NEXT GENERATION INTEL MPI PRODUCT FOR NEXT GENERATION SYSTEMS:
INTEL® MPI LIBRARY 2019 FEATURES AND OPTIMIZATION TECHNIQUES
Legal Disclaimer & Optimization Notice

Performance results are based on testing as of August and September 2018 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure.

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.

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.

Copyright © 2019, Intel Corporation. All rights reserved. Intel, Xeon, Core, VTune, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

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
**INTEL® MPI LIBRARY**

Intel® MPI Library is a multifabric message-passing library that implements the open-source MPICH specification. Use the library to create, maintain, and test advanced, complex applications that perform better on HPC clusters based on Intel® processors.

- Develop applications that can run on multiple cluster interconnects chosen by the user at run time.
- Quickly deliver maximum end-user performance without having to change the software or operating environment.
- Achieve the best latency, bandwidth, and scalability through automatic tuning for the latest Intel® platforms.
- Reduce the time to market by linking to one library and deploying on the latest optimized fabrics.

Learn More - [software.intel.com/intel-mpi-library](http://software.intel.com/intel-mpi-library)
Next generation product goals:

- Low instruction count on a critical path
- Better collective operations infrastructure
- Remove non scalable structures
- Better hybrid programming models support (MPI+X)

Intel® MPI Library 2019 key features:

- Based on a new MPICH/CH4/OFI architecture
- Enhanced support for hybrid programming models
- **New** collective operations
- **New** SHM transport
- **New** auto tuning capabilities
- **New** async progress
EXPOSE NATIVE HARDWARE SUPPORT TO THE MPI LAYER
• Reduction in number of instructions (1.5x lower instruction count on MPI levels)
  ▪ CH4 uses functions that can be inlined
  ▪ CH3 was based on function pointers
• Removal of non-scalable data structures
  ▪ Driven by Argonne National Laboratory
  ▪ Optimized data structures used to map ranks in a communicator to a global rank
• Enhanced path for MPI+X (threads) models
• OFI netmod
  ▪ Directly maps MPI constructs to OFI features as much as possible
INTEL® MPI LIBRARY 2018 SW STACK/ECOSYSTEM

MPI high level abstraction layer

CH3

MPI low level transport

DAPL RC
DAPL UD
OFA
TCP
TMI

Each transport layer required independent optimization

HW
Aries
Infiniband
iWarp, RoCE
usNIC
Eth, IPoIB, IPoOPA
OPA
TrueScale
**LIBFABRIC/OPEN FABRIC INTERFACES (OFI) STACK**

- **MPI**
- **SHMEM**
- **PGAS**

Libfabric Enabled Applications

**libfabric**
- **Control**
  - Discovery
- **Communication**
  - Connection Mgmt
  - Address Vectors
- **Completion**
  - Event Queues
  - Counters
- **Data Transfer**
  - Message Queues
  - Tag Matching
  - RMA
  - Atomics

**OFI Provider**
- **Discovery**
- **Connection Mgmt**
- **Address Vectors**
- **Event Queues**
- **Counters**
- **Message Queues**
- **Tag Matching**
- **RMA**
- **Atomics**

**NIC**
- **TX Command Queues**
- **RX Command Queues**

**OFI community**
http://libfabric.org/
INTEL® MPI LIBRARY 2019 SW STACK/ECOSYSTEM

OFI community
http://libfabric.org/

MPI high level abstraction layer

CH4

MPI low level transport

OFI

OFI provider

gni  mlx*  verbs  efa*  tcp  psm2  psm

HW

Aries  Infiniband  iWarp, RoCE  AWS EFA  Eth, IPoIB, IPoOPA  Intel OPA  TrueScale

IMPI 2019 U4 is shipped with prebuilt libfabric 1.7.2 (psm2, verbs, and tcp providers)

* - work in progress
NEW COLLECTIVE OPERATIONS INFRASTRUCTURE

Key Features

▪ New collective operation selection logic
▪ New topology-aware collective operations
▪ New OFI transport direct collective operations

Performance results are based on testing as of June 2019 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure. 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 http://www.intel.com/benchmarks.

Configuration:
Testing by Intel as of June, 2019. 16 nodes, 768 processes.
Hardware: Intel® Xeon® Platinum 8260L CPU @ 2.70GHz; 192 GB RAM. Intel® Turbo Boost Technology and Hyperthreading Technology enabled.
Interconnect: Intel® Omni-Path Host Fabric Interface
Software: RHEL* 7.6; IFS 10.9.0.0.210-957.12.2-2.10.6; Libfabric distributed with Intel® MPI 2019 Update 4; Intel® MPI Library 2019 Update 4
MPI Barrier results are included only in 8 bytes geomean data.
ENHANCED SUPPORT FOR HYBRID PROGRAMMING MODELS

- **New** MPI_THREAD_MULTIPLE model extension
  - Available with release_mt library version: I_MPI_THREAD_SPLIT=1
  - The extension allows to achieve the following:
    - Improve aggregated communication bandwidth & message rate
    - Communicate as soon as data is ready, not waiting for the slowest thread
    - Avoid implied bulk synchronization threading barriers, overhead on parallel sections start/stop
- **New** offload/asynchronous progress engine design

![Diagram of hybrid programming models](image.png)

- **OFI EP** - OFI endpoint
- **HW CX** - Independent HW context
USABILITY FEATURES

- **New** tuning data is separated from the library: https://software.intel.com/en-us/node/807848
- **New** auto tuning capability (**autotuner**)
- **New** spell checker logic
- **New** impi_info tool (**MPI_T** based)

```bash
$ impi_info | head -10
| NAME                                           | DEFAULT VALUE | DATA TYPE |
|==============================================================================|
|I_MPI_PIN                                       | on            | MPI_CHAR  |
|I_MPI_PIN_SHOW_REAL_MASK                        | on            | MPI_INT   |
|I_MPI_PIN_PROCESSOR_LIST                        | not defined   | MPI_CHAR  |
|I_MPI_PIN_PROCESSOR_EXCLUDE_LIST                | not defined   | MPI_CHAR  |
|I_MPI_PIN_CELL                                  | unit          | MPI_CHAR  |
|I_MPI_PIN_RESPECT_CPUSET                        | on            | MPI_CHAR  |
|I_MPI_PIN_RESPECT_HCA                           | on            | MPI_CHAR  |
|I_MPI_PIN_DOMAIN                                | auto:compact  | MPI_CHAR  |
```

```bash
$ I_MPI_PIN_DMAIN=socket mpirun -hosts host01,host02 -n 2 -ppn 1 IMB-MPI barrier
[0] MPI startup(): I_MPI_PIN_DMAIN environment variable is not supported.
[0] MPI startup(): Similar variables:
  I_MPI_PIN_UNIT
  I_MPI_PIN
  I_MPI_PIN_DOMAIN
[0] MPI startup(): To check the list of supported variables, use the impi_info utility or refer to https://software.intel.com/en-us/mpi-library/documentation/get-started.
```
AUTOTUNER
<table>
<thead>
<tr>
<th></th>
<th>mpitune</th>
<th>mpitune/ fast tuner</th>
<th>autotuner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Micro benchmark tuning</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application tuning</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Easy of use</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cluster time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adoption to environment</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**INTEL® MPI LIBRARY 2019 AUTOTUNER TUNING FLOW**

- **Execution timeline**
  - **MPI_Allreduce** → 1<sup>st</sup> invocation: \( I_{\text{MPI_ADJUST_ALLREDUCE}}=0 \)
  - **MPI_Allreduce** → 2<sup>nd</sup> invocation: \( I_{\text{MPI_ADJUST_ALLREDUCE}}=1 \)
  - ... 
  - **MPI_Allreduce** → k<sup>-th</sup> invocation: \( I_{\text{MPI_ADJUST_ALLREDUCE}}=\text{algo_id_max} \)
  - **MPI_Allreduce** → (k+1)-th invocation: \( I_{\text{MPI_ADJUST_ALLREDUCE}}=\text{best_algo_id} \)
  - ... 
  - **MPI_Allreduce** → N<sup>-th</sup> invocation: \( I_{\text{MPI_ADJUST_ALLREDUCE}}=\text{best_algo_id} \)

- No extra calls. Pure **application driven** tuning
- The procedure is performed for each message size and for each communicator
Each communicator has its own tuning. (E.g. COMM_1 and COMM_2 have independent tuning)
GET STARTED WITH AUTOTUNER

Step 1 – Enable autotuner and store results (store is optional):

$ export I_MPI_TUNING_AUTO=1
$ export I_MPI_TUNING_BIN_DUMP=./tuning_results.dat
$ mpirun -n 96 -ppn 48 IMB-MPI1 allreduce -iter 1000,800 -time 4800

Step 2 – Use the results of autotuner for consecutive launches (optional):

$ export I_MPI_TUNING_BIN=./tuning_results.dat
$ mpirun -n 96 -ppn 48 IMB-MPI1 allreduce -iter 1000,800 -time 4800

NOTE: You may adjust number of tuning iterations (minimal overhead/maximum precision balance) and use autotuner with every application run without results storing.
ENVIRONMENT VARIABLES. MAIN FLOW CONTROL

I_MPI_TUNING_AUTO=<0|1> Enable autotuner (disabled by default)

I_MPI_TUNING_AUTO_ITER_NUM=<number> Tuning iterations number (1 by default).

I_MPI_TUNING_AUTO_SYNC=<0|1> Call internal barrier on every tuning iteration (disabled by default)

I_MPI_TUNING_AUTO_WARMUP_ITER_NUM=<number> Warmup iterations number (1 by default).

NOTE: Assume that there are around 30 algorithms to be iterated. E.g. Application has 10000 invocations of MPI_Allreduce 8KB. For full tuning cycle I_MPI_TUNING_AUTO_ITER_NUM may be in 30 to 300 (if there is no warmup part) range. High value is recommended for the best precision. Iteration number for large messages may depend on I_MPI_TUNING_AUTO_ITER_POLICY_THRESHOLD.
I_MPI_TUNING_AUTO_SYNC is highly recommended for tuning file store scenario.
ENVIRONMENT VARIABLES. TUNING SCOPE AND STORAGE CONTROL

$I_{\text{MPI\_TUNING\_AUTO\_COMM\_LIST}}=<\text{comm\_id\_1, ... , comm\_id\_k}>$ List of communicators to be tuned (all communicators by default)

$I_{\text{MPI\_TUNING\_AUTO\_COMM\_USER}}=<0|1>$ Enable user defined comm_id through MPI_Info object. (disabled by default)

$I_{\text{MPI\_TUNING\_AUTO\_COMM\_DEFAULT}}=<0|1>$ Default/universal comm_ids. (disabled by default)

$I_{\text{MPI\_TUNING\_AUTO\_STORAGE\_SIZE}}=<\text{size}>$ Max per-communicator tuning storage size (512KB by default)

NOTE: You may use Intel® VTune™ Amplifier's Application Performance Snapshot for per communicator MPI cost analysis and narrow tuning scope.

$I_{\text{MPI\_TUNING\_AUTO\_COMM\_DEFAULT}}$ disables comm_id check (allows to get universal tuning)
ENVIRONMENT VARIABLES. TUNING POLICY

I_MPI_TUNING_AUTO_ITER_POLICY=<0|1> Adaptive iterations number mode. (enabled by default)

I_MPI_TUNING_AUTO_ITER_POLICY_THRESHOLD=<msg_size> Message size threshold for I_MPI_TUNING_AUTO_ITER_POLICY. (64KB by default).

I_MPI_TUNING_AUTO_POLICY=<max|min|avg> Autotuning strategy. (Use “max” time by default)

NOTE: I_MPI_TUNING_AUTO_ITER_POLICY_THRESHOLD halves number of iterations. E.g. If I_MPI_TUNING_AUTO_ITER_NUM=256, then for 512KB message size number of iterations will be 32
1. Source apsvars.sh:
   $ source <path_to_aps>/apsvars.sh

2. Gather APS statistics:
   $ export MPS_STAT_LEVEL=5
   $ export APS_COLLECT_COMM_ID=1
   $ mpirun -n 4 -ppn 2 aps IMB-MPI1 allreduce -iter 1000,800

3. Generate an APS report:
   $ aps-report aps_result_20190228/ -lFE

https://software.intel.com/sites/products/snapshots/application-snapshot/

Available with Intel® VTune™ Amplifier’s Application Performance Snapshot Update 4
**INTEL® VTUNE™ AMPLIFIER’S APPLICATION PERFORMANCE SNAPSHOT (APS) PER COMMUNICATOR ANALYSIS**

4. Get the results:

<table>
<thead>
<tr>
<th>Communicators used in the application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communicator Id</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>4611686018431582688</td>
</tr>
<tr>
<td>4611686018431582208</td>
</tr>
<tr>
<td>4611686018429485552</td>
</tr>
<tr>
<td>4611686018429485520</td>
</tr>
<tr>
<td>4611686018431582672</td>
</tr>
</tbody>
</table>
5. Specify communicators to be tuned:

$ export I_MPI_TUNING_AUTO_COMM_LIST=4611686018431582688
$ export I_MPI_TUNING_AUTO=1
$ mpirun -n 96 -ppn 48 IMB-MPI1 allreduce -iter 1000,800 -time 4800

NOTE: I_MPI_TUNING_AUTO_ITER_POLICY may impact tuning cycle for large messages. Please check that you have enough application level invocations
MULTIPLE ENDPOINTS/ASYNCRONOUS PROGRESS
MULTIPLE ENDPOINTS BASED FEATURES IN INTEL® MPI LIBRARY 2019

- **Thread-split**
  - Decrease threading computation imbalance - communicate as soon as data is ready, don't wait for the slowest thread
  - Improve interconnect saturation from single MPI rank (Intel® Omni Path Fabric, InfiniBand and Ethernet are supported)
  - Avoid implied bulk synchronization threading barriers and overhead on parallel sections start/stop

- **Asynchronous progress threads**
  - Allows to offload communication from application threads to MPI progress threads
  - Allows to improve computation/communication overlap
  - Allows to parallelize communication by multiple MPI progress threads

Both features are available only for:
- Linux
- `I_MPI_FABRICS=ofi`
- `release_mt` (non default version)
Thread-split

Asynchronous progress threads

AT<N> - Application Thread #N
MPT<N> - MPI Progress Thread #N
1. `#define N 2
2. 
3. int main() {
4.   int i;
5.   int buffer[N];
6. 
7. MPI_Init(NULL, NULL);
8. 
9. 
10. #pragma omp parallel for num_threads(N)
11. for (i = 0; i < N; i++) {
12.   // threaded partial computation
13.   // i-th thread contributes to buffer[i]
14. 
15. 
16. 
17. 
18. 
19. }
20. // single-threaded global communication
21. MPI_Allreduce(buffer, buffer, N, MPI_INT, 
22.                 MPI_SUM, MPI_COMM_WORLD);
23. 
24. MPI_Finalize();
25. return 0;
26. }

1. `#define N 2
2. 
3. int main() {
4.   int i, provided;
5.   int buffer[N];
6. 
7. MPI_Comm comm[N];
8. MPI_Init_thread(NULL, NULL, MPI_THREAD_MULTIPLE, &provided);
9. for (i = 0; i < N; i++)
10.   MPI_Comm_dup(MPI_COMM_WORLD, &comm[i]);
11. 
12.#pragma omp parallel for num_threads(N)
13. for (i = 0; i < N; i++) {
14.   // threaded partial computation
15.   // i-th thread contributes to buffer[i]
16. 
17. 
18. 
19. 
20.   // threaded partial communication
21.   MPI_Allreduce(&buffer[i], &buffer[i], 1, MPI_INT, 
22.                   MPI_SUM, comm[i]);
23. 
24. }
25. MPI_Finalize();
26. return 0;
27.}
ASYNCHRONOUS PROGRESS THREADS – STRONG SCALING CODE MODIFICATIONS

1. #define N (4)
2. #define CHUNK_SZ (1024)
3.
4. int main()
5. {
6.   MPI_Request request;
7.   MPI_Status status;
8.   int sbuf[N*CHUNK_SZ], rbuf[N*CHUNK_SZ];
9.   int idx;
10.  MPI_Init(NULL, NULL);
11.  
12.  MPI_INFO info;
13.  MPI_Comm comms[N];
14.  char thread_id_str[16];
15.  MPI_Info_create(&info);
16.  
17.  for (idx = 0; idx < N; idx++) {
18.     MPI_Comm_dup(MPI_COMM_WORLD, &comms[idx]);
19.     sprintf(thread_id_str, "%d", idx);
20.     MPI_Info_set(info, "thread_id", thread_id_str);
21.     MPI_Comm_set_info(comms[idx], info);
22.  }
23.  MPI_Info_free(&info);
24.  
25.  for (idx = 0; idx < N; idx++) {
26.     MPI_Iallreduce(send_buffer + idx * CHUNK_SZ,
27.                     recv_buffer + idx * CHUNK_SZ,
28.                     CHUNK_SZ, MPI_INT, MPI_SUM,
29.                     comms[idx], &requests[idx]);
30.  }
31.  
32.  MPI_Waitall(N, requests, statuses);
33.  MPI_Finalize();
34.  return 0;
DOCUMENTATION

Developer Guide

Developer Reference

Code examples
$1_MPI_ROOT/doc/examples
$ source <impi_2019_install_path>/intel64/bin/mpivars.sh release_mt

- 1 thread:

$ OMP_NUM_THREADS=1 I_MPI_THREAD_SPLIT=1 I_MPI_THREAD_RUNTIME=openmp mpirun -n 2 -ppn 1 -hosts host1,host2 IMB-MT -thread_level multiple -datatype char sendrecvmt -count 32768,524288,8388608 -repeat 1000

- 4 threads:

$ OMP_NUM_THREADS=4 I_MPI_THREAD_SPLIT=1 I_MPI_THREAD_RUNTIME=openmp mpirun -n 2 -ppn 1 -hosts host1,host2 IMB-MT -thread_level multiple -datatype char sendrecvmt -count 8192,131072,2097152 -repeat 1000

Performance results are based on testing as of June 2019 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure. 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 http://www.intel.com/benchmarks.

Configuration:
Testing by Intel as of June, 2019. 2 nodes
Hardware: Intel® Xeon® Gold 6252 CPU @ 2.10GHz; 192 GB RAM. Intel® Turbo Boost Technology and Hyperthreading Technology enabled.
Interconnect: Intel® Omni-Path Host Fabric Interface
Software: RHEL® 7.6; IFS 10.9.0.0.210-957.12.2-2.10.6; Libfabric distributed with Intel® MPI 2019 Update 4; Intel® MPI Library 2019 Update 4
$ source <impi_2019_install_path>/intel64/bin/mpivars.sh release_mt

- 1 thread:

$ OMP_NUM_THREADS=1 OMP_PLACES=cores I_MPI_THREAD_SPLIT=1 I_MPI_THREAD_RUNTIME=openmp mpirun -n 2 -ppn 1 -hosts host1,host2 IMB-MT -thread_level multiple bibandmt -count 4,8,16,64,256,1024,4096 -repeat 10000 -datatype char -window_size 64

- 4 threads:

$ OMP_NUM_THREADS=4 ... -window_size 16

Performance results are based on testing as of June 2019 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure. 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 http://www.intel.com/benchmarks.

Configuration:
Testing by Intel as of June, 2019. 2 nodes
Hardware: Intel® Xeon® Gold 6252 CPU @ 2.10GHz; 192 GB RAM. Intel® Turbo Boost Technology and Hyperthreading Technology enabled.
Interconnect: Intel® Omni-Path Host Fabric Interface
Software: RHEL® 7.6; IFS 10.9.0.0.210-957.12.2-2.10.6; Libfabric distributed with Intel® MPI 2019 Update 4; Intel® MPI Library 2019 Update 4
This is HPC on Intel

ASYNCHRONOUS PROGRESS THREADS. IALLREDUCE

$ source <impi_2019_install_path>/intel64/bin/mpivars.sh release_mt

▪ default

$ I_MPI_PIN_PROCESSOR_LIST=4 mpirun -n 2 -ppn 1 -hosts host1,host2 IMB-NBC -thread_level multiple iallreduce -msglog 15:20

▪ 1 thread:

$ I_MPI_ASYNC_PROGRESS=1 I_MPI_ASYNC_PROGRESS_PIN=10 ...

Performance results are based on testing as of June 2019 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure. 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 http://www.intel.com/benchmarks.

Configuration:
Testing by Intel as of June, 2019. 2 nodes
Hardware: Intel® Xeon® Gold 6252 CPU @ 2.10GHz; 192 GB RAM. Intel® Turbo Boost Technology and Hyperthreading Technology enabled.
Interconnect: Intel® Omni-Path Host Fabric Interface
Software: RHEL® 7.6; IFS 10.9.0.0.210-957.12.2-2.10.6; Libfabric distributed with Intel® MPI 2019 Update 4; Intel® MPI Library 2019 Update 4
USAGE IN APPLICATIONS

Thread-split:
- BQCD

Asynchronous progress threads:
- Intel® Machine Learning Scaling Library (https://github.com/intel/MLSL)
- GeoFEM
INTEL® MACHINE LEARNING SCALING LIBRARY (MLSL)
DISTRIBUTED DEEP LEARNING REQUIREMENTS

✓ Compute/communication overlap
✓ Choosing optimal communication algorithm
✓ Prioritizing latency-bound communication
✓ Portable / efficient implementation
✓ Ease of integration with quantization algorithms
✓ Integration with Deep Learning Frameworks
✓ Fusion, persistent operations, etc.
Some of the Intel® MLSL features include:

- Built on top of Intel MPI: transparently supports various interconnects: Intel® OPA, InfiniBand*, and Ethernet;
- Optimized to drive scalability of DL communication patterns
- Ability to trade off compute for communication performance – beneficial for communication-bound scenarios
- New domain-specific features are coming soon

Integrated into Horovod*, Intel® Optimization for Caffe* and nGraph

https://github.com/01org/MLSL/releases
DEEP LEARNING TRAINING IN CLOUD

Multi-node scaling with Intel® Optimization for Caffe

Higher is Better

<table>
<thead>
<tr>
<th>Relative throughput</th>
<th>1 node (1 x C5.18xlarge)</th>
<th>32 nodes (32 x C5.18xlarge)</th>
<th>64 nodes (64 x C5.18xlarge)</th>
<th>128 nodes (128 x C5.18xlarge)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception-v3 (Imagenet-2012)</td>
<td>1x 29.4x</td>
<td>58.4x</td>
<td>60.5x</td>
<td></td>
</tr>
<tr>
<td>Resnet-50 (Imagenet-2012)</td>
<td>1x 30.5x</td>
<td>60.5x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Intel® Xeon® Platinum 8124M CPU @ 3.00GHz

Performance results are based on testing as of Aug 21, 2018 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure. Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests 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: http://www.intel.com/performance

Optimization Notice: Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors or microprocessor-based devices. 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.
## Configuration Details

<table>
<thead>
<tr>
<th>Date of Testing</th>
<th>Intel tested as of Aug 21st 2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benchmark Segment</td>
<td>AI/ML</td>
</tr>
<tr>
<td>Benchmark Type</td>
<td>Training</td>
</tr>
<tr>
<td>Benchmark Metric</td>
<td>Images/Sec</td>
</tr>
<tr>
<td>Framework</td>
<td>Intel® Optimization for Caffe</td>
</tr>
<tr>
<td>Topology</td>
<td>Resnet-50 &amp; Inception-v3</td>
</tr>
<tr>
<td># of Nodes</td>
<td>1/32/64/128</td>
</tr>
<tr>
<td>Platform</td>
<td>Amazon EC2 C5.18xlarge instance</td>
</tr>
<tr>
<td>Sockets</td>
<td>25</td>
</tr>
<tr>
<td>Processor</td>
<td>Intel® Xeon® Platinum 8124M CPU @ 3.00GHz</td>
</tr>
<tr>
<td>BIOS</td>
<td>N/A</td>
</tr>
<tr>
<td>Enabled Cores</td>
<td>18 cores / socket</td>
</tr>
<tr>
<td>Platform</td>
<td>N/A</td>
</tr>
<tr>
<td>Slots</td>
<td>N/A</td>
</tr>
<tr>
<td>Total Memory</td>
<td>144GB</td>
</tr>
<tr>
<td>Memory Configuration</td>
<td>N/A</td>
</tr>
<tr>
<td>SSD</td>
<td>EBS Optimized 512GB, Provisioned IOPS SSD</td>
</tr>
<tr>
<td>OS</td>
<td>Amazon Linux 2 AMI (HVM)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network Configurations</th>
<th>Amazon Elastic Network Adapter (ENA)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>25 Gbps of aggregate network bandwidth</td>
</tr>
<tr>
<td></td>
<td>Placed all instances into the same placement group</td>
</tr>
<tr>
<td>HT</td>
<td>ON</td>
</tr>
<tr>
<td>Turbo</td>
<td>ON</td>
</tr>
<tr>
<td>Computer Type</td>
<td>Server</td>
</tr>
<tr>
<td>Framework Version</td>
<td>Intel® Optimization for Caffe version 1.1.1</td>
</tr>
<tr>
<td>Batch size</td>
<td>ResNet-50 : 128 x # of node</td>
</tr>
<tr>
<td></td>
<td>Inception-v3 : 64 x # of node</td>
</tr>
<tr>
<td>Dataset, version</td>
<td>Imagenet, ILSVRC 2012</td>
</tr>
<tr>
<td>MKLDNN</td>
<td>464c268e544bae26f9b85a2acb9122c766a4c396</td>
</tr>
<tr>
<td>MKL</td>
<td>mklml_lnx_2018.0.1.20171227</td>
</tr>
<tr>
<td>MLSL</td>
<td>l_mlsl_2018.0.0.003</td>
</tr>
<tr>
<td>Compiler</td>
<td>gcc/g++: 4.8.5</td>
</tr>
<tr>
<td></td>
<td>icc/icpc: 17.0.5</td>
</tr>
</tbody>
</table>