Scalable Machine Learning with OpenSHMEM

Gerard Taylor, ***David Ozog***, Md. Wasi-ur- Rahman, James Dinan

Parallel Applications Workshop, Alternatives to MPI (PAW-ATM)

November 17, 2019

Legal Disclaimers

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps. Intel provides these materials as-is, with no express or implied warranties.

This document contains information on products in the design phase of development which Intel may change at any time without notice. Do not finalize a design with this information. Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Intel® , Intel logo and Xeon® are trademarks or registered 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 technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com].

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

WARNING - This document contains technical data whose export is restricted by the Arms Export Control Act (Title 22, U.S.C., Sec 2751, etq.) or the Export Administration Act of 1979, as amended, Title 50, U.S.C., App. 2401 et seq. Violations of these export laws are subject to severe criminal penalties. Disseminate in accordance with provisions of DoD Directive 5230.25.

This document contains information on products in the design phase of development which Intel may change at any time without notice. Do not finalize a design with this information. Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Copyright © 2019 Intel Corporation.

Optimization Notice

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 SSE3 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

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors.

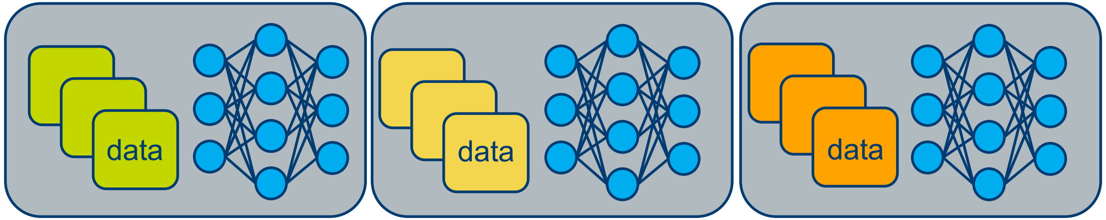
Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

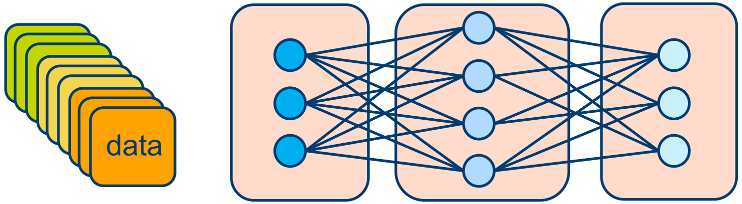
Test and System Configurations: Sandia OpenSHMEM running on Diamond cluster (Intel® Omni-Path 100 Series, Intel® Xeon Platinum 8170 (Skylake), and Cori, Intel®Xeon E5-2698 v3 (Haswell)

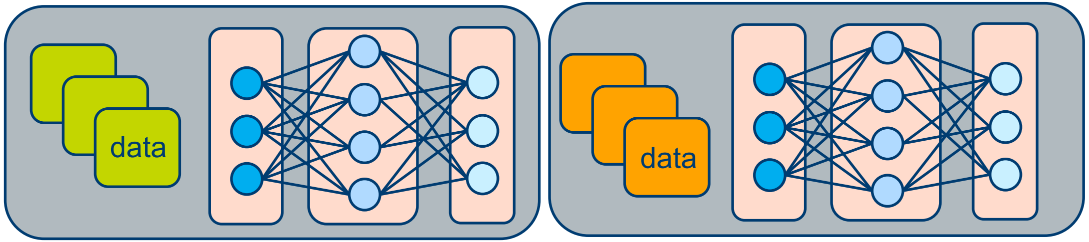
Performance results are based on testing as of Sept 19, 2019 and may not reflect all publicly available security updates. See configuration disclosure for details. No component or product can be absolutely secure.

Distributed Neural Network Training

- Three Types of Parallelism
  - Data Parallelism (Most Common, Memory Heavy)
  - Model Parallelism (Less Common, Communication Heavy)
  - Hybrid Parallelism (Combination of Model and Data Parallel)

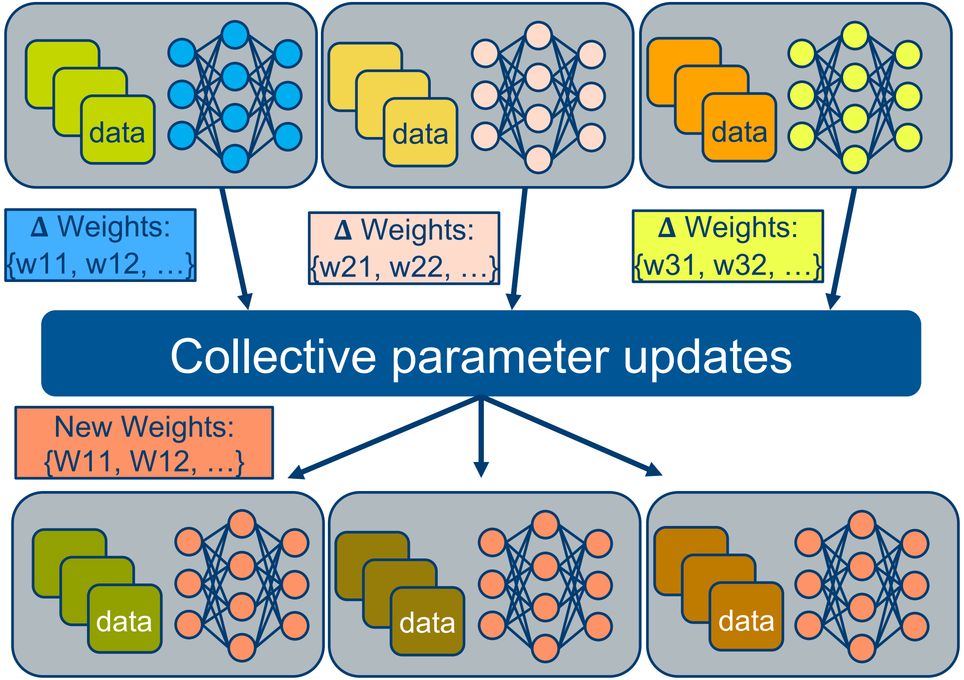






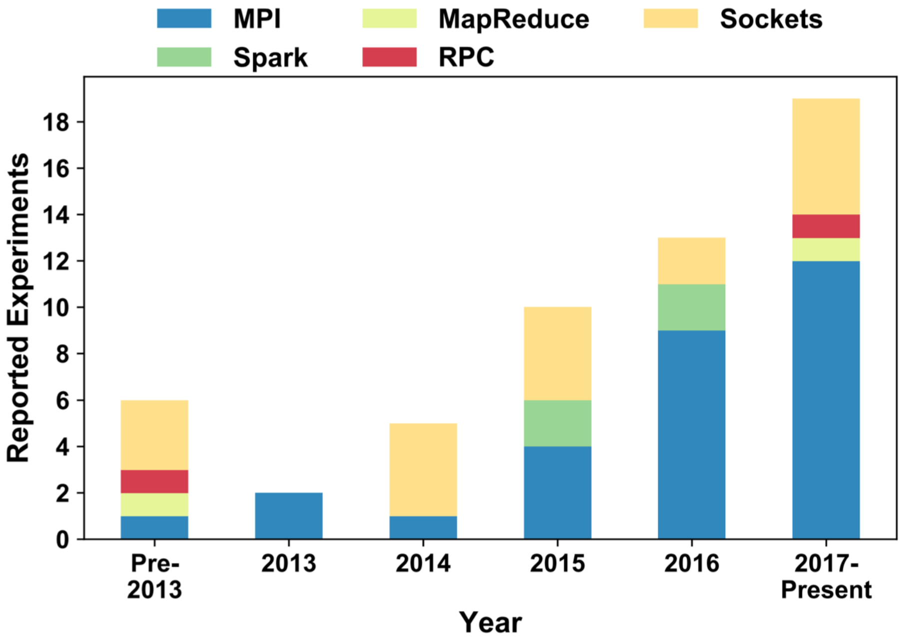
Distributed Stochastic Gradient Descent (SGD)

- SGD is an algorithm used in DNN training
  - Backpropagate error gradient(s)
  - Agree on model update(s)
- Various approaches:
  - Centralized vs. de-centralized
  - Parameter averaging vs. updating
  - Synchronous vs. asynchronous
- **Common theme:**
  - Large-scale training relies on the performance of *collective communication* (broadcast / all-reduce).



DNN Communication Models

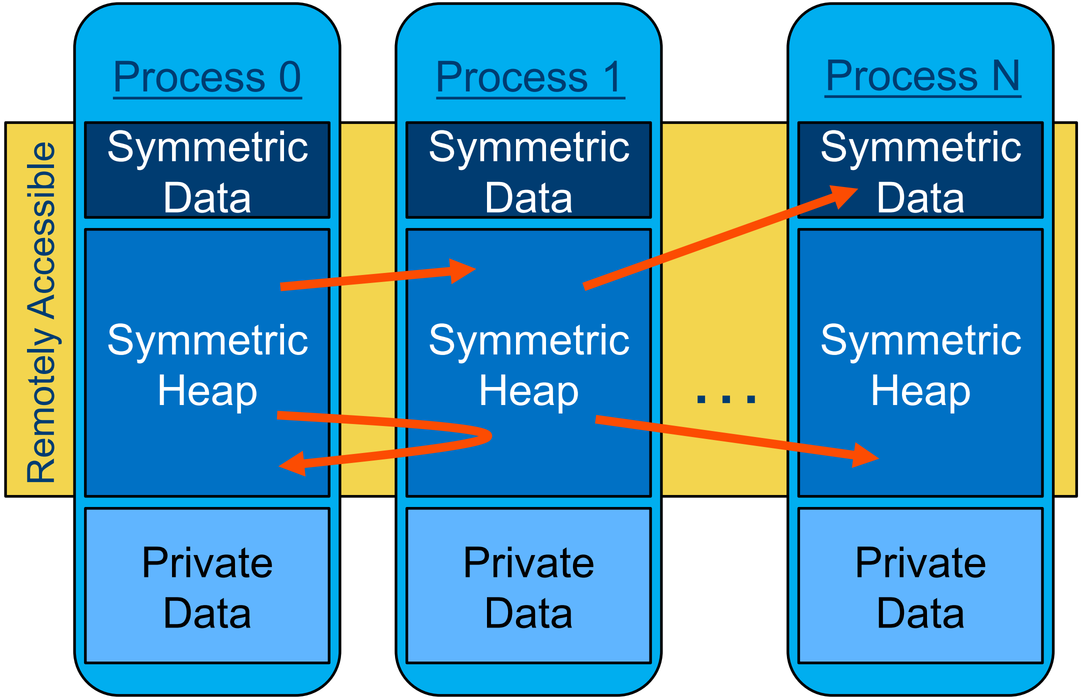
- In practice, frameworks like Tensorflow* and PyTorch* dominate DNN training usage.
- However, MPI is the leading HPC communication model, and has excellent collectives performance.
- MPI is being increasingly adopted in deep learning research (see figure).
- Other programming models also have excellent collectives performance!
- OpenSHMEM is gaining popularity in HPC usage and supports one-sided collectives operations.



“Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis,” Ben-Nun et al. arXiv:1802.09941

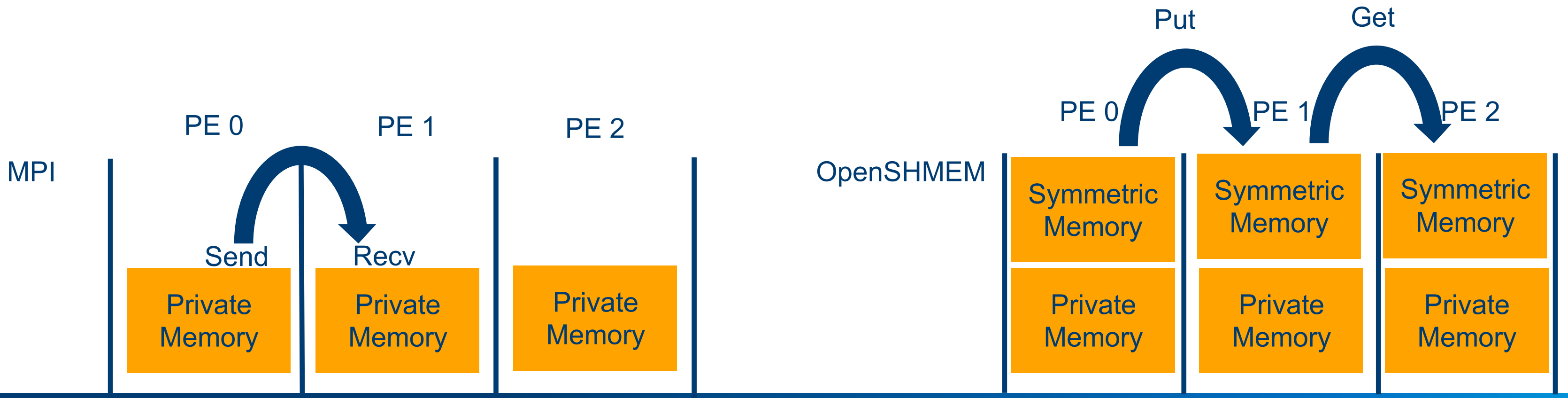
OpenSHMEM: Introduction / Overview

- OpenSHMEM:
  - Open standard PGAS model: emphasizes one-sided operations (put/get, atomics)
  - Symmetric memory exposed for remote access.
- Collective Operations:
  - Barrier, broadcast, collect, reductions, all-to-all, etc.
  - Most dominant communication in large-scale machine learning workloads.



OpenSHMEM vs. MPI For Scale-Out

- Collectives are implemented on top of point-to-point communication.
- OpenSHMEM performs collectives via one-sided RDMA operations (put/AMO based).
- MPI collectives are often two-sided in nature (matching send/receive ops).
  - MPI also provides single-sided interfaces, but are not as inherent to the programming model.
- Good algorithms and how they map to the system dictates collectives performance.



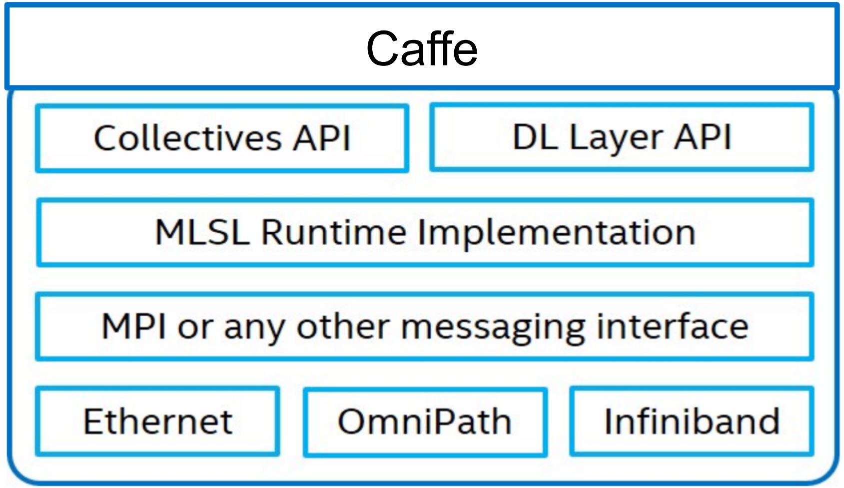
Intel® Distribution of Caffe*

Caffe is a Deep Learning Framework developed by Berkeley AI Research (BAIR) that enables efficient model development and testing.

- Optimized for image-recognition tasks but can be used for other domains.
- The Intel® distribution extends Caffe and optimizes for Intel® Xeon® processors.
  - Introduces multi-node training support using the Intel® Machine Learning Scaling Library (MLSL)

Intel® Machine Learning Scaling Library (MLSL)

- MLSL efficiently implements collective communication routines commonly found in parallel DNN Training.
- Common interface that many higher-level Deep Learning frameworks can hook into.
- Implemented using MPI for communication but can readily be modified for other messaging interfaces (i.e OpenSHMEM).



"On Scale-out Deep Learning Training for Cloud and HPC," Sridharan et al. arXiv:1801.08030.

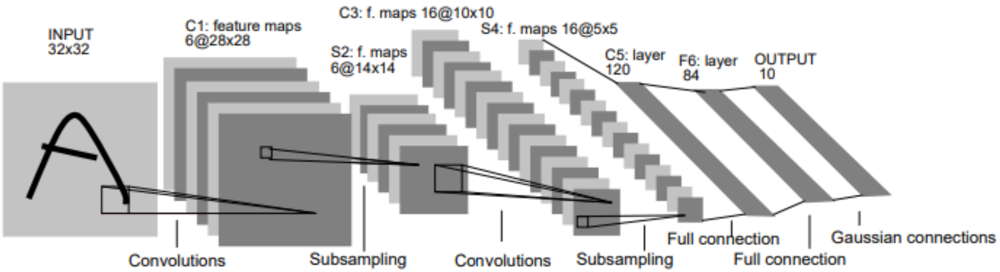
Design and Implementation

How do we test OpenSHMEM enabled DNN training?

- Extend MLSL to support both MPI and OpenSHMEM
  - Replace MPI Collectives with OpenSHMEM collectives
    - MPI_Bcast(void *buffer, …) -> shmem_broadcast(void *dest, void * src,…)
    - MPI_Barrier() -> shmem_barrier_all()
  - Modify all MLSL memory allocation calls to use shmem_malloc(…)
- Modify Intel® Caffe to use OpenSHMEM enabled MLSL.
  - Ensure that all memory allocation is symmetric instead of private
    - shmem_malloc(…) instead of malloc(…)
  - Ensure that collective communication calls adhere to updated MLSL API.

Design and Implementation

- The Intel® Distribution of Caffe comes with pre-defined models that can be used for testing and verification.
  - MNIST Handwritten Digit Data Set
    - Handwritten Digits from 0 -> 9
    - 60,000 Training Samples
    - 10,000 Test Samples
    - LeNet Network
      - Convolution/Pooling Layers (2)
      - Fully Connected ReLU Layer (1)
      - SoftMax Output Layer (1)
    - Communication dominates training time for small neural networks.
      - Expect OpenSHMEM to perform well for this type of problem



Performance Measurements

Characterizing performance of training.

- Wrap the train() method with start and stop timers.
- All-Reduce “sum” the time deltas across all PEs
- Divide accumulated sum by number of PEs

Experimental Setup:

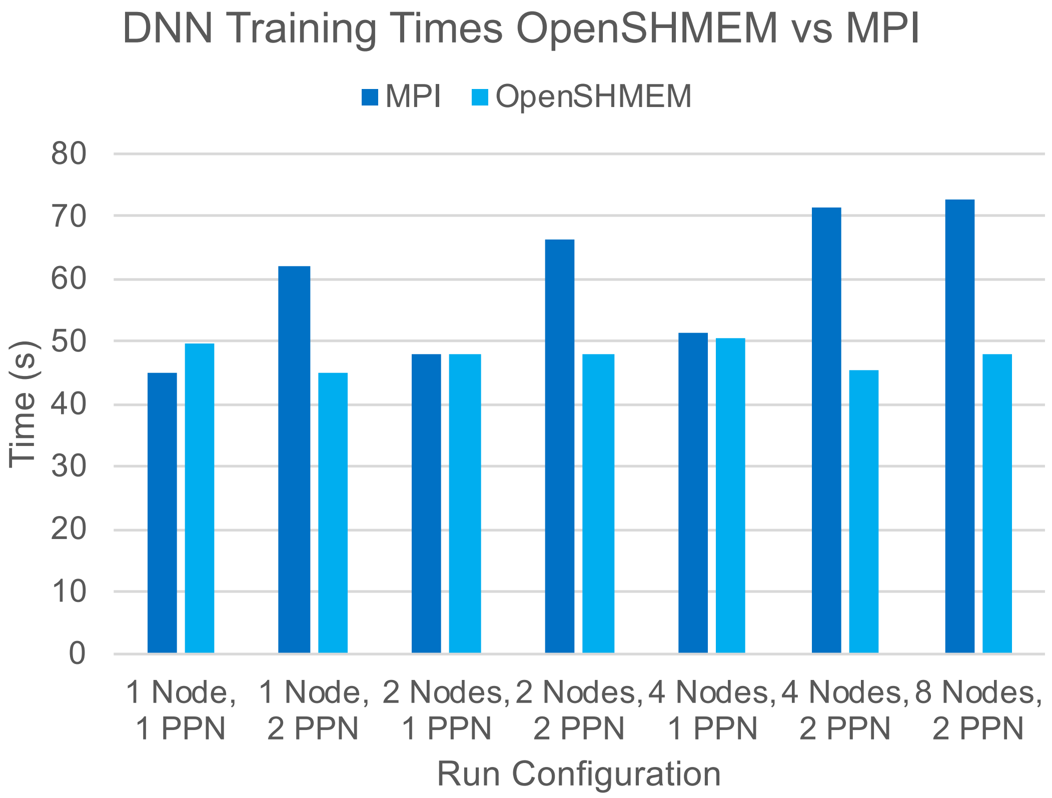
- Experimental Data was collected on NERSC’s Cori machine:
  - 2x 16-core Intel® Xeon™ Processor E5-2698 v3 (“Haswell”) at 2.3 GHz compute nodes
  - Aries Interconnect
  - Cray* MPICH 7.7.6, Cray* SHMEM 7.7.6

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

Results

Training Time Results.

- OpenSHMEM consistently shows lower training time than MPI.
- The largest margins are shown when OpenSHMEM can leverage on-node communication when PPN is 2.



Conclusion and Future Work

Conclusion:

- The performance of collective communication is important for DNN training.
- By using OpenSHMEM instead of MPI collectives, we see promising improvements in overall training time.

Future:

- Evaluate at larger scale.
- Investigate alternative algorithms (model parallel, pipelining, collectives, etc.)

intel®