Intel® MPI Library Reference Manual

Disclaimer and Legal Information

MPI Legal Notices

Contents

Overview

The Intel® MPI Library is a multi-fabric message passing library that implements the Message Passing Interface, v2 (MPI-2) specification. It enables you to switch interconnection fabrics without re-linking.

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.

Compiler Command	**Underlying Compiler**	**Supported Language(s)**	**Supported ABI(s)**
GNU* compilers			
`mpicc`	`gcc, cc`	C	32/64 bit

`mpicxx`	`g++ v3.x`
	`g++ v4.x`
		C/C++	32/64 bit

`mpicxx2`	`g++ v2.x`	C/C++	32/64 bit
`mpif77`	`g77`	F77	32/64 bit
`mpif90`	`gfortran`	F95	32/64 bit
Intel ® Fortran, C++ Compilers version 8.0, 8.1, 9.0 or 9.1			
`mpiicc`	`icc`	C	32/64 bit
`mpiicpc`	`icpc`	C++	32/64 bit
`mpiifort`	`ifort`	F77/F95	32/64 bit
Intel ® Fortran, C++ Compilers version 7.1			
`mpiicc7`	`icc`	C	32 bit
`mpiicpc7`	`icpc`	C++	32 bit
`mpiifc`	`ifc`	F77/F90	32 bit
`mpiecc`	`ecc`	C	64 bit
`mpiecpc`	`ecpc`	C++	64 bit
`mpiefc`	`efc`	F77/F90	64 bit

NOTES

- *Compiler commands are only available in the Intel® MPI Library Development Kit.*
- *Compiler commands are in the* `<installdir>/bin` *directory. For Intel® EM64T, 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.*

- *To compile and link without using the `mpicc` and related commands, run the appropriate command with the `-show` option added. The output will indicate the correct flags, options, includes, defines, and libraries to add to the compile and link lines. For example, use the following command to show the required compile flags, options, and then include paths for compiling source files:*
  `$ mpicc -show -c test.c`
- *Use the following command to show the required link flags, options, and libraries for linking object files:*
  `$ mpicc -show -o a.out test.o`

Compiler Command Options

-show

Use this option to display the compilation and linkage commands without actually running them. 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 program in debug mode, and link the resulting executable against the debugging versions of the libraries. See also `I_MPI_DEBUG`, in Section *Environment variables*, for information on how to use additional debug features with `-g` builds.

-O

Use this option to enable optimization. If `-g` is used, `-O` is not implied. Specify `-O` explicitly if you want to enable optimization.

-t or –trace

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

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

-dynamic_log

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

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

-profile=<*name*>

Use this option to specify MPI profiling library to be used. The `<name>` is:

- The name of the configuration file `<name>.conf`. The configuration file is located in `<installdir>/etc` directory.

- The name of the library `lib<name>.so` or `lib<name>.a` is located in the same directory as Intel MPI Library. This library is included before the Intel MPI Library at 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 Intel MPI Library. The thread safe libraries are provided at level `MPI_THREAD_MULTIPLE.`

NOTES

o *If you specify* `–openmp` *or* `–parallel` *options for Intel® C Compiler, the thread safe version of the library will be used.*

o *If you specify* `–openmp` *or* `–parallel` *or* `–threads` *or* `–reentrancy` *or* `-reentrancy:threaded` *options for Intel® Fortran Compiler, the thread safe version of the library will be used.*

-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

Use this option for compiler drivers `mpicxx` and `mpiicpc` to link an application for running in a particular GNU* C++ environment.

Use `-gcc-version=3` to build an application compatible to GNU* C++ version up to 3.3.

Use `-gcc-version=4` to build an application compatible to GNU* C++ version 3.4 or higher.

A library compatible with the detected version of the GNU*C++ compiler is used by default.

Configuration Files

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

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

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 options for profile library. Use the following name convention:

```
<installdir>/etc/<name>.conf
```

Use the `<name>` as a parameter to `–profile` option for compiler drivers.

The following variables can be defined in the profile 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"
```

Then 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=<*name*>

Specify a profile library and have the same effect as if '`-profile=$MPICC_PROFILE`' was used as an argument to `mpicc`.

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.out`, or `path/name` of executable, compiled with `mpicc` or related command
`<file>`	File with command-line options (see below)

Description

In the first form, run the specified `<executable>` with the specified options. All the global and/or local options apply to all MPI processes. A single arg-set is assumed.

In the second form, 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 the `<executable>` that is executed can be specified separately for each arg-set.

In the third form, read the command line from the 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.

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

Global Options

-version or -V

Use this option to output Intel MPI Library version information.

-nolocal

Use this option to avoid running the `<executable>` on the host where the `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 will be run on the first host, the next `<# of processes>` on the next host, 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 does not make sense for the second form of the `mpiexec` command.

-machinefile *<machine file>*

Use this option to place the ranks of processes in compliance with machine file. The `<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 name in compliance with quantity of starting processes on it. You can use the following format not to repeat the same host name: `<host name>:<number of processes>`. Set a # symbol at the beginning of line to comment it.

-genv *<ENVVAR> <value>*

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

-genvnone

Use this option to not propagate 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 also Section *Local Options* for local options.

-tv

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

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

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

-idb

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

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

Include the installation path of the Intel Debugger into the `IDB_HOME` environment variable to use this option.

-gdb

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

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

-l

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

-s *<spec>*

Use this option to direct standard input to specified ranks. Use "`all`" as a `<spec>` value to specify all processes or `1,3,5` to specify exact list of processes or `2-4,6` to specify range of processes. The default value is `0`.

-ifhn *<hostname>*

Use this option to specify the network interface which will be used for communications with MPD. The `<hostname>` should be IP address or 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 file `<filename>`.

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 on.

-env <*ENVVAR*> <*value*>

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

-host <*nodename*>

Use this option to specify the particular `<nodename>` on which the MPI processes for the current arg-set are to be run.

-path <*directory*>

Use this option to specify the path to find the `<executable>` that is to be executed for the current arg-set.

-wdir <*directory*>

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

-umask <*umask*>

Use this option to perform umask for remote process.

-envall

Use this option to pass all environment variables in current environment.

-envnone

Use this option not to pass environment variables.

-envlist <*list of env var names*>

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

-configfile <*filename*>

Get command line options from the file `<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 free-format text containing default `mpiexec` command-line options. Blank lines and lines that start with a `'#'` character in the very first column of 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 to the respective `mpiexec.conf` file:

```
-genv I_MPI_DEVICE <device>
```

Environment Variables

MPIEXEC_TIMEOUT

Set the `mpiexec` timeout.

Syntax

```
MPIEXEC_TIMEOUT=<timeout>
```

Arguments

`<timeout>`	Defines mpiexec timeout period in seconds
`> 0`	There is no default timeout value

Description

Set this variable to make `mpiexec` terminate the job `<timeout>` seconds after its launch.

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`	TCP/Ethernet/sockets
`shm`	shared-memory only (no sockets)
`ssm`	Combined TCP + 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 TCP + shared memory + DAPL* (for clusters with SMP nodes and RDMA-capable network fabrics)

Description

Set this variable to select 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 the shared-memory as fabric, use the following command:

```
$ mpiexec -n <#ranks> -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 `/etc/dat.conf` is used. If the `<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. will be used*
- *If you build the MPI program using `mpicc –t=log`, the trace-enabled version of the library will be 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

Control fallback upon 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 upon the `ssm` fabric if initialization of 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 upon the available fabric.

If the `I_MPI_FALLBACK_DEVICE` is set to `enable` and an attempt to initialize specified fabric fails, the library falls back upon the shared memory and/or socket fabrics. The exact combination depends on number of processes started 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 the `I_MPI_FALLBACK_DEVICE` is set to `disable` and an attempt to initialize 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>`	Indicates level of debug information provided
`(unset)`	Print no debugging information
`1`	Print warnings if the specified `I_MPI_DEVICE` could not be used for some reason
`2`	Use to positively confirm which `I_MPI_DEVICE` was used
`> 2`	Add extra levels of debug information

Description

Set this variable to control output of the debugging information.

The `I_MPI_DEBUG` mechanism augments the `MPICH_DBG_OUTPUT` debug mechanism from MPICH2*. `I_MPI_DEBUG` overrides and implies `MPICH_DBG_OUTPUT=stdout`.

Compiling with `mpicc -g` causes considerable amounts of additional debug information to be printed.

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

```
$ mpiexec –n <# of processes> -env I_MPI_DEBUG +2 ./a.out
I_MPI: [rank#pid@hostname]Debug message
```

TOTALVIEW

Select the particular TotalView* executable file to use.

Syntax

```
TOTALVIEW=<path>
```

Arguments

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

Description

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

MPD Daemon Commands

mpdboot

Syntax

```
mpdboot  [ -n <#nodes> ] [ -f <hostsfile> ] [ -h ] [ -r <rshcmd> ] \
         [ -u <user> ] [ -m <mpdcmd> ] [ --loccons ] [ --remcons ] \
         [ -s ] [ -d ] [ -v ] [ -1 ] [ --ncpus=<ncpus> ] [ -o ]
```

or

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

Arguments

`-h, --help`	Display help message
`-d, --debug`	Print debug information
`–v, --verbose`	Print extra verbose information. Show the rshcmd attempts
`-n <#nodes>` `--totalnum=<#nodes>`	Number of nodes in mpd.hosts on which daemons start
`-r <rshcmd>` `--rsh=<rshcmd>`	Specify remote shell to start daemons and jobs. The rsh is default value
`-f <hostsfile>` `--file=<hostsfile>`	Path/name of file that has the list of machine names on which daemons start.
`-1`	Remove a restriction of starting only one mpd per machine
`-m <mpdcmd>` `--mpd=<mpdcms>`	Specify the full path name of mpd on the remote hosts
`-s, --shell`	Specify shell
`-u <user>` `--user=<user>`	Specify user
`--loccons`	Do not create local MPD consoles
`--remcons`	Do not create remote MPD consoles
`--ncpus=<ncpus>`	Indicate how many processors to use on the local machine (other nodes are listed in the hosts file)
`-o, --ordered`	Start all the mpd daemons exactly in the order as in the mpd.hosts file

Description

Start `mpd` daemons on the specified number of nodes by providing a list of node machine 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 the `ssh`. Make sure that all nodes of the cluster can connect to each other via `rsh` command without password or, if the `–r ssh` option is used, via `ssh` command without password.

NOTES

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

mpd

Syntax

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

Arguments

`[ --help ]`	Display help message

`[ -h <host> -p <portnum>]`
`[ --host=<host> --port=<portnum>]`
	Specify host and port to be used for entering an existing ring. The `--host` and `--port` options must be specified together

`[ -n ]`
`[ --noconsole ]`
	Do not create console at startup.

`[ -t ]`
`[ --trace ]`
	Print a lot of trace information

`[ -e ]`
`[ --echo ]`
	Print port number at startup which other `mpd` may connect

`[ -d ]`
`[ --daemon ]`
	Start `mpd` in daemon mode

`[ --ifhn=<interface/hostname> ]`	Specify an `<interface/hostname>` for the host

`[ -l <listenport> ]`
`[ --listenport=<listenport> ]`
	Specify a port for this `mpd` to listen on.

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: `mpd –h masterhost -p 4268 –n` is equivalent to `mpd --host=masterhost --port=4268 –noconsole`

A file named `.mpd.conf` file must be present in the user's home directory with read and write access only for the user, and must contain at least a line with `secretword=<secretword>`. Install `mpd.conf` in the `/etc` directory to run `mpd` as root.

mpdtrace

Determine whether `mpd` is running.

Syntax

```
mpdtrace [ -l ]
```

Arguments

`-l`	Show MPD identifiers instead of the hostnames

Description

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

mpdlistjobs

List running processes of jobs.

Syntax

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

or

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

Arguments

`-u <username>` `--user=<username>`	List jobs of particular user
`-a <jobalias>` `--alias=<jobalias>`	List information about particular job specified by jobalias
`-j <jobid>` `--jobid=<jobid>`	List information about particular job specified by jobid

Description

Use this command to list running processes of jobs. All jobs are displayed by default.

mpdkilljobs

Kill the job.

Syntax

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

Arguments

`<jobnum>`	Kill job specified by `<jobnum>`
`-a <jobalias>`	Kill job specified by `<jobalias>`

Description

Use this command to kill the job specified by `<jobnum>`or by `<jobalias>`. Obtain `<jobnum>`and `<jobalias>`from `mpdlistjobs` command. `<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

`number of loops`	Number of loops

Description

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

mpdexit

Shut down a single `mpd.`

Syntax

```
mpdexit <mpdid>
```

Arguments

`<mpdid>`	Specify `mpd` daemon to kill

Description

Use this command to cause a single `mpd` to exit. Use `<mpdid>`obtained via `mpdtrace –l` command.

mpdallexit

Shut down all `mpd` daemons on all nodes.

Arguments

```
No arguments
```

Description

Use this command to shutdown all `mpd` rings.

mpdcleanup

Syntax

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

or

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

Arguments

`-f <hostsfile>`
`--file=<hostsfile>`
	Specify the file of machines to cleanup

`-r <rshcmd>`
`--rsh=<rshcmd>`
	Specify remote shell to use

`-u <user>`
`--user=<user>`
	Specify user

`-c <cleancmd>`
`--clean=<cleancmd>`
	Specify command to use for removing UNIX* socket

Description

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

mpdsigjob

Deliver a signal to the application process of a job.

Syntax

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

Arguments

`sygtype`	Specify signal to send
`-a <jobalias>`	Send a signal to job specified by `<jobalias>`
`-j <jobid>`	Send signal to job specified by `<jobid>`
`-s`	Delivery signal to the single user process

`-g`	Delivery signal to the group of processes. It is default behavior.

Description

Use this command to deliver a specific signal to the application processes of a job. Specified signal is the first argument. Specify only one of `-j` or `–a` options.

mpdhelp

Syntax

`mpdhelp`

Arguments

`No arguments`

Description

Use this command to get short help about mpd commands.

Configuration Files

$HOME/.mpd.conf

This file has the `mpd` daemon password. 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 any Linux* login password 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 in the cluster.

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

NOTES

o *The* `MPD_SECRETWORD` *is a synonym for* `secretword`.

mpd.hosts

This file has the list of node machine names which the `mpdboot` command uses.

Ensure that this file only needs to be accessible by the user who runs `mpdboot` on the node/machine where the `mpdboot` command is actually invoked.

Syntax

The format of the `mpd.hosts` file is a list of machine names, one name per line. Blank lines, and lines that start with a `'#'` character in the very first column of the line, are ignored.

Environment Variables

PATH

Make the `PATH` settings required for `mpdboot` and other `mpd` daemon commands.

NOTES

- *The `<installdir>/bin` directory (`<installdir>/bin64` directory for Intel® EM64T 64-bit mode) and the path to Python* version 2.2 or higher should be in the `PATH` in order for `mpd` daemon commands to succeed.*

MPD_CON_EXT

Set unique name of the `mpd` console file.

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. Once this variable is set you can start one `mpd` ring and work with it without affect to other available `mpd` rings. Each `mpd` ring is associated with one `MPD_CON_EXT` value. Set the appropriate `MPD_CON_EXT` value to work with particular `mpd` ring.

Normally, every new `mpd` ring totally replaces the older one.

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 will be used by the `mpdboot` script instead of the default value `${HOME}/.mpd.conf` .

I_MPI_MPD_CONNECTION_TIMEOUT

Set the `mpd` connection timeout.

Syntax

`I_MPI_MPD_CONNECTION_TIMEOUT=<timeout>`

Arguments

`<timeout>`	Defines MPD connection timeout period in seconds

`> 0`	The default *timeout* value is equal to 20 seconds

Description

Set this variable to make `mpd` terminate the job if another `mpd` cannot be connected to in at most *`<timeout>`* seconds.

Simplified Job Startup Command

mpirun

Syntax

`mpirun [ <mpdboot options> ] <mpiexec options>`

Arguments

`<mpdboot options>`	`mpdboot` options as described in the `mpdboot` section above, except `-n`
`<mpiexec options>`	`mpiexec` options as described in the `mpiexec` section above

Description

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

The set of hosts is defined by the following rules checked in 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, in case 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 fully automatically according to the above scheme.

In other words, if you work under one of the aforementioned job schedulers, you don’t 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.

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.

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

`<pinmode>`	Selects CPU pinning mode
`mpd`	Pin processes inside MPD
`lib`	Pin processes inside MPI library
`enable, yes, on, 1`	Pin processes inside MPD
`disable, no, off, 0`	Do not pin processes. This is default value

`<proclist>`	Defines process to CPU map
`all`	Use all CPUs
`allcores`	Use all CPU cores (or physical CPU)
`n`	Use only CPU number n (0,1, … , total number of CPUs - 1)
`m-n`	Use CPUs from m to n
`k,l-m,n`	Use CPUs k, l thru m, and n

Description

Set these variables to enable and control process pinning.

Set the variable `I_MPI_PIN_MODE` to `lib` to make the Intel MPI Library pin the processes.

Set the variable `I_MPI_PIN_MODE` to `mpd` to make `mpd` daemon pin processes via system specific means if they are available.

Set the `I_MPI_PIN_PROCS` variable to define the set of processors. If the pinning mode is not set the variable `I_MPI_PIN_PROCS` is ignored.

If the variable `I_MPI_PIN_MODE` is defined, the `I_MPI_PIN_PROCS` value `allcores` is assumed.

If no CPU set is defined in the system, the number and order of the processors corresponds 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, this process is bound to the next CPU out of the specified set.

Note that every host can be made to use their own value of an environment variable, or use a global value.

Device Control

I_MPI_SPIN_COUNT

Control the spin wait mode.

Syntax

`I_MPI_SPIN_COUNT=<scount>`

Arguments

`<scount>`	Defines the spin count for loop of polling fabric(s) in spin wait mode
`> 0`	The default `<scount>` value is equal to 1 for sock, shm and ssm devices and equal to 250 for rdma and rdssm devices.

Description

Change the spin count for loop of polling fabric(s) before freeing processor in case absent messages for processing.

I_MPI_EAGER_THRESHOLD

Change the eager/rendezvous cutover point for all devices.

Syntax

`I_MPI_EAGER_THRESHOLD=<nbytes>`

Arguments

`<nbytes>`	Defines eager/rendezvous cutover point
`> 0`	The default `<nbytes>` value is equal to 262144

Description

Set this variable to control the point-to-point protocol switchover point.

There are eager and rendezvous protocols for data transferred by the library. Messages shorter than or equal in size to `<nbytes>` are sent eagerly. Larger messages are sent by using more memory efficient rendezvous protocol.

I_MPI_INTRANODE_EAGER_THRESHOLD

Change the eager/rendezvous cutover point for intra-node communication mode.

Syntax

`I_MPI_INTRANODE_EAGER_THRESHOLD=<nbytes>`

Arguments

`<nbytes>`	Defines threshold for DAPL* intra-node communication
`> 0`	The default `<nbytes>` value is equal to 262144

Description

Set this variable to change threshold for intra-node communication mode.

There are eager and rendezvous protocols for data transferred by the library within the node. Messages shorter than or equal in size to `<nbytes>` are sent eagerly. Larger messages are sent by using 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 `shm` device.

Syntax

`I_MPI_SHM_PROC_THRESHOLD=<nproc>`

Arguments

`<nproc>`	Defines static/dynamic mode switch point for `shm` device.
`> 0, < 90`	The default `nproc` value is equal to 90

Description

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

There are static and dynamic modes for allocation shared memory segment(s) for `shm` device. The only one common shared memory segment is allocated for all processes in the static mode at initialization stage. The individual shared memory segments allocated for each connection in dynamic mode.

NOTES

o *The* `I_MPI_USE_DYNAMIC_CONNECTIONS` *environment variable does not make sense when static allocation mode is used.*

RDMA and RDSSM Device Control

RDMA_IBA_EAGER_THRESHOLD

Change the eager/rendezvous cutover point.

Syntax

`RDMA_IBA_EAGER_THRESHOLD=<nbytes>`

Arguments

`<nbytes>`	Defines eager/rendezvous cutover point
`> 0`	The default `<nbytes>` value is equal to 16512

Description

Set this variable to control low level point-to-point protocol switchover point.

There are low level eager and rendezvous protocols for data transferred by the `rdma` and `rdssm` devices. Messages shorter than or equal in size to `<nbytes>` are sent eagerly through internal pre-registered buffers. Larger messages are sent by using more memory efficient rendezvous protocol.

NOTES

o *This variable also determines the size of every pre-registered buffer. The higher it is, the more memory will be used for every established connection.*

NUM_RDMA_BUFFER

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

Syntax

`NUM_RDMA_BUFFER=<nbuf>`

Arguments

`<nbuf>`	Defines 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.

NOTES

o *The more pre-registered buffers are available, the more memory will be 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

`<nbytes>`	Defines the size of pre-registered buffers
`> 0`	The default `<nbytes>` value is equal to 16640

Description

Setting the value `<nbytes>` for this environment variable directs the `rdma` and `rdssm` devices to set the size of internal pre-registered buffer for each pair in a process group according to specified value. The actual size calculated by adjusting `<nbytes>` for alignment buffer to optimal value.

I_MPI_RDMA_TRANSLATION_CACHE

Turn on/off the mode of using a registration cache.

Syntax

`I_MPI_RDMA_TRANSLATION_CACHE=<arg>`

Arguments

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

Description

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

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

I_MPI_DAPL_IP_ADDR

I_MPI_DAPL_HOST

I_MPI_DAPL_HOST_SUFFIX

Specify the Interface Adapter (IA) address.

Syntax

`I_MPI_DAPL_IP_ADDR=<ipaddr>`

`I_MPI_DAPL_HOST=<hostname>`

`I_MPI_DAPL_HOST_SUFFIX=<hostsuff>`

Arguments

`<ipaddr>`	Defines the IA address as an explicit IP address. The value `<ipaddr>` should have IP address of the host in the usual convention

`<hostname>`	Defines the IA address using a `<hostname>`

`<hostsuff>`	Provides explicit hostname suffix that is prepended to the host name.

Description

Set the `I_MPI_DAPL_IP_ADDR`, `I_MPI_DAPL_HOST`, or `I_MPI_DAPL_HOST_SUFFIX` variables to control the identity of the Interface Adapter (IA).

NOTES

- *If none of these three variables is set, the IA address is determined automatically. This is the recommended mode of operation.*

I_MPI_DAPL_PORT

Specify the PSP (Public Service Point) value.

Syntax

`I_MPI_DAPL_PORT=<port>`

Arguments

`<port>`	Defines the port value
`Between 1024 and 65536`	The value of `<port>` must be an integer number between 1024 and 65536

Description

Set this variable to specify the PSP value.

NOTES

- *If this variable is not defined, the PSP port value is calculated automatically. This is the recommended mode of operation.*

I_MPI_USE_RENDEZVOUS_RDMA_WRITE

Turn on/off the use of rendezvous RDMA Write protocol instead of the default RDMA Read protocol.

Syntax

`I_MPI_USE_RENDEZVOUS_RDMA_WRITE=<arg>`

Arguments

`<arg>`	Binary indicator
`enable, yes, on, 1`	Turn the RDMA Write rendezvous protocol on
`disable, no, off, 0`	Turn the RDMA Write rendezvous protocol off. This is the default value

Description

Set this variable to select RDMA Write based rendezvous protocol.

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

I_MPI_RDMA_USE_EVD_FALLBACK

Turn on/off the Event Dispatcher (EVD) based polling fallback path.

Syntax

`I_MPI_RDMA_USE_EVD_FALLBACK=<arg>`

Arguments

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

Description

Set this variable to use 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

- *Note that the EVD path is typically substantially slower than the default algorithm.*

I_MPI_USE_DYNAMIC_CONNECTIONS

Turn on/off the dynamic connection establishment.

Syntax

`I_MPI_USE_DYNAMIC_CONNECTIONS=<arg>`

Arguments

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

Description

Set this variable to control dynamic connection establishment.

If this mode is enabled, connections are established upon first communication between each pair of processes. This is the default behavior. All connections are established upfront if this variable is off.

I_MPI_DYNAMIC_CONNECTIONS_MODE

Choose the algorithm of dynamic establishment of the DAPL* connections.

Syntax

`I_MPI_DYNAMIC_CONNECTION_MODE=<arg>`

Arguments

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

Description

Set this variable to choose the algorithm for handling dynamically established connections for DAPL* capable fabrics.

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 some broken providers.

I_MPI_DAPL_CONNECTION_TIMEOUT

Specify DAPL* connection timeout.

Syntax

`I_MPI_DAPL_CONNECTION_TIMEOUT=<value>`

Arguments

`<value>`	Defines DAPL* connection timeout value in microseconds
`> 0`	Default value is infinite

Description

Set this variable to specify timeout for DAPL* connection establishment operations.

NOTES

- *If this variable is not defined, infinite timeout is used. This is the recommended mode of operation.*

I_MPI_TWO_PHASE_BUF_ENLARGEMENT

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

Syntax

```
I_MPI_TWO_PHASE_BUF_ENLARGEMENT=<arg>
```

Arguments

`<arg>`	Binary indicator
`enable, yes, on, 1`	Turn the mode of using two-phrase buffer enlargement on
`disable, no, off,0`	Turn the mode of using two-phrase buffer enlargement off. This is the default value

Description

Set this variable to turn on/off the mode of using two- phase buffer enlargement.

If this mode is turned on, small size internal pre-registered RDMA buffers are allocated and enlarged later if the data to transfer exceeds some threshold. Two-phase buffer enlargement is off by default.

I_MPI_RDMA_SHORT_BUF_THRESHOLD

Change threshold for two-phrase buffer enlargement mode.

Syntax

```
I_MPI_RDMA_SHORT_BUF_THRESHOLD=<nbytes>
```

Arguments

`<nbytes>`	Defines threshold for starting RDMA buffers enlargement
`> 0`	The default *nbytes* value is equal to 580

Description

Set this variable to change threshold for two-phase buffer enlargement mode. This variable is used only if `I_MPI_TWO_PHASE_BUF_ENLARGEMENT` is set to `enable`.

I_MPI_USE_DAPL_INTRANODE

Turn on/off the DAPL* intra-node communication mode.

Syntax

`I_MPI_USE_DAPL_INTRANODE=<arg>`

Arguments

`<arg>`	Binary indicator
`enable, yes, on, 1`	Turn the DAPL* intra-node communication on
`disable, no, off, 0`	Turn the DAPL* intra-node communication off. This is the default value

Description

Set this variable to specify the intra-node communication mode for the universal device. If DAPL* intra-node communication mode is turned on then small messages are transferred using shared memory and large ones via the DAPL* layer. The threshold value for selecting communication layer of each message is determined by setting the `I_MPI_INTRANODE_EAGER_THRESHOLD` variable. It works only if neither shared memory nor DAPL* layer were turned off by setting `I_MPI_DEVICE` environment variable.

I_MPI_CONN_EVD_QLEN

Define event queue size of DAPL* event dispatcher.

Syntax

`I_MPI_CONN_EVD_QLEN=<size>`

Arguments

`<size>`	Defines length of event queue
`> 0`	The default value queried from DAPL provider.

Description

Set this variable to define event queue size of 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 provide a queue size that is at least equal to the calculated value, but the provider is free to provide a larger queue size.

I_MPI_DAPL_CHECK_MAX_RDMA_SIZE

Let DAPL provider control message segmentation threshold.

Syntax

`I_MPI_DAPL_CHECK_MAX_RDMA_SIZE=<arg>`

Arguments

`<arg>`	Binary indicator
`enable, yes, on, 1`	Allow to fragment messages

`disable, no, off, 0`	Forbid message fragmentation. This is the default value

Description

Set this variable to let DAPL provider control message segmentation threshold. This variable is set to *`disable`* by default to maximize performance. Certain DAPL providers set the value of *`max_rdma_size`* to an inappropriately small value. If you set the `I_MPI_DAPL_CHECK_MAX_RDMA_SIZE` to *`disable`*, Intel MPI will never fragment messages. If you set the `I_MPI_DAPL_CHECK_MAX_RDMA_SIZE` to *`enable`*, Intel MPI will fragment messages with size greater than the value of the DAPL attribute *`max_rdma_size`*.

Collective Operation Control

I_MPI_FAST_COLLECTIVES

Turn on/off the optimization of the collective operations.

Syntax

`I_MPI_FAST_COLLECTIVES=<arg>`

Arguments

`<arg>`	Binary indicator
`enable, yes, on, 1`	Turn the collective optimizations on
`disable, no, off, 0`	Turn the collective optimizations off. This is the default value

Description

Set this variable to controls optimization level of the collective operations.

The character of optimization depends upon internal package settings. All collective optimizations are turned off by default.

NOTES

- *If `I_MPI_FAST_COLLECTIVES` is turned on, then all other settings related to the collective operations (see `I_MPI_BCAST_NUM_PROCS`, `I_MPI_BCAST_MSG`, and so on) are not observed directly, because more suitable algorithms are chosen automatically in this case.*
- *Some optimizations of the collective operations may lead to violation of the MPI recommendation regarding the order of execution of the collective operations. Therefore results obtained in two different runs may differ depending on the process layout with respect to the processors and certain other factors.*
- *Some optimizations controlled by this variable may have an experimental character. In case of failure, turn the collective optimizations off 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

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

<nbytes1,nbytes2>	Defines the `MPI_Bcast` buffer size algorithm thresholds in bytes

`> 0`
`nbytes2 >= nbytes1`
	The default value is 12288,524288

Description

Set these variables to control selection of the `MPI_Bcast` algorithms according to the following scheme:

1. The first algorithm is selected if the message size is below *<nbytes1>*, or the number of processes in the operation is below *<nproc>*.
2. The second algorithm is selected if the message size lies between *<nbytes1>* and *<nbytes2>*, and the number of processes in the operation is a power of two.
3. The third algorithm is selected otherwise.

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

<nproc>	Defines the `MPI_Alltoall` number of processes algorithm thresholds
`> 0`	The default value is 8

<nbytes1,nbytes2>	Defines the `MPI_Alltoall` buffer size algorithm thresholds in bytes

`> 0`
`nbytes2 >= nbytes1`
	The default value is 256,32768

Description

Set these variables to control selection of the `MPI_Alltoall` algorithms according to the following scheme:

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

2. The second algorithm is selected if the message size lies between `<nbytes1>` and `<nbytes2>`, or if the message size lies below `<nbytes1>` and the number of processes in the operation is less than `<nproc>`.
3. The third algorithm is selected otherwise.

I_MPI_ALLTOALLV_MSG

Control optimized `MPI_Alltoallv` algorithm thresholds. The variable has an effect only if `I_MPI_FAST_COLLECTIVES` is turned on.

Syntax

`I_MPI_ALLTOALLV_MSG=<nbytes>`

Arguments

`<nbytes>`	Defines the `MPI_Alltoallv` buffer size algorithm thresholds in bytes
`> 0`	The default value is 4000

Description

Set this variable to select the optimized `MPI_Alltoallv` algorithm according to the following scheme:

The optimized algorithm is selected if the message size is above `<nbytes>`, and I_MPI_FAST_COLLECTIVES is turned on, and number of processes in the operation is power of two.

I_MPI_ALLGATHER_MSG

Control `MPI_Allgather` algorithm thresholds.

Syntax

`I_MPI_ALLGATHER_MSG=<nbytes1,nbytes2>`

Arguments

`<nbytes1,nbytes2>`	Defines the `MPI_Allgather` buffer size algorithm thresholds in bytes

`> 0`
`nbytes2 >= nbytes1`
	The default value is 81920,524288

Description

Set this variable to control selection of the `MPI_Allgather` algorithms according to the following scheme:

1. The first algorithm is selected if the message size lies below `<nbytes2>` and the number of processes in the operation is a power of two.
2. The second algorithm is selected if the message size lies below `<nbytes1>` and number of processes in the operation is not a power of two.
3. The third algorithm is selected otherwise.

I_MPI_ALLREDUCE_MSG

Control `MPI_Allreduce` algorithm thresholds.

Syntax

`I_MPI_ALLREDUCE_MSG=<nbytes>`

Arguments

`<nbytes>`	Defines the `MPI_Allreduce` buffer size algorithm threshold in bytes
`> 0`	The default value is 2048

Description

Set this variable to control selection of the `MPI_Allreduce` algorithms according to the following scheme:

1. The first algorithm is selected if the message size lies below `<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. The second algorithm is selected otherwise.

I_MPI_REDUCE_MSG

Control `MPI_Reduce` algorithm thresholds.

Syntax

`I_MPI_REDUCE_MSG=<nbytes>`

Arguments

`<nbytes>`	Defines the `MPI_Reduce` buffer size protocol threshold in bytes
`> 0`	The default value is 2048

Description

Set this variable to control selection of the `MPI_Reduce` algorithms according to the following scheme:

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

I_MPI_SCATTER_MSG

Control `MPI_Scatter` algorithm thresholds.

Syntax

`I_MPI_SCATTER_MSG=<nbytes>`

Arguments

`<nbytes>`	Defines the `MPI_Scatter` buffer size algorithm threshold in bytes
`> 0`	The default value is 2048

Description

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

1. The first algorithm is selected on intercommunicators if the message size lies above `<nbytes>`.
2. The second algorithm is selected otherwise.

I_MPI_GATHER_MSG

Control `MPI_Gather` algorithm thresholds.

Syntax

`I_MPI_GATHER_MSG=<nbytes>`

Arguments

`<nbytes>`	Defines the `MPI_Gather` buffer size algorithm threshold in bytes
`> 0`	The default value is 2048

Description

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

1. The first algorithm is selected on intercommunicators if the message size lies above `<nbytes>`.

2. The second algorithm is selected otherwise.

I_MPI_REDSCAT_MSG

Control `MPI_Reduce_scatter` algorithm thresholds.

Syntax

`I_MPI_REDSCAT_MSG=<nbytes1,nbytes2>`

Arguments

`<nbytes1,nbytes2>`	Defines the `MPI_Reduce_scatter` buffer size algorithm threshold in bytes

`> 0`
`nbytes2 >= nbytes1`
	The default value is 512,524288

Description

Set this variable to control selection of the `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 lies below `<nbytes2>`.
2. The second algorithm is selected if the reduction operation is commutative and message size lies above `<nbytes2>`, or if the reduction operation is not commutative and the message size lies above `<nbytes1>`.
3. The third algorithm is selected otherwise.

Miscellaneous

I_MPI_TIMER_KIND

Select the timer used by the `MPI_Wtime` and `MPI_Wtick` calls.

Syntax

`I_MPI_TIMER_KIND=<timername>`

Arguments

`<timername>`	Defines timer type
`gettimeofday`	`MPI_Wtime` and `MPI_Wtick` functions will work through the function `gettimeofday(2)`. This is a default value.
`rdtsc`	`MPI_Wtime` and `MPI_Wtick` functions will use the high resolution RDTSC timer

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.*