Using MPI Tuner for Intel® MPI Library

Tutorial for Linux® OS
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Legal Information</td>
<td>3</td>
</tr>
<tr>
<td>1. Overview</td>
<td>4</td>
</tr>
<tr>
<td>2. Getting Started</td>
<td>5</td>
</tr>
<tr>
<td>2.1. Accessing MPI Tuner</td>
<td>5</td>
</tr>
<tr>
<td>2.2. MPI Tuner Modes</td>
<td>5</td>
</tr>
<tr>
<td>2.2.1. Cluster-Specific Tuning Commands</td>
<td>6</td>
</tr>
<tr>
<td>2.2.2. Application-Specific Tuning Commands</td>
<td>6</td>
</tr>
<tr>
<td>3. Task 1: Minimize Tuning Time in Cluster-Specific Mode</td>
<td>7</td>
</tr>
<tr>
<td>3.1. Host Range</td>
<td>7</td>
</tr>
<tr>
<td>3.2. Numbers of Ranks per Host</td>
<td>7</td>
</tr>
<tr>
<td>3.3. Fabric Usage (I_MPI_FABRICS)</td>
<td>7</td>
</tr>
<tr>
<td>3.4. Message Sizes</td>
<td>7</td>
</tr>
<tr>
<td>3.5. Most Common MPI Functions</td>
<td>8</td>
</tr>
<tr>
<td>4. Task 2: Include Missing Values in the Default Parameters Grid during Cluster Tuning</td>
<td>9</td>
</tr>
<tr>
<td>5. Task 3: Application-Specific Tuning</td>
<td>11</td>
</tr>
<tr>
<td>5.1. Commands for Fast Application-Specific Tuning</td>
<td>11</td>
</tr>
<tr>
<td>5.2. Commands for Topology Awareness Rank Placement Optimization</td>
<td>11</td>
</tr>
<tr>
<td>6. Troubleshooting</td>
<td>13</td>
</tr>
</tbody>
</table>
Legal Information

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

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

Intel, Xeon Phi and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

© 2016 Intel Corporation.

Portions (PBS Library) are copyrighted by Altair Engineering, Inc. and used with permission. All rights reserved.

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 commonly seen 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 `mpirun` 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 four 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.

- **Fast application-specific.** In contrast to the previous mode, instead of launching a real application this mode performs micro-kernel tests during the tuning process. The set of tests is based on the Intel® MPI Library statistics collected from an application run. This approach is not very accurate but can speedup applications with non-typical patterns (non-typical rank placement per host or in a communicator, unusual messages sizes, and so on). This mode supports only regular collective operations that have related environment variables in the `I_MPI_ADJUST` family (excluding the `KN_RADIX` subset and `I_MPI_ADJUST_REDUCE_SEGMENT`). See the Intel® MPI Library Developer Reference for details.

- **Topology-aware rank placement optimization.** Evaluates cluster characteristics and rank-to-rank data transfers in an MPI application to identify the optimal rank placement. This mode is recommended for applications where the communication pattern has some kind of local groups, such as, stencils, collective operations on subsets of ranks, neighborhood operations, and so on.
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 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:
   
   $ mpirun -tune -ppn 8 -n 128 ./my_app
   $ mpirun -tune <path_to_result_directory> -ppn 8 -n 128 ./my_app

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 "mpirun -n 32 ./myprog\" -of ./myprog.conf

2. Use the -tune option to pick up the optimal recorded values for your application at runtime:
   
   $ mpirun -tune ./myprog.conf -n 32 ./myprog
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 \texttt{-hr} \texttt{<n:m>} option:

\texttt{$ mpitune ... \ -hr\ 4:16$}

The \texttt{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 \texttt{-pr} \texttt{<n:m>} option to set the number of ranks per host:

\texttt{$ mpitune ... \ -pr\ 1:16$}

Similarly, the \texttt{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 \texttt{mpitune} utility tunes the \texttt{ppn=24} case only:

\texttt{$ mpitune ... \ -pr\ 24:24$}

3.3. Fabric Usage (I_MPI_FABRICS)

Use the \texttt{-fl} option to specify which fabric to use during tuning:

\texttt{$ mpitune ... \ -fl\ shm:dapl,dapl,shm:ofa,ofa$}

The \texttt{mpitune} utility runs only on the specified fabrics.

3.4. Message Sizes

Use the \texttt{-mr} option to set the range of message sizes to be tuned (in bytes):

\texttt{$ mpitune ... \ -mr\ 16:2097152$}

The \texttt{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:**
   
   ```
   $ export 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:

```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>
      <set>@range_vars()</set>
      <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>
...```
Task 2: Include Missing Values in the Default Parameters Grid during Cluster Tuning

Benchmarks/imb.xml:

```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()
    <function title="msglen file" range_file(768:1536:+:256;"value[endl]")"/></function>
</test>
```

`Benchmarks/imb.xml:`

```xml
<launch_line>%mpiexec% %globals% %locals% %executable%</launch_line>
<requirements>
    <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,tmic" />
</requirements>

<options_filter filter="exclusive">
    <option type="global" name="I_MPI_EAGER_THRESHOLD" />
    <option type="global" name="I_MPI_INTRANODE_EAGER_THRESHOLD" />
</options_filter>

<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"
/>```

**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 `time` 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 mpirun 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.

5.1. Commands for Fast Application-Specific Tuning

To use the MPI tuner in the fast application-specific mode:

1. Use the --fast (-f) option to switch to this mode. Use the --application (-a) option to tune the specified workload for the provided environment and command line settings, or use --stats(-s) to pass previously gathered statistics. The tuner records the new optimal settings in the myprog.conf file:

   $ mpitune --fast --application "mpirun -n 32 ./myprog" -o ./myprog.conf

   Or use this command:

   $ mpitune -f -s ./stats.txt -o ./myprog.conf

   **NOTE**

   Use the --fast (-f) and --help(-h) options together to see other available options for this mode.

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

   $ mpirun -tune ./myprog.conf -n 32 ./myprog

5.2. Commands for Topology Awareness Rank Placement Optimization

To use the MPI tuner for rank placement optimization, you need a file with a list of hosts put in the order as MPI ranks are distributed on your system. For processes per host more than 1, duplicates are required. It can depend on specific parameters of the MPI process manager, cluster job scheduler, or resource manager. For example, for 4 hosts with 2 processes per host it usually looks like this:

$ cat hostfile.in
host1
host1
host2
host2
host3
host3
host4
host4

There are two possible scenarios. You can do the following:
1. **Use the** --rank-placement (-rp),--hostfile-in (-hi) **and** --config-out **options. The**
   **tuner** **records the new optimal settings in the** myprog.conf **file:**
   
   $ mpirun --rank-placement --application "mpirun -n 32 ./myprog" --hostfile-in hostfile.in --config-out ./myprog.conf

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

   $ mpirun --tune ./myprog.conf -n 32 ./myprog

Alternatively, do the following:

1. **Use the** --rank-placement (-rp),--hostfile-in (-hi) **and** --hostfile-out (-ho) **options. The**
   **tuner** **records the optimized hostlist to** hostfile.out **file:**
   
   $ mpirun --rank-placement --application "mpirun -n 32 ./myprog" --hostfile-out ./hostfile.out

2. **Use the** --machinefile **option to pick up the optimal recorded values for your application at runtime.**

   $ mpirun --machinefile ./myprog.ho -n 32 ./myprog

---

**NOTE**

Use the **--rank-placement** (-rp) **and** --help(-h) **options together to learn other available options for this mode.**

Also, this feature is available at runtime with the **-use-topology-app** option **of the Hydra process manager.** It may significantly increase startup time, but can be more effective because the cluster state at the moment of startup is taken into account (health, resource contentions, and so on). See the **Intel® MPI Library Developer Reference** for details.
## 6. Troubleshooting

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

<table>
<thead>
<tr>
<th>Issue</th>
<th>Cause and Possible Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>The scheduler of <code>mpitune</code> is empty.</strong></td>
<td>1. Check the arguments for <code>mpitune</code> and ensure they do not contradict each other. For example, values for <code>--options-set</code> and <code>--options-exclude</code> should not overlap.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td><strong><code>mpitune</code> runs very long.</strong></td>
<td>Check the projected schedule before the real launch, using the <code>-so</code> option.</td>
</tr>
<tr>
<td></td>
<td>Use methods described in task 1 and task 2 and/or their combinations to skip unnecessary jobs.</td>
</tr>
</tbody>
</table>