Disclaimer and Legal Information

Information in this document is provided in connection with Intel products. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document. EXCEPT AS PROVIDED IN INTEL’S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility application. Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

The software described in this Intel® MPI Library Reference Manual may contain software defects which may cause the product to deviate from published specifications. Current characterized software defects are available on request. This Intel® MPI Library Reference Manual, as well as the software described in it is furnished under license and may only be used or copied in accordance with the terms of the license. The information in this manual is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Intel Corporation. Intel Corporation assumes no responsibility or liability for any errors or inaccuracies that may appear in this document or any software that may be provided in association with this document. Except as permitted by such license, no part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without the express written consent of Intel Corporation.

Developers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Improper use of reserved or undefined features or instructions may cause unpredictable behavior or failure in developer’s software code when running on an Intel processor. Intel reserves these features or instructions for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from their unauthorized use.

Intel, the Intel logo, Intel Inside, the Intel Inside logo, Pentium, Itanium, Intel Xeon, Celeron, Intel SpeedStep, Intel Centrino, Intel NetBurst, Intel NetStructure, VTune, MMX, the MMX logo, Dialogic, i386, i486, iCOMP, Intel386, Intel486, Intel740, IntelDX2, IntelDX4 and IntelSX2 are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

Copyright © 2004-2007 Intel Corporation
MPI Legal Notices

Intel® MPI Library is based in part on the MPICH2* implementation of MPI from Argonne National Laboratory* (ANL).

Intel® MPI Library is also based in part on InfiniBand Architecture* RDMA drivers from MVAPICH2* from Ohio State University’s Network-Based Computing Laboratory.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disclaimer and Legal Information</td>
<td>2</td>
</tr>
<tr>
<td>MPI Legal Notices</td>
<td>3</td>
</tr>
<tr>
<td>Contents</td>
<td>4</td>
</tr>
<tr>
<td>Overview</td>
<td>5</td>
</tr>
<tr>
<td>Command Reference</td>
<td>6</td>
</tr>
<tr>
<td>Compiler Commands</td>
<td></td>
</tr>
<tr>
<td>Compiler Command Options</td>
<td>7</td>
</tr>
<tr>
<td>Configuration Files</td>
<td>8</td>
</tr>
<tr>
<td>Environment Variables</td>
<td>9</td>
</tr>
<tr>
<td>Job Startup Commands</td>
<td>10</td>
</tr>
<tr>
<td>Global Options</td>
<td>10</td>
</tr>
<tr>
<td>Local Options</td>
<td>13</td>
</tr>
<tr>
<td>Configuration Files</td>
<td>14</td>
</tr>
<tr>
<td>Environment Variables</td>
<td>15</td>
</tr>
<tr>
<td>MPD Daemon Commands</td>
<td>20</td>
</tr>
<tr>
<td>Configuration Files</td>
<td>25</td>
</tr>
<tr>
<td>Environment Variables</td>
<td>26</td>
</tr>
<tr>
<td>Simplified Job Startup Command</td>
<td>26</td>
</tr>
<tr>
<td>Tuning Reference</td>
<td>28</td>
</tr>
<tr>
<td>Process Pinning</td>
<td>28</td>
</tr>
<tr>
<td>Device Control</td>
<td>29</td>
</tr>
<tr>
<td>SOCK Device Control</td>
<td>31</td>
</tr>
<tr>
<td>RDMA and RDSSM Device Control</td>
<td>31</td>
</tr>
<tr>
<td>Collective Operation Control</td>
<td>37</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>40</td>
</tr>
</tbody>
</table>
Overview

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 MPD daemons and supporting utilities, shared (.so) libraries, Release Notes, a Getting Started Guide, and a Reference Manual.
- **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, static (.a) libraries, debug libraries, trace libraries, and test codes.

The goal of this Reference Manual is to provide you with a complete command and tuning reference for the Intel MPI Library.
# Command Reference

## 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>
<thead>
<tr>
<th>Compiler Command</th>
<th>Underlying Compiler</th>
<th>Supported Language(s)</th>
<th>Supported ABI(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpicc</td>
<td>gcc, cc</td>
<td>C</td>
<td>32/64 bit</td>
</tr>
<tr>
<td>mpicxx</td>
<td>g++ v3.x</td>
<td>C/C++</td>
<td>32/64 bit</td>
</tr>
<tr>
<td></td>
<td>g++ v4.x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpicxx2</td>
<td>g++ v2.x</td>
<td>C/C++</td>
<td>32/64 bit</td>
</tr>
<tr>
<td>mpif77</td>
<td>g77</td>
<td>F77</td>
<td>32/64 bit</td>
</tr>
<tr>
<td>mpif90</td>
<td>gfortran</td>
<td>F95</td>
<td>32/64 bit</td>
</tr>
<tr>
<td><strong>Intel® Fortran, C++ Compilers version 8.0, 8.1, 9.0 or 9.1</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpiicc</td>
<td>icc</td>
<td>C</td>
<td>32/64 bit</td>
</tr>
<tr>
<td>mpiicpc</td>
<td>icpc</td>
<td>C++</td>
<td>32/64 bit</td>
</tr>
<tr>
<td>mpiifort</td>
<td>ifort</td>
<td>F77/F95</td>
<td>32/64 bit</td>
</tr>
<tr>
<td><strong>Intel® Fortran, C++ Compilers version 7.1</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpiicc7</td>
<td>icc</td>
<td>C</td>
<td>32 bit</td>
</tr>
<tr>
<td>mpiicpc7</td>
<td>icpc</td>
<td>C++</td>
<td>32 bit</td>
</tr>
<tr>
<td>mpiifc</td>
<td>ifc</td>
<td>F77/F90</td>
<td>32 bit</td>
</tr>
<tr>
<td>mpiecc</td>
<td>ecc</td>
<td>C</td>
<td>64 bit</td>
</tr>
<tr>
<td>mpiecpc</td>
<td>ecpc</td>
<td>C++</td>
<td>64 bit</td>
</tr>
<tr>
<td>mpiefc</td>
<td>efc</td>
<td>F77/F90</td>
<td>64 bit</td>
</tr>
</tbody>
</table>

**NOTES**

- 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>/bin64` directory and 32-bit compiler commands are in the `<installdir>/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.
Compiler Command Options

-**show**

Use this option to display the compiler and link options without actually invoking `mpicc` and related commands. For example, use the following command to show the required compile flags, options, and include a path for compiling source files:

```bash
$ mpicc -show -c test.c
```

Use the following command to show the required link flags, options, and libraries for linking object files:

```bash
$ mpicc -show -o a.out test.o
```

This is useful for debugging, for submitting support issues, or for determining compile and link options for complex builds.

-**echo**

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

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

Use this option to set the path/name of the underlying compiler to be used.

-**g**

Use the `-g` option to compile a program in debug mode and link the resulting executable against the debugging versions of the libraries. See also `I_MPI_DEBUG`, in the `Environment variables` section for information on how to use additional debug features with the `-g` builds.

-**O**

Use this option to enable optimization. If `-g` is used, the `-O` functionality is disabled. Specify `-O` explicitly if you want to enable optimization.

-**t or –trace**

Use the `-t` option (or, equivalently, `–trace`) to link the resulting executable against the Intel® Trace Collector library. Use the `-t=log` option (or, equivalently, `–trace=log`) to link the resulting executable against logging versions of the Intel MPI libraries and the Intel Trace Collector. 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.

-**dynamic_log**

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

Include the `$VT_ROOT/slib` in the `LD_LIBRARY_PATH` environment variable to run the resulting programs.

-**profile=<profile_name>**

Use this option to specify the MPI profiling library to be used, where `<profile_name>` is:

- The name of the configuration file is `<profile_name>.conf`. The configuration file is located in the `<installdir>/etc` directory. Find the format description of this configuration file in the `Configuration Files` section.
The library `lib<profile_name>.so` or `lib<profile_name>.a` is located in the same directory as the Intel MPI Library. This library is included before the Intel MPI Library during the link stage.

**-static_mpi**

Use this option to link the `libmpi` library statically. This option does not affect the default linkage method for other libraries.

**-mt_mpi**

Use this option to link the thread safe version of the Intel MPI Library. Depending on desired level of thread support `MPI_Init()` returns support on requested level up to `MPI_THREAD_MULTIPLE`. Use the thread safe libraries if you request the thread support at the following levels `MPI_THREAD_FUNNELED`, `MPI_THREAD_SERIALIZED` or `MPI_THREAD_MULTIPLE`.

### NOTES

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

- If you specify one of the following options for the Intel® Fortran Compiler, the thread safe version of the library is used:
  - `-openmp`
  - `-parallel`
  - `-threads`
  - `-reentrancy`
  - `-reentrancy threaded`

**-nocompchk**

Use this option to disable compiler setup checks and to speed up compilation. By default, each compiler command performs checks to ensure that the appropriate underlying compiler is set up correctly.

**-gcc-version=<nnn>**

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

<table>
<thead>
<tr>
<th><code>&lt;nnn&gt;</code> value</th>
<th>GNU* C++ version</th>
</tr>
</thead>
<tbody>
<tr>
<td>320</td>
<td>3.2.x</td>
</tr>
<tr>
<td>330</td>
<td>3.3.x</td>
</tr>
<tr>
<td>340</td>
<td>3.4.x</td>
</tr>
<tr>
<td>400</td>
<td>4.0.x</td>
</tr>
<tr>
<td>410</td>
<td>4.1.x</td>
</tr>
</tbody>
</table>

By default, the library compatible with the detected version of the GNU* C++ compiler is used. Do not use this option if the GNU* C++ version is older than 3.2.

### Configuration Files

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

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

:name> = name of underlying compiler with spaces replaced by hyphens

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

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

Create a profile configuration file for setting the options for the profile library. Use the following naming convention:

<installdir>/etc/<profile_name>.conf

Use <profile_name> as a parameter to the -profile option for compiler drivers.

The following variables can be defined in the profile of the configuration file:

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 the file myprof.conf with the lines

PROFILE_PRELIB="-L<path_to_myprof>/lib -lmyprof"
PROFILE_INCPATHS="-I<paths_to_myprof>/include"

Use the command-line argument -profile=myprof for the relevant compile driver.

Environment Variables

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

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

CFLAGS=<flags>

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

LDFLAGS=<flags>

Set additional LDFLAGS to be used in the link step.

VT_ROOT=<path>

Set Intel® Trace Collector installation directory path.

IDB_HOME=<path>

Set Intel® Debugger installation directory path.

MPICC_PROFILE=<profile_name>

Specify a profile library. This has the same effect as if -profile=<profile_name> were used as an argument to mpicc.
Job Startup Commands

**mpiexec**

**Syntax**

\[
\text{mpiexec} \ <g\text{-options}> \ <l\text{-options}> \ <\text{executable}>
\]

or

\[
\text{mpiexec} \ <g\text{-options}> \ <l\text{-options}> \ <\text{executable}> : \ \\
\ <l\text{-options}> \ <\text{executable}>
\]

or

\[
\text{mpiexec} \ -\text{configfile} \ <\text{file}>
\]

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;g-options&gt;</code></td>
<td>Global options that apply to all MPI processes</td>
</tr>
<tr>
<td><code>&lt;l-options&gt;</code></td>
<td>Local options that apply to a single arg-set</td>
</tr>
<tr>
<td><code>&lt;executable&gt;</code></td>
<td><code>.a.out</code>, or path/name of executable, compiled with mpicc or any related command</td>
</tr>
<tr>
<td><code>&lt;file&gt;</code></td>
<td>File with command-line options</td>
</tr>
</tbody>
</table>

**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>`:

\[
\$ \text{mpiexec} \ -n \ <\# of processes> \ .a.out
\]

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:

\[
\$ \text{mpiexec} \ -n \ 2 \ -\text{host host1} \ .a.out : \\
\ -n \ 2 \ -\text{host host2} \ .b.out
\]

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

**NOTES**

- If `.`, " is not in the path on all nodes in the cluster, specify `<executable>` as `.a.out` rather than `a.out`.

**Global Options**

- `version` or `-V`

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

-perhost `<# of processes>`
Use this option to place the indicated number of consecutive MPI processes on every host.
The `mpiexec` command controls how the ranks of the processes are allocated to the nodes in the cluster. By default, `mpiexec` uses round-robin assignment of ranks to nodes. This placement algorithm may not be the best choice for your application, particularly for clusters with SMP nodes.
In order to change this default behavior, set the number of processes per host using the `-perhost` option, and set the total number of processes by using the `-n` option (see Local Options). Then the first `<# of processes>` indicated by the `-perhost` option is executed on the first host, the next `<# of processes>` is on the next one, and so on.
This is shorthand for using the multiple arg-sets that run the same number of processes on each indicated host. The `-perhost` option is equivalent to the second command-line syntax of the `mpiexec` command.

-machinefile `<machine file>`
Use this option to place the ranks of processes in compliance with the machine file. `<machine file>` has a list of host names, one per line. Use a short host name or a fully qualified domain name. Repeat the same host to match the number of processes executing on that host. You can use the following format to avoid repetition of the same host name: `<host name>:<number of processes>`. Blank lines and lines that start with `#' as the first character on the line are ignored.
Example:
The configuration file is as follows:
host1
host1
host2
host2
host3
Or, equivalently:
host1:2
host2:2
host3
Execute the following command to launch MPI application
`mpiexec -machinefile `<machine file>` -n `<# of processes>` ./a.out`

-genv `<ENVVAR>` `<value>`
Use this option to set the `<ENVVAR>` environment variable to the specified `<value>` for all MPI processes.
**-genvnone**

Use this option to suppress the propagation of any environment variables to any MPI processes. The default is to propagate the entire environment from which mpiexec was called.

**-g<l-option>**

Use this option to apply the named local option `<l-option>` globally. See section [Local Options](#Local_Options) for a list of all local options.

**-tv**

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

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

See also Section [Environment Variables](#Environment_Variables) for information on how to select the TotalView* executable file.

**-idb**

Use this option to run `<executable>` under the Intel® Debugger. For example:

```bash
$ mpiexec -idb -n <# of processes> <executable>
```

Include the installation path of the Intel Debugger in the `IDB_HOME` environment variable.

**-gdb**

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

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

**-gdba <jobid>**

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

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

**-l**

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

**-s <spec>**

Use this option to direct standard input to specified ranks.

<table>
<thead>
<tr>
<th><code>&lt;spec&gt;</code></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>Use all processes</td>
</tr>
<tr>
<td>m, n, k</td>
<td>Specify an exact list and use processes m, n, and k only. Default value is 0</td>
</tr>
<tr>
<td>k, l-m, n</td>
<td>Specify a range and use processes k, l thru m, and n</td>
</tr>
</tbody>
</table>

**-ifhn `<interface/hostname>`**

Use this option to specify the network interface which is used for communications with MPD on the local node. `<interface/hostname>` should be an IP address or a hostname associated with the alternative network interface.
-m
Use this option to merge output lines.

-a <alias>
Use this option to assign <alias> to the job.

-ecfn <filename>
Use this option to output xml exit codes to the file <filename>.

-ordered-output
Use this option to make sure that each line from all processes is printed correctly without mixing the lines with each other. This makes sense if your application prints a lot of output from each process.
This option affects output both from stdio and stderr.

NOTE
- At least one (the last) print operator MUST have the EOL “\n” character at the end.
  Otherwise, the application may hang because the stdio/stderr output gathering procedure waits for the ‘\n’ character to be sure the whole line has been received.

-trace [<libraries for preloading>]
Use this option to preload the indicated libraries to every process. The behavior of the option is the following:

1. If the -trace option is used without any parameters and MPIEXEC_TRACE_LIBS is not set, libVT.so will be preloaded (default case).
2. If the -trace option is used with a parameter, the library list for preloading will be taken from the option parameter.
3. If MPIEXEC_TRACE_LIBS is set and –trace is used without a parameter, MPIEXEC_TRACE_LIBS contains the list of libraries for preloading.
4. If MPIEXEC_TRACE_LIBS exists and –trace is used with a parameter, the environment variable is ignored and as in 2 above the library list for preloading is taken from the parameter.

If the option is used without any parameters, the libVT.so library will be preloaded. This is the default value.
The mpiexec command allows controlling the library preloading process in this case. It is not necessary to set the LD_PRELOAD environment variable to preload libraries. LD_PRELOAD is used to preload the ITC library to intercept MPI calls from each process. libVT.so is chosen as the default value in this case.

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.out on host host1 only:
$ mpiexec -n 2 -host host1 ./a.out

-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.

-umask <umask>
Use this option to perform the ’umask <umask>‘ command for the remote process.

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

-envnone
Use this option to suppress the propagation of any environment variables to the MPI processes in the current arg-set. The default is to propagate the entire environment from which mpiexec was called.

-envlist <list of env var names>
Use this option to pass a list of environment variables with their current values.

-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.out and b.out using the rdssm device over host1 and host2 respectively:
-host host1 -env I_MPI_DEBUG 2 -env I_MPI_DEVICE rdssm -n 2 ./a.out
-host host2 -env I_MPI_DEBUG 2 -env I_MPI_DEVICE rdssm -n 2 ./b.out
To launch the MPI application according to the above parameters, use:
mpiexec -configfile <filename>

Configuration Files
Create mpiexec configuration files using the following file-naming convention:

<installdir>/etc/mpiexec.conf
$HOME/.mpiexec.conf
$PWD/mpiexec.conf

Syntax
The format of the mpiexec.conf files is a free-format text containing default mpiexec command-line options. Blank lines and lines that start with ’#’ as the first character on the line are ignored.
Description
If these files exist, their contents are prepended to the command-line options for mpiexec in the following order:

1. System-wide <installdir>/etc/mpiexec.conf (if any)
2. User-specific $HOME/.mpiexec.conf (if any)
3. Session-specific $PWD/mpiexec.conf (if any)

This applies to all forms of the mpiexec command.

Use the mpiexec.conf files to specify the default options you will apply to all mpiexec commands. For example, to specify a default device, add the following line to the respective mpiexec.conf file:

-genv I_MPI_DEVICE <device>

Environment Variables

I_MPI_NETMASK
Choose the network interface for MPI communication over sockets.

Syntax
I_MPI_NETMASK=<arg>

Arguments

<table>
<thead>
<tr>
<th>&lt;arg&gt;</th>
<th>String parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;interface_name&gt;</td>
<td>Name of the network interface, usually the UNIX* driver name followed by the unit number</td>
</tr>
<tr>
<td>&lt;network_address&gt;</td>
<td>Network address. The trailing zero bits imply netmask</td>
</tr>
<tr>
<td>&lt;network_address/netmask&gt;</td>
<td>Network address. The &lt;netmask&gt; value specifies the netmask length</td>
</tr>
<tr>
<td>&lt;list of interfaces&gt;</td>
<td>A colon separated list of network addresses and interface names</td>
</tr>
</tbody>
</table>

Description
Set this variable to choose the network interface for MPI communication over sockets. If a list of interfaces is specified, the first available interface on the node is used for communication.

For example:

Specify <interface_name>:
export I_MPI_NETMASK=ib0

Specify <network_address>:
export I_MPI_NETMASK=192.169.0.0 (imply netmask 255.255.0.0)

Specify <network_address/netmask>:
export I_MPI_NETMASK=192.169.0.5/24

Specify <list of interfaces>:
export I_MPI_NETMASK=192.169.0.5/24:ib0:192.169.0.0

MPIEXEC_TIMEOUT
Set the mpiexec timeout.
Syntax
MPIEXEC_TIMEOUT=<timeout>

Arguments

<table>
<thead>
<tr>
<th>&lt;timeout&gt;</th>
<th>Define mpiexec timeout period in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>≥0</td>
<td>The default timeout value is zero corresponding to no timeout</td>
</tr>
</tbody>
</table>

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.

NOTES
- Set the MPIEXEC_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.

MPIEXEC_TIMEOUT_SIGNAL
Define a signal number.

Syntax
MPIEXEC_TIMEOUT_SIGNAL=<number>

Arguments

<table>
<thead>
<tr>
<th>&lt;number&gt;</th>
<th>A number of the signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 0</td>
<td>The default value is 9 (SIGKILL)</td>
</tr>
</tbody>
</table>

Description
Define a signal number for killing the processes of the task if the timeout pointed to by MPIEXEC_TIMEOUT is over. If a wrong signal number not supported by the system is set, mpiexec prints a warning message and continues task termination using the default signal number 9 (SIGKILL).

MPIEXEC_SIGNAL_PROPAGATION
Control signal propagation.

Syntax
MPIEXEC_SIGNAL_PROPAGATION=<arg>

Arguments

<table>
<thead>
<tr>
<th>&lt;arg&gt;</th>
<th>Binary indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable, yes, on, 1</td>
<td>Turn on propagation. This is the default value</td>
</tr>
<tr>
<td>disable, no, off, 0</td>
<td>Turn off propagation</td>
</tr>
</tbody>
</table>

Description
Set this variable to control signal propagation. If it is turned on the signal is applied to all processes of the task. If signal propagation is disabled, the only process with rank #0 is killed with the given signal. The remaining processes are killed with the default signal 9 (SIGKILL).
NOTES

- MPIEXEC_TIMEOUT_SIGNAL and MPIEXEC_SIGNAL_PROPAGATION can work independently.

I_MPI_PMI_EXTENSIONS

Turn on/off the use of the PMI extensions.

Syntax

I_MPI_PMI_EXTENSIONS=<arg>

Arguments

<table>
<thead>
<tr>
<th>&lt;arg&gt;</th>
<th>Binary indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable, yes, on, 1</td>
<td>Turn on the PMI extensions</td>
</tr>
<tr>
<td>disable, no, off, 0</td>
<td>Turn off the PMI extensions</td>
</tr>
</tbody>
</table>

Description

Use this variable to enable the use of the Intel MPI PMI extensions.

The extensions substantially decrease task startup time but they must be handled by the process manager. Set <arg> to disable if your process manager does not support these extensions.

NOTES

- Use the enable value with caution. Do not set it if your process manager does not support the PMI extensions. Otherwise, your application will not run.

I_MPI_PMI_FAST_STARTUP

Turn on/off the new internal mpd algorithm intended for faster application startup.

Syntax

I_MPI_PMI_FAST_STARTUP=<arg>

Arguments

<table>
<thead>
<tr>
<th>&lt;arg&gt;</th>
<th>Binary indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable, yes, on, 1</td>
<td>Turn on the algorithm for fast PMI startup. This is the default value</td>
</tr>
<tr>
<td>disable, no, off, 0</td>
<td>Turn off the algorithm for fast PMI startup</td>
</tr>
</tbody>
</table>

Description

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

I_MPI_DEVICE

Select the particular network fabric to be used.

Syntax

I_MPIDEVICE=<device>[:<provider>]

I_MPI_PMI_EXTENSIONS

Turn on/off the use of the PMI extensions.

Syntax

I_MPI_PMI_EXTENSIONS=<arg>

Arguments

<table>
<thead>
<tr>
<th>&lt;arg&gt;</th>
<th>Binary indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable, yes, on, 1</td>
<td>Turn on the PMI extensions</td>
</tr>
<tr>
<td>disable, no, off, 0</td>
<td>Turn off the PMI extensions</td>
</tr>
</tbody>
</table>

Description

Use this variable to enable the use of the Intel MPI PMI extensions.

The extensions substantially decrease task startup time but they must be handled by the process manager. Set <arg> to disable if your process manager does not support these extensions.

NOTES

- Use the enable value with caution. Do not set it if your process manager does not support the PMI extensions. Otherwise, your application will not run.

I_MPI_PMI_FAST_STARTUP

Turn on/off the new internal mpd algorithm intended for faster application startup.

Syntax

I_MPI_PMI_FAST_STARTUP=<arg>

Arguments

<table>
<thead>
<tr>
<th>&lt;arg&gt;</th>
<th>Binary indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable, yes, on, 1</td>
<td>Turn on the algorithm for fast PMI startup. This is the default value</td>
</tr>
<tr>
<td>disable, no, off, 0</td>
<td>Turn off the algorithm for fast PMI startup</td>
</tr>
</tbody>
</table>

Description

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

I_MPI_DEVICE

Select the particular network fabric to be used.

Syntax

I_MPIDEVICE=<device>[:<provider>]
Arguments

<table>
<thead>
<tr>
<th>&lt;device&gt;</th>
<th>One of {sock,shm,ssm}</th>
</tr>
</thead>
<tbody>
<tr>
<td>sock</td>
<td>Sockets</td>
</tr>
<tr>
<td>shm</td>
<td>Shared-memory only (no sockets)</td>
</tr>
<tr>
<td>ssm</td>
<td>Combined sockets + shared memory (for clusters with SMP nodes)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&lt;device&gt;</th>
<th>One of {rdma,rdssm}</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;provider&gt;</td>
<td>Optional DAPL* provider name</td>
</tr>
<tr>
<td>rdma</td>
<td>RDMA-capable network fabrics including InfiniBand*, Myrinet* (via DAPL*)</td>
</tr>
<tr>
<td>rdssm</td>
<td>Combined sockets + shared memory + DAPL* (for clusters with SMP nodes and RDMA-capable network fabrics)</td>
</tr>
</tbody>
</table>

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

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

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

```
$ mpiexec -n <# of processes> -env I_MPI_DEVICE rdssm:OpenIB-cma <executable>
```

For these devices, if <provider> is not specified, the first DAPL* provider in /etc/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.

NOTES

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

<table>
<thead>
<tr>
<th>&lt;arg&gt;</th>
<th>Binary indicator</th>
</tr>
</thead>
</table>
Fall back to the shared memory and/or socket fabrics if initialization of the DAPL* fabric fails. This is the default value

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

<table>
<thead>
<tr>
<th>&lt;level&gt;</th>
<th>Indicate level of debug information provided</th>
</tr>
</thead>
<tbody>
<tr>
<td>(unset)</td>
<td>Print no debugging information</td>
</tr>
<tr>
<td>1</td>
<td>Print warnings if specified I_MPI_DEVICE could not be used</td>
</tr>
<tr>
<td>2</td>
<td>Confirm which I_MPI_DEVICE was used</td>
</tr>
<tr>
<td>&gt; 2</td>
<td>Add extra levels of debug information</td>
</tr>
</tbody>
</table>

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 -n <# of processes> -env I_MPI_DEBUG +2 ./a.out
```

produces output debug messages in the following format:

I_MPI: [rank#pid@hostname]Debug message

NOTES

- Compiling with mpicc -g causes considerable amounts of additional debug information to be printed.
**I_MPI_DAT_LIBRARY**
Select the particular DAT library to be used.

**Syntax**

```plaintext
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.

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

**TOTALVIEW**
Select the particular TotalView* executable file to use.

**Syntax**

```plaintext
TOTALVIEW=<path>
```

**Arguments**

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

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

**MPD Daemon Commands**

**mpdboot**

**Syntax**

```plaintext
mpdboot [ -n <#nodes> ] [ -f <hostsfile> ] [ -h ] [ -r <rshcmd> ] \
[ -u <user> ] [ -m <mpdcmd> ] [ --loccons ] [ --remcons ] \
[ -s ] [ -d ] [ -v ] [ -l ] [ --ncpus=<ncpus> ] [ -o ]
```

or

```plaintext
mpdboot [ --totalnum=<#nodes> ] [ --file=<hostsfile> ] [ --help ] \
[ --rsh=<rshcmd> ] [ --user=<user> ] [ --mpd=<mpdcmd> ] \
[ --loccons ] [ --remcons ] [ --shell ] [ --debug ] \
[ --verbose ] [ -l ] [ --ncpus=<ncpus> ] [ --ordered ]
```

**Arguments**

| -h, --help | Display a help message |
| -d, --debug | Print debug information |
| -v, --verbose | Print extra verbose information. Show the <rshcmd> attempts |
**Description**

Start mpd daemons on the specified number of nodes by providing a list of node names in `<mpd.hosts>`. The mpd daemons are started using the `rsh` command by default. If the `rsh` connectivity is not enabled, use the `-r ssh` option to switch over to `ssh`. Make sure that all nodes in the cluster can connect to each other via the `rsh` command without a password or, if the `-r ssh` option is used, via the `ssh` command without a password.

**NOTES**

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

**mpd**

**Syntax**

```markdown
mpd [ --help ] [ --host=<host> --port=<portnum> ] [ --noconsole ] \ 
[ --trace ] [ --echo ] [ --daemon ] [ --bulletproof ] \ 
[ --ifhn <interface/hostname> ] [ --listenport <listenport> ]
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ --help ]</td>
<td>Display a help message</td>
</tr>
<tr>
<td>[ -h &lt;host&gt; -p &lt;portnum&gt;]</td>
<td>Specify the host and port to be used for entering an existing ring. The --host and --port options must be specified together</td>
</tr>
<tr>
<td>[ -n ]</td>
<td>Do not create console at startup</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-n</td>
<td>Number of nodes in <code>mpd.hosts</code> on which daemons are started</td>
</tr>
<tr>
<td>--totalnum=&lt;#nodes&gt;</td>
<td>Specify remote shell to start daemons and jobs. <code>rsh</code> is the default value</td>
</tr>
<tr>
<td>-r</td>
<td>Specify the full path name of <code>mpd</code> on the remote hosts</td>
</tr>
<tr>
<td>--rsh=&lt;rshcmd&gt;</td>
<td>Path/name of the file that has the list of machine names on which daemons are started</td>
</tr>
<tr>
<td>-f</td>
<td>Remote shell to start daemons and jobs. <code>rsh</code> is the default value</td>
</tr>
<tr>
<td>--file=&lt;hostsfile&gt;</td>
<td>Specify shell</td>
</tr>
<tr>
<td>-l</td>
<td>Remove the restriction of starting only one mpd per machine</td>
</tr>
<tr>
<td>-m</td>
<td>Specify shell</td>
</tr>
<tr>
<td>--mpd=&lt;mpdcmd&gt;</td>
<td>Specify user</td>
</tr>
<tr>
<td>-u</td>
<td>Do not create local MPD consoles</td>
</tr>
<tr>
<td>--user=&lt;user&gt;</td>
<td>Do not create remote MPD consoles</td>
</tr>
<tr>
<td>--ncpus=&lt;ncpus&gt;</td>
<td>Indicate how many processors to use on the local machine (other nodes are listed in the hosts file)</td>
</tr>
<tr>
<td>-o, --ordered</td>
<td>Start all the mpd daemons in the exact order as specified in the <code>mpd.hosts</code> file</td>
</tr>
</tbody>
</table>

---
### MPD Command Line Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--noconsole</td>
<td>Print internal MPD trace information</td>
</tr>
<tr>
<td>-t</td>
<td>Print internal MPD trace information</td>
</tr>
<tr>
<td>--trace</td>
<td>Print a port number at startup to which other mpds may connect</td>
</tr>
<tr>
<td>-e</td>
<td>Print a port number at startup to which other mpds may connect</td>
</tr>
<tr>
<td>--echo</td>
<td>Print a port number at startup to which other mpds may connect</td>
</tr>
<tr>
<td>-d</td>
<td>Start mpd in daemon mode</td>
</tr>
<tr>
<td>--daemon</td>
<td>Start mpd in daemon mode</td>
</tr>
<tr>
<td>--bulletproof</td>
<td>Turn MPD bulletproofing on</td>
</tr>
<tr>
<td>--ifhn=&lt;interface/hostname&gt;</td>
<td>Specify &lt;interface/hostname&gt; to use for MPD communications</td>
</tr>
<tr>
<td>-l &lt;listenport&gt;</td>
<td>Specify the mpd listening port</td>
</tr>
<tr>
<td>--listenport=&lt;listenport&gt;</td>
<td>Specify the mpd listening port</td>
</tr>
</tbody>
</table>

### Description

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

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

is equivalent to

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

The file named `.mpd.conf` must be present in the user's home directory with read and write access only for the user, and must minimally contain a line with `secretword=<secretword>`. To run `mpd` as root create the `mpd.conf` file in the `/etc` directory instead of `.mpd.conf` in the root's home directory. We do not recommend starting the MPD ring under the root account.

### mpdtrace

Determine whether mpd is running.

**Syntax**

```
mpdtrace [-l]
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-l</td>
<td>Show MPD identifiers instead of the hostnames</td>
</tr>
</tbody>
</table>

**Description**

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

### mpdlistjobs

List the running processes for a particular set of MPI jobs.

**Syntax**

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

or

```
mpdlistjobs [--user=<username>] [--alias=<jobalias>]
[ --jobid=<jobid> ]
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-u &lt;username&gt;</td>
<td>List jobs of a particular user</td>
</tr>
<tr>
<td>-a &lt;jobalias&gt;</td>
<td>List information about the particular job specified by &lt;jobalias&gt;</td>
</tr>
<tr>
<td>-j &lt;jobid&gt;</td>
<td>List information about the particular job specified by &lt;jobid&gt;</td>
</tr>
</tbody>
</table>

Description

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

**mpdkilljobs**

Kill a job.

Syntax

```
mpdkilljobs [ <jobnum> ] [ -a <jobalias> ]
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;jobnum&gt;</td>
<td>Kill the job specified by &lt;jobnum&gt;</td>
</tr>
<tr>
<td>-a &lt;jobalias&gt;</td>
<td>Kill the job specified by &lt;jobalias&gt;</td>
</tr>
</tbody>
</table>

Description

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

**mpdringtest**

Determine how much time is required for a ring loading.

Syntax

```
mpdringtest [ number of loops ]
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of loops</td>
<td>Number of loops</td>
</tr>
</tbody>
</table>

Description

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

**mpdexit**

Shut down a single mpd daemon.

Syntax

```
mpdexit <mpdid>
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;mpdid&gt;</td>
<td>Specify the mpd daemon to kill</td>
</tr>
</tbody>
</table>
**Description**
Use this command to cause the single **mpd** daemon to exit. Use `<mpdid>` obtained via the `mpdtrace -l` command in the form `<hostname>_<port number>`.

**mpdallexit**
Shut down all **mpd** daemons on all nodes.

**Arguments**
This command takes no arguments.

**Description**
Use this command to shutdown all **mpd** rings.

**mpdcleanup**

**Syntax**
```bash
mpdcleanup  
[-f <hostsfile>]  
[-r <rshcmd>]  
[-u <user>]  
[-c <cleancmd>]  
```
or

```bash
mpdcleanup  
[--file=<hostsfile>]  
[--rsh=<rshcmd>]  
[--user=<user>]  
[--clean=<cleancmd>]  
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-f &lt;hostsfile&gt;</code></td>
<td>Specify the file containing a list of machines to clean up</td>
</tr>
<tr>
<td><code>--file=&lt;hostsfile&gt;</code></td>
<td>Specify the file containing a list of machines to clean up</td>
</tr>
<tr>
<td><code>-r &lt;rshcmd&gt;</code></td>
<td>Specify the remote shell to use</td>
</tr>
<tr>
<td><code>--rsh=&lt;rshcmd&gt;</code></td>
<td>Specify the remote shell to use</td>
</tr>
<tr>
<td><code>-u &lt;user&gt;</code></td>
<td>Specify the user</td>
</tr>
<tr>
<td><code>--user=&lt;user&gt;</code></td>
<td>Specify the user</td>
</tr>
<tr>
<td><code>-c &lt;cleancmd&gt;</code></td>
<td>Specify the command to use for removing the UNIX* socket. The default command is <code>/bin/rm -f</code></td>
</tr>
<tr>
<td><code>--clean=&lt;cleancmd&gt;</code></td>
<td>Specify the command to use for removing the UNIX* socket. The default command is <code>/bin/rm -f</code></td>
</tr>
</tbody>
</table>

**Description**
Use this command to remove the UNIX* socket on local and remote machines.

**mpdsigjob**
Apply a signal to a process running an application.

**Syntax**
```bash
mpdsigjob sigtype [-j <jobid> | -a <jobalias>] [ -s | -g ]
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sigtype</code></td>
<td>Specify the signal to send</td>
</tr>
<tr>
<td><code>-a &lt;jobalias&gt;</code></td>
<td>Send a signal to the job specified by <code>&lt;jobalias&gt;</code></td>
</tr>
<tr>
<td><code>-j &lt;jobid&gt;</code></td>
<td>Send a signal to the job specified by <code>&lt;jobid&gt;</code></td>
</tr>
<tr>
<td><code>-s</code></td>
<td>Deliver a signal to the single user process</td>
</tr>
</tbody>
</table>
-g

| Deliver a signal to the group of processes. This is the default behavior |

**Description**

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

**mpdhelp**

**Syntax**

mpdhelp

**Arguments**

This command takes no arguments.

**Description**

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

**Configuration Files**

**$HOME/.mpd.conf**

This file contains the mpd daemon password and is required before setting up the mpd daemons. Use it to control access to the daemons by various Intel MPI Library users.

**Syntax**

The file has a single line:

secretword=<mpd password>

or

MPD_SECRETWORD=<mpd password>

**Description**

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

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

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

**NOTES**

- MPD_SECRETWORD is a synonym for secretword.

**mpd.hosts**

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

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

**Syntax**

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

PATH
Ensure that the PATH settings include the path to mpdboot and other mpd daemon commands.

NOTES
- The <installdir>/bin directory (<installdir>/bin64 directory for Intel® 64 in 64-bit mode) and the path to Python* version 2.2 or higher must be included in PATH in order for the mpd daemon commands to succeed.

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

Syntax
MPD_CON_EXT=<tag>

Arguments
| <tag> | Unique MPD identifier |

Description
Set this variable to different unique values to allow several mpd rings to co-exist. Each mpd ring is associated with a separate MPD_CON_EXT value. Once this variable is set, you can start one mpd ring and work with it without affecting other available mpd rings. Set the appropriate MPD_CON_EXT value to work with a particular mpd ring.

Normally, every new mpd ring completely replaces the older one but setting the appropriate MPD_CON_EXT value allows simultaneous existence of several mpd rings.

See section Simplified Job Startup Command to learn about an easier way to run several Intel MPI Library jobs at once.

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

Syntax
I_MPI_MPD_CONF=<path/name>

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

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

Simplified Job Startup Command

mpirun

Syntax
mpirun [ <mpdboot options> ] <mpiexec options>
Arguments

<table>
<thead>
<tr>
<th><code>&lt;mpdboot options&gt;</code></th>
<th><code>mpdboot</code> options as described in the <code>mpdboot</code> section above, except <code>-n</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;mpiexec options&gt;</code></td>
<td><code>mpiexec</code> options as described in the <code>mpiexec</code> section above</td>
</tr>
</tbody>
</table>

Description

Use this command to start an independent ring of `mpd` daemons, launch an MPI job, and shut down the `mpd` ring upon job termination.

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

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

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

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

The `mpirun` command also detects if the MPI job is submitted in a session allocated using a job scheduler like Torque*, PBS Pro*, LSF* or ParallelNavi* NQS*. In this case, the `mpirun` command extracts the host list from the respective environment and uses these nodes automatically according to the above scheme.

In such instances, you do not have to create the `mpd.hosts` file yourself. Just allocate the session you need using the particular job scheduler installed on your system, and use the `mpirun` command inside this session to run your MPI job.

See the product `Release Notes` for a complete list of the supported job schedulers.
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.

### Process Pinning

**I_MPI_PIN_MODE**

**I_MPI_PIN_PROCS**

Pin processes to the CPUs to prevent undesired process migration. Process pinning is performed if the operating system provides the necessary kernel interfaces.

#### Syntax

```
I_MPI_PIN_MODE=<pinmode>
I_MPI_PIN_PROCS=<proclist>
```

#### Arguments

<table>
<thead>
<tr>
<th>&lt;pinmode&gt;</th>
<th>Select CPU pinning mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpd, enable, yes, on, 1</td>
<td>Pin processes inside MPD. Default on SGI* Altix* platform</td>
</tr>
<tr>
<td>lib</td>
<td>Pin processes inside MPI library. Default on other platforms</td>
</tr>
<tr>
<td>disable, no, off, 0</td>
<td>Do not pin processes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&lt;proclist&gt;</th>
<th>Define mapping of process to CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>Use all CPUs</td>
</tr>
<tr>
<td>allcores</td>
<td>Use all CPU cores (or physical CPU). This is the default value</td>
</tr>
<tr>
<td>n</td>
<td>Use only CPU number n (where n is 0,1, ... , or total number of CPUs - 1)</td>
</tr>
<tr>
<td>m-n</td>
<td>Use CPUs from m to n</td>
</tr>
<tr>
<td>k,l-m,n</td>
<td>Use CPUs k, l thru m, and n</td>
</tr>
</tbody>
</table>

#### Description

Set these variables to enable and control process pinning. The Intel MPI Library pins the process to the CPU by default. It assumes the allcores settings for the I_MPI_PIN_PROCS variable. The mpd pinning mode is used on SGI* Altix*. The lib pinning mode is used on other platforms. Set the variable I_MPI_PIN_MODE to lib to make the Intel MPI Library pin the processes. The already running MPI process pinned to CPU. There is no chance to co-locate the process CPU and its memory.

Set the variable I_MPI_PIN_MODE to mpd to make the mpd daemon pin processes via system specific means, if they are available. The pinning is done before the MPI process is started. It is
possible to co-locate the process CPU and memory in this case. This pinning method has an advantage on the system with Non-Uniform memory Architecture (NUMA) like SGI* Altix*. Under NUMA, a processor can access its own local memory faster than non-local memory. To avoid additional overhead on the memory operations MPI process should be located in the local memory of the processor.

Set the I_MPI_PIN_PROCS variable to define the set of processors.

If no set of CPUs is defined in the system, the number and order of the processors correspond to the output of the cat /proc/cpuinfo command. If a CPU set is defined in the system, the I_MPI_PIN_PROCS value refers to the logical processors enabled in the current process set. 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_PROCS 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_PROCS 0,3 -n <# of processes> \ <executable> : -host host2 -env I_MPI_PIN_PROCS 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_PROCS 0,3 \ -n <# of processes> <executable> : -host host2 -env I_MPI_PIN_PROCS 1,2,3 \ -n <# of processes> <executable>

NOTES

\ o The values of I_MPI_PIN_MODE and I_MPI_PIN_PROCS can be defined locally, on a per host level, or globally, for the entire system.

Device Control

I_MPI_SPIN_COUNT
Control the spin count value.

Syntax

I_MPI_SPIN_COUNT=<scount>

Arguments

<table>
<thead>
<tr>
<th>&lt;scount&gt;</th>
<th>Define the loop spin count when polling fabric(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 0</td>
<td>The default &lt;scount&gt; value is equal to 1 time for sock, shm, and ssm devices, and equal to 250 times for rdma and rdssm devices</td>
</tr>
</tbody>
</table>

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. A smaller value for <scount> causes 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.

**NOTES**
- 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_EAGER_THRESHOLD**
Change the eager/rendezvous cutover point for all devices.

**Syntax**

```
I_MPI_EAGER_THRESHOLD=<nbytes>
```

**Arguments**

<table>
<thead>
<tr>
<th><code>&lt;nbytes&gt;</code></th>
<th>Define eager/rendezvous cutover point</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 0</td>
<td>The default <code>&lt;nbytes&gt;</code> value is equal to 262 144 bytes</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th><code>&lt;nbytes&gt;</code></th>
<th>Define threshold for DAPL* intranode communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 0</td>
<td>The default <code>&lt;nbytes&gt;</code> value is equal to 262 144 bytes</td>
</tr>
</tbody>
</table>

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

**Syntax**

```
I_MPI_SHM_PROC_THRESHOLD=<nproc>
```
Arguments

<table>
<thead>
<tr>
<th>&lt;nproc&gt;</th>
<th>Define static/dynamic mode switch point for the shm device</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 0, &lt; 90</td>
<td>The default &lt;nproc&gt; value is equal to 90</td>
</tr>
</tbody>
</table>

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.

NOTES

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

SOCK Device Control

I_MPI_WAIT_MODE

Turn on/off a wait mode.

Syntax

I_MPI_WAIT_MODE=<arg>

Arguments

<table>
<thead>
<tr>
<th>&lt;arg&gt;</th>
<th>Binary indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable, yes, on, 1</td>
<td>Turn on the wait mode</td>
</tr>
<tr>
<td>disable, no, off, 0</td>
<td>Turn off the wait mode. This is the default value</td>
</tr>
</tbody>
</table>

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.

NOTES

- The wait mode supports the sock device only.

RDMA and RDSSM Device Control

RDMA_IBA_EAGER_THRESHOLD

Change the eager/rendezvous cutover point.

Syntax

RDMA_IBA_EAGER_THRESHOLD=<nbytes>
### NUM_RDMA_BUFFER

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

#### Syntax

```
NUM_RDMA_BUFFER=<nbuf>
```

#### Arguments

<table>
<thead>
<tr>
<th><code>&lt;nbuf&gt;</code></th>
<th>Define the number of buffers for each pair in a process group</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 0</td>
<td>The default <code>&lt;nbuf&gt;</code> value ranges between 8 and 40 depending on the cluster size and platform</td>
</tr>
</tbody>
</table>

#### Description

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

#### NOTES

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

### I_MPI_RDMA_VBUF_TOTAL_SIZE

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

#### Syntax

```
I_MPI_RDMA_VBUF_TOTAL_SIZE=<nbytes>
```

#### Arguments

<table>
<thead>
<tr>
<th><code>&lt;nbytes&gt;</code></th>
<th>Define the size of pre-registered buffers</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 0</td>
<td>The default <code>&lt;nbytes&gt;</code> value is equal to 16 640 bytes</td>
</tr>
</tbody>
</table>

#### 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_TWO_PHASE_BUF_ENLARGEMENT**

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

**Syntax**

```
I_MPI_TWO_PHASE_BUF_ENLARGEMENT=<arg>
```

**Arguments**

<table>
<thead>
<tr>
<th>&lt;arg&gt;</th>
<th>Binary indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable, yes, on, 1</td>
<td>Turn on the mode of using two-phase buffer enlargement</td>
</tr>
<tr>
<td>disable, no, off, 0</td>
<td>Turn off the mode of using two-phase buffer enlargement. This is the default value</td>
</tr>
</tbody>
</table>

**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_SHORT_BUF_THRESHOLD`.
- Two-phase buffer enlargement is turned off by default.

**I_MPI_RDMA_SHORT_BUF_THRESHOLD**

Change threshold for two-phase buffer enlargement mode.

**Syntax**

```
I_MPI_RDMA_SHORT_BUF_THRESHOLD=<nbytes>
```

**Arguments**

<table>
<thead>
<tr>
<th>&lt;nbytes&gt;</th>
<th>Define the threshold for starting enlargement of the RDMA buffers</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 0</td>
<td>The default value is 580</td>
</tr>
</tbody>
</table>

**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_TWO_PHASE_BUF_ENLARGEMENT` is enabled.

**I_MPI_RDMA_TRANSLATION_CACHE**

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

**Syntax**

```
I_MPI_RDMA_TRANSLATION_CACHE=<arg>
```

**Arguments**

<table>
<thead>
<tr>
<th>&lt;arg&gt;</th>
<th>Binary indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable, yes, on, 1</td>
<td>Turn on the memory registration cache. This is the default value</td>
</tr>
<tr>
<td>disable, no, off, 0</td>
<td>Turn off the memory registration cache</td>
</tr>
</tbody>
</table>

**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_USE_RENDEZVOUS_RDMA_WRITE**

Turn on/off the rendezvous RDMA Write protocol.

**Syntax**

\[ I\_MPI\_USE\_REND\_RDMA\_WRITE=\langle\text{arg}\rangle \]

**Arguments**

<table>
<thead>
<tr>
<th>&lt;arg&gt;</th>
<th>Binary indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable, yes, on, 1</td>
<td>Turn on the RDMA Write rendezvous protocol</td>
</tr>
<tr>
<td>disable, no, off, 0</td>
<td>Turn off the RDMA Write rendezvous protocol. This is the default value (the RDMA Read protocol is used instead)</td>
</tr>
</tbody>
</table>

**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_USE_EVD_FALLBACK**

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

**Syntax**

\[ I\_MPI\_RDMA\_USE\_EVD\_FALLBACK=\langle\text{arg}\rangle \]

**Arguments**

<table>
<thead>
<tr>
<th>&lt;arg&gt;</th>
<th>Binary indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable, yes, on, 1</td>
<td>Turn on the EVD based fallback</td>
</tr>
<tr>
<td>disable, no, off, 0</td>
<td>Turn off the EVD based fallback. This is the default value</td>
</tr>
</tbody>
</table>

**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.

**NOTES**

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

**I_MPI_USE_DYNAMIC_CONNECTIONS**

Turn on/off the dynamic connection establishment.

**Syntax**

\[ I\_MPI\_USE\_DYN\_CON\_ECTIONS=\langle\text{arg}\rangle \]
Arguments

<table>
<thead>
<tr>
<th>&lt;arg&gt;</th>
<th>Binary indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable, yes, on, 1</td>
<td>Turn on the dynamic connection establishment. This is the default value</td>
</tr>
<tr>
<td>disable, no, off, 0</td>
<td>Turn off the dynamic connection establishment</td>
</tr>
</tbody>
</table>

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_CONNECTIONS_MODE**

Choose the algorithm for establishing of the DAPL* connections.

Syntax

```
I_MPI_DYNAMIC_CONNECTIONS_MODE=<arg>
```

Arguments

<table>
<thead>
<tr>
<th>&lt;arg&gt;</th>
<th>Mode selector</th>
</tr>
</thead>
<tbody>
<tr>
<td>reject</td>
<td>Deny one of the two simultaneous connection requests. This is the default value</td>
</tr>
<tr>
<td>disconnect</td>
<td>Deny one of the two simultaneous connection requests after both connections have been established</td>
</tr>
</tbody>
</table>

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_USE_DAPL_INTRANODE**

Turn on/off the DAPL* intranode communication mode.

Syntax

```
I_MPI_USE_DAPL_INTRANODE=<arg>
```

Arguments

<table>
<thead>
<tr>
<th>&lt;arg&gt;</th>
<th>Binary indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable, yes, on, 1</td>
<td>Turn on the DAPL* intranode communication</td>
</tr>
<tr>
<td>disable, no, off, 0</td>
<td>Turn off the DAPL* intranode communication. This is the default value</td>
</tr>
</tbody>
</table>
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.

NOTES
- 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_CONN_EVD_QLEN
Define the event queue size of the DAPL* event dispatcher.

Syntax
I_MPI_CONN_EVD_QLEN=<size>

Arguments

<table>
<thead>
<tr>
<th>&lt;size&gt;</th>
<th>Define the length of the event queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 0</td>
<td>The default value is queried from the DAPL provider</td>
</tr>
</tbody>
</table>

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.

I_MPI_DAPL_CHECK_MAX_RDMA_SIZE
Control the message fragmentation based on the value of the DAPL* attribute `max_rdma_size`.

Syntax
I_MPI_DAPL_CHECK_MAX_RDMA_SIZE=<arg>

Arguments

<table>
<thead>
<tr>
<th>&lt;arg&gt;</th>
<th>Binary indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable, yes, on, 1</td>
<td>Allow message fragmentation</td>
</tr>
<tr>
<td>disable, no, off, 0</td>
<td>Forbid message fragmentation. This is the default value</td>
</tr>
</tbody>
</table>

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

- If set to `disable`, the Intel MPI Library never fragments the messages to maximize performance.
- If set to `enable`, the Intel MPI Library fragments messages of size greater than the value of the DAPL* attribute `max_rdma_size`. 
Collective Operation Control

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

<table>
<thead>
<tr>
<th>&lt;arg&gt;</th>
<th>Binary indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable, yes, on, 1</td>
<td>Fast collective algorithms are used. This is the default value</td>
</tr>
<tr>
<td>disable, no, off, 0</td>
<td>Slower and safer collective algorithms are used</td>
</tr>
</tbody>
</table>

Description
The Intel MPI Library uses by default advanced collective algorithms designed for better application performance. 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.

NOTES
- 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
I_MPI_BCAST_MSG
Control MPI_Bcast algorithm thresholds.

Syntax
I_MPI_BCAST_NUM_PROCS=<nproc>
I_MPI_BCAST_MSG=<nbytes1,nbytes2>

Arguments

<table>
<thead>
<tr>
<th>&lt;nproc&gt;</th>
<th>Define the number of processes threshold for choosing the MPI_Bcast algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 0</td>
<td>The default value is 8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&lt;nbytes1,nbytes2&gt;</th>
<th>Define the message size threshold range (in bytes) for choosing the MPI_Bcast algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 0</td>
<td>The default pair of values is 12288,524288</td>
</tr>
<tr>
<td>nbytes2 &gt;= nbytes1</td>
<td></td>
</tr>
</tbody>
</table>
Description
Set these variables to control the selection of the three possible MPI_Bcast algorithms according to the following scheme:

1. The first algorithm is selected if the message size is less than `<nbytes1>`, or the number of processes in the operation is less than `<nproc>`.
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**

**I_MPI_ALLTOALL_MSG**
Control MPI_Alltoall algorithm thresholds.

**Syntax**

- I_MPI_ALLTOALL_NUM_PROCS=<nproc>
- I_MPI_ALLTOALL_MSG=<nbytes1,nbytes2>

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;nproc&gt;</code></td>
<td>Define the number of processes threshold for choosing the MPI_Alltoall algorithm</td>
</tr>
<tr>
<td>&gt; 0</td>
<td>The default value is 8</td>
</tr>
<tr>
<td><code>&lt;nbytes1,nbytes2&gt;</code></td>
<td>Defines the message size threshold range (in bytes) for choosing the MPI_Alltoall algorithm</td>
</tr>
<tr>
<td>&gt; 0</td>
<td>nbytes2 &gt;= nbytes1 The default pair of values is 256,32768</td>
</tr>
</tbody>
</table>

**Description**
Set these variables to control the selection of the three possible MPI_Alltoall algorithms according to the following scheme:

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**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;nbytes1,nbytes2&gt;</code></td>
<td>Define the message size threshold range (in bytes) for choosing the MPI_Allgather algorithm</td>
</tr>
<tr>
<td>&gt; 0</td>
<td>The default pair of values is 81920,524288</td>
</tr>
</tbody>
</table>
Set this variable to control the selection of the three possible `MPI_Allgather` algorithms according to the following scheme:

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

<table>
<thead>
<tr>
<th><code>&lt;nbytes&gt;</code></th>
<th>Define the message size threshold (in bytes) for choosing the <code>MPI_Allreduce</code> algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 0</td>
<td>The default value is 2048</td>
</tr>
</tbody>
</table>

#### Description
Set this variable to control the selection of the two possible `MPI_Allreduce` algorithms according to the following scheme:

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_REDScatid_MSG
Control the `MPI_Reduce_scatter` algorithm thresholds.

#### Syntax

```
I_MPI_REDScatid_MSG=<nbytes1,nbytes2>
```

#### Arguments

<table>
<thead>
<tr>
<th><code>&lt;nbytes1,nbytes2&gt;</code></th>
<th>Define the message size threshold range (in bytes) for choosing the <code>MPI_Reduce_scatter</code> algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 0</td>
<td>The default pair of values is 512,524288</td>
</tr>
</tbody>
</table>

#### Description
Set this variable to control the selection of the three possible `MPI_Reduce_scatter` algorithms according to the following scheme:

1. The first algorithm is selected if the reduction operation is commutative and the message size is less than `<nbytes2>`.  

nbytes2 >= nbytes1
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**

\[
\text{I_MPI_SCATTER_MSG} = <nbytes>
\]

**Arguments**

<table>
<thead>
<tr>
<th>&lt;nbytes&gt;</th>
<th>Define the buffer size threshold range (in bytes) for choosing the MPI_Scatter algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 0</td>
<td>The default value is 2048</td>
</tr>
</tbody>
</table>

**Description**

Set this variable to control the selection of the two possible MPI_Scatter algorithms according to the following scheme:

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**

\[
\text{I_MPI_GATHER_MSG} = <nbytes>
\]

**Arguments**

<table>
<thead>
<tr>
<th>&lt;nbytes&gt;</th>
<th>Define the buffer size threshold range (in bytes) for choosing the MPI_Gather algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 0</td>
<td>The default value is 2048</td>
</tr>
</tbody>
</table>

**Description**

Set this variable to control the selection of the two possible MPI_Gather algorithms according to the following scheme:

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.

**Miscellaneous**

**I_MPI_TIMER_KIND**

Select the timer used by the MPI_Wtime and MPI_Wtick calls.

**Syntax**

\[
\text{I_MPI_TIMER_KIND} = <timername>
\]
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;timename&gt;</td>
<td>Define the timer type</td>
</tr>
<tr>
<td>gettimeofday</td>
<td>If this setting is chosen, the MPI_Wtime and MPI_Wtick functions will work through the function gettimeofday(2). This is the default value</td>
</tr>
<tr>
<td>rdtsc</td>
<td>If this setting is chosen, the MPI_Wtime and MPI_Wtick functions will use the high resolution RDTSC timer</td>
</tr>
</tbody>
</table>

Description

Set this variable to select either the ordinary or RDTSC timer.

NOTES

- The resolution of the default gettimeofday(2) timer may be insufficient on certain platforms.