
it
starts
with
intel

Bringing AI Everywhere in HPC

MPI@Intel

Maria J. Garzaran

Supercomputing 2023

it starts with

Intel® MPI 2021.11 Update

- What's new:
  - MPI-3 RMA GPU (host and device initiated)
    - Device initiated mode: I_MPI_OFFLOAD_ONESIDED_DEVICE_INITIATED
  - MPI 4.0 features: MPI Sessions support
    - Large count/native ILP64 is available since 2021.10
  - OFI/cxi support and optimizations (Technical preview)
- Enhancements:
  - GPU buffer aware pt2pt and collective operations optimization
    - MPI_Alltoall, MPI_Allreduce
  - CPU/GPU platforms scalability optimizations

Intel® MPI 2021.11 MPI-3 RMA GPU (SYCL example)

Host initiated

```
q.submit([&](auto & h) {
        h.parallel_for(sycl::range(my_subarray.x_size), [ =] (auto index)
{
...
        });
    }).wait();
  }
  /* Perform 1D halo-exchange with neighbours */
  if (my_subarray.rank != 0) {
    int idx = XY_2_IDX(0, 0, my_subarray);
     MPI_Put(&a_out[idx], my_subarray.x_size,
     MPI_DOUBLE, my_subarray.rank - 1,
      my_subarray.l_nbh_offt,
       my_subarray.x_size, MPI_DOUBLE, cwin);
  }
  if (my_subarray.rank != (my_subarray.comm_size - 1)) {
    int idx = XY_2_IDX(0, my_subarray.y_size - 1, my_subarray);
     MPI_Put(&a_out[idx],
      my_subarray.x_size,MPI_DOUBLE,
      my_subarray.rank + 1, 1,
      my_subarray.x_size, MPI_DOUBLE, cwin);
  }
  /* Recalculate internal points  in parallel with communications */
  {
    q.submit([&](auto & h) {
        h.parallel_for(sycl::range(my_subarray.y_size - 2,
my_subarray.x_size), [ =] (auto index) {
...
        });
    }).wait();
   }
```

MPI invoked from Host

Device initiated*

```
q.submit([&](auto & h) {
      h.parallel_for(sycl::nd_range<1>(work_group_size, work_group_size),
            [=](sycl::nd_item<1> item) {
...
        /* Calculate values on borders to initiate communications early */
        for (int column = my_x_lb; column < my_x_ub;  column ++) {
...
        }
        item.barrier(sycl::access::fence_space::global_space);
        if (local_id == 0) {
        /* Perform 1D halo-exchange with neighbours */
          if (my_subarray.rank != 0) {
            int idx = XY_2_IDX(0, 0, my_subarray);
              MPI_Put(&a_out[idx], my_subarray.x_size,
              MPI_DOUBLE,
              my_subarray.rank - 1, my_subarray.l_nbh_offt,
             my_subarray.x_size, MPI_DOUBLE, cwin);
          }
          if (my_subarray.rank != (my_subarray.comm_size - 1)) {
            int idx = XY_2_IDX(0, my_subarray.y_size - 1, my_subarray);
              MPI_Put(&a_out[idx], my_subarray.x_size,
              MPI_DOUBLE,
               my_subarray.rank + 1, 1,
               my_subarray.x_size, MPI_DOUBLE, cwin);
          }
        }
        /* Recalculate internal points in parallel with comunications */
        for (int row = 1; row < my_subarray.y_size - 1; ++row) {
          for (int column = my_x_lb; column < my_x_ub;  column ++) {
...
          }
        }
        item.barrier(sycl::access::fence_space::global_space);
      });
    }).wait();
```

MPI invoked from GPU

- MPI-3 RMA GPU subset of supported functions:
  - *MPI_Put*
  - *MPI_Get*
  - *MPI_Win_lock / MPI_Win_lock_all***
  - *MPI_Win_unlock / MPI_Win_unlock_all***
  - *MPI_Win_flush / MPI_Win_flush_all***
  - *MPI_Win_fence***
- Supported for scale up and scale out
- Supported for **SYCL and OpenMP** (C/C++ only***)

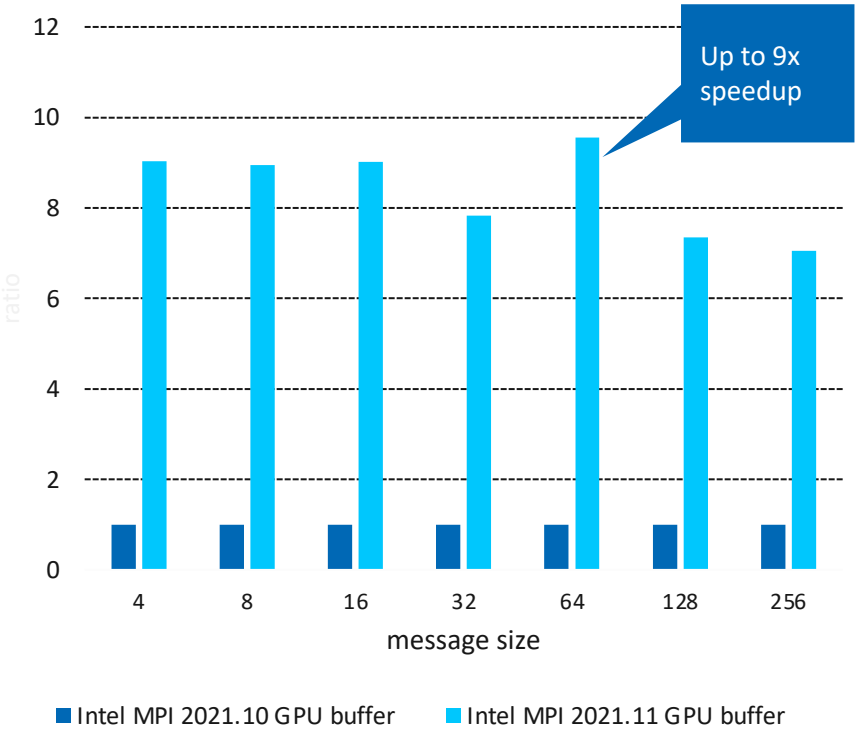
Legend:
- GPU kernel code
- HOST code

* - Host initiated is available out of the box (**I_MPI_OFFLOAD** path)
Device initiated requires **I_MPI_OFFLOAD_ONESIDED_DEVICE_INITIATED=1**
** - Synchronization primitives require kernel level serialization
*** - Fortran support is work in progress

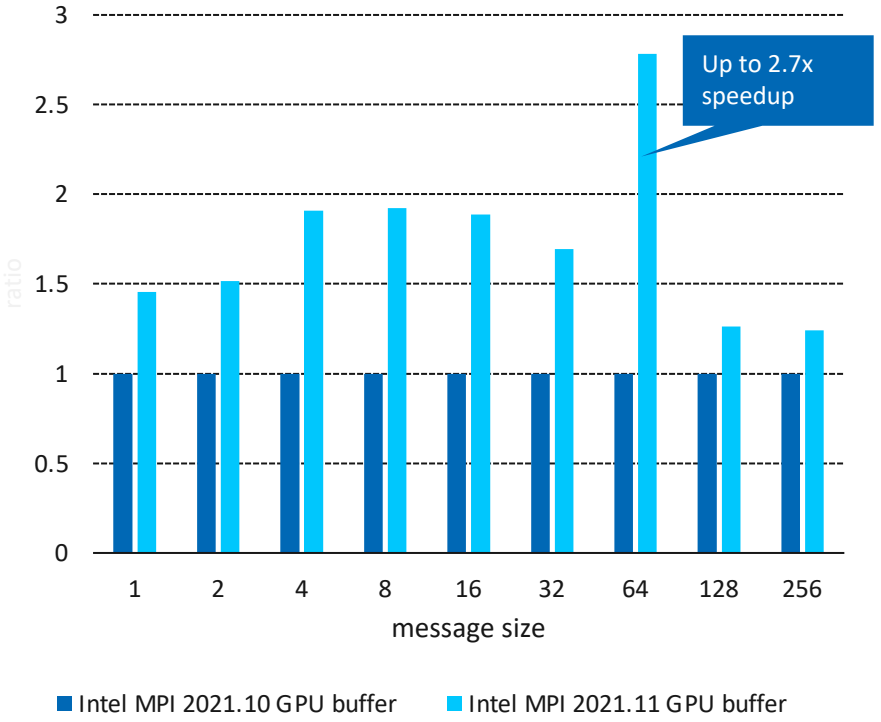
Examples: https://github.com/oneapi-src/oneAPI-samples/tree/development/Libraries/MPI/jacobian_solver

Intel® MPI 2021.11 GPU buffer path latency optimizations

IMB-MPI1-GPU **allreduce**. 4 nodes, 32 ranks total: 2 x Intel® Xeon® Platinum 8480+ Processor + 4 x Intel® Data Center GPU Max 1550. Latency ratio. Higher is better



IMB-MPI1-GPU **pingpong**. 1 node, 2 ranks total: 2 x Intel® Xeon® Platinum 8480+ Processor + 2 x Intel® Data Center GPU Max 1550. Latency ratio. Higher is better



- New latency and BW optimizations for GPU aware pt2pt and collective operations: allreduce/alltoall

- New highly efficient support for alltoall with XeLink and GPU RDMA* support (I_MPI_OFFLOAD_RDMA*)

* I_MPI_OFFLOAD_RDMA is fully supported with IEFS + OFI/psm3 (available for ConnectX-6+ interconnects family as well): https://www.intel.com/content/www/us/en/support/articles/000088090/ethernet-products/intel-ethernet-software.html

MPICH for Aurora Update - I

- Improvements for Intel® Data Center GPU Max Series
  - Optimizations for intra-node communication
    - Uses oneAPI Level Zero Inter Process Communication (IPC) primitives
    - For latency and bandwidth, and for a variety of message sizes
  - Optimized collective algorithms for GPU buffers
    - AllReduce, Broadcast, Alltoall, and Allgather
    - They benefit from XeLinks for large messages
  - Support for GPU RDMA (point to point and RMA)
  - Optimizations for RMA for GPU buffers
  - Optimizations leverage the underlying hardware features
  - Support for using a tile as a device or two tiles (in a single GPU) as a device

MPICH for Aurora Update - II

- Contributions to Collectives
  - Topology (dragonfly) aware collectives for Broadcast, Allreduce, and Reduce
  - Hardware-offload Collectives for Broadcast, Allreduce, and Barrier
    - Triggered-based operations
    - Switched-based
  - Collectives with GPU buffers (in previous slide)
  - Past Contributions
    - High radix algorithms for large scale system
    - Multi-leader algorithms to leverage multiple NICs
    - Non-blocking algorithms
    - Intra-node collectives that leverage shared-memory in the node
- Validation and bug fixes

Intel® SHMEM

- Device-initiated OpenSHMEM operations on Intel GPUs with SYCL
- Supports OpenSHMEM 1.5 style Remote Memory Access (RMA), Atomics, Collectives, Synchronization and Ordering operations
- Includes work-group and sub-group extensions for co-operative thread execution
- GPU RDMA enabled Sandia OpenSHMEM as host back-end through CPU proxy
- Fast synchronization algorithm between CPU and GPU threads
- >= 90% scale-up performance efficiency with 2D stencil kernel
- First open-source release is expected by next week)

  https://github.com/oneapi-src/ishmem

Legal Notices and Disclaimers

Performance varies by use, configuration and other factors. Learn more on the Performance Index site.

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.

Your costs and results may vary.

Intel technologies may require enabled hardware, software or service activation.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

Thank you!

it
starts
with