-MPI1 allreduce -iter 1000,800 -time 4800
• > export I_MPI_TUNING_BIN=./tuning_results.dat
  > mpirun -n 128 -ppn 64 IMB-MPI1 allreduce -iter 1000,800 -time 4800
```

> **NOTE** To tune collectives on a communicator identified with the help of Application Performance Snapshot (APS), execute the following variable at step 1:
> `I_MPI_TUNING_AUTO_COMM_LIST=comm_id_1, … , comm_id_n.`

See Also

I_MPI_TUNING_AUTO Family Environment Variables

Make HPC Clusters More Efficient Using Intel® MPI Library Tuning Utilities

I_MPI_TUNING_AUTO Family Environment Variables

NOTE You must set I_MPI_TUNING_MODE to use any of the I_MPI_TUNING_AUTO family environment variables.

NOTE The I_MPI_TUNING_AUTO family environment variables support only Intel processors, and cannot be used on other platforms.

I_MPI_TUNING_AUTO_STORAGE_SIZE

Define size of the per-communicator tuning storage.

Syntax

I_MPI_TUNING_AUTO_STORAGE_SIZE=*<size>*

Argument

<size>	Specify size of the communicator tuning storage. The default size of the storage is 512 Kb.

Description

Set this environment variable to change the size of the communicator tuning storage.

I_MPI_TUNING_AUTO_ITER_NUM

Specify the number of autotuner iterations.

Syntax

I_MPI_TUNING_AUTO_ITER_NUM=*<number>*

Argument

<number>	Define the number of iterations. By default, it is 1.

Description

Set this environment variable to specify the number of autotuner iterations. The greater iteration number produces more accurate results.

NOTE To check if all possible algorithms are iterated, make sure that the total number of collective invocations for a particular message size in a target application is at least equal the value of I_MPI_TUNING_AUTO_ITER_NUM multiplied by the number of algorithms.

I_MPI_TUNING_AUTO_WARMUP_ITER_NUM

Specify the number of warmup autotuner iterations.

Syntax

I_MPI_TUNING_AUTO_WARMUP_ITER_NUM=*<number>*

Argument

<number>	Define the number of iterations. By default, it is 1.

Description

Set this environment variable to specify the number of autotuner warmup iterations. Warmup iterations do not impact autotuner decisions and allow to skip additional iterations, such as infrastructure preparation.

I_MPI_TUNING_AUTO_SYNC

Enable the internal barrier on every iteration of the autotuner.

Syntax

I_MPI_TUNING_AUTO_SYNC=*<arg>*

Argument

<arg>	**Binary indicator**
enable | yes | on | 1	Align the autotuner with the IMB measurement approach.
disable | no | off | 0	Do not use the barrier on every iteration of the autotuner. This is the default value.

Description

Set this environment variable to control the IMB measurement logic. Setting this variable to 1 may lead to overhead due to an additional MPI_Barrier call.

I_MPI_TUNING_AUTO_COMM_LIST

Control the scope of autotuning.

Syntax

I_MPI_TUNING_AUTO_COMM_LIST=*<comm_id_1, ..., comm_id_n>*

Argument

<comm_id_n, ...>	Specify communicators to be tuned.

Description

Set this environment variable to specify communicators to be tuned using their unique id. By default, the variable is not specified. In this case, all communicators in the application are involved into the tuning process.

> **NOTE** To get the list of communicators available for tuning, use the Application Performance Snapshot (APS) tool, which supports per communicator profiling starting with the 2019 Update 4 release.

I_MPI_TUNING_AUTO_COMM_DEFAULT

Mark all communicators with the default value.

Syntax

I_MPI_TUNING_AUTO_COMM_DEFAULT=*<arg>*

Argument

<arg>	**Binary indicator**
enable | yes | on | 1	Mark communicators.
disable | no | off | 0	Do not mark communicators. This is the default value.

Description

Set this environment variable to mark all communicators in an application with the default value. In this case, all communicators will have the identical default comm_id equal to -1.

I_MPI_TUNING_AUTO_COMM_USER

Enable communicator marking with a user value.

Syntax

```
I_MPI_TUNING_AUTO_COMM_USER=<arg>
```

Argument

\<arg>	**Binary indicator**
enable | yes | on | 1	Enable marking of communicators.
disable | no | off | 0	Disable marking of communicators. This is the default value.

Description

Set this environment variable to enable communicator marking with a user value. To mark a communicator in your application, use the MPI_Info object for this communicator that contains a record with the comm_id key. The key must belong the `0...UINT64_MAX range`.

I_MPI_TUNING_AUTO_ITER_POLICY

Control the iteration policy logic.

Syntax

```
_MPI_TUNING_AUTO_ITER_POLICY=<arg>
```

Argument

\<arg>	**Binary indicator**
enable | yes | on | 1	Reduce the number of iterations with a message size increase after 64Kb (by half). This is the default value.
disable | no | off | 0	Use the `I_MPI_TUNING_AUTO_ITER_NUM` value. `This value affects warmup iterations.`

Description

Set this environment variable to control the autotuning iteration policy logic.

I_MPI_TUNING_AUTO_ITER_POLICY_THRESHOLD

Control the message size limit for the `I_MPI_TUNING_AUTO_ITER_POLICY` environment variable.

Syntax

```
I_MPI_TUNING_AUTO_ITER_POLICY_THRESHOLD=<arg>
```

Argument

\<arg>	Define the value. By default, it is 64KB.

Description

Set this environment variable to control the message size limit for the autotuning iteration policy logic (`I_MPI_TUNING_AUTO_ITER_POLICY`).

I_MPI_TUNING_AUTO_POLICY

Choose the best algorithm identification strategy.

Syntax

```
I_MPI_TUNING_AUTO_POLICY=<arg>
```

Argument

\<arg>	**Description**
max	Choose the best algorithm based on a maximum time value. This is the default value.

min	Choose the best algorithm based on a minimum time value.
avg	Choose the best algorithm based on an average time value.

Description

Set this environment variable to control the autotuning strategy and choose the best algorithm based on the time value across ranks involved into the tuning process.

Main Thread Pinning

Use this feature to pin a particular MPI thread to a corresponding set of CPUs within a node and avoid undesired thread migration. This feature is available on operating systems that provide the necessary kernel interfaces.

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

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.

Logical Enumeration

0	4	1	5	2	6	3	7

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

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.

Default Settings

If you do not specify values for any main thread 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_RESPECT_CPUSET=on
- I_MPI_PIN_RESPECT_HCA=on
- I_MPI_PIN_CELL=unit

- `I_MPI_PIN_DOMAIN=auto:compact`
- `I_MPI_PIN_ORDER=bunch`

> **NOTE** If hyperthreading is on, the number or processes on the node is greater than the number of cores and no one process pinning environment variable is set. For better performance, the "spread" order will automatically be used instead of the default "compact" order.

Environment Variables for Main Thread Pinning

I_MPI_PIN

Turn on/off main thread pinning.

Syntax

`I_MPI_PIN=<arg>`

Arguments

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

Description

Set this environment variable to control the main thread 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 main threads within this subset.

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 Forms

`I_MPI_PIN_PROCESSOR_LIST=<value>`

The environment variable value has three syntax forms:

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

The following paragraphs provide detailed descriptions for each of these syntax forms.

> **NOTE** The `postoffset` keyword has `offset` alias.

> **NOTE** The second form of the pinning procedure has three steps:
>
> 1. Circular 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. Circular 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.

Syntax 1: *<proclist>*

`I_MPI_PIN_PROCESSOR_LIST=<proclist>`

Arguments

<proclist>	A comma-separated list of logical processor numbers and/or ranges of processors. The main thread with the i-th rank is pinned to the i-th processor in the list. The number should not exceed the number 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 2: **[*<procset>*][:[grain=*<grain>*][,shift=*<shift>*][,preoffset=**

`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 circular 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 circular 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 3: **[*<procset>*][:`map`=*<map>*]**

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

Arguments

<map>	The mapping pattern used for main thread placement.
`bunch`	The main threads are mapped proportionally to sockets and the threads are ordered as close as possible on the sockets.
`scatter`	The main threads are mapped as remotely as possible so as not to share common resources: FSB, caches, and core.
`spread`	The main threads 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 main thread is an index to the processor list such that the i-th main thread is pinned on i-th list member. This permits the definition of any main thread placement on the CPUs.

  For example, main thread 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 circular shift of a defined `grain` along the processor list with steps equal to `shift*grain` and a single shift on `offset*grain` at the end. This shifting action is repeated `shift` times.

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

Legend:

gray - MPI process grains

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

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

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

D) Final map table ordered by MPI ranks

A)

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

B)

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

C)

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

D)

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

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

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

In the `scatter` scenario the main threads 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

cy an - 1ˢᵗ socket processors

gre en - 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:

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

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

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

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

```
$ mpirun -genv I_MPI_DEBUG=4 -m -host host1 \
-env I_MPI_PIN_PROCESSOR_LIST=0,3 -n <number-of-processes> <executable> :\
-host host2 -env I_MPI_PIN_PROCESSOR_LIST=1,2,3 -n <number-of-processes> <executable>
```

> **NOTE** If the number of processes is greater than the number of CPUs used for pinning, the process list is wrapped around to the start of the processor list.

Examples

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

```
> mpiexec -genv I_MPI_PIN_PROCESSOR_LIST=0,3 -n <number-of-main-threads><executable>
```

To pin the main thread 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 <number-of-main-threads>
<executable> :^
-host host2 -env I_MPI_PIN_PROCESSOR_LIST=1,2,3 -n <number-of-main-threads> <executable>
```

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

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

> **NOTE** If the number of main threads is greater than the number of CPUs used for pinning, the thread list is wrapped around to the start of the processor list.

Interoperability with OpenMP* API

I_MPI_PIN_DOMAIN

Intel® MPI Library provides an additional environment variable to control main thread 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 1 Domain Example
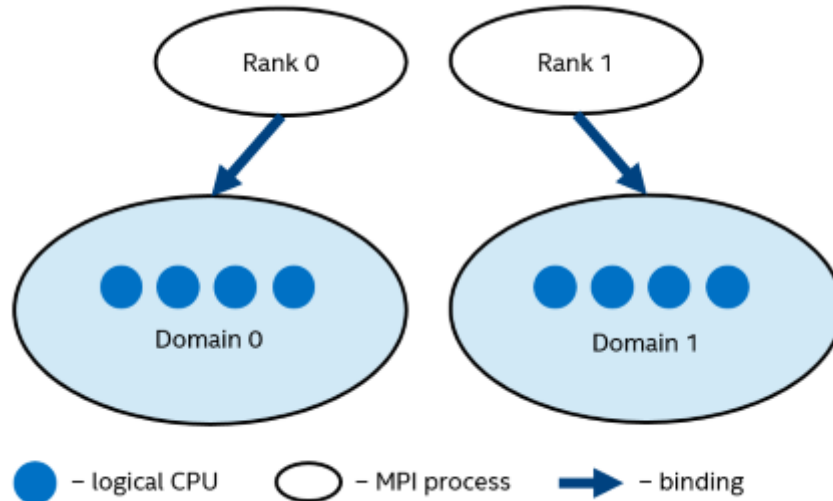
Each MPI process can create a number of children threads for running within the corresponding domain. The process threads can freely migrate from one logical processor to another within the particular domain.

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 main threads 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)
$[m_1, \ldots, m_n]$	For `<masklist>`, each m_i is a hexadecimail bit mask defining an individual domain. The following rule is used: the ith 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=[55,aa]`

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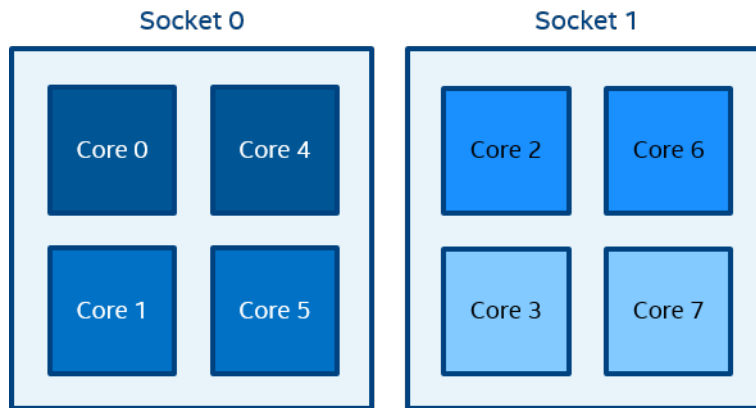
The figure above represents the SMP node model with a total of 8 cores on 2 sockets. Intel® Hyper-Threading Technology is disabled. Core pairs of the same color share the L2 cache.

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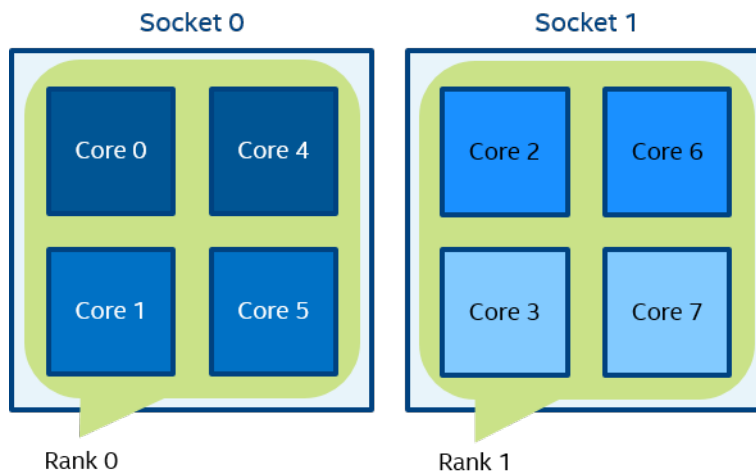

In Figure 3, two domains are defined according to the number of sockets. Process rank 0 can migrate on all cores on the 0-th socket. Process rank 1 can migrate on all cores on the first socket.

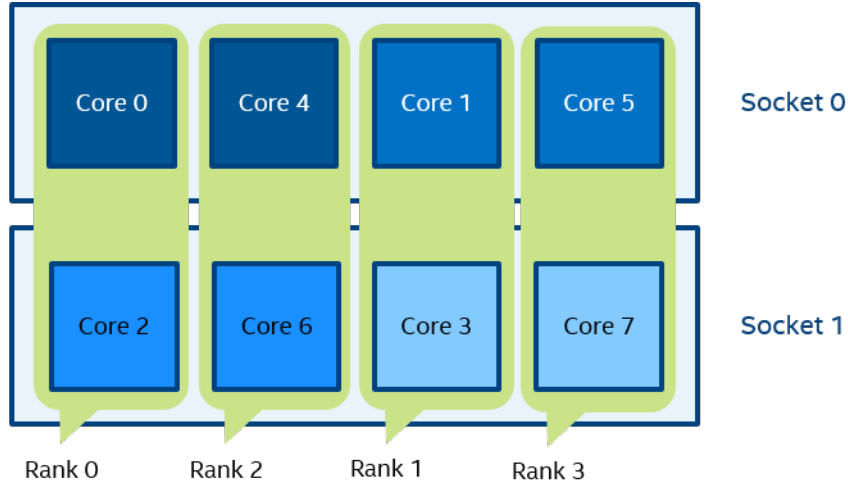*Figure 4* `mpiexec -n 4 -env I_MPI_PIN_DOMAIN cache2 test.exe`
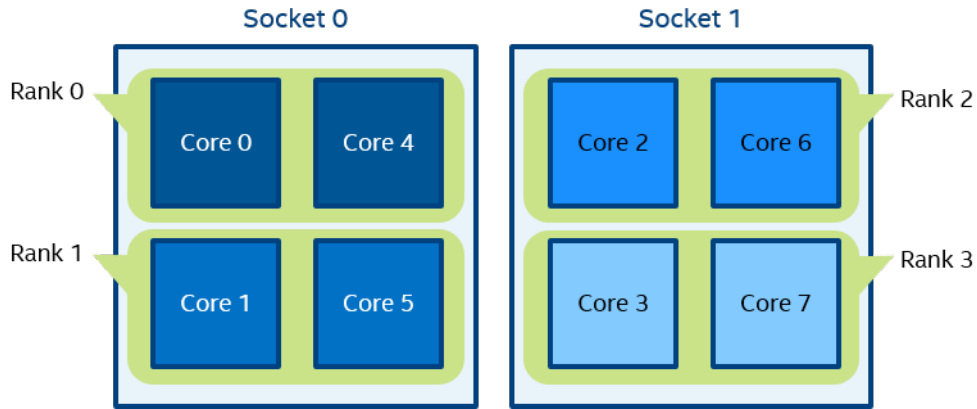
In Figure 4, four domains are defined according to the amount of common L2 caches. Process rank 0 runs on cores {0,4} that share an L2 cache. Process rank 1 runs on cores {1,5} that share an L2 cache as well, and so on.

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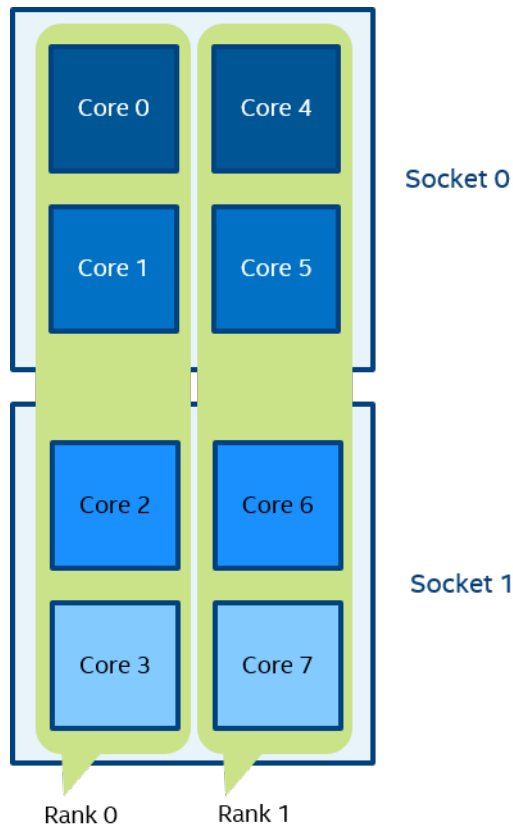

In Figure 5, two domains with size=4 are defined. The first domain contains cores {0,1,2,3}, and the second domain contains cores {4,5,6,7}. Domain members (cores) have consecutive numbering as defined by the `platform` option.

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

In Figure 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 7 `set OMP_NUM_THREADS=2 mpiexec -n 4 -env I_MPI_PIN_DOMAIN omp:platform test.exe`
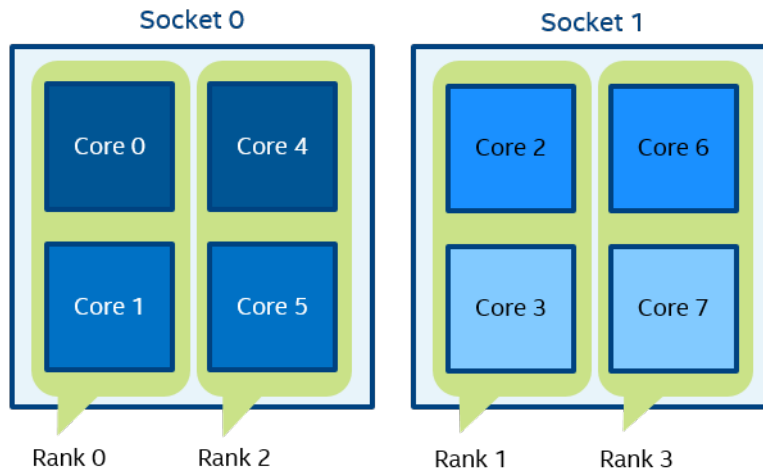
In Figure 7, domain size=2 (defined by `OMP_NUM_THREADS=2`), `platform` layout. Four domains {0,1}, {2,3}, {4,5}, {6,7} are defined. Domain members (cores) have consecutive numbering.

Figure 8 `mpiexec -n 2 -env I_MPI_PIN_DOMAIN [55,aa] test.exe`
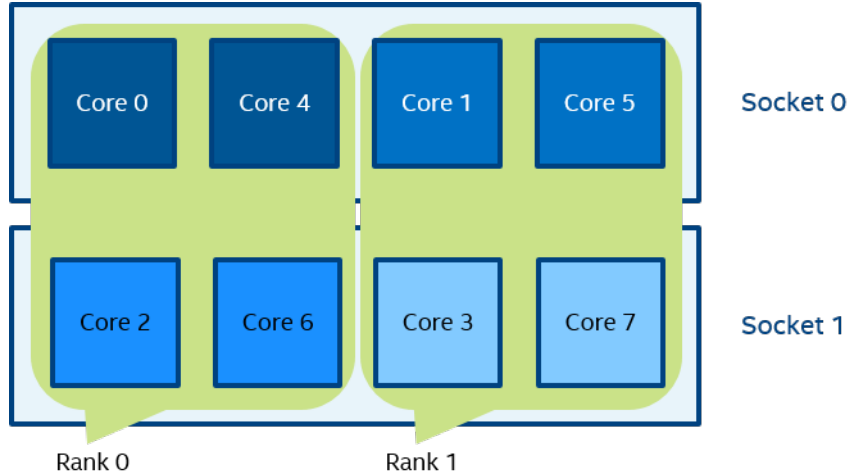
In Figure 8 (the example for `I_MPI_PIN_DOMAIN=<masklist>`), the first domain is defined by the 55 mask. It contains all cores with even numbers {0,2,4,6}. The second domain is defined by the AA mask. It contains all cores with odd numbers {1,3,5,7}.

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, whenever possible.
`compact`	The domains are ordered so that adjacent domains share common resources as much as possible.
`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. This is the default value.

Description

The optimal setting for this environment variable is application-specific. If adjacent MPI processes prefer to share common resources, such as cores, caches, sockets, FSB, use the `compact` 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
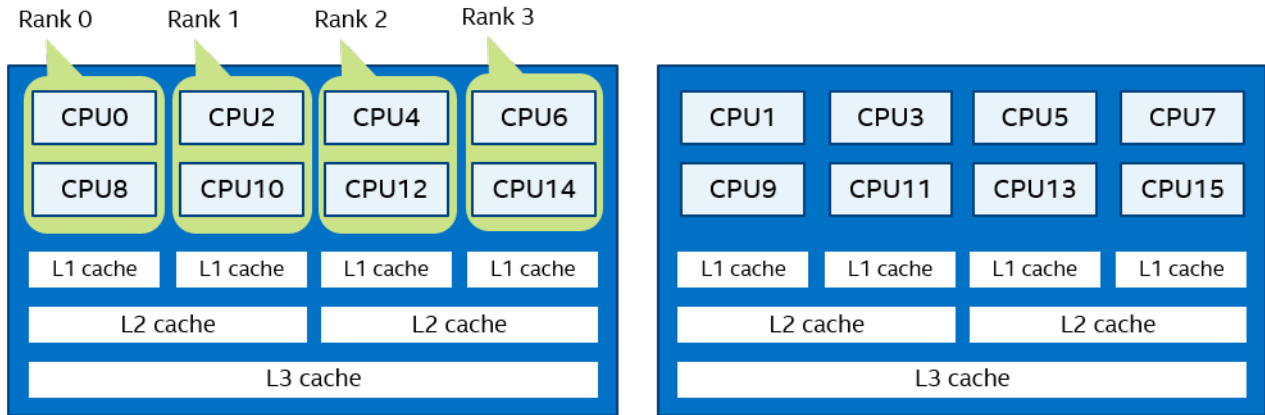
For the following configuration:

- 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:

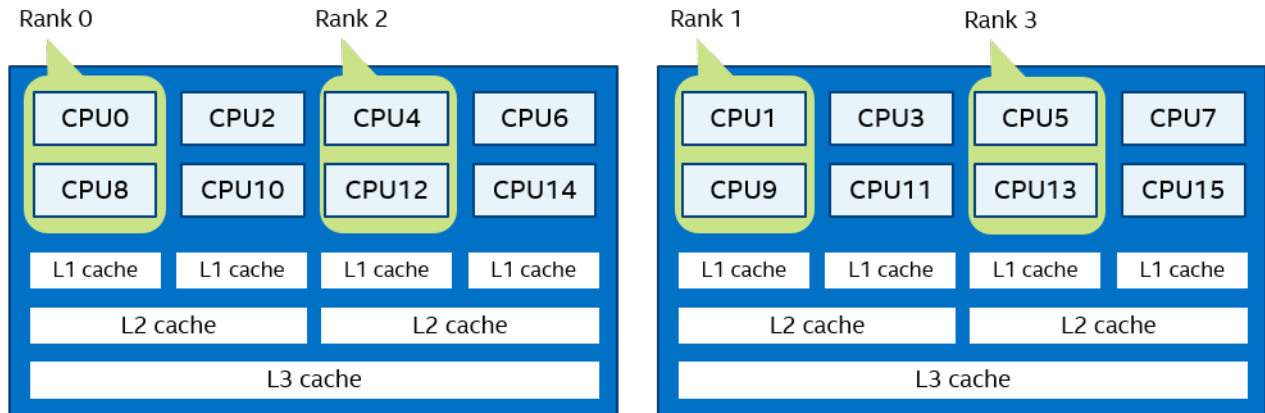`I_MPI_PIN_DOMAIN=2 I_MPI_PIN_ORDER=compact`

Figure 9 Compact Order Example

Scatter order:

`I_MPI_PIN_DOMAIN=2 I_MPI_PIN_ORDER=scatter`
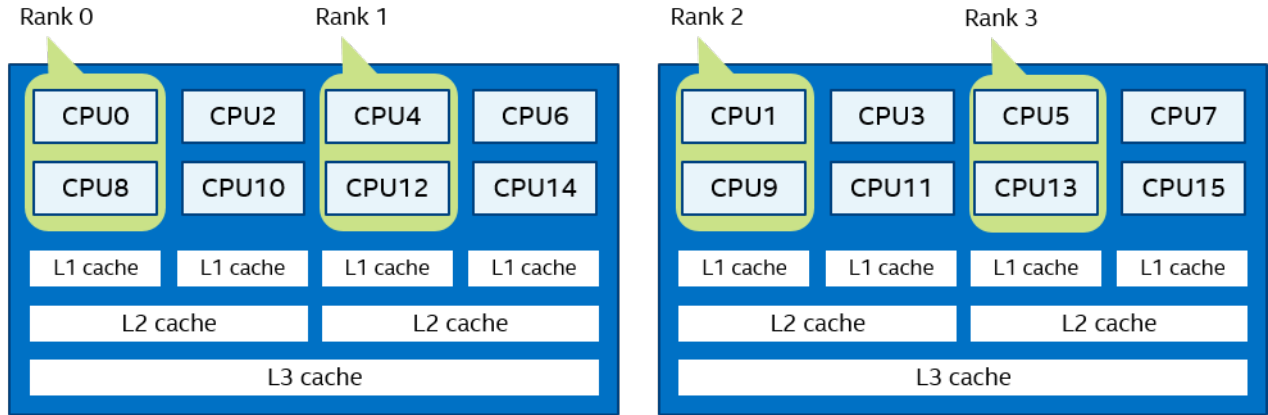
Figure 10 Scatter Order Example

Spread order:

`I_MPI_PIN_DOMAIN=2 I_MPI_PIN_ORDER=spread`

> **NOTE** For I_MPI_PIN_ORDER=spread, the order will be switched to 'compact' if there are not enough CPUs to emplace all domains.
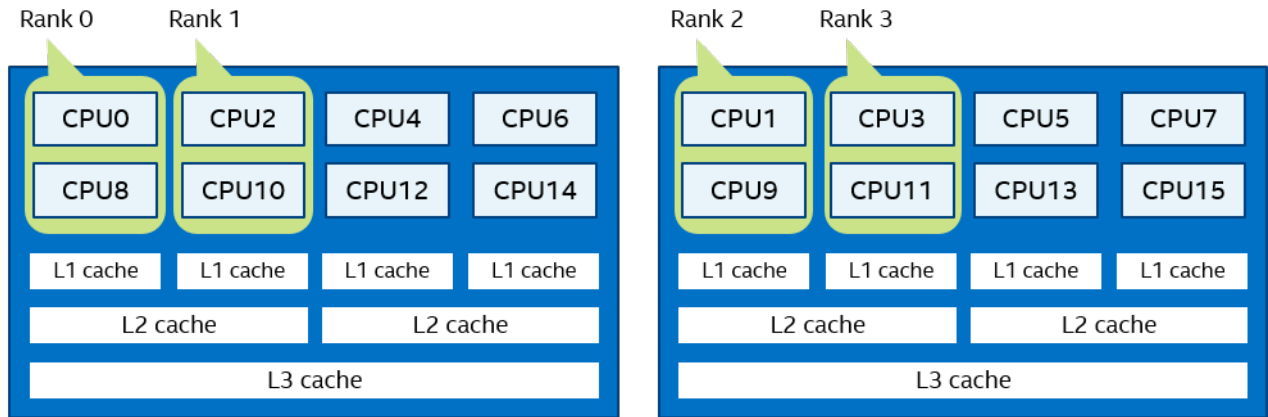
Figure 11 Spread Order Example

Bunch order:

I_MPI_PIN_DOMAIN=2 I_MPI_PIN_ORDER=bunch

Figure 12 Bunch Order Example

Environment Variables for Fabrics Control

This section provides description of the general environment variables for controlling fabrics, as well as description of variables for controlling specific fabrics:

- Communication Fabrics Control

Communication Fabrics Control

I_MPI_FABRICS

Select the particular fabrics to be used.

Syntax

I_MPI_FABRICS=ofi | shm

Arguments

<fabric>	Define a network fabric.
shm	Shared memory transport (used for intra-node communication only).
ofi	OpenFabrics Interfaces* (OFI)-capable network fabrics, such as Intel® Omni-Path Architecture, InfiniBand*, and Ethernet (through OFI API).

Description

Set this environment variable to select a specific fabric combination.

NOTE

This option is not applicable to `slurm` and `pdsh` bootstrap servers.

Shared Memory Control

I_MPI_SHM

Select a shared memory transport to be used.

Syntax

```
I_MPI_SHM=<transport>
```

Arguments

`<transport>`	Define a shared memory transport solution.
`disable	no	off	0`	Do not use shared memory transport.
`auto`	Select a shared memory transport solution automatically.
`bdw_sse`	The shared memory transport solution tuned for Intel® microarchitecture code name Broadwell. The SSE4.2. instruction set is used.
`bdw_avx2`	The shared memory transport solution tuned for Intel® microarchitecture code name Broadwell. The AVX2 instruction set is used.
`skx_sse`	The shared memory transport solution tuned for Intel® Xeon® processors based on Intel® microarchitecture code name Skylake. The CLFLUSHOPT and SSE4.2 instruction set is used.
`skx_avx2`	The shared memory transport solution tuned for Intel® Xeon® processors based on Intel® microarchitecture code name Skylake. The CLFLUSHOPT and AVX2 instruction set is used.
`skx_avx512`	The shared memory transport solution tuned for Intel® Xeon® processors based on Intel® microarchitecture code name Skylake. The CLFLUSHOPT and AVX512 instruction set is used.
`clx_sse`	The shared memory transport solution tuned for Intel® Xeon® processors based on Intel® microarchitecture code name Cascade Lake. The CLFLUSHOPT and SSE4.2 instruction set is used.
`clx_avx2`	The shared memory transport solution tuned for Intel® Xeon® processors based on Intel® microarchitecture code name Cascade Lake. The CLFLUSHOPT and AVX2 instruction set is used.
`clx_avx512`	The shared memory transport solution tuned for Intel® Xeon® processors based on Intel® microarchitecture code name Cascade Lake. The CLFLUSHOPT and AVX512 instruction set is used.
`clx-ap`	The shared memory transport solution tuned for Intel® Xeon® processors based on Intel® microarchitecture code name Cascade Lake Advanced Performance.
`icx`	The shared memory transport solution tuned for Intel® Xeon® processors based on Intel® microarchitecture code name Ice Lake.

Description

Set this environment variable to select a specific shared memory transport solution.

Automatically selected transports:

- `icx` for Intel® Xeon® processors based on Intel® microarchitecture code name Ice Lake
- `clx-ap` for Intel® Xeon® processors based on Intel® microarchitecture code name Cascade Lake Advanced Performance
- `bdw_avx2` for Intel® microarchitecture code name Haswell, Broadwell and Skylake
- `skx_avx2` for Intel® Xeon® processors based on Intel® microarchitecture code name Skylake
- `ckx_avx2` for Intel® Xeon® processors based on Intel® microarchitecture code name Cascade Lake
- `knl_mcdram` for Intel® microarchitecture code name Knights Landing and Knights Mill
- `bdw_sse` for all other platforms

The value of `I_MPI_SHM` depends on the value of `I_MPI_FABRICS` as follows: if `I_MPI_FABRICS` is `ofi`, `I_MPI_SHM` is disabled. If `I_MPI_FABRICS` is `shm:ofi`, `I_MPI_SHM` defaults to `auto` or takes the specified value.

I_MPI_SHM_CELL_FWD_SIZE

Change the size of a shared memory forward cell.

Syntax

`I_MPI_SHM_CELL_FWD_SIZE=<nbytes>`

Arguments

`<nbytes>`	The size of a shared memory forward cell in bytes
`> 0`	The default `<nbytes>` value depends on the transport used and should normally range from 64K to 1024K.

Description

Forward cells are in-cache message buffer cells used for sending small amounts of data. Lower values are recommended. Set this environment variable to define the size of a forward cell in the shared memory transport.

I_MPI_SHM_CELL_BWD_SIZE

Change the size of a shared memory backward cell.

Syntax

`I_MPI_SHM_CELL_BWD_SIZE=<nbytes>`

Arguments

`<nbytes>`	The size of a shared memory backward cell in bytes
`> 0`	The default `<nbytes>` value depends on the transport used and should normally range from 64K to 1024K.

Description

Backward cells are out-of-cache message buffer cells used for sending large amounts of data. Higher values are recommended. Set this environment variable to define the size of a backwrad cell in the shared memory transport.

I_MPI_SHM_CELL_EXT_SIZE

Change the size of a shared memory extended cell.

Syntax

`I_MPI_SHM_CELL_EXT_SIZE=<nbytes>`

Arguments

`<nbytes>`	The size of a shared memory extended cell in bytes

> 0	The default `<nbytes>` value depends on the transport used and should normally range from 64K to 1024K.

Description

Extended cells are used in the imbalanced applications when forward and backward cells are run out. An extended cell does not have a specific owner - it is shared between all ranks on the computing node. Set this environment variable to define the size of an extended cell in the shared memory transport.

I_MPI_SHM_CELL_FWD_NUM

Change the number of forward cells in the shared memory transport (per rank).

Syntax

`I_MPI_SHM_CELL_FWD_NUM=<num>`

Arguments

`<num>`	The number of shared memory forward cells
> 0	The default value depends on the transport used and should normally range from 4 to 16.

Description

Set this environment variable to define the number of forward cells in the shared memory transport.

I_MPI_SHM_CELL_BWD_NUM

Change the number of backward cells in the shared memory transport (per rank).

Syntax

`I_MPI_SHM_CELL_BWD_NUM=<num>`

Arguments

`<num>`	The number of shared memory backward cells
> 0	The default value depends on the transport used and should normally range from 4 to 64.

Description

Set this environment variable to define the number of backward cells in the shared memory transport.

I_MPI_SHM_CELL_EXT_NUM_TOTAL

Change the total number of extended cells in the shared memory transport.

Syntax

`I_MPI_SHM_CELL_EXT_NUM_TOTAL=<num>`

Arguments

`<num>`	The number of shared memory backward cells
> 0	The default value depends on the transport used and should normally range from 2K to 8K.

Description

Set this environment variable to define the number of extended cells in the shared memory transport.

NOTE

This is not "per rank" number, it is total number of extended cells on the computing node.

I_MPI_SHM_CELL_FWD_HOLD_NUM

Change the number of hold forward cells in the shared memory transport (per rank).

Syntax

`I_MPI_SHM_CELL_FWD_HOLD_NUM=<num>`

Arguments

`<num>`	The number of shared memory hold forward cells
`> 0`	The default value depends on the transport used and must be less than `I_MPI_SHM_CELL_FWD_NUM`.

Description

Set this environment variable to define the number of forward cells in the shared memory transport a rank can hold at the same time. Recommended values are powers of two in the range between 1 and 8.

I_MPI_SHM_MCDRAM_LIMIT

Change the size of the shared memory bound to the multi-channel DRAM (MCDRAM) (size per rank).

Syntax

`I_MPI_SHM_MCDRAM_LIMIT=<nbytes>`

Arguments

`<nbytes>`	The size of the shared memory bound to MCDRAM per rank
1048576	This is the default value.

Description

Set this environment variable to define how much MCDRAM memory per rank is allowed for the shared memory transport. This variable takes effect with `I_MPI_SHM=knl_mcdram` only.

I_MPI_SHM_SEND_SPIN_COUNT

Control the spin count value for the shared memory transport for sending messages.

Syntax

`I_MPI_SHM_SEND_SPIN_COUNT=<count>`

Arguments

`<count>`	Define the spin count value. A typical value range is between 1 and 1000.

Description

If the recipient ingress buffer is full, the sender may be blocked until this spin count value is reached. It has no effect when sending small messages.

I_MPI_SHM_RECV_SPIN_COUNT

Control the spin count value for the shared memory transport for receiving messages.

Syntax

`I_MPI_SHM_RECV_SPIN_COUNT=<count>`

Arguments

<count>	Define the spin count value. A typical value range is between 1 and 1000000.

Description

If the receive is non-blocking, this spin count is used only for safe reorder of expected and unexpected messages. It has no effect on receiving small messages.

OFI*-capable Network Fabrics Control

I_MPI_OFI_DRECV

Control the capability of the direct receive in the OFI fabric.

Syntax

I_MPI_OFI_DRECV=*<arg>*

Arguments

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

Description

Use the direct receive capability to block MPI_Recv calls only. Before using the direct receive capability, ensure that you use it for single-threaded MPI applications and check if you have selected OFI as the network fabric by setting I_MPI_FABRICS=ofi.

I_MPI_OFI_MATCH_COMPLETE

Control the capability of the match complete in the OFI fabric for all providers if applicable.

Syntax

I_MPI_OFI_MATCH_COMPLETE=*<arg>*

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Enable match complete if applicable. This is the default value
disable | no | off | 0	Disable match complete

I_MPI_OFI_LIBRARY_INTERNAL

Control the usage of libfabric* shipped with the Intel® MPI Library.

Syntax

I_MPI_OFI_LIBRARY_INTERNAL=*<arg>*

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Use libfabric from the Intel MPI Library
disable | no | off | 0	Do not use libfabric from the Intel MPI Library

Description

Set this environment variable to disable or enable usage of libfabric from the Intel MPI Library. The variable must be set before sourcing the vars.bat script.

Example

```
> set I_MPI_OFI_LIBRARY_INTERNAL=1
> call <installdir> \env\vars.bat
```

Setting this variable is equivalent to passing the `-ofi_internal` option to the `vars.bat` script.

For more information, refer to the Intel® MPI Library Developer Guide, section Libfabric* Support.

I_MPI_OFI_TAG_DYNAMIC

Enable dynamic tag partitioning.

Syntax

I_MPI_OFI_TAG_DYNAMIC=*<arg>*

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Enable automatic OFI tag partitioning
disable | no | off | 0	Use static OFI tag layout. This is the default value

Description

Set this environment variable to enable dynamic OFI Netmod tag partitioning based on the run configuration. You can use it to get larger MPI tag space or to improve scalability in large-scale runs.

Environment Variables for Memory Policy Control

Intel® MPI Library supports non-uniform memory access (NUMA) nodes with high-bandwidth (HBW) memory (MCDRAM) on Intel® Xeon Phi™ processors (codenamed Knights Landing). Intel® MPI Library can attach memory of MPI processes to the memory of specific NUMA nodes. This section describes the environment variables for such memory placement control.

I_MPI_HBW_POLICY

Set the policy for MPI process memory placement for using HBW memory.

Syntax

I_MPI_HBW_POLICY=<user memory policy>[,<mpi memory policy>][,<win_allocate policy>]

In the syntax:

- `<user memory policy>` - memory policy used to allocate the memory for user applications (required)
- `<mpi memory policy>` - memory policy used to allocate the internal MPI memory (optional)
- `<win_allocate policy>` - memory policy used to allocate memory for window segments for RMA operations (optional)

Each of the listed policies may have the values below:

Arguments

<value>	The memory allocation policy used.
hbw_preferred	Allocate the local HBW memory for each process. If the HBW memory is not available, allocate the local dynamic random access memory.
hbw_bind	Allocate only the local HBW memory for each process.
hbw_interleave	Allocate the HBW memory and dynamic random access memory on the local node in the round-robin manner.

Description

Use this environment variable to specify the policy for MPI process memory placement on a machine with HBW memory.

By default, Intel MPI Library allocates memory for a process in local DDR. The use of HBW memory becomes available only when you specify the `I_MPI_HBW_POLICY` variable.

Examples

The following examples demonstrate different configurations of memory placement:

* `I_MPI_HBW_POLICY=hbw_bind,hbw_preferred,hbw_bind`

    Only use the local HBW memory allocated in user applications and window segments for RMA operations. Use the local HBW memory internally allocated in Intel® MPI Library first. If the HBW memory is not available, use the local DDR internally allocated in Intel MPI Library.
* `I_MPI_HBW_POLICY=hbw_bind,,hbw_bind`

    Only use the local HBW memory allocated in user applications and window segments for RMA operations. Use the local DDR internally allocated in Intel MPI Library.
* `I_MPI_HBW_POLICY=hbw_bind,hbw_preferred`

    Only use the local HBW memory allocated in user applications. Use the local HBW memory internally allocated in Intel MPI Library first. If the HBW memory is not available, use the local DDR internally allocated in Intel MPI Library. Use the local DDR allocated in window segments for RMA operations.

I_MPI_BIND_NUMA

Set the NUMA nodes for memory allocation.

Syntax

`I_MPI_BIND_NUMA=<value>`

Arguments

<value>	Specify the NUMA nodes for memory allocation.
`localalloc`	Allocate memory on the local node. This is the default value.
`Node_1,…,Node_k`	Allocate memory according to `I_MPI_BIND_ORDER` on the specified NUMA nodes.

Description

Set this environment variable to specify the NUMA node set that is involved in the memory allocation procedure.

I_MPI_BIND_ORDER

Set this environment variable to define the memory allocation manner.

Syntax

`I_MPI_BIND_ORDER=<value>`

Arguments

<value>	Specify the allocation manner.
`compact`	Allocate memory for processes as close as possible (in terms of NUMA nodes), among the NUMA nodes specified in `I_MPI_BIND_NUMA`. This is the default value.
`scatter`	Allocate memory among the NUMA nodes specified in `I_MPI_BIND_NUMA` using the round-robin manner.

Description

Set this environment variable to define the memory allocation manner among the NUMA nodes specified in `I_MPI_BIND_NUMA`. The variable has no effect without `I_MPI_BIND_NUMA` set.

I_MPI_BIND_WIN_ALLOCATE

Set this environment variable to control memory allocation for window segments.

Syntax

`I_MPI_BIND_WIN_ALLOCATE=<value>`

Arguments

`<value>`	Specify the memory allocation behavior for window segments.
`localalloc`	Allocate memory on the local node. This is the default value.
`hbw_preferred`	Allocate the local HBW memory for each process. If the HBW memory is not available, allocate the local dynamic random access memory.
`hbw_bind`	Allocate only the local HBW memory for each process.
`hbw_interleave`	Allocate the HBW memory and dynamic random access memory on a local node in the round-robin manner.
`<NUMA node id>`	Allocate memory on the given NUMA node.

Description

Set this environment variable to create window segments allocated in HBW memory with the help of the `MPI_Win_allocate_shared` or `MPI_Win_allocate` functions.

MPI_Info

You can control memory allocation for window segments with the help of an `MPI_Info` object, which is passed as a parameter to the `MPI_Win_allocate` or `MPI_Win_allocate_shared` function. In an application, if you specify such an object with the `numa_bind_policy` key, window segments are allocated in accordance with the value for `numa_bind_policy`. Possible values are the same as for `I_MPI_BIND_WIN_ALLOCATE`.

A code fragment demonstrating the use of `MPI_Info`:

```
MPI_Info info;
...
MPI_Info_create( &info );
MPI_Info_set( info, "numa_bind_policy", "hbw_preferred" );
...
MPI_Win_allocate_shared( size, disp_unit, info, comm, &baseptr, &win );
```

> **NOTE**
> When you specify the memory placement policy for window segments, Intel MPI Library recognizes the configurations according to the following priority:

1.   Setting of `MPI_Info`.
2.   Setting of `I_MPI_HBW_POLICY`, if you specified `<win_allocate policy>`.
3.   Setting of `I_MPI_BIND_WIN_ALLOCATE`.

Other Environment Variables

I_MPI_DEBUG

Print out debugging information when an MPI program starts running.

Syntax

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

Arguments

Argument	Description
<level>	Indicate the level of debug information provided.
0	Output no debugging information. This is the default value.
1	Output libfabric* version and provider.
2	Output information about the tuning file used.
3	Output effective MPI rank, pid and node mapping table.
4	Output process pinning information.
5	Output environment variables specific to the Intel® MPI Library.
> 5	Add extra levels of debug information.

Argument	Description
<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.
nousrwarn	Suppress warnings for improper use case (for example, incompatible combination of controls).
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,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

Argument	Description
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_DEBUG_COREDUMP

Controls core dump files generation in case of failure during MPI application execution.

Syntax

`I_MPI_DEBUG_COREDUMP=<arg>`

Arguments

Argument	Description
enable|yes|on|1	Enable coredump files generation.
disable|no|off|0	Do not generate coredump files. Default value.

Description

Set this environment variable to enable coredump files dumping in case of termination caused by segmentation fault. Available for both release and debug builds.

I_MPI_PMI_VALUE_LENGTH_MAX

Control the length of the value buffer in PMI on the client side.

Syntax

`I_MPI_PMI_VALUE_LENGTH_MAX=<length>`

Arguments

Argument	Description
`<length>`	Define the value of the buffer length in bytes.
`<n> > 0`	The default value is -1, which means do not override the value received from the `PMI_KVS_Get_value_length_max()` function.

Description

Set this environment variable to control the length of the value buffer in PMI on the client side. The length of the buffer will be the lesser of `I_MPI_PMI_VALUE_LENGTH_MAX` and `PMI_KVS_Get_value_length_max()`.

I_MPI_REMOVED_VAR_WARNING

Print out a warning if a removed environment variable is set.

Syntax

`I_MPI_REMOVED_VAR_WARNING=<arg>`

Arguments

Argument	Description
`enable | yes | on | 1`	Print out the warning. This is the default value
`disable | no | off | 0`	Do not print the warning

Description

Use this environment variable to print out a warning if a removed environment variable is set. Warnings are printed regardless of whether `I_MPI_DEBUG is set`.

I_MPI_VAR_CHECK_SPELLING

Print out a warning if an unknown environment variable is set.

Syntax

`I_MPI_VAR_CHECK_SPELLING=<arg>`

Arguments

Argument	Description
`enable | yes | on | 1`	Print out the warning. This is the default value
`disable | no | off | 0`	Do not print the warning

Description

Use this environment variable to print out a warning if an unsupported environment variable is set. Warnings are printed in case of removed or misprinted environment variables.

I_MPI_LIBRARY_KIND

Specify the Intel® MPI Library configuration.

Syntax

`I_MPI_LIBRARY_KIND=<value>`

Arguments

Value	Description
`release`	Multi-threaded optimized library. This is the default value
`debug`	Multi-threaded debug library

Description

Use this variable to set an argument for the `vars. bat`script. This script establishes the Intel® MPI Library environment and enables you to specify the appropriate library configuration. To ensure that the desired configuration is set, check the `LD_LIBRARY_PATH` variable.

Example

```
> export I_MPI_LIBRARY_KIND=debug
```

Setting this variable is equivalent to passing an argument directly to the `vars.[c]sh` script:

Example

```
> <installdir> \env\vars.bat release
```

I_MPI_PLATFORM

Select the intended optimization platform.

Syntax

`I_MPI_PLATFORM=<platform>`

Arguments

Argument	Description
`<platform>`	Intended optimization platform (string value)
`auto`	Use only with heterogeneous runs to determine the appropriate platform across all nodes. May slow down MPI initialization time due to collective operation across all nodes.
`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.
`skx`	Optimize for the Intel Xeon Processors E3 V5 and Intel Xeon Scalable Family series, and other Intel Architecture processors formerly code named Skylake.
`clx`	Optimize for the 2nd Generation Intel Xeon Scalable Processors, and other Intel® Architecture processors formerly code named Cascade Lake.
`clx-ap`	Optimize for the 2nd Generation Intel Xeon Scalable Processors, and other Intel Architecture processors formerly code named Cascade Lake AP Note: The explicit `clx-ap` setting is ignored if the actual platform is not Intel.

Description

Set this environment variable to use the predefined platform settings. The default value is a local platform for each node.

The variable 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_MALLOC

Control the Intel® MPI Library custom allocator of private memory.

Syntax

`I_MPI_MALLOC=<arg>`

Argument

Argument	Description
1	Enable the Intel MPI Library custom allocator of private memory.
	Use the Intel MPI custom allocator of private memory for `MPI_Alloc_mem/MPI_Free_mem`.
0	Disable the Intel MPI Library custom allocator of private memory.
	Use the system-provided memory allocator for `MPI_Alloc_mem/MPI_Free_mem`.

Description

Use this environment variable to enable or disable the Intel MPI Library custom allocator of private memory for `MPI_Alloc_mem/MPI_Free_mem`.

By default, `I_MPI_MALLOC` is enabled if `I_MPI_ASYNC_PROGRESS` and `I_MPI_THREAD_SPLIT` are disabled.

NOTE If the platform is not supported by the Intel MPI Library custom allocator of private memory, a system-provided memory allocator is used and the `I_MPI_MALLOC` variable is ignored.

I_MPI_WAIT_MODE

Control the Intel® MPI Library optimization for oversubscription mode.

Syntax

`I_MPI_WAIT_MODE=<arg>`

Arguments

Argument	Description
0	Optimize MPI application to work in the normal mode (1 rank on 1 CPU). This is the default value if the number of processes on a computation node is less than or equal to the number of CPUs on the node.
1	Optimize MPI application to work in the oversubscription mode (multiple ranks on 1 CPU). This is the default value if the number of processes on a computation node is greater than the number of CPUs on the node.

Description

It is recommended to use this variable in the oversubscription mode.

I_MPI_THREAD_YIELD

Control the Intel® MPI Library thread yield customization during MPI busy wait time.

Syntax

I_MPI_THREAD_YIELD=*<arg>*

Arguments

Argument	Description
0	Do nothing for thread yield during the busy wait (spin wait). This is the default value when I_MPI_WAIT_MODE=0
1	Do the pause processor instruction for I_MPI_PAUSE_COUNT during the busy wait.
2	Do the SwitchToThread() system call for thread yield during the busy wait. This is the default value when I_MPI_WAIT_MODE=1
3	Do the Sleep() system call for I_MPI_THREAD_SLEEP number of milliseconds for thread yield during the busy wait.

Description

I_MPI_THREAD_YIELD=0 or I_MPI_THREAD_YIELD=1 in the normal mode and I_MPI_THREAD_YIELD=2 or I_MPI_THREAD_YIELD=3 in the oversubscription mode.

I_MPI_PAUSE_COUNT

Control the Intel® MPI Library pause count for the thread yield customization during MPI busy wait time.

Syntax

I_MPI_PAUSE_COUNT=*<arg>*

Argument

Argument	Description
>=0	Pause count for thread yield customization during MPI busy wait time. The default value is 0. Normally, the value is less than 100.

Description

This variable is applicable when I_MPI_THREAD_YIELD=1. Small values of I_MPI_PAUSE_COUNT may increase performance, while larger values may reduce energy consumption.

I_MPI_SPIN_COUNT

Control the spin count value.

Syntax

I_MPI_SPIN_COUNT=*<scount>*

Argument

Argument	Description
<scount> >=0	Define the loop spin count when polling fabric(s). The default <scount> value is equal to 1 when more than one process runs per processor/core. Otherwise the value equals 2000. 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. Smaller values for *<scount>* cause the Intel® MPI Library to release the processor more frequently.

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

I_MPI_THREAD_SLEEP

Control the Intel® MPI Library thread sleep milliseconds timeout for thread yield customization while MPI busy wait progress.

Syntax

`I_MPI_THREAD_SLEEP=<arg>`

Argument

Argument	Description
>=0	Thread sleep microseconds timeout. The default value is 0. Normally, the value is less than 100.

Description

This variable is applicable when `I_MPI_THREAD_YIELD=3`. Small values of `I_MPI_PAUSE_COUNT` may increase performance in the normal mode, while larger values may increase performance in the oversubscription mode

I_MPI_EXTRA_FILESYSTEM

Control native support for parallel file systems.

Syntax

`I_MPI_EXTRA_FILESYSTEM=<arg>`

Argument

Argument	Description
`enable	yes	on	1`	Enable native support for parallel file systems.
`disable	no	off	0`	Disable native support for parallel file systems. This is the default value.

Description

Use this environment variable to enable or disable native support for parallel file systems. This environment variable is deprecated.

I_MPI_EXTRA_FILESYSTEM_FORCE

Syntax

`I_MPI_EXTRA_FILESYSTEM_FORCE=<ufs|nfs|gpfs|panfs|lustre>`

Description

Force filesystem recognition logic. Setting this variable is equivalent to prefixing all paths in MPI-IO calls with the selected filesystem plus colon. This environment variable is deprecated.

I_MPI_FILESYSTEM

Turn on/off native parallel file systems support. If set, I_MPI_EXTRA_FILESYSTEM is ignored.

Syntax

I_MPI_FILESYSTEM=<*arg*>

Argument

Argument	Description
disable | no | off | 0	Disable native support for parallel file. This is the default value.
enable | yes | on | 1	Enable native support for parallel file.

I_MPI_FILESYSTEM_FORCE

Force Intel MPI to use a specific driver for a file system. If set, I_MPI_EXTRA_FILESYSTEM_FORCE is ignored.

Syntax

I_MPI_FILESYSTEM_FORCE=<*ufs|nfs|gpfs|panfs|lustre|daos*>

I_MPI_FILESYSTEM_CB_NODES

Explicitly set the MPI-IO hint cb_nodes for all MPI-IO file handles, overriding user info set at runtime. Non-positive values are ignored.

Syntax

I_MPI_FILESYSTEM_CB_NODES=<*arg*>

Argument

Argument	Description
Any positive integer	Maximum number of collective I/O aggregators for all collective I/O operations.
Non-positive integer	Ignored. The default value is -1.

I_MPI_FILESYSTEM_CB_CONFIG_LIST

Explicitly set the MPI-IO hint cb_config_list for all MPI-IO file handles, which overrides user information set at runtime.

Syntax

I_MPI_FILESYSTEM_CB_CONFIG_LIST=<*arg*>

Argument

Argument	Description
"*:<proc>"	Place <*proc*> number of I/O aggregators per node. <*proc*> should be a positive integer.
""	Ignored. This is the default value.

I_MPI_FILESYSTEM_NFS_DIRECT

Enable NFS bypassing cache to achieve sequential consistency among all accesses using a single file handle. If set, I_MPI_FILESYSTEM_NFS_DIRECT is ignored.

Syntax

```
I_MPI_FILESYSTEM_NFS_DIRECT=<arg>
```

Argument

Argument	Description
disable | no | off | 0	Disable native support for parallel file systems.
enable | yes | on | 1	Enable native support for parallel file systems. This is the default value.

I_MPI_FILESYSTEM_GPFS_DIRECT

Enable GPFS bypassing cache to achieve sequential consistency among all accesses using a single file handle.

Syntax

```
I_MPI_FILESYSTEM_GPFS_DIRECT=<arg>
```

Argument

Argument	Description
disable | no | off | 0	Disable native support for GPFS sequential consistency. This is the default value.
enable | yes | on | 1	Enable native support for GPFS sequential consistency.

I_MPI_MULTIRAIL

Syntax

```
I_MPI_MULTIRAIL=<arg>
```

Argument

Argument	Description
1	Enable multi-rail capability.
0	Disable multi-rail capability. This is the default value.

Description

Set this variable to enable multi-rail capability and identify NICs serviced by the provider. Pick this variable on the same NUMA.

I_MPI_SPAWN

Syntax

```
I_MPI_SPAWN=<arg>
```

Argument

Argument	Description
enable | yes | on | 1	Enable support of dynamic processes.
disable | no | off | 0	Disable support of dynamic processes. This is the default value.

Description

Use this environment variable to enable or disable dynamic processes and MPI-port support.

When dynamic processes infrastructure conflicts with optimization or require extra communication during bootstrap, this feature is disabled by default. This control is mandatory for applications that use dynamic processes.

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

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

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

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

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

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.