
IWOCL 2024
The 12th International Workshop on OpenCL and SYCL
Intel® SHMEM: an OpenSHMEM Runtime with GPU-initiated Operations using SYCL
Lawrence Stewart, Intel Corporation
David Ozog, Md. Wasi-ur- Rahman, Lawrence Stewart
APRIL 8-11, 2024 | CHICAGO, USA | IWOCL.ORG

Legal Disclosures

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.

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex. Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure. See backup for configuration details. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks

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.

Outline

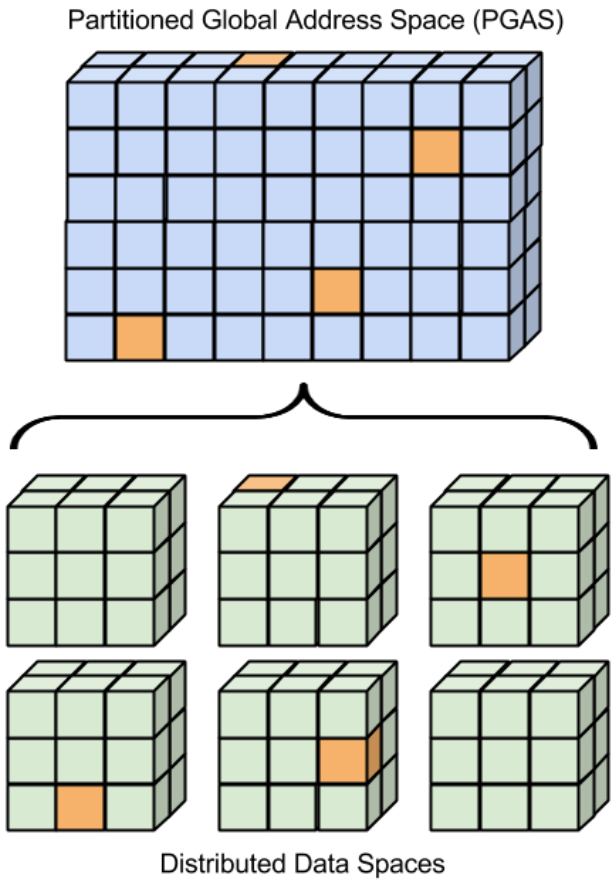
- Overview
- PGAS and OpenSHMEM
  - Background
  - Current State and Future Directions
- Intel® SHMEM
  - Release 1.0.0
- Intel® SHMEM with SYCL
- Performance
- OneCCL and Intel® MPI

Overview

- What is Intel® SHMEM?
  - A library for one-sided and collective communication
  - Implements the OpenSHMEM API, with extensions for GPU
- Operations Supported
  - Remote memory access, remote atomic memory operations, collective operations, synchronization of multiple processing elements (ranks)
  - API available on host and inside SYCL kernels (device initiated)
- Benefits
  - Lightweight, high-performance communications for distributed GPU applications

Partitioned Global Address Space

- Processing Elements (Pes) have access to global memory, and are aware what data is where.
- Operations:
  - Put(), Get(), Add(), Put-Signal(), Barrier(), …
- Distributed Memory model:
  - “Shared” memory with a process ID
- SPMD Execution Model:
  - SPMD = Same Program Multiple Data
  - All PEs execute the same program



OpenSHMEM: Basic Operations

Initialization

- `shmem_init()` sets up *symmetric* data regions

Memory Managment

- "Symmetric" memory is remotely accessible
- `shmem_{malloc|calloc|realloc}`

Remote Memory Access (RMA):

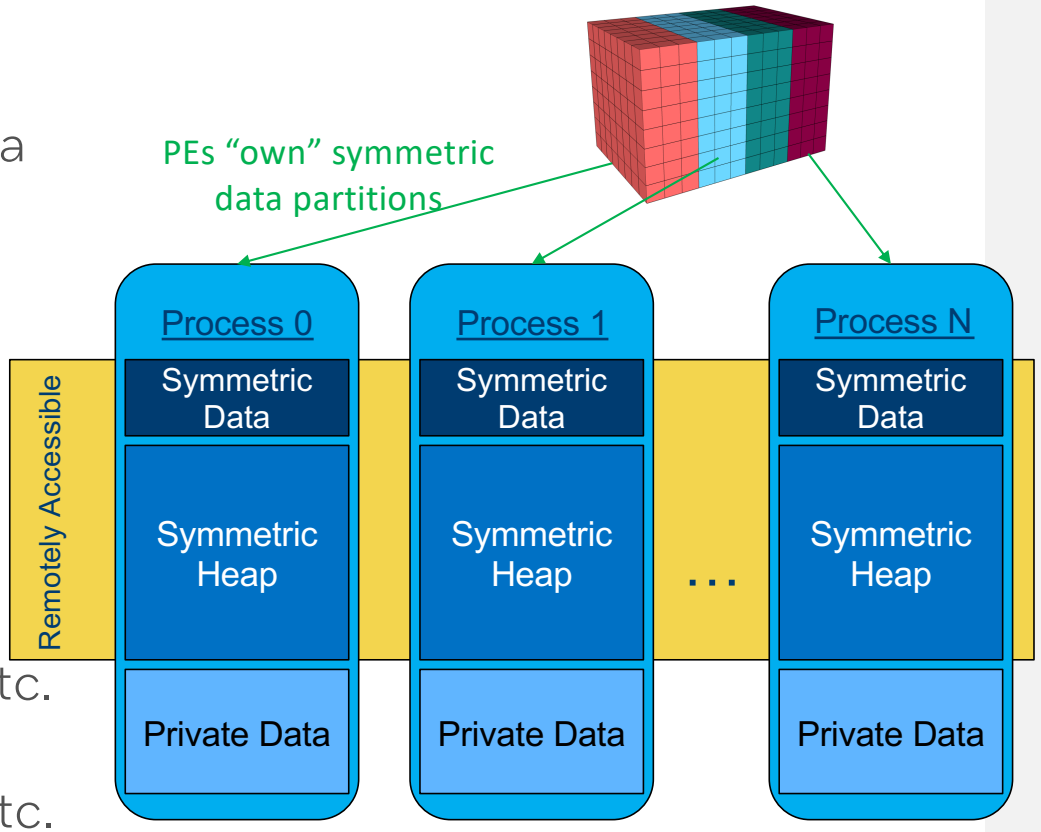
- Put / Get / Put-with-signal

Atomic Memory Operations (AMO):

- Add, compare-swap, fetch, bitwise, etc.

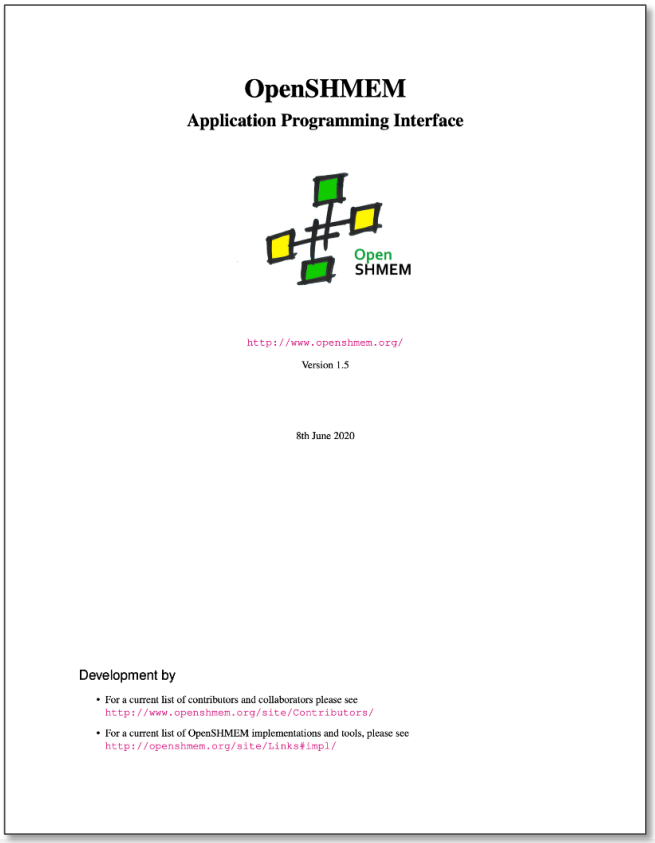
Collective Operations:

- Barrier, Broadcast, Reduce, Alltoall, etc.



OpenSHMEM Specification 1.5

- Active community
- Current (1.5) features
  - Threads, Teams
  - Profiling Interface
- Proposed (1.6, 1.7)
  - Completions
  - Non-blocking collectives
  - Aggregation
  - GPU support
  - …



OpenSHMEM

Application Programming Interface

Open SHMEM

http://www.openshmem.org/

Version 1.5

8th June 2020

Development by

- For a current list of contributors and collaborators please see http://www.openshmem.org/site/Contributors/
- For a current list of OpenSHMEM implementations and tools, please see http://openshmem.org/site/Links#impl/

*names and images may be claimed as the property of others

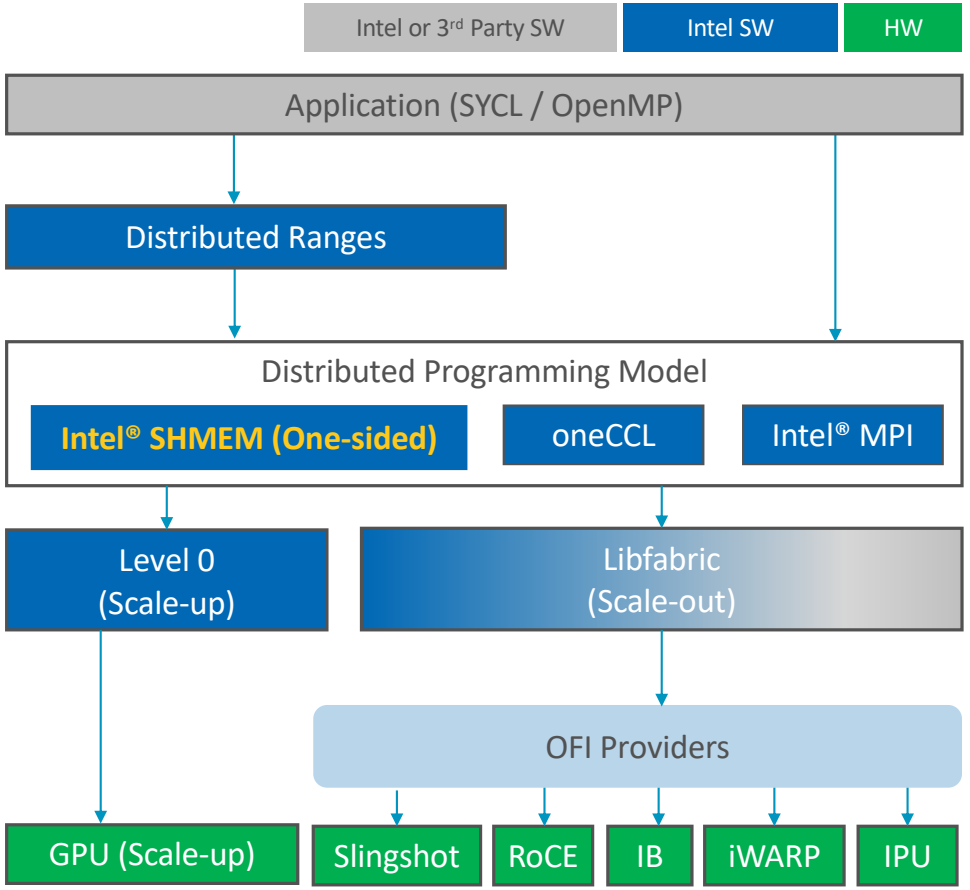
GPUs and OpenSHMEM

- Distributed Multi-GPU Clusters growing quickly
- Some Intel-based systems
  - Aurora has 12 Ponte Vecchio (PVC) GPU tiles connected w/ Xe-Link, dual Sapphire Rapids (SPR)
  - Dawn Phase 1[1], SuperMUC-NG Phase 2[2], Clementina XX1[3], TACC Stampede 3
- Other GPU-initiated SHMEM libraries
  - NVIDIA NVSHMEM supports OpenSHMEM 1.5 features with CUDA and NVIDIA GPU specific extensions only
  - AMD ROC-SHMEM supports a subset of OpenSHMEM 1.5 APIs with HIP and AMD GPU specific extensions only
- Current OpenSHMEM standard does not enable accelerators
- Intel® SHMEM is envisioned to provide open GPU-initiated one-sided programming based on OpenSHMEM and SYCL

1 https://www.intel.com/content/www/us/en/newsroom/news/intel-dell-power-uk-fastest-ai-supercomputer.html
2 https://doku.lrz.de/supermuc-ng-10745965.html
3 https://www.datacenterdynamics.com/en/tags/clementina-xxi/

Intel® SHMEM

- GPU-initiated OpenSHMEM APIs on Intel Data Center GPUs using C++/SYCL
- SPMD model for developers on heterogeneous platforms
- OpenSHMEM standard device- and host-initiated point-to-point (RMA, AMO) and collective operations
- Device Extension APIs for GPU capabilities (SYCL Groups)
- Promote adoption of GPUs into OpenSHMEM standard
- Leverage efficient host runtimes, such as Sandia OpenSHMEM, Intel® MPI, oneCCL
- Open-source release 1.0.0 is available



Release 1.0.0

- Open-source Release
  - GitHub repo: https://github.com/oneapi-src/ishmem
  - Specification: https://oneapi-src.github.io/ishmem/intro.html
- Tested on Intel® Data Center Max Series GPUs with
  - Xe-Link (intra-node)
  - CXI/Verbs;RXM OFI providers (inter-node)
- Major Features
  - RMA, AMO, Collectives, Signaling, Synchronization, Memory Ordering
  - Sandia OpenSHMEM as CPU-side backend (GPU heap and GPU RDMA support)
  - Configurable with Unified Shared Memory host heap
  - Supports fast GPU and host interaction
  - Unit tests, intranode performance tests, examples
- Blog post on Intel® SHMEM: "Introducing the New Intel® SHMEM" by Rahman, Stewart, Ozog

Intel® SHMEM with SYCL Example

red = Intel® SHMEM | blue = comment
black = SYCL / C++

1. Include header files
2. Initialize with `ishmem_init()`
3. Allocate symmetric objects using memory management APIs
4. Query APIs can be used within the kernel to retrieve runtime info
5. Within the kernel, communication APIs will be translated to load/store (intra-node) or corresponding host-based API through the proxy (inter-node)
5. Device-extension APIs optimizes by utilizing all the work-items in a work-group co-operatively
6. `ishmem_finalize()` finalizes the runtime and destroys all allocated resources

```
1 #include <ishmem.h>
  #include <ishmemx.h>

  int main() {
    /* Initialize Intel SHMEM */
2   ishmem_init();
    /* Allocate objects on symmetric heap */
3   int *src = (int *) ishmem_malloc(N * sizeof(int));
    int *dst = (int *) ishmem_malloc(N * sizeof(int));
    /* Launch kernel */
    auto e = q.parallel_for(nd_range<3>(range<3>(x_s, y_s, z_s),
                                        range<3>(x_s, y_s, z_s)),
                                        [=](nd_item<3> idx) {
4     int pe   = ishmem_my_pe();
      int npes = ishmem_n_pes();
      auto group = idx.get_group();
      /* Put data to neighbor device in "ring" fasion */
5     ishmemx_int_put_work_group(dst, src, N, (pe + 1) % npes,
                                 group);
    }); e.wait();
    /* Finalize Intel SHMEM */
    ishmem_free(src);
    ishmem_free(dst);
6   ishmem_finalize();
  }
```

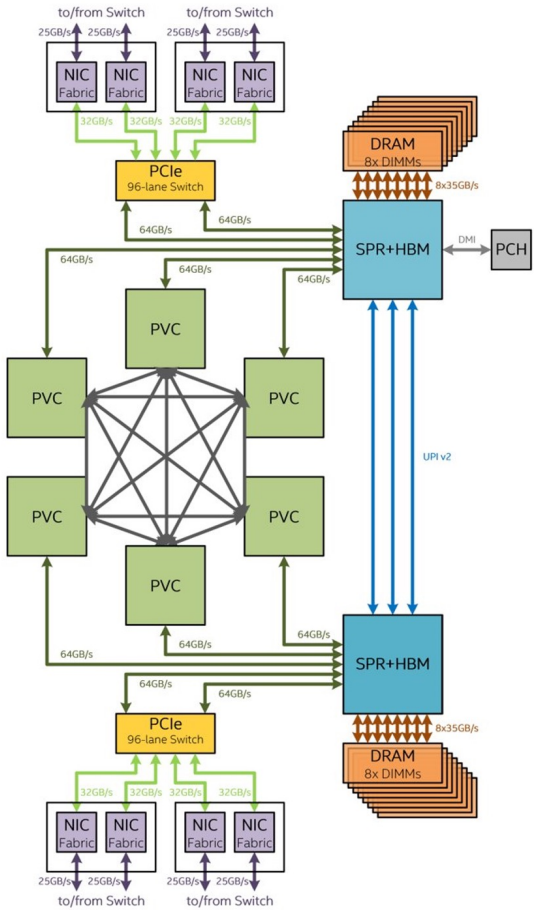
Jacobi: 2D Stencil

- Iterative approach of solving a set of linear equations
- Each entry in the 2D matrix is computed using the surrounding elements
- Exchange of data between iterations are performed using **`ishmem_float_p`**
- Further optimization opportunities with device extension APIs

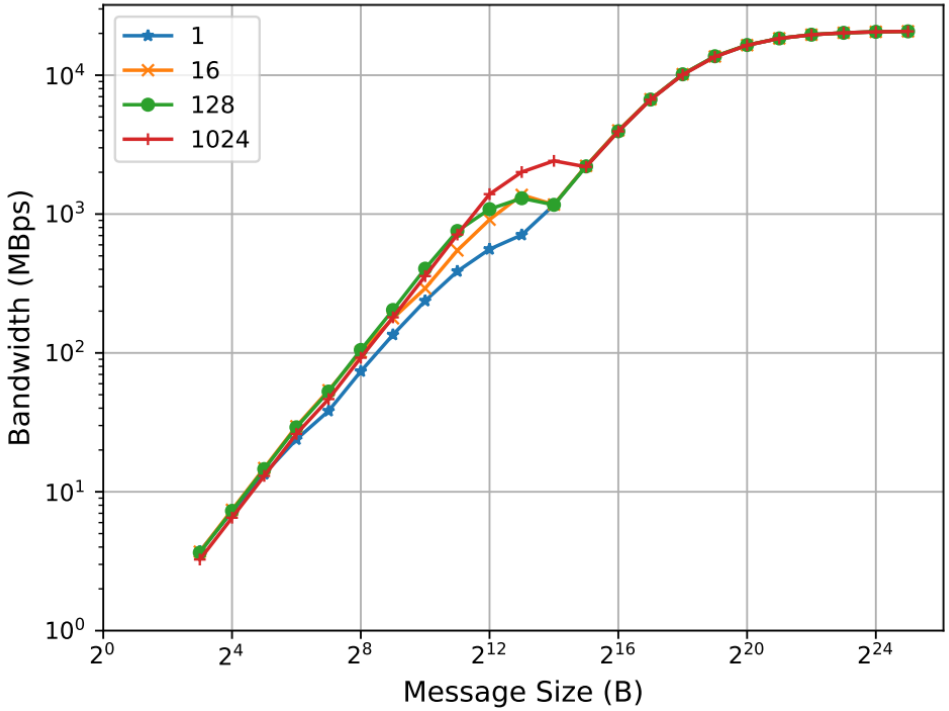
```
auto sum_red = sycl::reduction(l2_norm_d, sycl::plus<>());
auto ret = q.parallel_for(
  sycl::range<1>{global_range}, sum_red, [=](sycl::id<1> idx,
                auto &sumr) {
    int iy = idx / nx + iy_start;
    int ix = idx % nx + 1;
    real local_norm = 0.0;

    if (iy < iy_end && ix < (nx - 1)) {
      const real new_val = (0.25) * (a[iy * nx + ix + 1] +
        a[iy * nx + ix - 1] + a[(iy + 1) * nx + ix] +
        a[(iy - 1) * nx + ix]);
      a_new[iy * nx + ix] = new_val;

      if (iy_start == iy)
        ishmem_float_p((real *) (a_new + top_iy * nx + ix),
                       new_val, top_pe);

      if (iy_end - 1 == iy)
        ishmem_float_p((real *) (a_new + bottom_iy * nx + ix),
                       new_val, bottom_pe);

      real residue = a[iy * nx + ix] - new_val;
      local_norm = residue * residue;
      sumr += local_norm;
    }
  }
);
```

Performance Test Setup

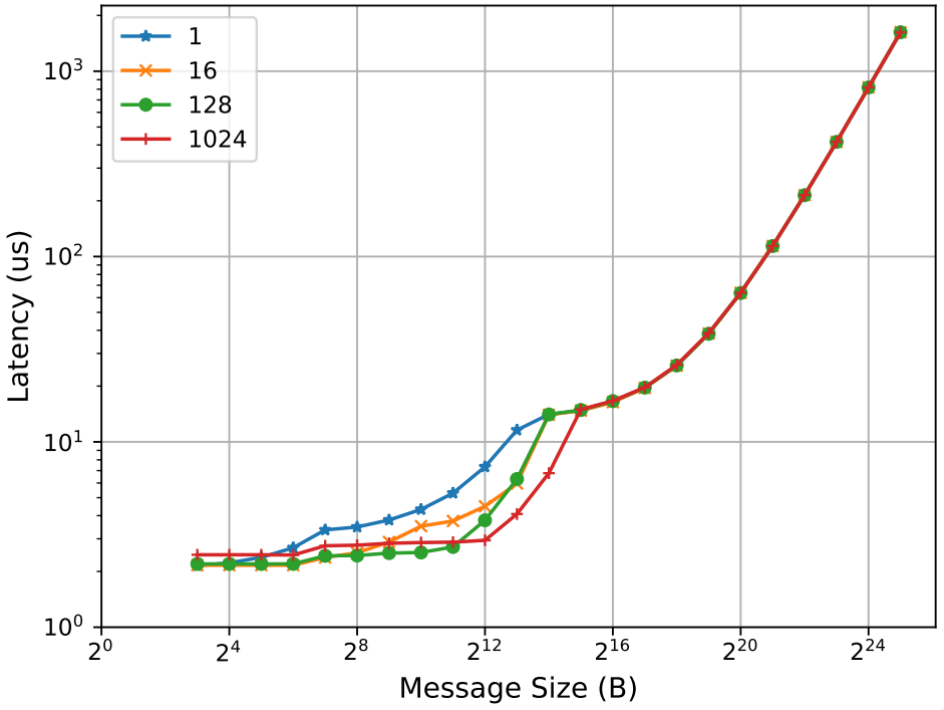
- Measurements done on Intel internal Aurora validation system
- Hardware details:
  - 6 Intel® Data Center GPU Max Series (PVC) devices
  - 6 GPUs are fully connected by XeLink fabric
  - 2x Intel® Xeon® CPU Max 9470C (SPR) CPUs
    - 2.0 GHz, each socket with 52 cores and 2 threads per core
  - 8x Slingshot 11 NICs (200 Gbps each)
- Software Details:
  - Intel® SHMEM 1.0.0
  - Sandia OpenSHMEM (main branch)
  - Intel® oneAPI DPC++/C++ Compiler 2024.1.0
- Measurements taken March 2024



Performance



Put_WG Bandwidth (Device-initiated)



Put_WG Latency (Device-initiated)

Intel® SHMEM Takeaways

- Available Now
  - Opensource Release
- Light weight (low impact on source code)
- GPU and Host initiated operations on GPU memory
- Small operations are low latency
- Large operations achieve full hardware bandwidth

oneCCL and Intel® MPI

Intel® oneAPI Collective Communications Library

Key Features

Enables efficient implementations of collectives used for deep learning training: all-gather, all-reduce, and more

oneCCL is designed for easy integration into deep learning (DL) frameworks (PyTorch, TensorFlow)

Provides C++ API and interoperability with SYCL

Deep Learning Optimizations include:

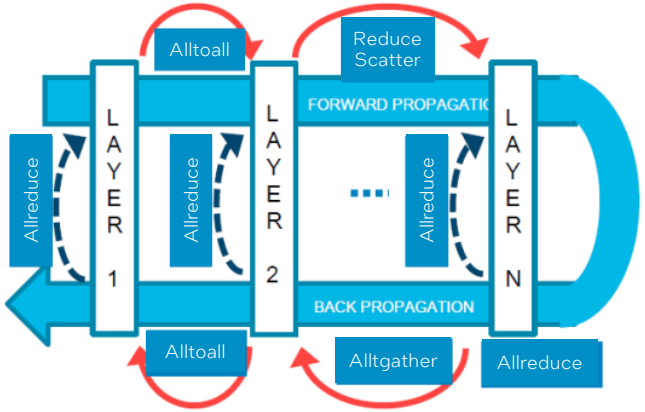
- Asynchronous progress for compute communication overlap
- Dedication of cores to ensure optimal network use
- Optimizations for persistent collectives
- Collectives in low-precision data types
- Point to Point Operations

Supported Collectives

- Allgatherv
- Allreduce
- Alltoall/Alltoallv
- Broadcast
- Reduce
- ReduceScatter

Point to Point

- Send/Recv

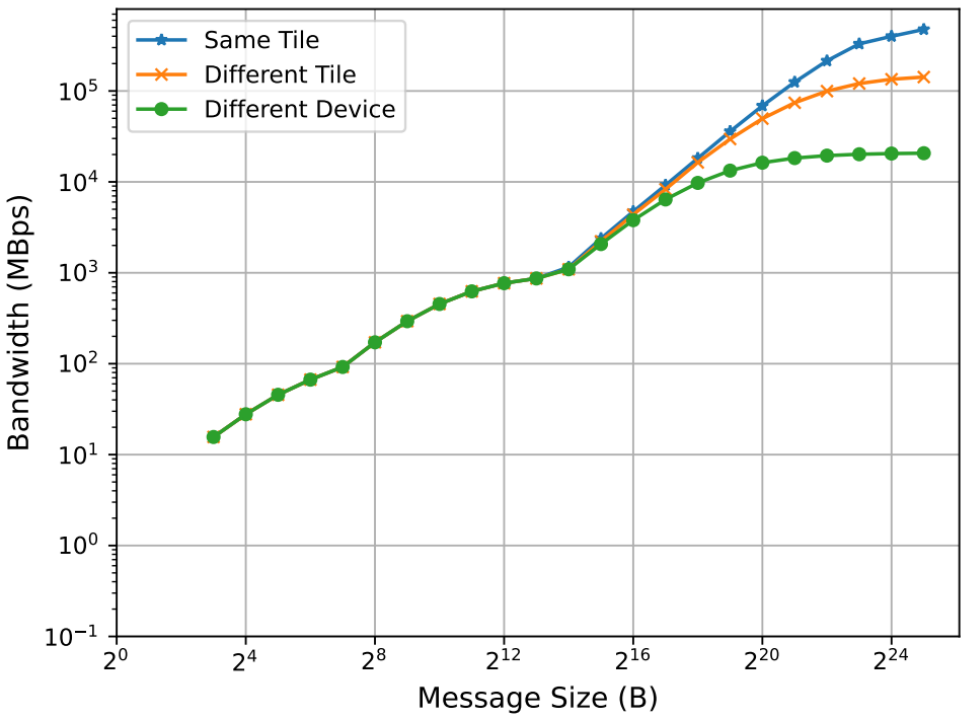


Intel® MPI 2021.12 Update

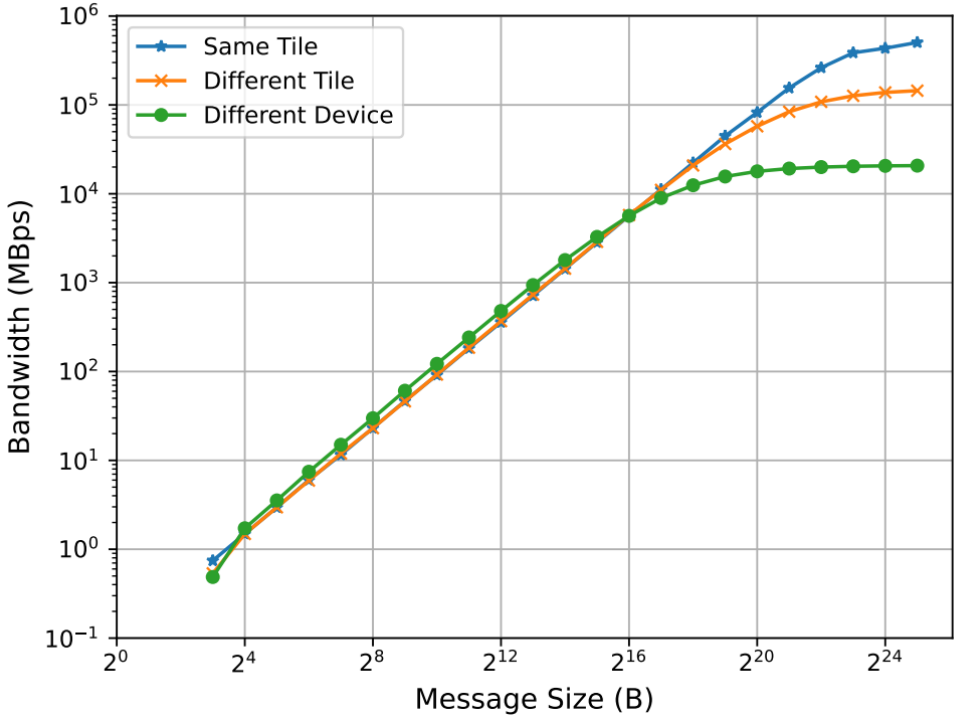
- Intel® MPI is a portable message-passing library implementing the open MPI Standard: https://www.intel.com/content/www/us/en/developer/tools/oneapi/mpi-library.htm
- What's new:
  - MPI-3 RMA GPU (host and device initiated)
    - Including device-initiated kernel level load/store
  - MPI 4.0 features
  - Enhancements:
    - GPU scalability optimizations


intel

Performance



Put Bandwidth (Device-initiated)



Put Bandwidth (Host-initiated)

12th International Workshop on OpenCL and SYCL

Intel® SHMEM: an OpenSHMEM Runtime with GPU-initiated Operations using SYCL

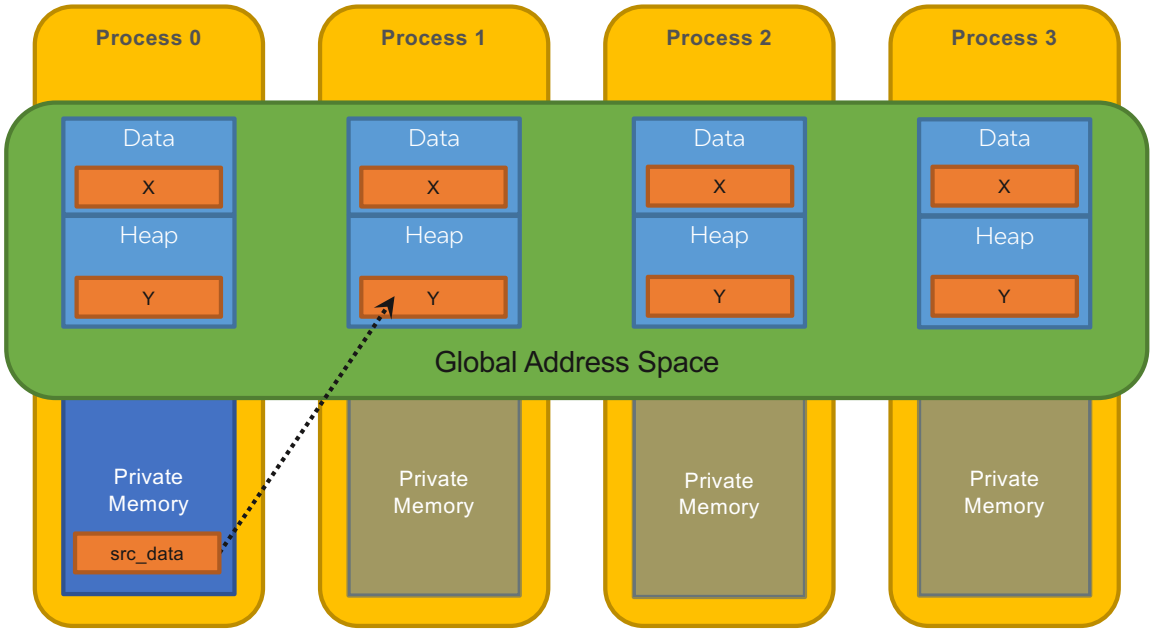
David Ozog, Md. Wasi-ur- Rahman, Lawrence Stewart

April 10, 2024

intel®

OpenSHMEM: Symmetric Memory

- Same size and layout at each process:
  - Instances of symmetric objects present on every PE
  - TYPE *Y = shmem_malloc(sizeof(TYPE))
- Put data to remote objects
  - shmem_put(&Y, &src_data, nelems=1, pe=1)
- Remotely accessible memory is comprised of two segments
  - Data segment (statically allocated)
  - Heap segment (dynamically allocated)



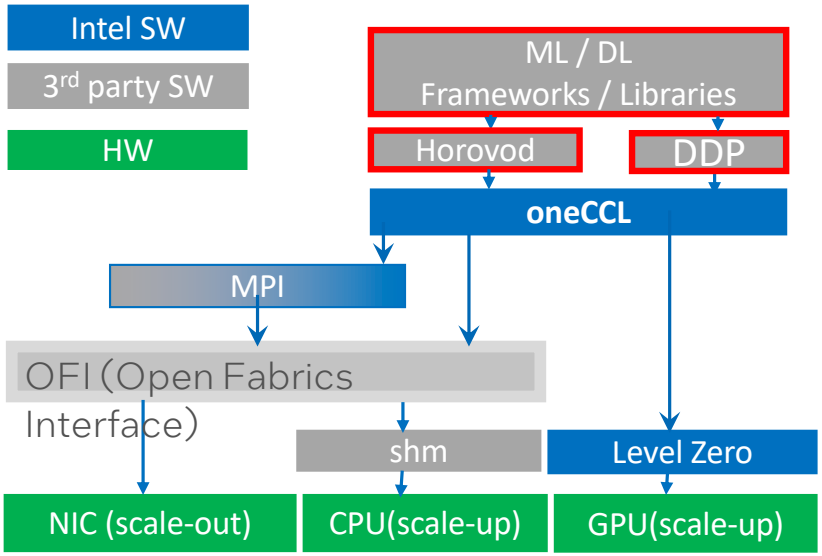
APIs

OpenSHMEM standard (`ishmem.h`)	▪ Library setup and query ▪ Memory management ▪ RMA, AMO, Signaling, Collective, Synchronization, Memory ordering (Host and Device) ▪ Host operations with symmetric memory objects resided in device memory
Device Extensions (`ishmemx.h`)	▪ Work-group and sub-group level RMA, Signaling, Collectives, Synchronization, Memory ordering ▪ Host RMA operations using Level Zero ▪ Utility functions: print, timestamp
Not yet released	▪ Threading Model ▪ Context (further clarifications needed) and Team management ▪ Non-blocking AMOs, vector-based wait-until / test ▪ Profiling interface

Intel® oneAPI Collective Communications Library

Optimize Communication Patterns

- oneCCL provides optimized communication patterns for high performance on Intel CPUs & GPUs to distribute model training across multiple nodes
- Transparently supports many interconnects, such as Intel® Omni-Path Architecture, InfiniBand, & Ethernet
- Built on top of lower-level communication middleware-MPI & libfabrics
- Enables efficient implementations of collectives used for deep learning training: all-gather, all-reduce, and reduce-scatter, among others



[Intel® oneAPI Base Toolkit]

[oneCCL Product Page]