Getting Started with the Intel® MPI Library

## 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 Getting Started with Intel® MPI Library guide may contain software defects which may cause the product to deviate from published specifications. Current characterized software defects are available on request.

This Getting Started with Intel® MPI Library guide, 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 © 2004-2006 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

Disclaimer and Legal Information 2			
	MPI Legal Notices	3	
Contents	4		
Overview	5		
Using the Intel® MPI Library 6			
	Before You Begin	6	
	Usage Model	6	
	Quick Start	7	
	1. Compiling and Linking	8	
	2. Setting up MPD Daemons	8	
	3. Selecting a Network Fabric	9	
	4. Running an MPI Program	9	
Troubleshooting 11			
	Testing Installation	11	
	Troubleshooting MPD Setup	11	
	Compiling and Running a Test Program	12	

# 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 goal of this *Getting Started Guide* is to explain how to use Intel MPI Library to compile and run a simple MPI program. This guide also includes basic usage examples and troubleshooting tips.

This release of the Intel MPI Library supports the following major features:

- MPI-1 and MPI-2 specification conformance with some limitations
- Support for any combination of the following interconnection fabrics:
  - o Shared memory
  - o RDMA-capable network fabrics via DAPL\*, such as InfiniBand\* and Myrinet\*
  - o Sockets, for example, TCP/IP over Ethernet\*, Gigabit Ethernet\*, and other interconnects
- Support for IA-32 and Itanium® architecture clusters using:
  - Intel® C++ Compiler for Linux\* version 7.1 and higher
  - o Intel® Fortran Compiler for Linux\* version 7.1 and higher
  - o GNU\* C, C++ and Fortran 95 compilers
- Support for Intel® Extended Memory 64 Technology (Intel® EM64T) using:
  - o the Intel C++ Compiler for Linux\* version 8.1 and higher
  - o Intel Fortran Compiler for Linux\* version 8.1 and higher
  - o GNU\* C, C++ and Fortran 95 compilers
- C, C++, Fortran-77 and Fortran-90 language bindings
- Dynamic or static linking
- Clusters with homogeneous processor architectures and operating environments only.

The MPI-2 specification provides full support for MPI-1, as well as the following new functionality:

- One-sided communication (RDMA reads and writes)
- Extended collective operations
- Enhanced, standardized I/O functionality
- Standardized job startup mechanism via MPD daemons (Multi-Purpose Daemons) and the mpiexec command

See the product *Release Notes, Known Limitations* section for information on MPI-2 implementation limitations.

# Using the Intel® MPI Library

## **Before You Begin**

Before using the Intel® MPI Library, ensure that the library, scripts, and utility applications are installed. See the product *Release Notes* for installation instructions.

## Usage Model

Using the Intel MPI Library involves the following steps:



Figure 1: Flowchart representing the usage model for working with Intel MPI Library.

## **Quick Start**

- 1. Use the source shell command for getting proper environment settings from the mpivars.[c]sh scripts included with the Intel MPI Library. It located in the <*installdir>*/bin directory, or, for Intel EM64T 64-bit mode, the <*installdir>*/bin64 directory.
- 2. Create a \$HOME/.mpd.conf file. To set up your MPD password, enter the following into this file:

```
secretword=<mpd secret word>
```

Do not use any Linux\* login password. An arbitrary <mpd secret word> string only controls access to the MPD daemons by various cluster users.

3. Set protection on the \$HOME/.mpd.conf file using the chmod command so that only you have read and write privileges:

```
$ chmod 600 $HOME/.mpd.conf
```

4. Verify that you can observe the PATH settings and .mpd.conf contents through rsh on all nodes of the cluster. For example, use the following commands with each <*node*> in the cluster:

```
$ rsh <node> env
```

```
$ rsh <node> cat $HOME/.mpd.conf
```

- 5. Create an mpd.hosts text file that lists the nodes in the cluster using one host name per line.
- 6. Make sure that you have compiler in your PATH.
- 7. Compile test program using appropriate compiler driver. For instance: \$ mpicc -o test <installdir>/test/test.c
- 8. Execute the test using the mpirun command.

\$ mpirun -n <# of processes> ./test

9. See the rest of this document and *Reference Manual* for more details.

## 1. Compiling and Linking

#### To compile and link an MPI program with Intel MPI Library:

- 1. Ensure that the underlying compiler and related software appear in your PATH.
- If you are using Intel compilers, ensure that the compiler library directories appear in LD\_LIBRARY\_PATH environment variable.
   For example, for Intel C++ Compiler version 9.1 and Intel Fortran Compiler version 9.1, execute the appropriate setup scripts:

/opt/intel\_cc\_91/bin/iccvars.[c]sh, and

/opt/intel\_fc\_91/bin/ifortvars.[c]sh

 Compile your MPI program via the appropriate mpi command. For example, use the mpicc command to compile C code using the GNU\* C compiler as follows:

\$ mpicc <installdir>/test/test.c

where *<installdir>* is a full path to installed package.

All supported compilers have equivalent commands that use the prefix mpi on the standard compiler command. For example, the Intel MPI Library command for Intel Fortran Compiler from version 8.0 up (ifort) is mpiifort.

## 2. Setting up MPD Daemons

The Intel MPI Library uses a Multi-Purpose Daemon (MPD) job startup mechanism. In order to run programs compiled with mpicc (or related) commands, set up MPD daemons.

Always start and maintain your own set of MPD daemons, as opposed to having the system administrator start up the MPD daemons once for use by all users on the system. This setup enhances system security and gives you flexibility in controlling your execution environment.

#### To set up MPD daemons:

- 1. Set up environment variables with appropriate values and directories, for example, in the .cshrc or .bashrc files.
  - Ensure that the PATH variable includes the *<installdir>/bin* directory, or, for Intel EM64T 64-bit mode, the *<installdir>/bin64* directory. Use the mpivars.[c]sh scripts included with the Intel MPI Library to set up this variable.
  - Ensure that the PATH variable includes the directory for Python\* version 2.2 or greater.
  - If you are using Intel compilers, ensure that the LD\_LIBRARY\_PATH variable contains the directories for the compiler library. Set this variable by using the \*vars.[c]sh scripts included with the compiler.
  - Set any additional environment variables your application uses.
- 2. Create a \$HOME/.mpd.conf file. To set up your MPD password, enter the following into this file:

```
secretword=<mpd secret word>
```

Do not use any Linux\* login password. An arbitrary *mpd* secret word> string only controls access to the MPD daemons by various cluster users.

3. Set protection on the \$HOME/.mpd.conf file using the chmod command so that only you have read and write privileges:

\$ chmod 600 \$HOME/.mpd.conf

4. Verify that you can observe the PATH settings and .mpd.conf contents through rsh on all nodes of the cluster. For example, use the following commands with each <*node*> in the cluster:

```
$ rsh <node> env
```

```
$ rsh <node> cat $HOME/.mpd.conf
```

Make sure that every node, rather than only one of them, can connect to any other node. If your cluster uses **ssh** instead of **rsh**, look into the *Notes* section below.

- 5. Create an mpd.hosts text file that lists the nodes in the cluster using one host name per line.
- 6. Shut down the eventual MPD daemons using the mpdallexit command:

\$ mpdallexit

7. Use the mpdboot command to start up the MPD daemons:

```
$ mpdboot -n <#nodes>
```

The file \$PWD/mpd.hosts will be used by default if it is present. If there is no host file, the mpdboot command will start one MPD daemon on the local machine.

8. Use the mpdtrace command to determine the status of the MPD daemons:

#### \$ mpdtrace

The output should be a list of nodes that are currently running MPD daemons. This list should match the contents of the mpd.hosts file.

#### NOTES

- If your cluster uses ssh instead of rsh, make sure that every node can connect to any other node via ssh without a password. For details of the ssh setup, look into your system manuals.
- If your cluster uses ssh instead of rsh, add the -r ssh option to the mpdboot invocation string.

## 3. Selecting a Network Fabric

The Intel MPI Library dynamically selects different fabrics for communication between MPI processes.

To select specific fabric combination, set the **I\_MPI\_DEVICE** environment variable to one of the following values:

I_MPI_DEVICE values	Supported fabric
sock	TCP/Ethernet*/sockets
shm	Shared memory only (no sockets)
ssm	TCP + shared memory (for SMP clusters connected via Ethernet*)
rdma[: <provider>]</provider>	InfiniBand*, Myrinet*, etc. (via specified DAPL* provider)
rdssm[: <provider>]</provider>	TCP + shared memory + DAPL* (for SMP clusters connected via RDMA-capable fabrics)

Ensure that the selected fabric is available. For example, use the shm only when all the processes can communicate with each other via shared memory. Use the rdma only when all processes can communicate with each other via a single DAPL provider.

### 4. Running an MPI Program

To launch programs linked with the Intel MPI Library, use the mpiexec command:

\$ mpiexec -n <# of processes> ./myprog

Use the -n option to set the number of processes. This is the only obligatory option for the mpiexec command.

If you are using a network fabric as opposed to the default fabric, use the -genv option to specify a value to be assigned to the I\_MPI\_DEVICE variable.

For example, to run an MPI program using the shm fabric, type in the following command:

\$ mpiexec -genv I\_MPI\_DEVICE shm -n <# of processes> ./a.out

For the rdma capable fabric, use the following command:

\$ mpiexec -genv I\_MPI\_DEVICE rdma -n <# of processes> ./a.out

You can select any supported device. For more information, see Section <u>Selecting a Network</u> <u>Fabric</u> above.

If you successfully ran your application using the Intel MPI Library, it should run as with other MPI libraries. You can now move your application from one cluster to another and use different fabrics between the nodes without re-linking. If you encounter problems, see <u>*Troubleshooting*</u> for possible solutions.

## Troubleshooting

Use the following sections to troubleshoot problems with installation, setup, and running applications using the Intel MPI Library.

## **Testing Installation**

To ensure that the Intel MPI Library is installed and functioning, complete the general testing, compile and run a test program.

#### To test the installation:

1. Verify that you have Python\* v2.2 or higher in your PATH:

```
$ rsh <nodename> python -V
```

If this command returns an error message or a value lower than 2.2, install Python\* v2.2 or higher, and make sure that you have it in your PATH.

2. Check for the presence of a Python\* XML module such as python-xml\* or libxml2-python\*:

```
$ rpm -qa | grep python-xml
$ rpm -qa | grep libxml2-python
```

Install the missing module if the output does not include the name "python-xml" or "libxml2python" and a version number.

3. Check for the presence of an XML parser such as expat\* or pyxml\*:

```
$ rpm -qa | grep expat
```

```
$ rpm -qa | grep pyxml
```

Install the missing module if the output does not include the name "expat" or "pyxml" and a version number.

4. Verify that *<installdir>/bin(<installdir>/bin64* for Intel EM64T 64-bit mode) is in your PATH:

\$ rsh <nodename> which mpiexec

You should see the correct path for each node you test.

5. If you use Intel compilers, verify that the appropriate directories are included in the PATH and LD\_LIBRARY\_PATH environment variables:

```
$ mpiexec -n <# of processes> env | grep PATH
```

You should start mpd ring before executing mpiexec command. You should see the correct directories for these path variables for each node you test. If you do not, call the appropriate \*vars.[c]sh scripts. For example, for Intel C++ Compiler version 9.1 use the following source command:

\$ . /opt/intel\_cc\_91/bin/iccvars.sh

6. In some unusual circumstances, you may need to include the *<installdir>/lib* directory (*<installdir>/lib*64 for Intel EM64T 64-bit mode) in your LD\_LIBRARY\_PATH. To verify your LD\_LIBRARY\_PATH settings, use the command:

```
$ mpiexec -n <# of processes> env | grep PATH
```

### **Troubleshooting MPD Setup**

Check if it is possible to run the mpd command on the local machine. Do the following:

```
# mpd &
# mpdtrace
```

```
# mpdallexit
```

The output of mpdtrace should show the hostname of the machine you are running on. If this is not the case, or if you cannot start up the MPD, check that the installation was correct and the environment was set up properly.

The next troubleshooting steps assume that MPD daemons are set up and running. To start your diagnosis, verify that MPD daemons are running on all expected nodes using:

#### mpdtrace

The output lists all MPD daemons running or indicates an error. If some desired nodes are missing from the output list of mpdtrace, do the following:

- 1. Try to restart the MPD daemons using the following commands:
  - a. Kill all running MPD daemons:\$ mpdallexit
  - b. For each node, ensure all daemons were killed:
    - \$ rsh <nodename> ps -ael | grep python
      - \$ rsh <nodename> kill -9 <remaining python processes>
  - c. Reboot the MPD daemons. Be sure to use the appropriate configuration options and host file:
    - \$ mpdboot [<options>]
  - d. Confirm that all expected MPD daemons are now running:
     \$ mpdtrace
- 2. If the output of the mpdtrace command is not still indicating that all expected MPD daemons are running, follow the next steps:
  - a. Kill and restart the MPD daemons as described in step 1, adding the debug and verbose options to the mpdboot command:
    \$ mpdboot -d -v [<options>]

```
Note the rsh commands in the output from step a. For example:
```

```
launch cmd= rsh  -n  <nodename> '<installdir>/bin/mpd  \
```

```
-h <nodename> -p <port-number> --ncpus=<ncpus> -e -d'
```

b. Copy and paste the line of the output from the rsh command up to the end of line. For example:

```
$ rsh -n <nodename> '<installdir>/bin/mpd \
```

```
-h <nodename> -p <port-number> --ncpus=<ncpus> -e -d'
```

- c. Execute the edited rsh command. Use the resulting output to diagnose and correct the underlying problem. For example, the most common problems include:
  - Failure of the rsh command to contact < nodename >.
  - Other failure of the rsh command, for example, a system setup problem.
  - The <installdir>/bin/mpd command could not be found or could not be executed.
  - The .mpd.conf file could not be found or read (access error).

### **Compiling and Running a Test Program**

To compile and run a test program, do the following:

- 1. Compile a test program included with the product release as follows:
  - \$ cd <installdir>/test
  - \$ mpicc test.c
- 2. If you are using InfiniBand\*, Myrinet\*, or other RDMA-capable network hardware and software, verify that everything is functioning.
- 3. Run the test program with all available configurations on your cluster.

a) Test the **sock** device using:

```
$ mpiexec -n 2 -env I_MPI_DEBUG 2 -env I_MPI_DEVICE sock \
./a.out
```

You should see one line of output for each rank, as well as debug output indicating the sock device used.

b) Test the **ssm** devices using:

\$ mpiexec -n 2 -env I\_MPI\_DEBUG 2 -env I\_MPI\_DEVICE ssm ./a.out You should see one line of output for each rank, as well as debug output indicating the ssm device used.

c) Test any other fabric devices using:

\$ mpiexec -n 2 -env I\_MPI\_DEBUG 2 -env I\_MPI\_DEVICE <device> \
./a.out

where <device> can be shm, rdma, or rdssm.

For each of the mpiexec commands used, you should see one line of output for each rank, as well as debug output indicating which device was used. The device(s) should agree with the I\_MPI\_DEVICE setting.

#### NOTE

The <installdir>/test directory in the Intel MPI Library Development Kit contains other test programs in addition to test.c that you can use for testing.