HIGH-PERFORMANCE THREADED INTEL® MPI LIBRARY ON INTEL® OMNI-PATH ARCHITECTURE: UNDER THE HOOD

Intel® HPC Developer Conference 2017, Denver
Michael Chuvelev, Intel Corporation
Larry Meadows, Intel Corporation
Agenda

• Hybrid MPI + threads Application Programming Models
• New MPI-MT Programming Model Introduction
• High Performance MPI-MT Stack
• Performance Results on Intel® Omni-Path Architecture
MPI + Threads Programming Models

Funneled Model

MPI_Init_thread(MPI_THREAD_SINGLE)
The best MPI can do now

True Multi-threaded Model (MPI-MT)

MPI_Init_thread(MPI_THREAD_MULTIPLE)
Why is MPI so slow ???

Communication thread(s) model

MPI_Init_thread(MPI_THREAD_SINGLE) or
MPI_Init_thread(MPI_THREAD_MULTIPLE)

Timeline diagram: Light green arrow - rank, Dotted arrow - thread, Blue - computation, Red - MPI communication, Connectors - parallel region
MPI in Multi-threads - What's Wrong

MPI 3.1 Standard: "threads are not separately addressable: a rank in a send or receive call identifies a process, not a thread"

- Any thread may send/receive a message to a peer
- Single global receive queue has to be supported
  - Global, or fine grained lock imposed
  - Overheads are more than just serialization
- MPI objects are global
  - Object management overhead incurred
MPI-MT: How to Get Performance

- Reduced/narrowed MPI_THREAD_MULTIPLE model introduced: `MPI_THREAD_SPLIT` (or *thread-split* model)
- Non-standard, available in Intel MPI with `I_MPI_THREAD_SPLIT=1`
- Threads are addressed with a dedicated communicator - now separate per-thread queues possible
- Matching the same message from concurrent threads not allowed
- Explicit/implicit thread addressing sub-models apply (next slide)
- MPI request objects can't be accessed from >1 thread - now can be in TLS
- No multi-threads access penalty
Thread-split Model: Explicit vs. Implicit (1/2)

• Explicit: using communicator info key to address particular thread
  • General thread support: OpenMP* (incl. tasks), pthreads, Intel® Threading Building Blocks (Intel® TBB) etc

• Implicit: using OpenMP* thread number to address particular thread
  • OpenMP* parallel support only
  • communicator & tag might be used for matching at different threads
Thread-split Model: Explicit vs. Implicit (2/2)

**Implicit sub-model**
Each thread communicates via dedicated comm (Comm A, B)
Each comm is associated with specific thread number
No info key needs to be passed to identify peer threads

**Explicit sub-model**
Each thread communicates via dedicated comm (Comm A, B)
Comm is not associated with specific thread number
Info key needs to be passed to identify peer threads
Implicit sub-model

```c
#define n 2
MPI_Comm split_comm[n];
int main() {
    int i, j, provided;
    MPI_Init_thread(NULL, NULL, MPI_THREAD_MULTIPLE, &provided);
    for (i = 0; i < n; i++)
        MPI_Comm_dup(MPI_COMM_WORLD, &split_comm[i]);
    #pragma omp parallel for private(j) num_threads(n)
    for (i = 0; i < n; i++)
        MPI_Allreduce(&i, &j, 1, MPI_INT, MPI_SUM, split_comm[i]);
    MPI_Finalize();
}
export OMP_NUM_THREADS=2
export I_MPI_THREAD_SPLIT=1
```

Explicit sub-model

```c
#define n 2
int thread_id[n];
MPI_Comm split_comm[n];
pthread_t thread[n];
void *worker(void *arg) {
    int i = *((int*)arg), j;
    MPI_Allreduce(&i, &j, 1, MPI_INT, MPI_SUM, split_comm[i]);
}
int main() {
    ...,
    MPI_Init_thread(NULL, NULL, MPI_THREAD_MULTIPLE, &provided);
    MPI_Info_create(&info);
    for (i = 0; i < n; i++)
        MPI_Comm_dup(MPI_COMM_WORLD, &split_comm[i]);
    for (i = 0; i < n; i++)
        MPI_Comm_set_info(split_comm[i], info);
    for (i = 0; i < n; i++)
        pthread_create(&thread[i], NULL, worker, (void*) &thread_id[i]);
    ...,
    export I_MPI_THREAD_MAX=2
    export I_MPI_THREAD_SPLIT=1
```
Application Performance Impact

• What happens if applications could call MPI from threads with single-threaded efficiency?

• Improved aggregated communication bandwidth & message rate, including leverage of multiple HFIs (Host Fabric Interface)/ports, even with 1 rank per node

• Hiding threading computation imbalance - communicating as soon as data is ready, not waiting for the slowest thread

• Avoiding implied bulk synchronization threading barriers, overhead on parallel sections start/stop
High Performance MPI-MT Stack

- Intel® MPI Library 2019 (MPICH*-CH4 based)
- TO be implemented
- OFI 1.5.0 Scalable EP over PSM2 Multi-EP
- IFS 10.5 PSM2 Multi-EP
- OFI - Open Fabrics Interfaces (Open Fabrics, libfabric); IFS - Intel® Omni-Path Host Fabric Interface Products; PSM2 - Intel® Performance Scaled Messaging 2
MPI-MT Stack on Intel® Omni-Path Architecture

- PSM2: Lowest-level interface to Intel® OPA hardware
  - New library allows multiple endpoints (Multi-EP) to be opened within a single process
  - User is responsible for ensuring one endpoint per thread
- Open Fabrics
  - Multi-EP is implemented with Scalable Endpoints
  - Uses PSM2, similar constraints on threading
- Intel® MPI Library
  - Multi-EP support at the MPI level currently implemented only in the Intel® MPI Library.
Intel® MPI Benchmarks: IMB-MT Exchange

The new IMB 2019 set supports true multi-threaded parallelism under the thread-split model. Exchange benchmark represents a one-dimensional exchange (every rank exchanges with a left-hand and a right-hand neighbor) on an 8 node system. The message size is 1MB.

Weak scaling (on the left): total data size per thread remains constant.

Strong scaling (on the right): total data size per node remains constant.

Hardware:
- Intel® Xeon Phi™ 7250 processor
- Intel® Omni-Path Host Fabric Interface, dual-rail

Software:
- Intel® MPI Library 2018
- Intel® MPI Library 2019 Technical Preview
- OFI 1.5.0
- PSM2 w/ Multi-EP support
- Intel® MPI Benchmarks 2019

1 The new IMB 2019 set supports true multi-threaded parallelism under the thread-split model. Exchange benchmark represents a one-dimensional exchange (every rank exchanges with a left-hand and a right-hand neighbor) on an 8 node system. The message size is 1MB.

Weak scaling (on the left): total data size per thread remains constant.

Strong scaling (on the right): total data size per node remains constant.

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/performance. Copyright © 2017, Intel Corporation
Grid QCD Performance Benchmark (courtesy of Prof. Peter Boyle)

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/performance. Copyright © 2017, Intel Corporation

Hardware:
- Intel® Xeon® Platinum 8170 processor
- Intel® Omni-Path Host Fabric Interface, dual-rail Software:
- Intel® MPI Library 2017 Update 4
- Intel® MPI Library 2019 Technical Preview
- OFI 1.5.0
- PSM2 w/ Multi-EP
- Grid benchmark\(^1\)
- huge memory pages

\(^1\) The benchmark software uses the Grid QCD library https://github.com/paboyle/Grid and the performance benchmark benchmarks/Benchmark_comms.cc is used. The tests are run on 16 nodes system is divided into a four dimensional Cartesian communicator with 24 ranks, and one rank per node. These dimensions are called \((x, y, z, t)\). Each node is given a four dimensional volume \(L^4\), and the global volume is \(G^4 = (2L)^4\). The code mimics the communications pattern for a halo exchange PDE arising in quantum chromodynamics. Each node sends packets of size the surface \(L^3\) to neighbors in each of \(+x, -x, +y, -y, +z, -z, +t, \text{ and } -t\) directions, while concurrently receiving the neighbors data.
Baidu Research SGD Reduction
(courtesy of Prof. Peter Boyle)

Wall clock time per reduction call vs. vector length before and after optimization

Hardware:
- Intel® Xeon® Platinum 8170 processor
- Intel® Omni-Path Host Fabric Interface, dual-rail
Software:
- Intel® MPI Library 2017 Update 4
- Intel® MPI Library 2019 Technical Preview
- OFI 1.5.0
- PSM2 w/ Multi-EP
- Baidu optimized reduction¹
- huge memory pages

Summary

Intel® MPI Library with Multi-EP allows a single multi-threaded rank per node to achieve near line rate on Intel® Omni-Path Architecture single- and dual-rail over a variety of communication patterns.

Intel® MPI Library with Multi-EP provides efficient multi-threaded support, allowing significant improvement in hybrid MPI+OpenMP* applications by using MPI calls inside of parallel regions.

Open Fabrics Scalable Endpoints expose convenient interface for any provider to leverage the best MPI performance from multiple threads.
Legal Disclaimer & Optimization Notice

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.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

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

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

© 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
BACKUP
Useful Links

Intel® MPI Library: https://software.intel.com/en-us/intel-mpi-library

MPICH*: https://www.mpich.org


Open Fabrics Interfaces (OFI, libfabric): https://github.com/ofiwg/libfabric

Intel® MPI Benchmarks (IMB): https://github.com/intel/mpi-benchmarks