



Tutorial: MPI Tuner for Intel® MPI Library

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 which may cause deviations from published specifications.

MPEG-1, MPEG-2, MPEG-4, H.261, H.263, H.264, MP3, DV, VC-1, MJPEG, AC3, AAC, G.711, G.722, G.722.1, G.722.2, AMRWB, Extended AMRWB (AMRWB+), G.167, G.168, G.169, G.723.1, G.726, G.728, G.729, G.729.1, GSM AMR, GSM FR are international standards promoted by ISO, IEC, ITU, ETSI, 3GPP and other organizations. Implementations of these standards, or the standard enabled platforms may require licenses from various entities, including Intel Corporation.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Intel, the Intel logo, BlueMoon, BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino Inside, Cilk, Core Inside, E-GOLD, Flexpipe, i960, Intel, the Intel logo, Intel AppUp, Intel Atom, Intel Atom Inside, Intel Core, Intel Inside, Intel Insider, the Intel Inside logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel Sponsors of Tomorrow., the Intel Sponsors of Tomorrow. logo, Intel StrataFlash, Intel vPro, Intel XScale, Intel True Scale Fabric, InTru, the InTru logo, the InTru Inside logo, InTru soundmark, Itanium, Itanium Inside, MCS, MMX, MPSS, Moblin, Pentium, Pentium Inside, Puma, skool, the skool logo, SMARTi, Sound Mark, Stay With It, The Creators Project, The Journey Inside, Thunderbolt, Ultrabook, vPro Inside, VTune, Xeon, Xeon Phi, Xeon Inside, X-GOLD, XMM, X-PMU and XPOSYS are trademarks of Intel Corporation in the U.S. and/or other countries.

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

Microsoft, Windows, and the Windows logo are trademarks, or registered trademarks of Microsoft Corporation in the United States and/or other countries.

Java is a registered trademark of Oracle and/or its affiliates.

Bluetooth is a trademark owned by its proprietor and used by Intel Corporation under license.

Intel Corporation uses the Palm OS* Ready mark under license from Palm, Inc.

OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos.

© 2015 Intel Corporation.

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

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	<p>This tutorial demonstrates various methods to optimize the performance of Intel® MPI Library for your own cluster and applications, including:</p> <ul style="list-style-type: none">• Minimize the time spent tuning the cluster• 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	<p>After you complete this tutorial, you should be able to:</p> <ul style="list-style-type: none">• 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	<p>To get more information about the MPI Tuner for Intel® MPI Library, see the following resources:</p> <p>Product Web Site</p> <p>Intel® MPI Library Support</p> <p>Intel® Cluster Tools Products</p> <p>Intel® Software Development Products</p>

Prerequisites

Before using the MPI Tuner for Intel® MPI Library, ensure that the library, scripts, and utility applications are installed. See the *Intel® MPI Library for Linux® OS Installation Guide* for installation instructions.

Navigation Quick Start

To use the MPI tuner:

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.

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.`

MPI Tuner Access

To access the MPI Tuner: `$ mpitune`

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

MPI Tuner Commands

The MPI tuner utility operates in four modes:

- Cluster-specific, evaluating a given cluster environment using either the Intel® MPI Benchmarks or a user-provided benchmarking program to find the most suitable configuration of the Intel® MPI Library. This mode is used by default.
- Application-specific, evaluating the performance of a given MPI application to find the best configuration for the Intel® MPI Library for the particular application.
- Fast application-specific mode. In contrast to previous mode, this mode performs micro-kernel tests based on Intel® MPI Library statistics instead of real application launches. This approach is not so careful but can speedup applications with non-typical patterns where non-optimal settings have a chance to be chosen out of the box, such as rank placement, including, process number per host or communicator, messages sizes, and so on. The scope of tuning possibilities is relevant to regular collective operations which have related environment variables in `I_MPI_ADJUST` family, excluding `KN_RADIX` subset and `I_MPI_ADJUST_REDUCE_SEGMENT`. See *the Intel® MPI Library Reference Manual* for details.
- Topology aware rank placement optimization mode, evaluating rank to rank data transfers in the MPI application and the cluster characteristics to find optimal rank placement. This approach has a reputation for applications where communication pattern has some kind of local groups, such as, stencils, collective operations on subsets of ranks, neighborhood operations, and so on.

Cluster-Specific Tuning Commands

To use the MPI tuner under cluster-specific mode:

1. Run the following command to create the tuned configuration files in the default `<installdir>/<arch>/<etc>` directory
`$ mpitune -hf <hostfile>`
or use the `-odr` option instead to create the tuned configuration files in a result directory of your choice
`$ mpitune -hf <hostfile> -odr <path_to_result_directory>`
2. Use the `-tune` option without an argument to pick up config files from the default directory, or with the path to the results directly. For example:
`$ mpirun -tune -ppn 8 -n 128 ./my_app`
`$ mpirun -tune <path_to_result_directory> -ppn 8 -n 128`
`./my_app`

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 will record 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
```

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 regards to:

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

Host Range

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

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

The `mpitune` utility will build all host ranges that are powers of 2 between 4 to 16. Here, it'll create tuned settings for 4 hosts, 8 hosts, and 16 hosts.

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 will build a range of ranks that are powers of 2 between 1 and 16. For example, it will create tuned settings for all cases where the number of ranks is 1, 2, 4, 8, 16.

```
$ mpitune ... -pr 24:24
```

If lower and upper bounds are the same, the `mpitune` utility will tune for the `ppn=24` case only.

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 will run only the enumerated fabrics.

Message Sizes

Use the `-mr` option to set the range of message sizes to be tuned:

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

The `mpitune` utility will tune MPI operations with message sizes that are a power of 2 between the specified bounds of 16 to 2097152 bytes.

Most Common MPI Functions

If you have statistics by usage and performance of MPI functions, you can adjust the tuning scope with regard to your needs. You can find out compliance of various tuning options with MPI functions in *Intel® MPI Library User's Guide* before start.

You can look at the most widely used MPI routines and go from simple to more complex functions. For example, perform p2p tuning before the tuning of collective operations.

Start with the p2p-sensitive options first:

1. Congregate 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 Intel® MPI Library cluster settings based only on the environment variables provided above.

```
$ mpitune ... -os $option_set
```

3. Now tune the collectives:

```
$ 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 further reduce the tuning time, you can specify the percentage improvement needed or exclude those options which show acceptable performance.

Task 2: Include Missed Values in the Default Parameters Grid during Cluster Tuning

The `mpitune` utility enumerates and only tunes values of most variables that are powers of 2. 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 self-configuration. There are two main configuration files which describe *what* and *how* tuning is implemented for 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,tmi" />
        </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="thtime"

                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>

...

```

Task 3: Application-Specific Tuning

Now that you have completed the cluster-specific tuning, you can focus on how to optimize the 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`.

Fast Application-Specific Tuning Commands

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

1. Use the `--fast (-f)` option to switch in this mode, the `--application (-a)` option to tune the specified workload for the provided environment and command line settings or `--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

```
$ 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
```

Topology Awareness Rank Placement Optimization Commands

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

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

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:

```
$ mpitune --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
```

Or

1. Use the `--rank-placement (-rp)`, `--hostfile-in (-hi)` and `--hostfile-out (-ho)` options. The tuner records the optimized hostlist to `hostfile.out` file:

```
$ mpitune --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 right at runtime with `-use-topology-app` option of hydra process manager. It may significantly increase startup time, but can be more effective because cluster state (health, resource contentions, etc) at the moment of startup is taken into account. See *Intel® MPI Library Reference Manual* for details.

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.	<ol style="list-style-type: none">1. Check the arguments for <code>mpitune</code> and ensure they are not contradicting with each other. For example <code>--options-set</code> and <code>--options-exclude</code> should not overlap.2. If no one fabric or device passes checking, try run any MPI test applications on the same configuration for details. The issue might be caused by wrong hostfile or incorrect cluster configuration.
The mpitune running time is very long.	<ol style="list-style-type: none">1. Check the projected schedule before the real launch by using the <code>-so</code> option. Use methods described in task1 and task 2 and/or their combinations to skip unnecessary jobs