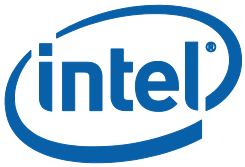

Using MPI Tuner for Intel® MPI Library

Tutorial for Windows* OS

Contents

Legal Information

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

1. Overview

Discover how to use the MPI Tuner for Intel® MPI Library to get optimized configuration files for the runtime library automatically. You can also get basic usage examples and troubleshooting tips from this tutorial.

About This Tutorial	This tutorial demonstrates various methods to optimize the performance of Intel® MPI Library for your own cluster and applications, including: • Minimize the cluster tuning time • Include missed values in the default parameter grid during cluster tuning • Configure the optimal settings during application tuning • Troubleshoot commonly seen issues when using the MPI tuner
Estimated Duration	15-20 minutes.
Learning Objectives	After you complete this tutorial, you should be able to: • Use the MPI tuner to get optimal settings for the Intel® MPI Library relevant to your cluster or your application configuration • Troubleshoot common issues when using the MPI tuner
More Resources	To get more information about the MPI Tuner for Intel® MPI Library, see the following resources: • Product Web Site • Intel® MPI Library Support • Intel® Cluster Tools Products • Intel® Software Development Products

2. Getting Started

Before using the MPI tuner for Intel® MPI Library, ensure that the library, scripts, and utility applications are installed. See the *Intel® MPI Library Getting Started Guide* for instructions.

The general workflow of using the MPI tuner is as follows:

1. Create optimized configuration files through the `mpitune` utility.
2. Use the configuration files through the `-tune` option of the `mpiexec` command during regular execution.

Usage instructions for specific tasks are given in detail further in this tutorial and rely on these general steps.

NOTE

Before you use the MPI tuner, you can check the tasks to be executed. Use the `--scheduler-only (-so)` option to see the scope of mpitune work before the real run: `> mpitune ... -so`.

2.1. Accessing MPI Tuner

To access the MPI Tuner, enter the command:

```
> mpitune
```

NOTE

This command is not available from Intel® Xeon Phi™ Coprocessor natively but can be launched from the host to tune MPI applications for any platforms supported by Intel® MPI Library.

2.2. MPI Tuner Modes

The MPI tuner utility can operate in two modes:

- **Cluster-specific.** Evaluates a given cluster environment using either Intel® MPI Benchmarks, or a user-provided benchmarking program to find the most suitable configuration of Intel® MPI Library. This mode is used by default.
- **Application-specific.** Evaluates performance of a given MPI application to find the best configuration of Intel® MPI Library for a particular application.

2.2.1. Cluster-Specific Tuning Commands

To use the MPI tuner in the cluster-specific mode:

1. Run the following command to create the tuned configuration files in the default `<installdir>\<arch>\<etc>` directory:

   ```
   > mpitune -hf <hostfile>
   ```

   To create configuration files in a different directory, use the `-odr` option:

   ```
   > mpitune -hf <hostfile> -odr <path_to_result_directory>
   ```

2. Use the `-tune` option without an argument to pick up the configuration files from the default directory, or use the path to the results as an argument for `-tune`. For example:

   ```
   > mpiexec -tune -ppn 8 -n 128 my_app.exe
   ```

```
> mpiexec -tune <path_to_result_directory> -ppn 8 -n 128 my_app.exe
```

2.2.2. Application-Specific Tuning Commands

To use the MPI tuner under application-specific mode:

1. Use the `--application (-a)` option to tune the specified workload for the provided environment and command line settings. The tuner records the new optimal settings in the `myprog.conf file`:

   ```
   > mpitune --application \"mpiexec -n 32 myprog.exe\" -of myprog.conf
   ```

2. Use the `-tune` option to pick up the optimal recorded values for your application at runtime:

   ```
   > mpiexec -tune myprog.conf -n 32 myprog.exe
   ```

3. Task 1: Minimize Tuning Time in Cluster-Specific Mode

To reduce the cluster tuning time, think about which are the most common and widely used MPI workloads on your cluster. Make a note about how they are typically run in regard to:

- The range of hosts used
- Numbers of ranks per host
- Fabric used (`I_MPI_FABRICS`)
- Common message sizes
- Most popular MPI functions

3.1. Host Range

For example, if the majority of workloads on the cluster use between `4` and `16` hosts, set these numbers as the lower and the upper bounds, through the `-hr <n:m>` option:

```
> mpitune ... -hr 4:16
```

The `mpitune` utility scans all hosts in the rage between `4` and `16` whose numbers are powers of `2`. For the example above, it creates tuned settings for `4`, `8`, and `16` hosts.

3.2. Numbers of Ranks per Host

Use the `-pr <n:m>` option to set the number of ranks per host:

```
> mpitune ... -pr 1:16
```

Similarly, the `mpitune` utility affects the ranks in the rage between `4` and `16` whose numbers are powers of `2`. For the example above, it creates tuned settings for all cases where the number of ranks is `1`, `2`, `4`, `8`, `16`.

If the lower and the upper bounds are the same, the `mpitune` utility tunes for the `ppn=24` case only:

```
> mpitune ... -pr 24:24
```

3.3. Fabric Usage (I_MPI_FABRICS)

Use the `-fl` option to specify which fabric to use during tuning:

```
> mpitune ... -fl shm:dapl,dapl,shm:ofa,ofa
```

The `mpitune` utility runs only on the specified fabrics.

3.4. Message Sizes

Use the `-mr` option to set the range of message sizes to be tuned (in bytes):

```
> mpitune ... -mr 16:2097152
```

The `mpitune` utility tunes MPI operations with message sizes that are powers of `2`, between the specified bounds of `16` and `2097152` bytes.

3.5. Most Common MPI Functions

If you have statistics on usage and performance of MPI functions, you can adjust the tuning scope according to your needs. You can find the correspondence of various tuning options with MPI functions in *Intel® MPI Library Developer Guide* before you start.

You can look at the most widely used MPI routines and go from simple to more complex functions. For example, you can perform point-to-point tuning before tuning collective operations.

Start with the P2P-sensitive options first:

1. Specify the most common MPI functions under the `option_set` variable:

```
> set option_set=I_MPI_RDMA_TRANSLATION_CACHE
,I_MPI_DAPL_RNDV_BUFFER_ALIGNMENT
,I_MPI_SHM_FBOX_SIZE
,I_MPI_SHM_CELL_SIZE
,I_MPI_SSHM_BUFFER_SIZE
,I_MPI_EAGER_THRESHOLD
,I_MPI_DAPL_BUFFER_SIZE
,I_MPI_INTRANODE_EAGER_THRESHOLD
,I_MPI_DAPL_DIRECT_COPY_THRESHOLD
```

2. Run a tuning session on `option_set`. This will create a set of optimal cluster settings based only on the environment variables provided above:

   ```
   > mpitune ... -os %option_set%
   ```

3. Proceed to tuning collective operations:

   ```
   > mpitune ... --collective-only
   ```

Once complete, merge both configuration files into a single one and use it with the `-tune` or `-config` runtime options.

NOTE

To reduce the tuning time in future, you can specify the percentage improvement needed, or exclude the options that show acceptable performance.

---

4. Task 2: Include Missing Values in the Default Parameters Grid during Cluster Tuning

The `mpitune` utility has a predefined range of variable values to be scanned. If you know that your applications use *atypical* layouts or data sizes, you can overwrite the `mpitune` defaults to run with a customized set.

Ensure you have write access to the `<installdir>\<arch>\<etc>` directory.

The `mpitune` utility uses `*.xml` files from `<installdir>\<arch>\<etc>` for its configuration. There are two main configuration files that describe what is tuned and how tuning is performed in the cluster-specific mode: `options.xml` and `Benchmarks\imb.xml`, respectively.

For example, if you would like to customize the tuning of the `I_MPI_EAGER_THRESHOLD` variable, see the highlighted text below for appropriate changes.

```
options.xml:
...
        <option name="I_MPI_EAGER_THRESHOLD" type="global" group="collective"
weight="1.0">
            <actions>
                <step order="1" storage="first">
                    <additive>
                        <env name="I_MPI_FALLBACK_DEVICE" type="global"
value="disable" />
                    </additive>
                    <range name="range vars">int range(8192:524288:*:2)</range> <!-
- explicit range from 8k to 512k with power of 2 -->

                    <format>@range_vars()</format>
                    <result format="[msg_size]" limit="1" separator="" />
                </step>
            </actions>
            <requirements>
                <param name="hosts" value="2:2" /> <!--use 2 hosts -->

                <param name="perhost" value="1:1" /> <!-- with 1 process on host --
>

                <param name="processes" value="2:2" /> <!-- and 2 processes total -
->

                <param name="devices" value="shm:dapl,shm:tmi" /> <!-- for shm:dapl
and shm tmi fabrics (I_MPI_FABRICS) -->

            </requirements>
            <result <!-- internal format description -->

                format="#first#"
                quotes="no"
                quotesInline="no"
            />
        </option>
...
```

```
Benchmarks/imb.xml:
        <test title="IMB Sendrecv" weight="1.0">
            <description>Sendrecv test from IMB benchmark for OUTPUT
mode</description>
            <executable>"IMB-MPI1" -npmin %proc% -iter 5 -msglen @msglen_file()
Sendrecv</executable>
            <function
title="msglen file">range file(768:1536:+:256;"value[endl]")</function> <!-- msg
len file of IMB with range: 768, 1024, 1280 and 1536 bytes -->

            <launch_line>%mpiexec%  %globals%  %locals%  %executable%</launch line>
            <requirements> <!-- values for requirements section are calculated as
intersection with the same block from options.xml file. Results are in the mpitune
schedule -->

                <param name="hosts" value="1:-1" />
                <param name="perhost" value="1:-1" />
                <param name="processes" value="2:-1" />
                <param name="devices"
value="rdssm,rdma,shm,ssm,sock,shm:dapl,shm:tcp,dapl,tcp,shm,shm:ofa,shm:tmi,ofa,tm
i" />
            </requirements>
            <options filter filter="exclusive"> <!--  this section enumerates
options to tune by this benchmark-->

                <option type="global"  name="I_MPI_EAGER_THRESHOLD" />
                <option type="global"  name="I_MPI_INTRANODE_EAGER_THRESHOLD" />
            </options filter>
            <result <!-- format to parse benchmark output -->

                source="brtime"
                paramGroup="4"
                paramTitle="t[usec]"
                paramTarget="min"
                paramLeftMarginGroup="2"
                paramRightMarginGroup="3"
                paramChooseMode="heaviest"
                paramDiffDelta="0.001"
                msgGroup="0"
                msgTitle="Bytes"
                iterationCompare="min"
                startline=".*(\#bytes\s+\#repetitions).*"
                dataline="\s+(\d+)\s+(\d+)\s+([\d\.]+)\s+([\d\.]+)\s+([\d\.]+)"
                solidatalines="1"
            />
        </test>
...
```

NOTE

When you define a custom range for tuning the option, take the following parameter into account:

`test->result->source`

This parameter is described in the configuration file of the benchmark you used. For example, your explicitly defined range is used when `thtime` value is set. However, the `brtime` parameter requires that you set the bottom and upper boundaries, while all intermediate values of the range are calculated automatically.

5. Task 3: Application-Specific Tuning

Now that you have completed the cluster-specific tuning, you can focus on optimizing Intel® MPI Library for your application. You can use the above cluster-specific methods and apply them to your application-specific tuning, with the following modifications:

1. The layout of your application (such as hosts and process count per host) is set on your `mpiexec` command line, outside of `mpitune`.
2. Use your application instead of the micro benchmarks mentioned above.
3. Use the `app.xml` configuration file instead of `imb.xml`.

6. Troubleshooting

This topic explains how to troubleshoot common issues seen when running with the MPI Tuner.

Issue	Cause and Possible Solutions
The scheduler of `mpitune` is empty.	1. Check the arguments for `mpitune` and ensure they do not contradicting each other. For example, values for `--options-set` and `--options-exclude` should not overlap. 2. If no fabrics or devices pass checking, try running an MPI test application with the same configuration. The issue might be caused by an incorrect host file or cluster configuration.
`mpitune` runs very long.	Check the projected schedule before the real launch, using the `-so` option. Use methods described in task 1 and task 2 and/or their combinations to skip unnecessary jobs