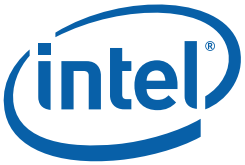


Intel® MPI Library for Windows* Reference Manual

Contents

Disclaimer and Legal Notices

Revision History

Document Number	Revision Number	Description	Revision Date
315399-001	3.1 Beta	Initial release	/07/10/2007
315399-002	3.1	New variables were added. New section “Processor Information Utility” was added. Updated and reviewed for style	/10/02/2007

1 About this Document

This *Reference Manual* provides you with a complete command and tuning reference for the Intel MPI Library.

The Intel® MPI Library enables you to deliver maximum end user performance as soon as new processor and interconnect technology become available. The Intel® MPI Library is a multi-fabric message passing library that implements the Message Passing Interface, v2 (MPI-2) specification. It provides a standard library across Intel® platforms that:

- Delivers best in class performance for enterprise, divisional, departmental and workgroup high performance computing. Intel® MPI Library focuses on making applications perform better on IA based clusters.
- Enables to adopt MPI -2 functions as their needs dictate.

The Intel® MPI Library enables you to change or upgrade interconnects as the technology becomes available without major changes to the software or to the operating environment.

The library is included in the following kits:

- *Intel® MPI Library Runtime Environment* has the tools you need to run programs including SMPD daemons and supporting utilities, dynamic (.dll) libraries and documentation.
- *Intel® MPI Library Development Kit* includes all of the Runtime Environment components plus compilation tools including compiler commands such as `mpicc`, include files and modules, debug libraries, trace libraries, and test codes.

1.1 Intended Audience

This *Reference Manual* helps an experienced user understand full functionality of the Intel® MPI Library and get the best possible application performance.

1.2 Using Doc Type Field

This *Reference Manual* contains the following sections

Table 1-1 Document Organization

Section	Description
Section 1 About this Document	Section 1 introduces this document
Section 2 Command Reference	Section 2 describes options and variables for compiler commands, job startup commands and MPD daemon commands as well
Section 3 Tuning Reference	Section 3 describes environment variables used to influence program behavior and performance at run time
Section 4 Statistics Gathering Mode	Section 4 describes statistics about usage of MPI communication operations
Section 5 Unified Memory Management	Section 5 describes unified Intel memory management subsystem (`i_malloc`)

Section 6 GUI utilities	Section 6 describes graphical user interface (GUI) utilities distributed with the Intel® MPI Library

1.3 Conventions and Symbols

The following conventions are used in this document.

Table 1-2 Conventions and Symbols used in this Document

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

1.4 Related Information

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

Product Web Site

Intel® MPI Library support

Intel® Cluster Tools Products

Intel® Software Development Products

2 Command Reference

2.1 Compiler Commands

The following table lists available MPI compiler commands and the underlying compilers, compiler families, languages, and application binary interfaces (ABIs) that they support.

Table 2-1 Intel® MPI Library compiler drivers

Compiler command	Underlying compiler	Supported language(s)	Supported ABI(s)
Common Compilers			
mpicc.bat	cl.exe	C	32/64 bit
mpicxx.bat	cl.exe	C++	32/64 bit
mpifc.bat	ifort.exe	Fortran 77/Fortran 95	32/64 bit
Microsoft* Visual C++* Compilers			
mpicl.bat	cl.exe	C/C++	32/64 bit
Intel® Fortran, C++ Compilers versions 9.1, 10.0 or 10.1			
mpiicc.bat	icl.exe	C	32/64 bit
mpiicpc.bat	icpc.exe	C++	32/64 bit
mpiifort.bat	ifort.exe	Fortran 77/Fortran 95	32/64 bit

NOTE:

- Compiler commands are available only in the Intel® MPI Library Development Kit.
- Compiler commands are in the *<installdir>*\bin directory. For Intel® 64 in 64-bit-enabled compiler commands are in the *<installdir>*\em64t\bin directory and 32-bit compiler commands are in the *<installdir>*\ia32\bin directory.
- Ensure that the corresponding underlying compilers (32-bit or 64-bit, as appropriate) are already in your PATH.
- To port existing, MPI-enabled applications to Intel MPI Library, recompile all sources.
- To display mini-help of a compiler command, execute it without any parameters.

2.1.1 Compiler Command Options

-profile=<*profile_name*>

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

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

-t or –trace

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

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

Include the installation path of the Intel® Trace Collector in the `VT_ROOT` environment variable to use this option.

/Zi or /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 *Environment Variables*, `I_MPI_DEBUG` for information on how to use additional debugging features with the `/Zi`, `/Z7`, or `/ZI` builds.

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

-O

Use this option to enable optimization.

-echo

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

-show

Use this option to learn how the underlying compiler is invoked. For example, use the following command to see the required compiler flags and options:

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

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

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

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

-show_env

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

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

Use this option to select the underlying compiler.

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

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

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

NOTE: This option works only with the `mpicc.bat` and `mpifc.bat` commands.

2.1.2 Configuration Files

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

`<installdir>\etc\mpi<compiler>-<name>.conf`

where:

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

`<name>` = name of underlying compiler

Source this file, if it exists, prior to compiling or linking to enable changes to the environment on a per-compiler-command basis.

2.1.3 Profiles

You can select a profile library through the `-profile` option of the Intel® MPI Library compiler drivers. You can also create your own profile as `<installdir>\etc\<profile_name>.conf`

The following variables can be defined there:

`PROFILE_PRELIB` - libraries (and paths) to include before the Intel® MPI library

`PROFILE_POSTLIB` - libraries to include after the Intel® MPI library

`PROFILE_INCPATHS` - C preprocessor arguments for any include files

For instance, create a file `<installdir>\etc\myprof.conf` with the following lines:

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

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

2.1.4 Environment Variables

I_MPI_{CC, CXX, FC}_PROFILE

Specify a default profiling library.

Syntax

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

Arguments

`<profile_name>`	Specify a default profiling library

Description

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

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

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

Syntax

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

Arguments

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

Description

Set this 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.

CFLAGS

Add additional CFLAGS to be used in compile and/or link steps.

Syntax

CFLAGS=<*flags*>

Arguments

<*flags*>	Specify the flags to be used in compile and/or link steps

Description

Set this variable to modify compiler behavior.

LDFLAGS

Set additional LDFLAGS to be used in the link step.

Syntax

CFLAGS=<*flags*>

Arguments

<*flags*>	Specify the flags to be used in the link step

Description

Set this variable to modify linker behavior.

I_MPI_ROOT

Set Intel® MPI Library installation directory path.

Syntax

I_MPI_ROOT=<*path*>

Arguments

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

Description

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

VT_ROOT

Set Intel® Trace Collector installation directory path.

Syntax

VT_ROOT=*<path>*

Arguments

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

Description

Set this variable to specify 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 is *<installdir>*\etc

Description

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

2.2 Job Startup Commands

mpiexec

Syntax

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

or

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

<l-options> <executable>

or

mpiexec –configfile *<file>*

Arguments

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

Description

In the first command-line syntax, run the specified *<executable>* with the specified options. All global and/or local options apply to all MPI processes. A single arg-set is assumed. For example, the following command executes a.out over the specified *<# of processes>*:

> mpiexec –n *<# of processes>* a.exe

In the second command-line syntax, divide the command line into multiple arg-sets, separated by colon characters. All the global options apply to all MPI processes, but the various local options and *<executable>* can be specified separately for each arg-set. For example, the following command would run each given executable on a different host:

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

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

SMPD service must already be running in order for `mpiexec` to succeed.

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

2.2.1 Global Options

-machinefile *<machine file>*

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

A machine file is a list of fully qualified or short host names, one name per line. Blank lines and lines that start with `'#'` as the first character are ignored.

By repeating a host name you will place additional processes on this host. You can also use the following format to avoid repetition of the same host name: *<host name>:<number of processes>*. For example, the following machine file:

```
host1
host1
host2
host2
host3
```

is equivalent to:

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

-g*<l-option>*

Use this option to apply the named local option *<l-option>* globally. See section *Local Options* for a list of all local options.

-genvlist *<list of env var names>*

Use this option to pass a list of environment variables with their current values.

-l

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

2.2.2 Local Options

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

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

-env <ENVVAR> <value>

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

-host <nodename>

Use this option to specify particular <nodename> on which the MPI processes in the current arg-set are to be run. For example, the following will run the executable a.exe on host host1 only:

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

-configfile <filename>

Use this option to specify the file <filename> that contains command-line options. For example, the configuration file contains the following commands to run the executables a.exe and b.exe using the rdssm device over host1 and host2 respectively:

```
-host host1 -env I_MPI_DEBUG 2 -env I_MPI_DEVICE rdssm -n 2 a.exe
-host host2 -env I_MPI_DEBUG 2 -env I_MPI_DEVICE rdssm -n 2 b.exe
```

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

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

-path <directory>

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

-wdir <directory>

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

2.2.3 Environment Variables

I_MPI_DEVICE

Select the particular network fabric to be used.

Syntax

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

Arguments

<device>	One of {sock, shm, ssm}
sock	Sockets

shm	Shared-memory only (no sockets)
ssm	Combined sockets + shared memory (for clusters with SMP nodes)

<device>	One of {rdma,rdssm}
<provider>	Optional DAPL* provider name
rdma	RDMA-capable network fabrics including InfiniBand*, Myrinet* (via DAPL*)
rdssm	Combined sockets + shared memory + DAPL* (for clusters with SMP nodes and RDMA-capable network fabrics)

Description

Set this variable to select a specific fabric combination. If the I_MPI_DEVICE variable is not defined, Intel® MPI Library selects the most appropriate fabric combination automatically.

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

```
> mpiexec.exe -n <# of processes> -env I_MPI_DEVICE shm <executable>
```

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

For these devices, if *<provider>* is not specified, the first DAPL* provider in dat.conf is used. If *<provider>* is set to none, the rdssm device establishes sockets connections between the nodes without trying to establish DAPL* connections first.

NOTE: If you build the MPI program using mpicc.bat /Zi or /Z7, the debug-enabled version of the library is used.

NOTE: If you build the MPI program using mpicc.bat -t=log, the trace-enabled version of the library is used.

NOTE: The debug-enabled and trace-enabled versions of the library are only available when you use the Intel® MPI Library Development Kit.

I_MPI_FALLBACK_DEVICE

Set this environment variable to enable fallback to the available fabric. It is valid only for rdssm and rdma modes.

Syntax

I_MPI_FALLBACK_DEVICE=*<arg>*

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Fall back to the shared memory and/or socket fabrics if initialization of the DAPL* fabric fails. This is the default value
disable | no | off | 0	Terminate the job if the fabric selected by the I_MPI_DEVICE environment variable cannot be initialized

Description

Set this variable to control fallback to the available fabric.

If I_MPI_FALLBACK_DEVICE is set to enable and an attempt to initialize the specified fabric fails, the library falls back to the shared memory and/or socket fabrics. The exact combination of devices

depends on the number of processes started per node. For example, the library can use only sockets or a mix of sockets plus shared memory (ssm) per node. This device ensures that the job will run but it may not provide the highest possible performance for the given cluster configuration.

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

I_MPI_DEBUG

Print out debugging information when an MPI program starts running.

Syntax

I_MPI_DEBUG=<*level*>

Arguments

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

Description

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

The I_MPI_DEBUG mechanism extends the MPICH2* MPICH_DBG_OUTPUT debug mechanism by overriding the current value and setting MPICH_DBG_OUTPUT=stdout.

In order to simplify process identification add the '+' or '-' sign in front of the numerical value for I_MPI_DEBUG. This setting produces debug output lines prepended with the MPI process rank, a UNIX process id, and a host name as defined at the process launch time. For example, the command:

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

produces output debug messages in the following format:

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

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

I_MPI_JOB_TIMEOUT

Set the mpiexec timeout.

Syntax

I_MPI_JOB_TIMEOUT=<*timeout*>

Arguments

<timeout>	Define mpiexec timeout period in seconds
≥0	The default timeout value is zero corresponding to no timeout

Description

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

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

I_MPI_DAT_LIBRARY

Select the particular DAT library to be used.

Syntax

I_MPI_DAT_LIBRARY=*<library>*

Arguments

<library>	Specify the exact library to be used instead of default libdat.so

Description

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

NOTE: Use this variable only if you are going to utilize a DAPL provider.

2.3 SMPD Daemon

smpd

Simple multi-purpose daemon.

Syntax

```
smpd.exe   [-port <port>] [-phrase <passphrase>] [-d] [-noprompt] \
           [-install | -start | -stop | -shutdown <hostname> | -status | \
           -restart <hostname>] [-console <hostname>] [-anyport] [-hosts] \
           [-sethosts] [-set <option_name> <option_value>] \
           [-get <option_name>] [-query <domain>] [-register_spn] \
           [-remove_spn] [-traceon <logfilename> [<hostA> <hostB> ...]] \
           [-traceoff [<hostA> <hostB> ...]] [-help]
```

Arguments

-p *<port>* | -port *<port>*	Specify the port that the smpd is listening on
-d | -debug	Start smpd in debug mode
-restart *<hostname>*	Restart smpd on specified *<hostname>*

-shutdown <hostname>	Shutdown smpd on specified <hostname>
-status <hostname>	Get smpd status on specified <hostname>
-help	Print help
-install | -regserver	Install smpd service
-remove | -unregserver | -uninstall	Remove smpd service
-start	Start smpd service
-stop	Start smpd service
-traceon <logfilename> [<hostA> <hostB> ...]	Restart smpd and store the output into provided <logfilename>
-traceoff [<hostA> <hostB> ...]	Restart smpd without logging the output

Description

SMPD is a process management system for starting parallel jobs. Before running a job, start smpd service on each host and connect them into a ring.

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

Examples:

1. Use the following command to install SMPD service:
   ```
   > smpd.exe -install
   ```

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

2. Use the following command to start the SMPD service in debug mode:
   ```
   > smpd.exe -d
   ```

2.4 Processor Information Utility

cpuinfo

Use the cpuinfo utility to display processor architecture information.

Syntax

```
cpuinfo
```

Description

The cpuinfo utility prints out processor architecture information that can be used for more suitable process pinning settings.

The output contains a number of tables:

1. General data.
   Architecture – one of “i686”, “x86_64”, “ia64”
   Hyperthreading – one of “enabled”, “disabled”, “not supported”
   Packages – the number of physical packages (sockets)
   Cores – the number of all cores
   Processors – the number of logical processors (cpu)

2. Processor identification table. The table represents three-level (thread, core, and package) identification of each logical processor.
   Thread – unique processor identifier within a core.
   Core – unique core identifier within a package.
   Package – unique package identifier within a node.
3. Processor placement table. The table represents a map of processor placement by packages and cores (inversion of previous processor identification table). Each entry contains:
   Package – a physical package identifier.
   Cores – a list of core identifiers that belong to this package.
   Processors – a list of processors that belong to this package. This list order directly corresponds to the core list. A group of processors enclosed in the brackets belongs to one core.
4. Cache sharing table. For each cache level the table contain:
   Size – cache size in bytes.
   Processors – a list of processor groups (enclosed in the brackets) that shared this cache or “no sharing” otherwise.

NOTE: Only the architecture information is printed for Itanium-2 based machines.

Examples:

1. `cpuinfo` output for Dual-Core Intel® Xeon® Processor 5100 series:

```
Architecture  : x86_64
Hyperthreading: disabled
Packages    : 2
Cores       : 4
Processors : 4

=====  Processor identification  =====
Processor       Thread  Core    Package
0               0       0       0
1               0       0       3
2               0       1       0
3               0       1       3

=====  Processor placement  =====
Package Cores           Processors
0       0,1             0,2
3       0,1             1,3

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

2. `cpuinfo` output for the Quad-Core Intel® Xeon® processor 5300 series:

```
Architecture  : x86_64
Hyperthreading: disabled
Packages    : 2
Cores       : 8
Processors : 8

=====  Processor identification  =====
Processor       Thread  Core    Package
0               0       0       0
1               0       2       0
2               0       0       1
3               0       2       1
```

```
4               0       1       0
5               0       3       0
6               0       1       1
7               0       3       1

=====  Processor placement  =====
Package Cores           Processors
0       0,2,1,3         0,1,4,5
1       0,2,1,3         2,3,6,7

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

3. `cpuinfo` output for machine with Hyper-Threading Technology enabled:

```
Architecture  : x86_64
Hyperthreading: enabled
Packages   : 2
Cores      : 2
Processors : 4

=====  Processor identification  =====
Processor       Thread  Core    Package
0               0       0       0
1               1       0       0
2               0       0       3
3               1       0       3

=====  Processor placement  =====
Package Cores           Processors
0       0               (0,1)
3       0               (2,3)

===== Cache sharing =====
Cache  Size        Processors
L1     16  KB      (0,1)(2,3)
L2     1   MB      (0,1)(2,3)
```

3 Tuning Reference

The Intel® MPI Library provides many environment variables that can be used to influence program behavior and performance at run time. These variables are described below.

3.1 Process Pinning

I_MPI_PIN

Turn on/off process pinning feature of the Intel® MPI Library.

Syntax

I_MPI_PIN=<*arg*>

Arguments

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

Description

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

I_MPI_PIN_PROCESSOR_LIST (I_MPI_PIN_PROCS)

Identify set of processors to be used for process pinning.

I_MPI_PIN_PROCESSOR_LIST=<*proclist*>

Arguments

<*proclist*>	Define mapping of process to CPU
all	Use all CPUs
allcores	Use all CPU cores (or physical CPU). This is the default value
<*l*>	Use only CPU number <*l*> (where <*l*> is 0,1, … , or total number of CPUs - 1)
<*l*>-<*m*>	Use CPUs from <*l*> to <*m*>
<*k*>,<*l*>-<*m*>,<*n*>	Use CPUs <*k*>, <*l*> through <*m*>, and <*n*>

Description

Set the I_MPI_PIN_PROCESSOR_LIST variable to define the set of processors. This variable is valid only if I_MPI_PIN is enabled.

The number and order of the processors correspond to the output of the cpuinfo utility. See *Processor Information Utility*.

This variable does not influence the process placement that is controlled by the mpdboot and mpiexec commands. However, when this variable is defined and a process is placed upon the node, that process is bound to the next CPU out of the specified set.

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

```
$ mpirun -genv I_MPI_PIN on -genv I_MPI_PIN_PROCESSOR_LIST 0,3 -n \
<# of processes> <executable>
```

To pin the processes to different CPUs on each node individually, use the following command:

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

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

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

NOTE: The value of `I_MPI_PIN_PROCESSOR_LIST` can be defined locally, on a per host level, or globally, for the entire system.

3.2 Device Control

I_MPI_EAGER_THRESHOLD

Change the eager/rendezvous cutover point for all devices.

Syntax

`I_MPI_EAGER_THRESHOLD=`*`<nbytes>`*

Arguments

`<nbytes>`	Define eager/rendezvous cutover point
`> 0`	The default *`<nbytes>`* value is equal to 262 144 bytes

Description

Set this variable to control the point-to-point protocol switchover point. Data transfer algorithms are selected based on the following scheme:

- Messages shorter than or equal in size to *`<nbytes>`* are sent using the eager protocol.
- Larger messages are sent by using the more memory efficient rendezvous protocol.

I_MPI_INTRANODE_EAGER_THRESHOLD

Change the eager/rendezvous cutover point for intranode communication mode.

Syntax

I_MPI_INTRANODE_EAGER_THRESHOLD=<*nbytes*>

Arguments

<*nbytes*>	Define the threshold for DAPL* intranode communication
> 0	The default <*nbytes*> value is equal to 262 144 bytes

Description

Set this variable to change the threshold for communication within the node. Data transfer algorithms are selected based on the following scheme:

- Messages shorter than or equal in size to <*nbytes*> are sent using the eager protocol.
- Larger messages are sent by using the more memory efficient rendezvous protocol.

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

I_MPI_WAIT_MODE

Turn on/off a wait mode.

Syntax

I_MPI_WAIT_MODE=<*arg*>

Arguments

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

Description

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

NOTE: The wait mode supports the sock, shm and ssm devices.

I_MPI_SPIN_COUNT

Control the spin count value.

Syntax

I_MPI_SPIN_COUNT=<*scount*>

Arguments

<*scount*>	Define the loop spin count when polling fabric(s)
> 0	The default <*scount*> value is equal to 1 time for sock, shm, and ssm devices, and equal to 250 times for rdma and rdssm devices

Description

Set the spin count limit. The loop for polling the fabric(s) will spin <*scount*> times before freeing the processes if no incoming messages are received for processing. Smaller values for <*scount*> cause the Intel® MPI Library to release the processor more frequently.

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

NOTE: Use the I_MPI_SPIN_COUNT environment variable with caution. Keep in mind that three different effects are possible: no effect, performance improvement, or performance degradation.

I_MPI_CACHE_BYPASS

Control a message transfer algorithm for shm device.

Syntax

I_MPI_CACHE_BYPASS=<arg>

Arguments

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

Description

Set this variable to control message transfer algorithm for shm device. By default messages greater than or equal in size to value specified by the I_MPI_CACHE_BYPASS_THRESHOLD environment variable are sent bypass cache. This feature is enabled on IA-32 and Intel® 64 architectures by default. It does not affect Itanium® 2-based systems.

I_MPI_CACHE_BYPASS_THRESHOLDS

Change messages copying algorithm cutover point.

Syntax

I_MPI_CACHE_BYPASS_THRESHOLD=<nb_send>,[<nb_recv>]

Arguments

<nb_send>	Define cutover point for sent messages
≥ 16384	Copying bypass cache is disabled by default
<nb_recv>	Define cutover point for received messages
≥ 16384	The default <nb_send> value is 2097152 bytes (2 Megabytes)

Description

Set this variable to control the switchover point for a message copying algorithm. Messages greater than or equal in size to defined thresholds value are copied bypass cache. “-1” value disables coping bypass cache. This variable is valid only if I_MPI_CACHE_BYPASS is enabled.

I_MPI_SHM_SINGLE_SEGMENT_THRESHOLD

Change the static/dynamic shared memory segment(s) allocation mode for the shm device.

Syntax

I_MPI_SHM_SINGLE_SEGMENT_THRESHOLD=<nproc>

Arguments

<nproc>	Define static/dynamic mode switch point for the shm device
> 0, < 90	The default *<nproc>* value is equal to 90

Description

Set this variable to change the allocation mode for the shm device.

The following modes are available for the allocation of the shared memory segment(s) for the shm device:

- If the number of processes started on the system is less than the value specified by *<nproc>*, the static mode is used. In that case only one common shared memory segment is allocated for all processes during the initialization stage.
- Otherwise, the dynamic mode is used and the shared memory segments are allocated for each connection individually.

NOTE: The dynamic connection establishment mode does not make sense when the static allocation mode is used. The I_MPI_DYNAMIC_CONNECTION environment variable is not applicable in this case.

3.3 RDMA and RDSSM Device Control

I_MPI_RDMA_TRANSLATION_CACHE

Turn on/off the use of a memory registration cache.

Syntax

I_MPI_RDMA_TRANSLATION_CACHE=*<arg>*

Arguments

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

Description

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

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

I_MPI_RDMA_EAGER_THRESHOLD

Change the eager/rendezvous cutover point.

Syntax

I_MPI_RDMA_EAGER_THRESHOLD=*<nbytes>*

Arguments

<nbytes>	Define eager/rendezvous cutover point

> 0	The default <nbytes> value is equal to 16 512 bytes

Description

Set this variable to control low-level point-to-point protocol switchover point. Data transfer algorithms for the rdma and rdssm devices are selected based on the following scheme:

- Messages shorter than or equal to <nbytes> are sent using the eager protocol through internal pre-registered buffers.
- Larger messages are sent by using the more memory efficient rendezvous protocol.

NOTE: This variable also determines the size of each pre-registered buffer. The higher it is the more memory is used for each established connection.

I_MPI_DYNAMIC_CONNECTION

Turn on/off the dynamic connection establishment.

Syntax

I_MPI_DYNAMIC_CONNECTION=<arg>

Arguments

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

Description

Set this variable to control dynamic connection establishment.

- If enabled, connections are established at the time of the first communication between each pair of processes. This is the default behavior.
- Otherwise, all connections are established upfront.

I_MPI_DYNAMIC_CONNECTION_MODE

Choose the algorithm for establishing of the DAPL* connections.

Syntax

I_MPI_DYNAMIC_CONNECTION_MODE=<arg>

Arguments

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

Description

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

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

I_MPI_RDMA_SCALABLE_PROGRESS

Turn on/off scalable algorithm for RDMA read progress.

Syntax

I_MPI_RDMA_SCALABLE_PROGRESS=<*arg*>

Arguments

<*arg*>	Binary indicator
enable | yes | on | 1	Turn on scalable algorithm
disable | no | off | 0	Turn off scalable algorithm. This is the default value

Description

Set this variable to select scalable algorithm for RDMA read progress. In some cases it provides advantages for large number of processes.

I_MPI_INTRANODE_SHMEM_BYPASS

Turn on/off the DAPL* intranode communication mode.

Syntax

I_MPI_INTRANODE_SHMEM_BYPASS=<*arg*>

Arguments

<*arg*>	Binary indicator
enable | yes | on | 1	Turn on the DAPL* intranode communication
disable | no | off | 0	Turn off the DAPL* intranode communication. This is the default value

Description

Set this variable to specify the communication for the universal device within the node. If the DAPL* intranode communication mode is enabled, data transfer algorithms are selected based on the following scheme:

- Messages shorter than or equal in size to the threshold value of the I_MPI_INTRANODE_EAGER_THRESHOLD variable are transferred using shared memory.
- Large messages are transferred via the DAPL* layer.

NOTE: This variable is applicable only when shared memory and the DAPL* layer are turned on by setting the I_MPI_DEVICE environment variable to the rdssm value.

I_MPI_RDMA_EVD_POLLING

Turn on/off the use of the Event Dispatcher (EVD) as a fallback method when polling for messages.

Syntax

I_MPI_RDMA_EVD_POLLING=<*arg*>

Arguments

<*arg*>	Binary indicator
enable | yes | on | 1	Turn on the EVD based fallback
disable | no | off | 0	Turn off the EVD based fallback. This is the default value

Description

Set this variable to use the DAPL* Event Dispatcher (EVD) for detecting incoming messages.

Use this method instead of the default method of buffer polling if the DAPL* provider does not guarantee the delivery of the transmitted data in order from low to high addresses.

NOTE: The EVD method of message detection is typically substantially slower than the default algorithm.

I_MPI_RDMA_BUFFER_NUM

Change the number of internal pre-registered buffers for each pair in a process group.

Syntax

`I_MPI_RDMA_BUFFER_NUM=<nbuf>`

Arguments

<nbuf>	Define the number of buffers for each pair in a process group
> 0	The default *<nbuf>* value ranges between 8 and 40 depending on the cluster size and platform

Description

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

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

I_MPI_RDMA_BUFFER_SIZE

Change the size of internal pre-registered buffers for each pair in a process group.

Syntax

`I_MPI_RDMA_BUFFER_SIZE=<nbytes>`

Arguments

<nbytes>	Define the size of pre-registered buffers
> 0	The default *<nbytes>* value is equal to 16 640 bytes

Description

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

I_MPI_RDMA_BUFFER_ENLARGEMENT

Turn on/off the use of two-phase buffer enlargement.

Syntax

`I_MPI_RDMA_BUFFER_ENLARGEMENT =<arg>`

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Turn on the mode of using two-phase buffer enlargement

disable | no | off | 0	Turn off the mode of using two-phrase buffer enlargement. This is the default value

Description

Set this variable to control the use of the two-phase buffer enlargement according to the following algorithm:

- If enabled, small size internal pre-registered RDMA buffers are allocated and enlarged later if data size exceeds the threshold defined by I_MPI_RDMA_BUFFER_ENLARGEMENT_THRESHOLD
- Two-phase buffer enlargement is turned off by default.

I_MPI_RDMA_BUFFER_ENLARGEMENT_THRESHOLD

Change threshold for two-phase buffer enlargement mode.

Syntax

I_MPI_RDMA_BUFFER_ENLARGEMENT_THRESHOLD=*<nbytes>*

Arguments

<nbytes>	Define the threshold for starting enlargement of the RDMA buffers
> 0	The default value is 580

Description

Set this variable to define the threshold for increasing the size of the two-phase RDMA buffers. This variable is valid only if I_MPI_RDMA_BUFFER_ENLARGEMENT is enabled.

I_MPI_RDMA_TINY_PACKET

Turn on/off the use of small packets.

Syntax

I_MPI_RDMA_TINY_PACKET=*<arg>*

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Turn on the use of small packets
disable | no | off | 0	Turn off the use of small packets. This is the default value (the regular packet sizes is used instead)

Description

Set this variable to use the small packets for short messages. The regular packet sizes are used by default.

Certain DAPL* providers are sensitive to the packet size on certain hardware. Switching on small packets for short messages may increase performance in these cases.

I_MPI_RDMA_RNDV_WRITE

Turn on/off the rendezvous RDMA Write protocol.

Syntax

I_MPI_RDMA_RNDV_WRITE=*<arg>*

Arguments

<arg>	Binary indicator
enable | yes | on | 1	Turn on the RDMA Write rendezvous protocol
disable | no | off | 0	Turn off the RDMA Write rendezvous protocol. This is the default value (the RDMA Read protocol is used instead)

Description

Set this variable to select the RDMA Write-based rendezvous protocol.

Certain DAPL* providers have a slow RDMA Read implementation on certain platforms. Switching on the rendezvous protocol based on the RDMA Write operation may increase performance in these cases.

I_MPI_RDMA_CHECK_MAX_RDMA_SIZE

Check the value of the DAPL* attribute max_rdma_size.

Syntax

I_MPI_RDMA_CHECK_MAX_RDMA_SIZE=*<arg>*

Arguments

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

Description

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

- If set to disable, the Intel® MPI Library does not take into account the value of the DAPL* attribute max_rdma_size for message fragmentation.
- If set to enable, the Intel® MPI Library fragments messages of size greater than the value of the DAPL* attribute max_rdma_size.

I_MPI_RDMA_CONN_EVD_SIZE

Define the event queue size of the DAPL* event dispatcher.

Syntax

I_MPI_RDMA_CONN_EVD_SIZE=*<size>*

Arguments

<size>	Define the length of the event queue
> 0	The default value is queried from the DAPL provider

Description

Set this variable to define the event queue size of the DAPL event dispatcher. If this variable is set, the minimum value between *<size>* and the value obtained from the provider is used as the size of the event queue. The provider is required to supply a queue size that is at least equal to the calculated value, but it can also provide a larger queue size.

3.4 Collective Operation Control

Each collective operation in the Intel® MPI Library supports a number of communication algorithms. In addition to reasonable default settings, the library provides two ways to control the algorithm selection explicitly: the novel I_MPI_ADJUST environment variable family and the deprecated I_MPI_MSG environment variable family.

3.4.1 I_MPI_ADJUST family

I_MPI_ADJUST_<opname>

Control collective operation algorithm selection.

Syntax

I_MPI_ADJUST_<opname>=<algid>[:<conditions>][;<algid>:<conditions>[…]]

Arguments

<algid>	Algorithm identifier
≥ 0	The default value of zero selects reasonable default settings

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

Description

Set this variable to select the desired algorithm(s) for the collective operation <opname> under particular conditions. Each collective operation has its own environment variable and algorithms (see Table 3-1).

Table 3-1 Environment variables, collective operations, and algorithms

Environment variable	Collective operation	Algorithms
I_MPI_ADJUST_ALLGATHER	MPI_Allgather	1. Recursive doubling algorithm 2. Bruck's algorithm 3. Ring algorithm 4. Topology aware Gatherv + Bcast algorithm

I_MPI_ADJUST_ALLGATHERV	MPI_Allgatherv	1. Recursive doubling algorithm 2. Bruck's algorithm 3. Ring algorithm 4. Topology aware Gatherv + Bcast algorithm
I_MPI_ADJUST_ALLREDUCE	MPI_Allreduce	1. Recursive doubling algorithm 2. Rabenseifner's algorithm 3. Reduce + Bcast algorithm 4. Topology aware Reduce + Bcast algorithm
I_MPI_ADJUST_ALLTOALL	MPI_Alltoall	1. Bruck's algorithm 2. Isend/Irecv + waitall algorithm 3. Pair wise exchange algorithm
I_MPI_ADJUST_ALLTOALLV	MPI_Alltoallv	1. Isend/Irecv + waitall algorithm
I_MPI_ADJUST_ALLTOALLW	MPI_Alltoallw	1. Isend/Irecv + waitall algorithm
I_MPI_ADJUST_BARRIER	MPI_Barrier	1. Dissemination algorithm 2. Recursive doubling algorithm 3. Topology aware dissemination algorithm 4. Topology aware recursive doubling algorithm
I_MPI_ADJUST_BCAST	MPI_Bcast	1. Binomial algorithm 2. Recursive doubling algorithm 3. Ring algorithm 4. Topology aware binomial algorithm
I_MPI_ADJUST_EXSCAN	MPI_Exscan	1. Partial results gathering algorithm 2. Partial results gathering regarding algorithm layout of processes
I_MPI_ADJUST_GATHER	MPI_Gather	1. Binomial algorithm 2. Topology aware binomial algorithm
I_MPI_ADJUST_GATHERV	MPI_Gatherv	1. Linear algorithm 2. Topology aware linear algorithm
I_MPI_ADJUST_REDUCE_SCATTER	MPI_Reduce_scatter	1. Recursive having algorithm 2. Pair wise exchange algorithm 3. Recursive doubling algorithm 4. Reduce + Scatterv algorithm 5. Topology aware Reduce + Scatterv algorithm

I_MPI_ADJUST_REDUCE	MPI_Reduce	1. Shumilin's algorithm 2. Binomial algorithm 3. Topology aware Shumilin's algorithm 4. Topology aware binomial algorithm
I_MPI_ADJUST_SCAN	MPI_Scan	1. Partial results gathering algorithm 2. Topology aware partial results gathering algorithm
I_MPI_ADJUST_SCATTER	MPI_Scatter	1. Binomial algorithm 2. Topology aware binomial algorithm
I_MPI_ADJUST_SCATTERV	MPI_Scatterv	1. Linear algorithm 2. Topology aware linear algorithm

The message size calculation rules for the collective operations are described in Table 3-2. Here, "n/a" means that the corresponding interval <*l*>-<*m*> should be omitted.

Table 3-2

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	Ecv_count*recv_type_size if MPI_IN_PLACE is used, otherwise recv_count*recv_type_size
MPI_Gatherv	n/a
MPI_Reduce_scatter	total_recv_count*type_size
MPI_Reduce	count*type_size
MPI_Scan	count*type_size
MPI_Scatter	send_count*send_type_size if MPI_IN_PLACE is used, otherwise recv_count*recv_type_size
MPI_Scatterv	n/a

Examples:

1. Use the following settings to select second algorithm for `MPI_Reduce` operation:
   `I_MPI_ADJUST_REDUCE=2`
2. Use the following settings to define algorithms for `MPI_Scatter` operation:
   `I_MPI_ADJUST_REDUCE_SCATTER=4:0-100,5001-10000;1:101-3200,2:3201-5000;3`

   In this case algorithm four will be used for the message sizes from 0 up to 100 bytes and from 5001 to 10000 bytes, algorithm one will be used for the message sizes from 101 up to 3200 bytes, algorithm two will be used for the message sizes from 3201 up to 5000 bytes, and algorithm three will be used for all other messages.

3.4.2 I_MPI_MSG family

These variables are deprecated and supported mostly for backward compatibility. Use the `I_MPI_ADJUST` environment variable family whenever possible.

I_MPI_FAST_COLLECTIVES

Control default library behavior during selection of appropriate collective algorithm for specific execution situation.

Syntax

`I_MPI_FAST_COLLECTIVES=<arg>`

Arguments

<arg>	Binary indicator
`enable | yes | on | 1`	Fast collective algorithms are used. This is the default value
`disable | no | off | 0`	Slower and safer collective algorithms are used

Description

The Intel® MPI Library uses advanced collective algorithms designed for better application performance by default. The implementation makes the following assumptions:

- It is safe to utilize the flexibility of the MPI standard regarding the order of execution of the collective operations to take advantage of the process layout and other opportunities.
- There is enough memory available for allocating additional internal buffers.

Set the `I_MPI_FAST_COLLECTIVES` variable to `disable` if you need to obtain results that do not depend on the physical process layout or other factors.

NOTE: Some optimizations controlled by this variable are of an experimental nature. In case of failure, turn off the collective optimizations and repeat the run.

I_MPI_BCAST_NUM_PROCS

Control `MPI_Bcast` algorithm thresholds.

Syntax

`I_MPI_BCAST_NUM_PROCS=<nproc>`

Arguments

<nproc>	Define the number of processes threshold for choosing the `MPI_Bcast` algorithm
`> 0`	The default value is 8

I_MPI_BCAST_MSG

Control MPI_Bcast algorithm thresholds.

Syntax

I_MPI_BCAST_MSG=<*nbytes1,nbytes2*>

Arguments

<*nbytes1,nbytes2*>	Define the message size threshold range (in bytes) for choosing the MPI_Bcast algorithm
> 0 nbytes2 >= nbytes1	The default pair of values is 12288,524288

Description

Set these variables to control the selection of the three possible MPI_Bcast algorithms according to the following scheme (see Table 3-1 for algorithm descriptions):

1. The first algorithm is selected if the message size is less then <*nbytes1*>, or the number of processes in the operation is less than <*nproc*>.
2. The second algorithm is selected if the message size is greater than or equal to <*nbytes1*> and less than <*nbytes2*>, and the number of processes in the operation is a power of two.
3. If none of the above conditions is satisfied, the third algorithm is selected.

I_MPI_ALLTOALL_NUM_PROCS

Control MPI_Alltoall algorithm thresholds.

Syntax

I_MPI_ALLTOALL_NUM_PROCS=<*nproc*>

Arguments

<*nproc*>	Define the number of processes threshold for choosing the MPI_Alltoall algorithm
> 0	The default value is 8

I_MPI_ALLTOALL_MSG

Control MPI_Alltoall algorithm thresholds.

Syntax

I_MPI_ALLTOALL_MSG=<*nbytes1,nbytes2*>

Arguments

<*nbytes1,nbytes2*>	Defines the message size threshold range (in bytes) for choosing the MPI_Alltoall algorithm
> 0 nbytes2 >= nbytes1	The default pair of values is 256,32768

Description

Set these variables to control the selection of the three possible MPI_Alltoall algorithms according to the following scheme (see Table 3-1 for algorithm descriptions):

1. The first algorithm is selected if the message size is greater than or equal to <*nbytes1*>, and the number of processes in the operation is not less than <*nproc*>.

2. The second algorithm is selected if the message size is greater than <nbytes1> and less than or equal to <nbytes2>, or if the message size is less than <nbytes2> and the number of processes in the operation is less than <nproc>.
3. If none of the above conditions is satisfied, the third algorithm is selected.

I_MPI_ALLGATHER_MSG

Control MPI_Allgather algorithm thresholds.

Syntax

I_MPI_ALLGATHER_MSG=<nbytes1,nbytes2>

Arguments

<nbytes1,nbytes2>	Define the message size threshold range (in bytes) for choosing the MPI_Allgather algorithm
> 0 nbytes2 >= nbytes1	The default pair of values is 81920,524288

Description

Set this variable to control the selection of the three possible MPI_Allgather algorithms according to the following scheme (see Table 3-1 for algorithm descriptions):

1. The first algorithm is selected if the message size is less than <nbytes2> and the number of processes in the operation is a power of two.
2. The second algorithm is selected if the message size is less than <nbytes1> and number of processes in the operation is not a power of two.
3. If none of the above conditions is satisfied, the third algorithm is selected.

I_MPI_ALLREDUCE_MSG

Control MPI_Allreduce algorithm thresholds.

Syntax

I_MPI_ALLREDUCE_MSG=<nbytes>

Arguments

<nbytes>	Define the message size threshold (in bytes) for choosing the MPI_Allreduce algorithm
> 0	The default value is 2048

Description

Set this variable to control the selection of the two possible MPI_Allreduce algorithms according to the following scheme (see Table 3-1 for algorithm descriptions):

1. The first algorithm is selected if the message size is less than or equal <nbytes>, or the reduction operation is user-defined, or the count argument is less than the nearest power of two less than or equal to the number of processes.
2. If the above condition is not satisfied, the second algorithm is selected.

I_MPI_REDSCAT_MSG

Control the MPI_Reduce_scatter algorithm thresholds.

Syntax

I_MPI_REDSCAT_MSG=<nbytes1,nbytes2>

Arguments

<nbytes>	Define the message size threshold range (in bytes) for choosing the MPI_Reduce_scatter algorithm
> 0	The default value is 512,524288

Description

Set this variable to control the selection of the three possible MPI_Reduce_scatter algorithms according to the following scheme (see Table 3-1 for algorithm descriptions):

1. The first algorithm is selected if the reduction operation is commutative and the message size is less than *<nbytes2>*.
2. The second algorithm is selected if the reduction operation is commutative and the message size is greater than or equal to *<nbytes2>*, or if the reduction operation is not commutative and the message size is greater than or equal to *<nbytes1>*.
3. If none of the above conditions is satisfied, the third algorithm is selected.

I_MPI_SCATTER_MSG

Control MPI_Scatter algorithm thresholds.

Syntax

I_MPI_SCATTER_MSG=*<nbytes>*

Arguments

<nbytes>	Define the buffer size threshold range (in bytes) for choosing the MPI_Scatter algorithm
> 0	The default value is 2048

Description

Set this variable to control the selection of the two possible MPI_Scatter algorithms according to the following scheme (see Table 3-1 for algorithm descriptions):

1. The first algorithm is selected on the intercommunicators if the message size is greater than *<nbytes>*.
2. If the above condition is not satisfied, the second algorithm is selected.

I_MPI_GATHER_MSG

Control MPI_Gather algorithm thresholds.

Syntax

I_MPI_GATHER_MSG=*<nbytes>*

Arguments

<nbytes>	Define the buffer size threshold range (in bytes) for choosing the MPI_Gather algorithm
> 0	The default value is 2048

Description

Set this variable to control the selection of the two possible MPI_Gather algorithms according to the following scheme (see Table 3-1 for algorithm descriptions):

1. The first algorithm is selected on the intercommunicators if the message size is greater than *<nbytes>*.
2. If the above condition is not satisfied, the second algorithm is selected.

4 Statistics Gathering Mode

Intel® MPI Library has a built-in statistics gathering facility that collects essential performance data without disturbing the application execution. The collected information is output onto an easy to interpret text file. This section describes the environment variables used to control the built-in statistics gathering facility and provides example output files.

I_MPI_STATS

Control the statistics information output.

Syntax

I_MPI_STATS=<*level*>

Arguments

<*level*>	Indicate level of statistics information provided
0	Do not collect any statistics. This is the default value
1	Output the amount of data sent by each process
2	Output the number of calls and amount of transferred data
> 2	Output collective operation statistics for all communication contexts

Description

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

I_MPI_STATS_SCOPE

Select the subsystem(s) for output statistics.

Syntax

I_MPI_STATS_SCOPE=<*subsystem*>[:<*ops*>][;<*subsystem*>[:<*ops*>][…]]

Arguments

<*subsystem*>	Define the target subsystem(s)
all	Collect statistics data for all operations. This is default value
coll	Collect statistics data for all collective operations
p2p	Collect statistics data for all point to point operations

<*ops*>	Define the target operations as a comma separated list
Allgather	MPI_Allgather
Allgatherv	MPI_Allgatherv
Allreduce	MPI_Allreduce

Alltoall	MPI_Alltoall
Alltoallv	MPI_Alltoallv
Alltoallw	MPI_Alltoallw
Barrier	MPI_Barrier
Bcast	MPI_Bcast
Exscan	MPI_Exscan
Gather	MPI_Gather
Gatherv	MPI_Gatherv
reduce_scatter	MPI_Reduce_scatter
Reduce	MPI_Reduce
Scan	MPI_Scan
Scatter	MPI_Scatter
Scatterv	MPI_Scatterv
Send	Standard transfers (MPI_Send, MPI_Isend, MPI_Send_init)
Bsend	Buffered transfers (MPI_Bsend, MPI_Ibsend, MPI_Bsend_init)
Csend	Point to point operations inside the collectives
Rsend	Ready transfers (MPI_Rsend, MPI_Irsend, MPI_Rsend_init)
Ssend	Synchronous transfers (MPI_Ssend, MPI_Issend, MPI_Ssend_init)

Description

Set this variable to select the target subsystem. All collective and point to point operations, including point to point support of the collectives are covered by default.

Examples:

3. The default settings are equivalent to:
   I_MPI_STATS_SCOPE=coll;p2p
4. Use the following settings to collect statistics for MPI_Bcast, MPI_Reduce, and all point to point operations:
   I_MPI_STATS_SCOPE=p2p;coll:bcast,reduce
5. Use the following settings to collect statistics for point to point operations inside the collectives:
   I_MPI_STATS_SCOPE=p2p:csend

I_MPI_STATS_FILE

Define the statistics output file name

Syntax

I_MPI_STATS_FILE=*<name>*

Arguments

<name>	Define the statistics output file name

Description

Set this variable to define the statistics output file. The stats.txt file is created in the current directory by default.

The statistics data is blocked and ordered according to the process ranks in the MPI_COMM_WORLD communicator. The timing data is presented in microseconds.

For example, with the following settings in effect

```
I_MPI_STATS=2
I_MPI_STATS_SCOPE=p2p;coll:bcast
```

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

```
Intel(R) MPI Library Version 3.1
____ MPI Communication Statistics ____

Stats level: 2
P2P scope: <FULL>
Collective scope: <Bcast>

~~~~ Process 0 of 2 on node host1

Data Transfers
Src    Dst    Volume(MB)  Transfers
----------------------------------------
000 --> 000  0.000000e+00      0
000 --> 001  9.384155e-02      6
========================================
Totals 9.384155e-02      6

Communication Activity
Operation    Volume(MB)  Calls
----------------------------------------
P2P
Csend        9.384155e-02      6
Send         0.000000e+00      0
Bsend        0.000000e+00      0
Rsend        0.000000e+00      0
Ssend        0.000000e+00      0
Collectives
Bcast        9.384155e-02      6
========================================

~~~~ Process 1 of 2 on node host1

Data Transfers
Src    Dst    Volume(MB)  Transfers
----------------------------------------
001 --> 000  0.000000e+00      0
```

```
001 --> 001  0.000000e+00      0
=========================================
Totals 0.000000e+00      0

Communication Activity
Operation    Volume(MB)  Calls
-----------------------------------------
P2P
Csend        0.000000e+00      0
Send         0.000000e+00      0
Bsend        0.000000e+00      0
Rsend        0.000000e+00      0
Ssend        0.000000e+00      0
Collectives
Bcast        9.384155e-02      6
=========================================

____ End of stats.txt file ____
```

5 Unified memory management

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

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

These pointers also affect the C++ new and delete operators.

The respective standard C library functions are used by default.

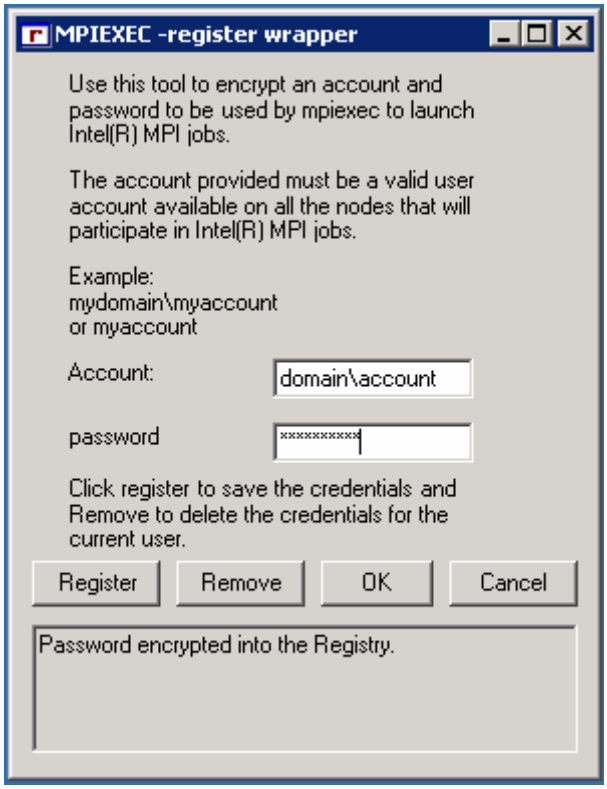
To use the unified memory management subsystem, link your application against the `libimalloc.dll`.

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

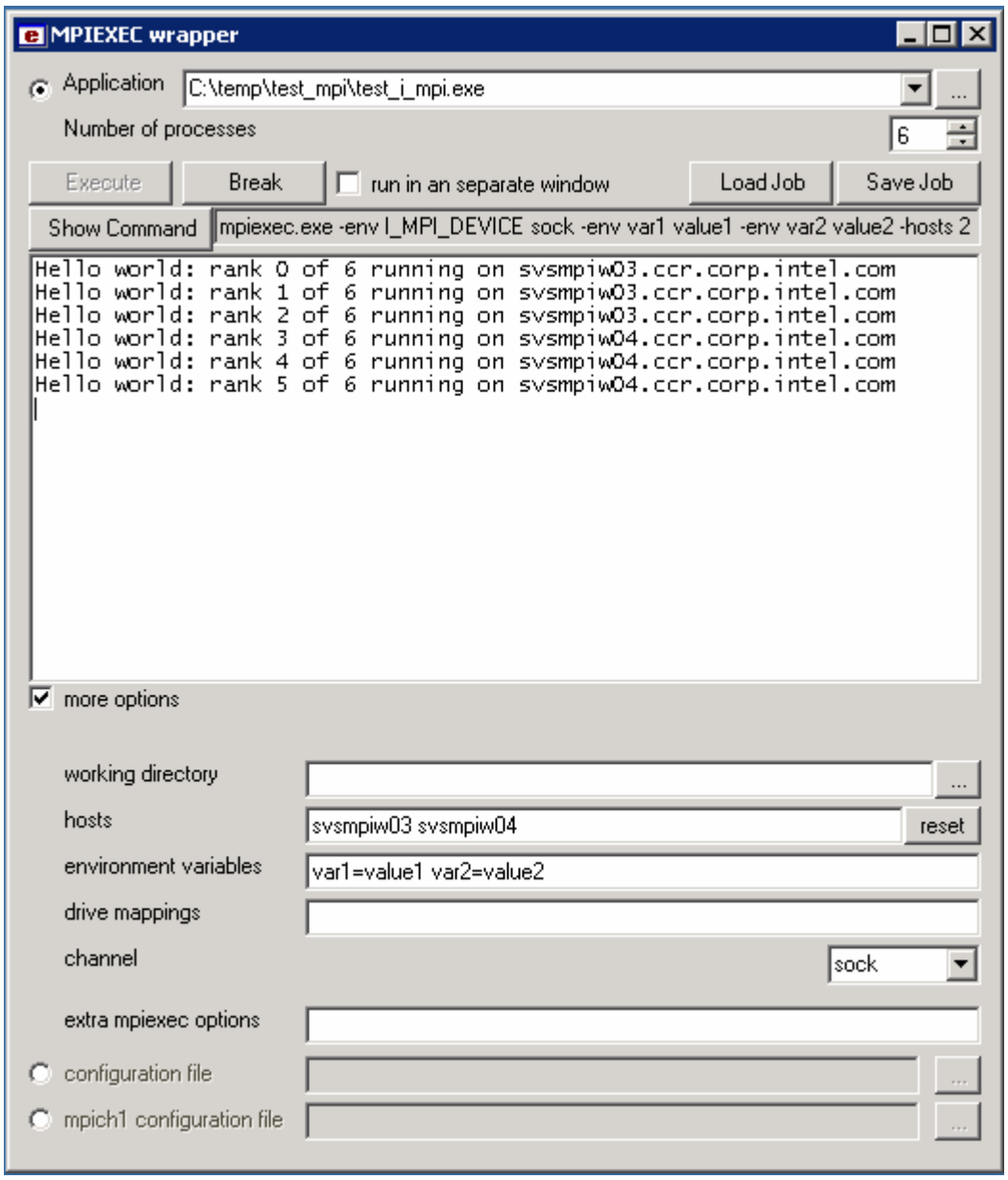
```
#include <i_malloc.h>
#include <my_malloc.h>

int main( int argc, int argv )
{
    // override normal pointers
    i_malloc = my_malloc;
    i_calloc = my_calloc;
    i_realloc = my_realloc;
    i_free = my_free;

#ifdef _WIN32
    // also override pointers used by DLLs
    i_malloc_dll = my_malloc;
    i_calloc_dll = my_calloc;
    i_realloc_dll = my_realloc;
    i_free_dll = my_free;
#endif

    // now start using Intel(R) libraries
}
```

6 Graphical utilities

The Intel® MPI Library provides three graphical utilities: `wmpiregister`, `wmpiexec`, and `wmpiconfig`.

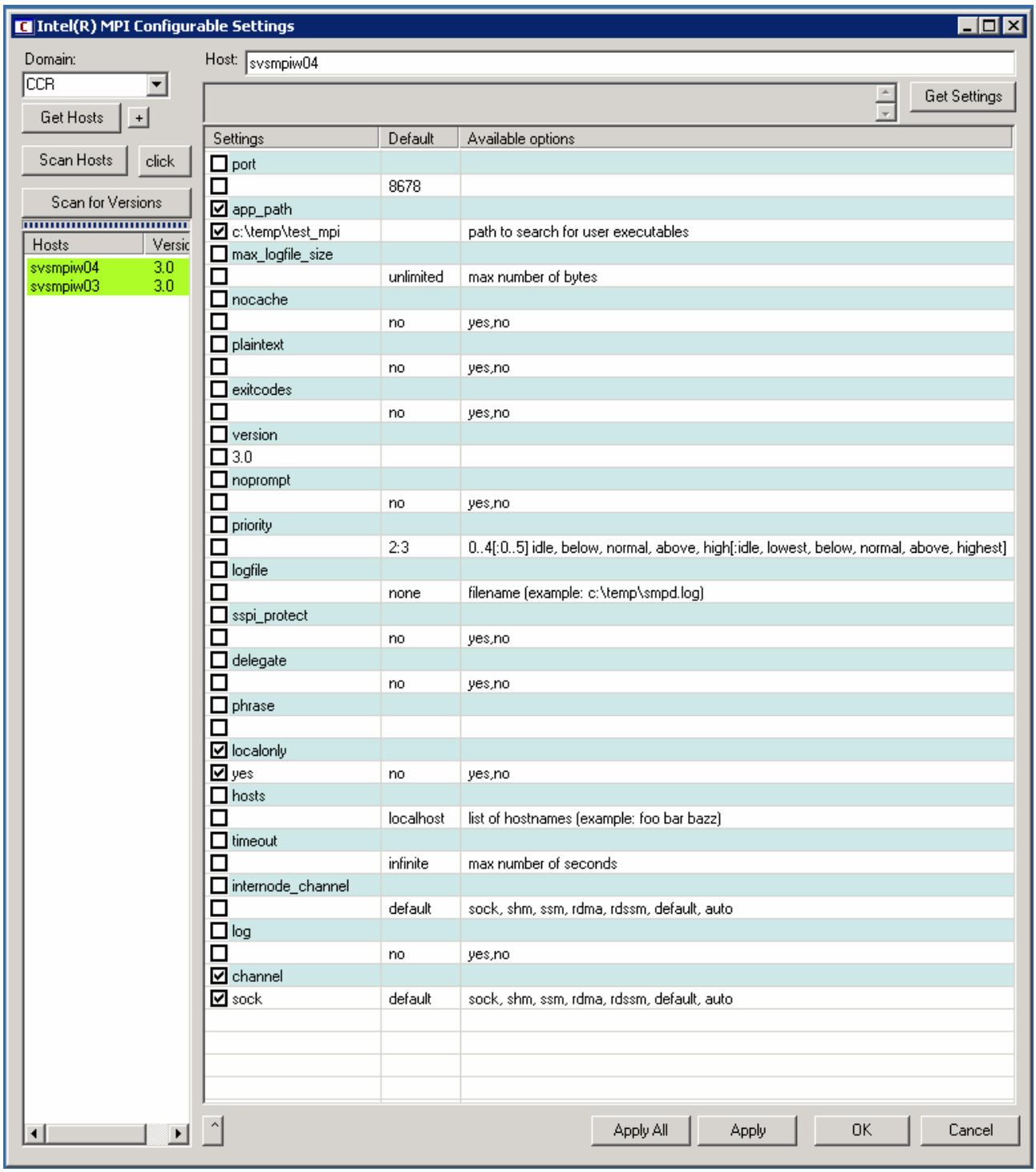


Use the `wmpiregister` utility to encrypt and store your account name and password for all subsequent MPI jobs. If you do not use this utility, you will have to enter your account name and password during each `mpiexec` invocation.



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

1. Describe the job by specifying the application, the number of instances, the machines, the communication channel, and other details in the main `wmpiexec` window.
2. Save the created job description using the ‘Save Job’ button (optional).
3. Load the job description using the ‘Load Job’ button (optional).
4. View the actual `mpiexec` command line using the ‘Show Command’ button.
5. Launch the job using the ‘Execute’ button.
6. Break the job execution using the ‘Break’ button.



Use the `wmpiconfig` utility to select an Intel® MPI host and get or set its SMPD settings. This will affect every job run on that host. Do the following:

1. Select the host(s) for which you want to change the SMPD settings. Use one of the following methods:
   a. Host manipulation
      - For each host, type the host name in the ‘Host’ text box and press the ‘Get Settings’ button. The specified host will be added to the list, and its options will be displayed.
   b. Domain manipulation
      - Select the domain using the ‘Domain’ combo box and press the ‘Get Hosts’ button. All hosts from the specified domain will be added to the list.
      - Toggle the ‘click’ button. Now you will get the host settings when that host name is selected.

- Select a host name from the list. The SMPD settings for the selected host will be displayed.

2. Change the SMPD settings.
   - Double click the desired option. In-place editing will be started.
   - Enter a new value for the option. An empty value will remove this option.
   - Press the ‘Apply’ button to apply changes to the selected host.
   - Press ‘Apply All’ button to apply changes to all hosts on the list.