MPI TUNING

Michael Steyer

Technical Consulting Engineer

Intel Corporation

Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

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. For more complete information visit www.intel.com/benchmarks.

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

Prerequisite –

Make sure your cluster is properly configured

- Install the latest Intel® MPI Library
- Have the documentation at hand
- When benchmarking try to run on the same subset of nodes
- Understand the performance characteristics of your cluster
  - What fabrics are available, how do they perform
    - What is the lowest achievable latency
    - What is the maximum achievable bandwidth
  - Communication speeds differ
    - Intra-socket / Inter-socket
    - Intra-node / Inter-node
  - Use IMB to determine performance

Prerequisite –

Understand the performance characteristics of your Application

- Simple code internal timings
- ITAC traces – use MPI_Pcontrol() to manage overhead
- APS statistics
- VTune traces

-> know what MPI routines to tune for

Choose the best collective algorithm

Use one of the I_MPI_ADJUST_<opname> knobs to change the algorithm

- Focus on the most critical collective operations (see statistics)
- Use the Intel® MPI Benchmarks by selecting various algorithms to find out the right algorithms for collective operations
- Or use mpitune for automatic (/fast) tuning!

```
I_MPI_ADJUST_<collective>=<algorithm#>
```

Select a proper process pinning 1/3

- The default pinning is suitable for most scenarios
- Set I_MPI_PERHOST or use the –perhost (/-ppn) option to override the default process layout:

```
$ mpirun –ppn <#processes per node> -n <#processes> …
```

- Intel® MPI Library respects the batch scheduler settings - to overwrite use:

```
I_MPI_JOB_RESPECT_PROCESS_PLACEMENT=0
```

- Per-node pinning can also be achieved using a “machinefile”
- Custom processor core pinning can be achieved by two different environment variables

`I_MPI_PIN_PROCESSOR_LIST` - for pure MPI applications

`I_MPI_PIN_DOMAIN` - for Hybrid – MPI + Threading applications

Select a proper process pinning 2/3

- The '**`cpuinfo`**' utility from Intel MPI Library can be used to observe the processor topology
- Threads of Hybrid applications are not pinned by default
- Threads can migrate along the cores of a rank defined by **`I_MPI_PIN_DOMAIN`**
- Therefore, threads should be pinned as well using e.g. **`OMP_PLACES`**
- Further information can be found in the "Process Pinning" section of the Intel MPI Library reference manual

Select a proper process pinning 3/3

Default Intel Library MPI pinning	Impact
I_MPI_PIN=on	Pinning Enabled
I_MPI_PIN_MODE=pm	Use Hydra for Pinning
I_MPI_PIN_RESPECT_CPUSET=on	Respect process affinity mask
I_MPI_PIN_RESPECT_HCA=on	Pin according to HCA socket
I_MPI_PIN_CELL=unit	Pin on all logical cores
I_MPI_PIN_DOMAIN=auto:compact	Pin size #lcores/#ranks : compact
I_MPI_PIN_ORDER=compact	Order domains adjacent

Tuning for numerical stability

Repeatable	Provides consistent results if the application is launched under exactly the same conditions – repeating the run on the same machine- and configuration.
Reproducible (conditionally)	Provides consistent results even if the distribution of ranks differs, while the number of ranks (& #threads for hybrid applications) involved has to be stable. Also, the runtime including the microarchitecture has to be consistent.

In order to achieve conditional reproducibility with the IMPI library:

1) Do not use topologically-aware algorithms inside the collective reduction operations
2) Avoid the “Recursive doubling” algorithm for the MPI_Allreduce operation
3) Avoid MPI_Reduce_scatter_block - as well as the MPI-3 Non-Blocking-Collective operations

Tuning for numerical stability

Which algorithms of the collective reduction operations are topologically-aware?

Collective MPI Operation using Reductions	Intel MPI Library Collective Operation Control Environment	Non-Topologically Aware Algorithms
MPI_Allreduce	I_MPI_ADJUST_ALLREDUCE	(1) , 2 , 3 , 5 , 7 , 8, 9
MPI_Exscan	I_MPI_ADJUST_EXSCAN	1
MPI_Reduce_scatter	I_MPI_ADJUST_REDUCE_SCATTER	1 , 2 , 3 , 4
MPI_Reduce	I_MPI_ADJUST_REDUCE	1 , 2 , 5, [7]
MPI_Scan	I_MPI_ADJUST_SCAN	1

Further information can be found in the Parallel Universe Magazine Issue 21

Notes:
(1) – The first algorithm of MPI_Allreduce is not topologically aware, it does however not guarantee to provide conditionally reproducible results
[7] – The Knomial algorithm provides reproducible results, only if the I_MPI_ADJUST_<COLLECTIVE-OP-NAME>_KN_RADIX environment is kept stable – or unmodified

Tuning for numerical stability

```
program rep
 use mpi
 implicit none
 integer :: n_ranks,rank,errc
 real*8 :: global_sum,local_value

 call MPI_Init(errc)
 call MPI_Comm_size(MPI_COMM_WORLD, n_ranks, errc)
 call MPI_Comm_rank(MPI_COMM_WORLD, rank, errc)

 local_value = 2.0 ** -60

 if(rank.eq.15) local_value= +1.0
 if(rank.eq.16) local_value= -1.0

 call MPI_Reduce(local_value,global_sum,1,MPI_DOUBLE_PRECISION, &
      MPI_SUM,0,MPI_COMM_WORLD, errc)

 if(rank.eq.0) write(*,'(f22.20)') global_sum

 call MPI_Finalize(errc)
end program rep
```

```
$ cat ${machinefile_A}
ehk248:16
ehs146:16
ehs231:16
ehs145:16
$ cat ${machinefile_B}
ehk248:32
ehs146:32
ehs231:0
ehs145:0
$ mpiifort -fp-model strict -o ./rep.x ./rep.f90

$ export I_MPI_ADJUST_REDUCE=3
$ mpirun -n 64 -machinefile ${machinefile_A} ./rep.x
0.00000000000000000000
$ mpirun -n 64 -machinefile ${machinefile_B} ./rep.x
0.00000000000000004163

$ export I_MPI_ADJUST_REDUCE=1
$ mpirun -n 64 -machinefile ${machinefile_A} ./rep.x
0.00000000000000004163
$ mpirun -n 64 -machinefile ${machinefile_B} ./rep.x
0.00000000000000004163
```

I_MPI_CBWR

ARGUMENTS

<arg>	CBWR compatibility mode	Description
0	None	Do not use CBWR in a library-wide mode. CNR-safe communicators may be created with `MPI_Comm_dup_with_info` explicitly. This is the default value.
1	Weak mode	Disable topology aware collectives. The result of a collective operation does not depend on the rank placement. The mode guarantees results reproducibility across different runs on the same cluster (independent of the rank placement).
2	Strict mode	Disable topology aware collectives, ignore CPU architecture, and interconnect during algorithm selection. The mode guarantees results reproducibility across different runs on different clusters (independent of the rank placement, CPU architecture, and interconnection)

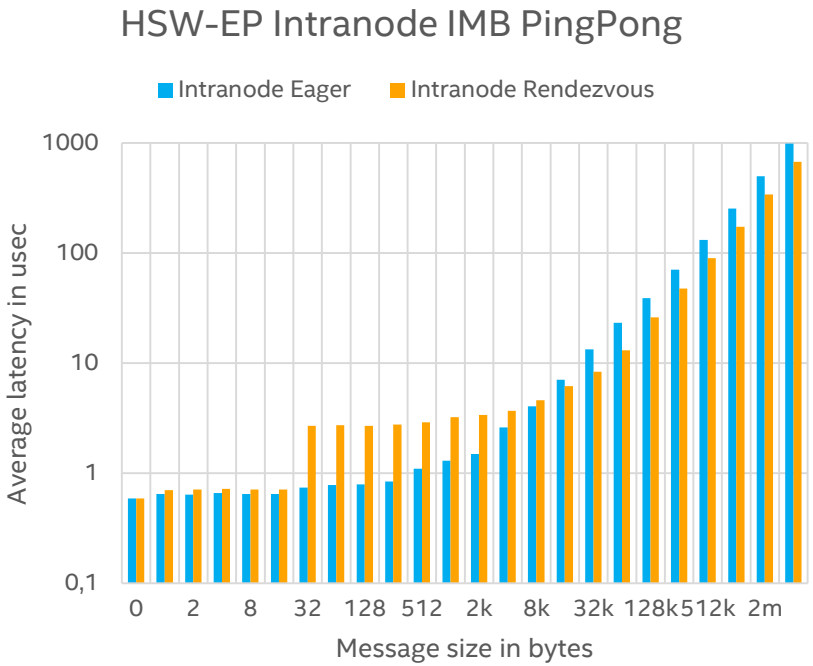
Adjust the eager / rendezvous protocol threshold

Two communication protocols:

1. “Eager” sends data immediately regardless of receive request availability - used for short messages
2. “Rendezvous” notices receiving site on data pending and transfers when receive request is set (RTS/CTS)

I_MPI_EAGER_THRESHOLD

Default is 256k bytes



Enforce asynchronous message transfer for - non-blocking operations

- Overlapping communication and computation is possible by spawning a helper thread
- Can cause oversubscription - therefore deactivated by default

Enabling asynchronous progress

```
I_MPI_ASYNC_PROGRESS=1
```

Pinning the helper thread - takes one rank out of the pinning mask

```
I_MPI_ASYNC_PROGRESS_PIN=1
```

Tune for reduced initialization times at scale 1/2

Make sure to use the latest Intel MPI library as well as the latest PSM2 version

If all ranks work on the same Intel Architecture generation, switch off the platform check:

```
I_MPI_PLATFORM_CHECK=0
```

Specify the processor architecture being used to tune the collective operations:

```
I_MPI_PLATFORM=uniform
```

Alternative PMI data exchange algorithm can help to speed up the startup phase:

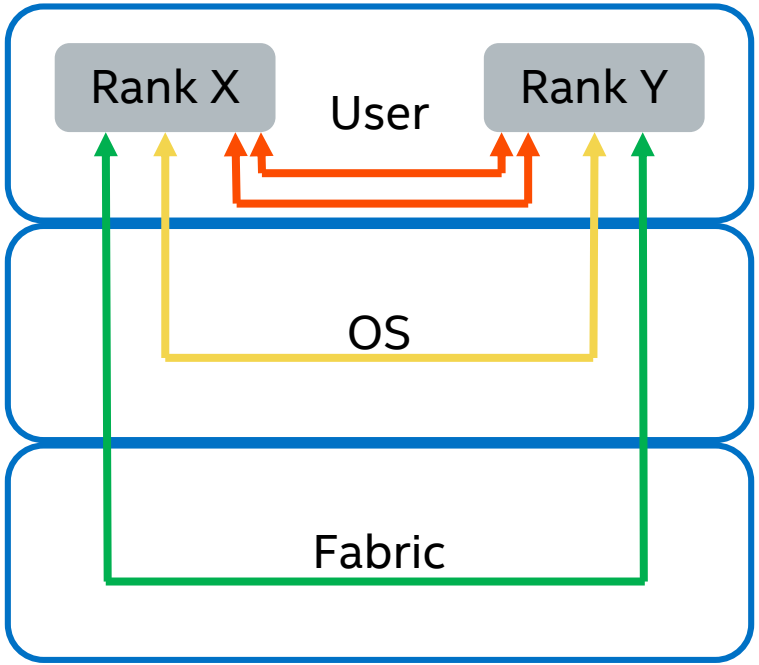
```
I_MPI_HYDRA_PMI_CONNECT=alltoall
```

Customizing the branching may also help startup times (default is 32 for over 127 nodes):

```
I_MPI_HYDRA_BRANCH_COUNT=<n>
```

For further information, read "Reducing Initialization Times of the Intel MPI® Library"

Tuning shared memory



User space – double copy – cache / dram - fast for small messages (eager) [**I_MPI_SHM_CACHE_BYPASS_THRESHOLDS /I_MPI_INTRANODE_EAGER_THRESHOLD**]

Kernel assisted – single copy (CMA – Linux* 3.2) – fast for medium / large (rendezvous) [**I_MPI_SHM_LMT/ I_MPI_INTRANODE_EAGER_THRESHOLD**]

Loopback Fabric – DMA can be performed by HW – might be faster for large messages in certain situations [**I_MPI_SHM_BYPASS/ I_MPI_INTRANODE_EAGER_THRESHOLD**]

Tuning the fabric spinning

Significant portions of CPU cycles spent in IMPI progress engine (CH3/CH4), can be addressed reducing the spin count:

I_MPI_SPIN_COUNT

#times the progress engine spins waiting for a message or connection requests before yielding to the OS. Default - 250

Multi-core platforms with intranode communication dominated executions may benefit from a customized value of the intranode spin count:

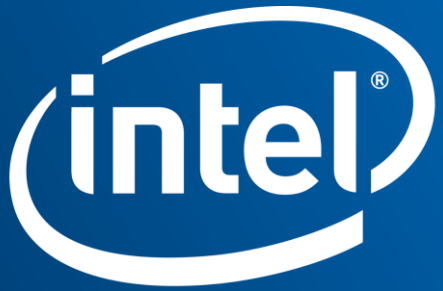
I_MPI_SHM_SPIN_COUNT

Summary

- Intel MPI Library out of box configuration sufficient for most scenarios
- Automatic tuning can help to optimize settings, specifically the collective communication parameter
- Manual tuning takes less time than automatic tuning but has to be applied specifically

```
$ fortune|cowsay
 ________________________________________
/ Tomorrow, this will be part of the     \
| unchangeable past but fortunately, it |
\ can still be changed today.            /
 ----------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Therefore - MPI tuning today!



Software