OpenSHMEM over MPI: A Performance Contender

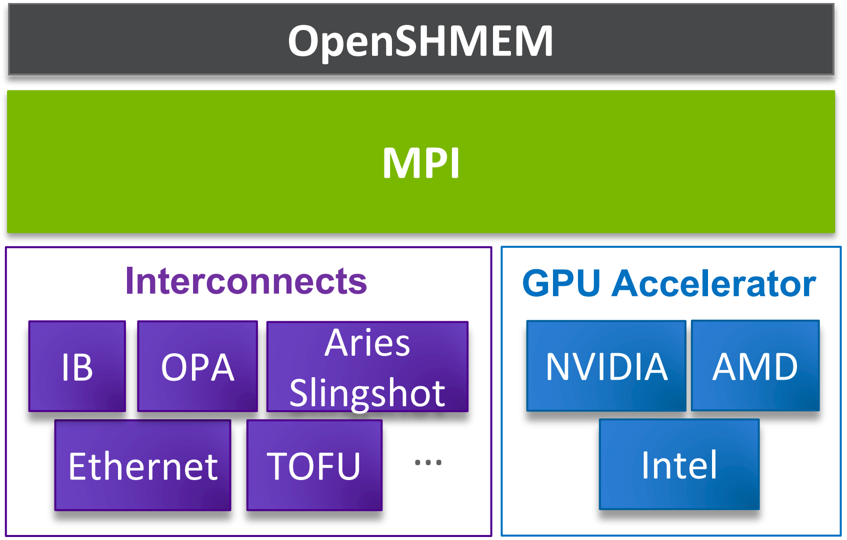
Min Si, Ken Raffenetti, Yanfei Guo, Pavan Balaji

Programming Models and Runtime Systems Group

Argonne National Laboratory, USA

Overview of OpenSHMEM over MPI

- **OpenSHMEM**
  - Specialized API designed for fast one-sided and collective communication
  - Directly mapping to low-level network API to ensure high performance
    - *Any overhead is too much overhead!*
- **MPI**
  - Low level library focusing on completeness of feature (e.g., p2p, one-sided, collectives, various reduction operation types, various data types)
- **OpenSHMEM over MPI**
  - OSHMPI: a ***portable*** implementation of OpenSHMEM but ***extra software overheads may exist***
  - As a serious performance contender
    - *What are the software overheads in OpenSHMEM over MPI?*
    - *Can we optimize them? How much?*
  - As a GPU-aware OpenSHMEM implementation
    - *Support CPU-initiated GPU communication*
    - *Leverage highly-optimized GPU-aware MPI implementations*



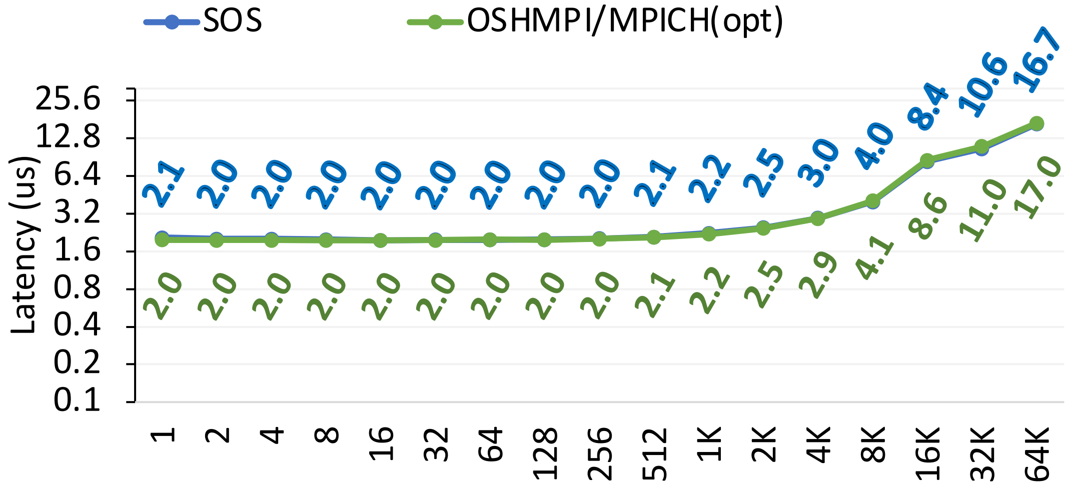
Systemic Software Overhead Analysis & Optimizations in RMA Path

- Datatype decoding
  - Datatype is a constant in each SHMEM op but becomes a variable when passing down to MPI
    - Compiler cannot optimize, result in **14 additional instructions** at PUT fast path
  - **Optimization**: leverage **compiler IPO** (already provided by mainstream compilers) to optimize code across OSHMPI and MPI libraries at link-time
    - All instructions can be eliminated by compiler
- Window metadata access
  - MPI internal win obj stores metadata, e.g., **comm (MPI-specific)**, network ep, remote mr_rkey...
    - Access to MPI **win->comm's attributes** causes expensive pointer dereferences at RMA /AMO fast-path
  - **Optimization**: Identify win with COMM_WORLD at win creation and avoid win->comm dereferences at OSHMPI RMA fast path (All OSHMPI windows use dup of COMM_WORLD)
- Virtual address translation for remote buffers
  - MPI requires **relative offset (displacement)** of remote buffers
    - Cause extra translations in OSHMPI (vaddr->disp) and MPI (disp->vaddr) at RMA/AMO fast path
  - **Optimization**: introduce MPI extension (MPIX_PUT_ABS|MPIX_GET_ABS) to handle vaddr directly
- Expensive MPI full progress
  - Ensure prompt progress for all MPI communication types (i.e., P2P, coll, AM-based)
    - Cause expensive overhead in SHMEM blocking operations and fence/quiet which may be unnecessary for OSHMPI
  - **Optimization:** progress polling with low freq when no AM occurs; exclude unnecessary polling for P2P/coll in RMA path

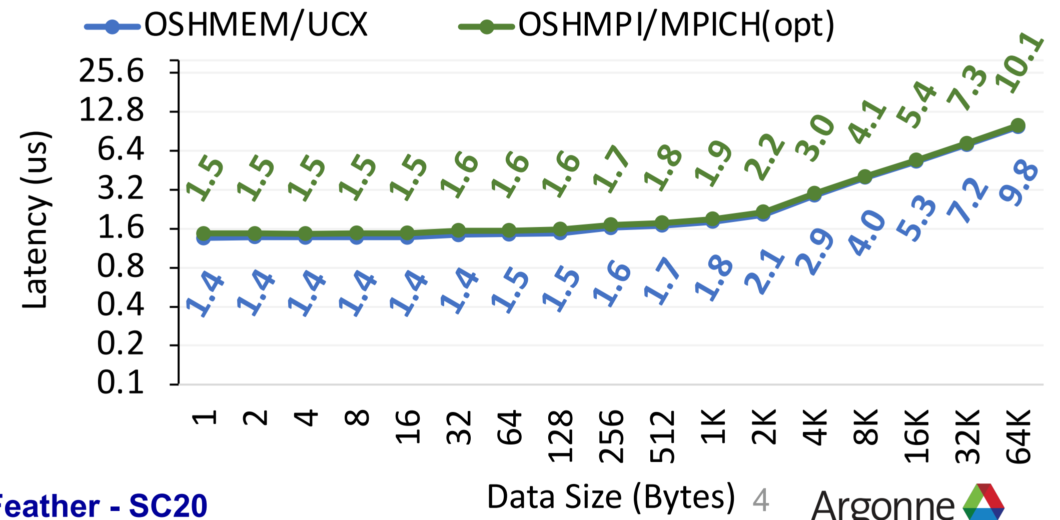
OSHMPI Performance Evaluation

- OSU benchmark osu_oshm_put
- Over OFI/Intel Omni-Path:
  - Optimized OSHMPI/MPICH delivers similar results as that of SOS in internode latency
  - No visible gap in internode message rate (graph omitted)
- Over UCX/Mellanox ConnectX-5:
  - OSHMPI/MPICH delivers only ~5% additional overhead compared to OSHMEM in internode latency
  - No visible gap in internode message rate (graph omitted)

Internode Latency on Argonne Bebop (OFI, Intel Broadwell, Omni-Path)



Internode Latency on Argonne JLSE (UCX, Intel Xeon Gold, ConnectX-5 EDR)



GPU-Aware OpenSHMEM with Memory Space Prototype

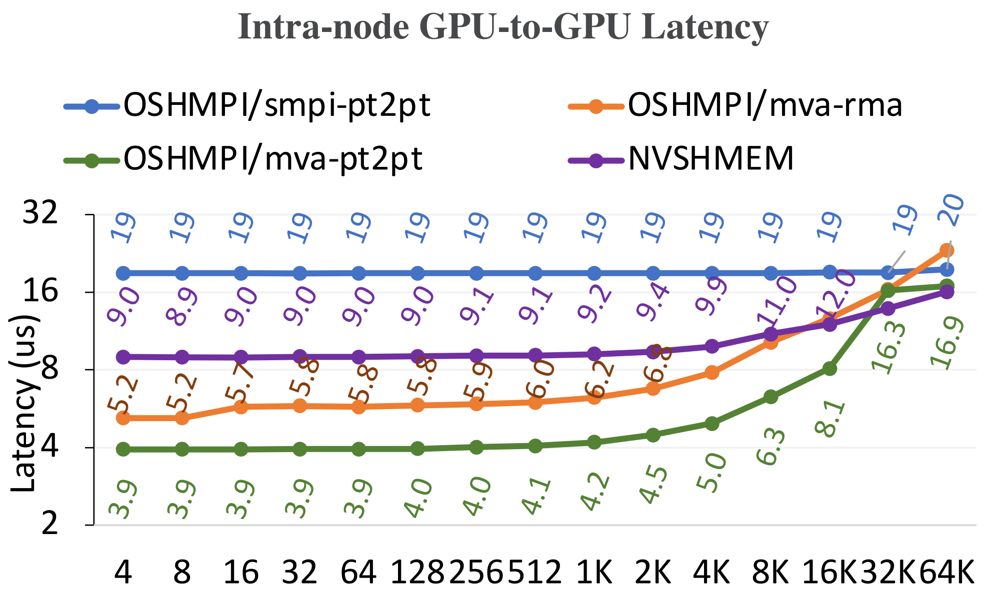
- Developed memory space prototype in OSHMPI (subset of the entire proposal)
  - Omit teams in this prototype, but flexible to extend
- Communication schemes with memory space
  - AMO/RMA with a space context
    - Dedicated internal window (i.e., communication resource + remote mem) for each space context
  - AMO/RMA without specific context (CTX_DEFAULT)
    - Attach default symmetric heap, global data, all space heaps to a single dynamic window as shared communication resource
- Create GPU memory space
  - E.g., specify CUDA mem_kind to allocate space heap by internally using cudaMalloc

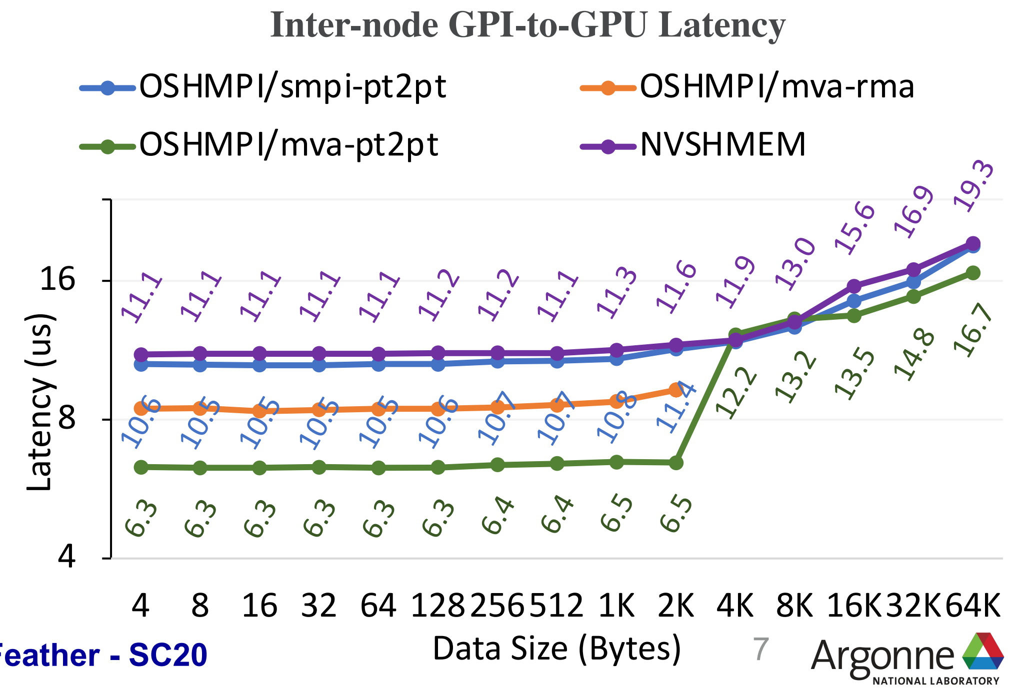
CPU-Initiated GPU-Aware OpenSHMEM RMA

- Why leverage GPU-aware MPI implementations?
  - Most GPU-specific optimizations are already provided by MPI
  - Portable support for wide range of GPUs (e.g., NVIDIA GPU, AMD GPU, Intel GPU)
- Limitations of GPU-aware MPI implementations
  - Some MPI impls provide GPU-awareness only for PT2PT, RMA simply segfaults (e.g., Spectrum MPI, OpenMPI/UCX)
  - Some MPI impls supports GPU-aware RMA but have to internally utilize active message (AM) for internode data transfer (e.g., MPICH/UCX)
- **Design Strategies in OSHMPI**
  - Support both MPI-PT2PT based path and MPI-RMA based path for RMA operations
  - Require the user to specify the GPU features (value is subset of “pt2pt,put,get,acc”) of the underlying MPI implementation
  - Choose the appropriate RMA path at runtime

GPU-Aware OpenSHMEM Evaluation (1)

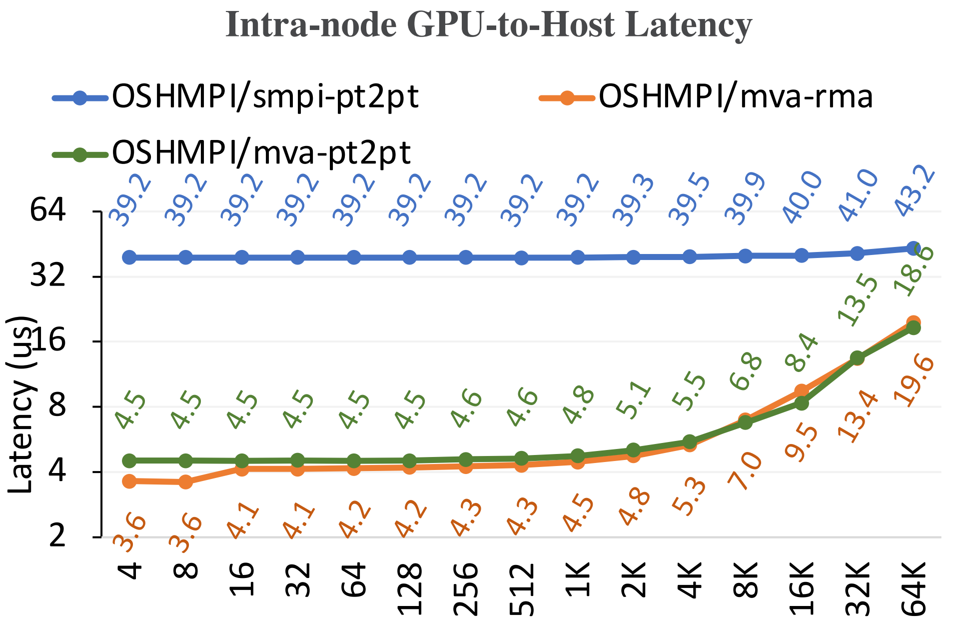
- Extended OSU benchmark osu_oshm_put with memory space and CUDA memkind
  - Experiments: GPU-to-GPU, GPU-to-Host, Host-to-GPU for both intranode and internode latency
- All experiments were performed on Summit
- OSHMPI can portably support ***CPU-initiated*** mode by leveraging various GPU-aware MPI implementations
  - IBM Spectrum MPI (smpi): supports GPU only for PT2PT
  - MVAPICH-GDR (mva): supports GPU for both PT2TP and RMA, but segfaults at internode transfer when size >= 4Kbytes
- NVSHMEM: as reference of *GPU-initiated* SHMEM
  - Support only GPU-to-GPU and Host-to-GPU in version 1.0.1

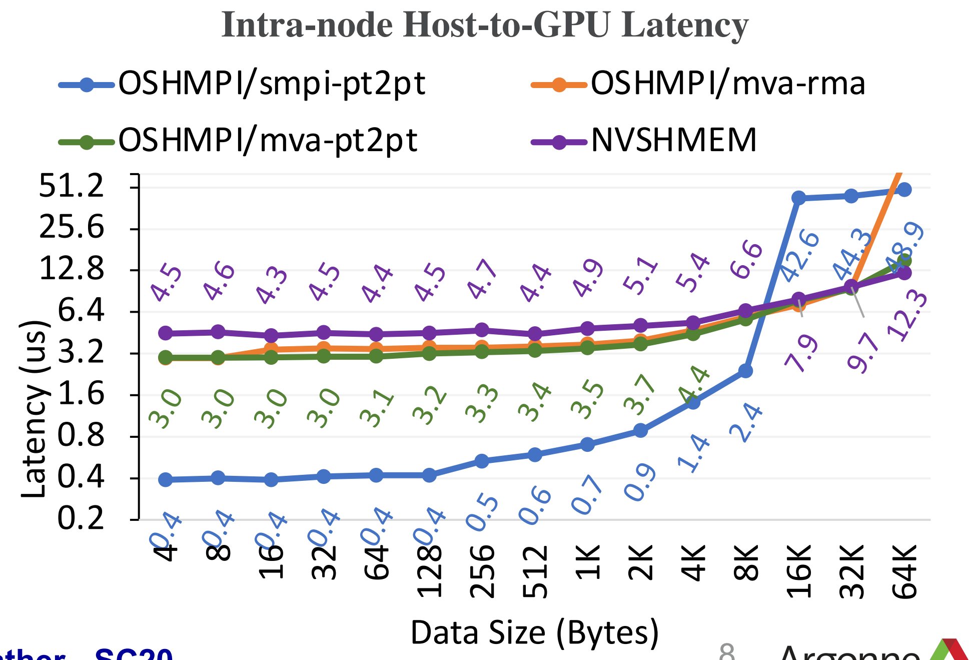




GPU-Aware OpenSHMEM Evaluation (2)

- **Observations**
  - **OSHMPI over MPI***:* performance trend of each option varies in different data transfer direction
    - OSHMPI/MVA-pt2pt delivered the lowest latency in *GPU-to-GPU*, *GPU-to-Host* directions
    - But OSHMPI/SMPI-pt2pt performs better in the *Host-to-GPU* direction
    - Analysis for the root cause of such performance diversity is still ongoing
  - **NVSHMEM:** delivered relatively high latency
    - Might be caused by high software overhead since the data transfer is performed by a low-frequency GPU thread





* Inter-node GPU-to-Host and Host-to-GPU results share a similar trend. Graphs are omitted.

Summary

- OSHMPI as a ***serious performance contender***
  - Analysis & optimizations focused on essential RMA operations (optimizations are also valid for AMO)
  - **Optimized OSHMPI/MPICH can deliver similar performance as that of the native impls**
  - **No visible gap compared to SOS/OPA, ~5% overhead compared to OSHMEM/IB !**
- OSHMPI as a ***GPU-aware OpenSHMEM implementation***
  - Explored memory space extension for supporting GPU space heap
  - Portably support ***CPU-initiated*** communication by leveraging both GPU-aware MPI PT2PT and RMA
- Ongoing / next step:
  - Overhead analysis & optimizations for GPU-aware OpenSHMEM
  - Automatic MPI GPU feature detection without user hints
  - OpenSHMEM 1.5 support (e.g., team, nonblocking AMO...)
  - Thorough analysis and optimization for team-based collectives and AMO

- *All optimizations and new features are available on GitHub:*
  - *OSHMPI: https://github.com/pmodels/oshmpi*
  - *MPICH: https://github.com/pmodels/mpich*
- *Will be included in the upcoming releases of OSHMPI and MPICH.*