OpenSHMEM over MPI: Capabilities and Challenges

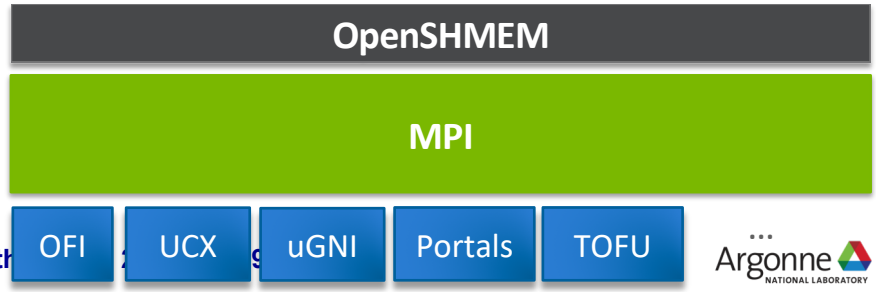
Min Si, Pavan Balaji

Programming Models and Runtime Systems Group

Argonne National Laboratory, USA

Overview of OpenSHMEM over MPI

- **OpenSHMEM**
  - Specialized API 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 overhead cannot be avoided
  - ***Why does OpenSHMEM over MPI not perform well?***
    - MPI implementation does not optimize the RMA routines
    - Over-generalized functionality of MPI



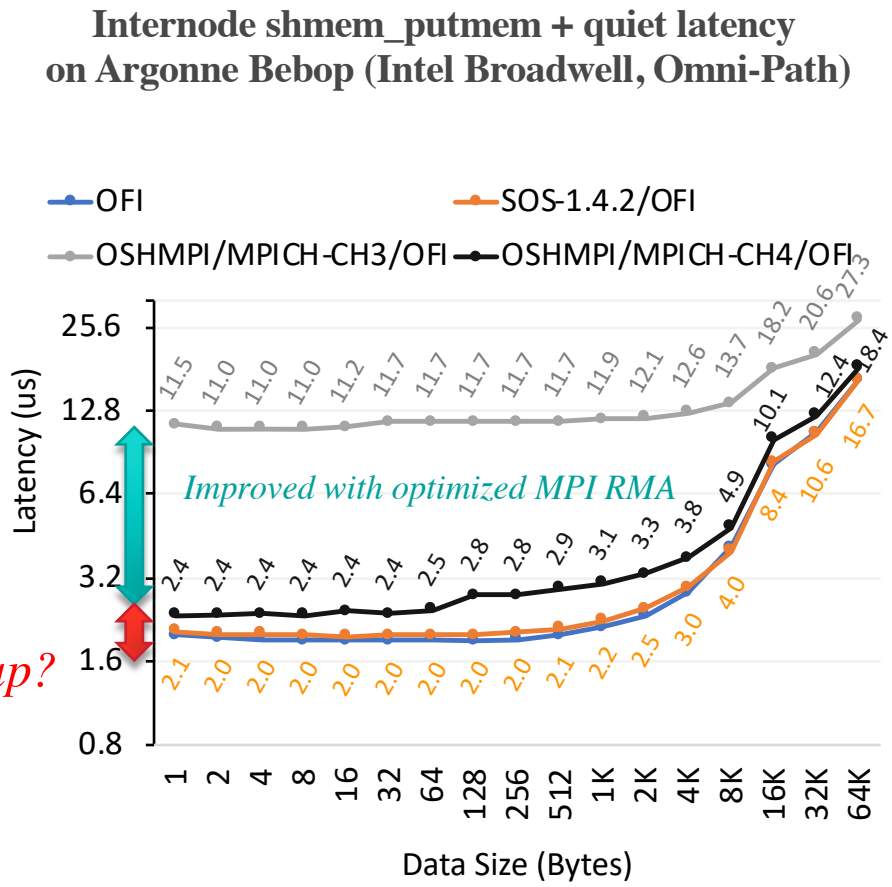
Demonstrating OSHMPI Software Overhead

- Measured SHMEM latency between two PEs

```
shmem_putmem(dest, source, nelems, pe);
shmem_quiet();
```

- Evaluated implementations:
  - **OFI**: ideal performance
  - **SOS**: native implementation
  - **OSHMPI/MPICH-CH3/OFI**: original implementation
  - **OSHMPI/MPICH-CH4/OFI**: with highly optimized MPI RMA

What causes the remaining gap?



Instruction Count Analysis and Optimizations (SHMEM_PUTMEM)

OSHMPI/MPICH-CH4

	Original	OPT
OSHMPI: calling overhead	6	
OSHMPI: obtain MPI parameters (win, disp)	13	
MPI_Put: calling overhead	9	
MPI_Put: obtain window object	14	
MPI_Put: translate rank to network addr	17	2
MPI_Put: decode MPI datatype	22	14
MPI_Put: obtain dest virtual address	8	
MPI_Put: prepare OFI parameters	14	
MPI_Put: check OFI conditions	13	10
MPI_Put: other	15	13
Flush_local: calling overhead	8	
Flush_local: obtain window object	7	
Flush_local: wait OFI completion	38	
Flush_local: MPI full progress	81	24
Flush_local: find target AM object	59	7
Flush_local: other	9	7
TOTAL	**333**	**194**

SOS

Calling overhead	6
Prepare OFI parameters	24
Check OFI conditions	7
Other	34
TOTAL	**71**

Necessary functionality

MPI semantics requirement

Implementation dependent

Instruction Count Analysis and Optimizations (SHMEM_QUIET)

OSHMPI/MPICH-CH4

	Original	OPT
OSHMPI: calling overhead	2	
Flush_all: calling overhead	4	
Flush_all: obtain window object	7	
Flush_all: wait OFI completion	14	
Flush_all: MPI full progress	81	24
Flush_all: traverse target AM objects	130	15
Flush_all: other	29	
(Flush_all for symm_data_win)	**267**	**2**
Win_sync: calling overhead	4	
Win_sync: memory fence	1	
(Win_sync for symm_data_win)	**5**	**2**
TOTAL	**544**	**104**

SOS

Calling overhead	2
Wait OFI completion	51
Other	38
TOTAL	**91**

Necessary functionality

MPI semantics requirement

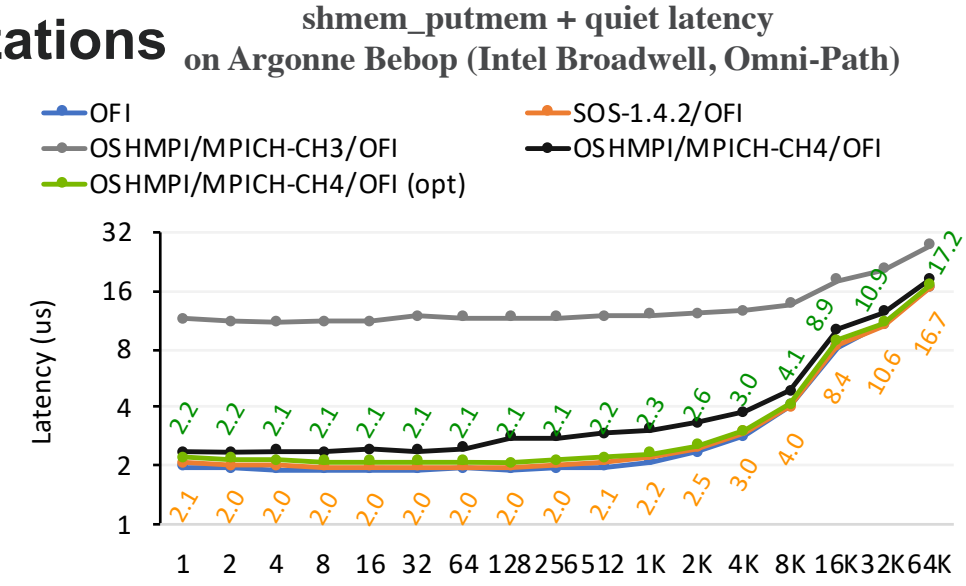
Implementation dependent

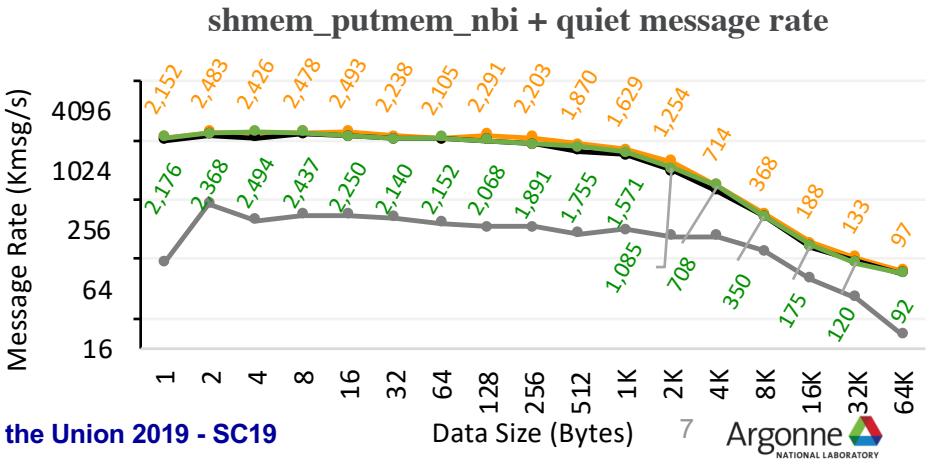
Key Bottlenecks & Optimizations

- Datatype decoding
  - OSHMPI uses only basic MPI datatypes (e.g., MPI_INT), and same type for src and dest
  - **Optimization**: provide fast-path for basic datatypes in MPICH
- Rank to network address translation
  - Expensive lookup overhead because rank can be arbitrarily reordered in each communicator
  - OSHMPI uses only COMM_WORLD (or dup of COMM_WORLD)
  - **Optimization**: provide fast-path for COMM_WORLD rank-to-network-address translation
- Window synchronization (i.e., MPI_Win_flush_all, MPI_Win_sync)
  - OSHMPI creates separate windows for symmetric heap and global/static variables
  - MPI requires per-win flush & sync to ensure completion, trigger expensive “MPI full progress”
  - **Optimization**: skip window synchronization if no outstanding OP exists
- Expensive MPI full progress
  - Ensure prompt progress for all MPI communication types (i.e., P2P, coll, AM-based)
  - MPI full progress may be unnecessary for OSHMPI
  - **Optimization-1**: refactor MPICH code to reduce some progress overhead
  - **Ongoing**: generic approach to eliminate expensive “MPI full progress”

Performance Evaluation with Optimizations

- **Latency**:
  - Remaining gap between optimized OSHMPI/MPICH and SOS is less than 5%
  - Improvement mainly contributed by reduced MPI progress overhead in both shmem_putmem and quiet
- **Message rate**:
  - Optimized OSHMPI/MPICH delivers similar message rate as that of SOS
  - Improvement contributed by fast-path optimizations in MPI_Put

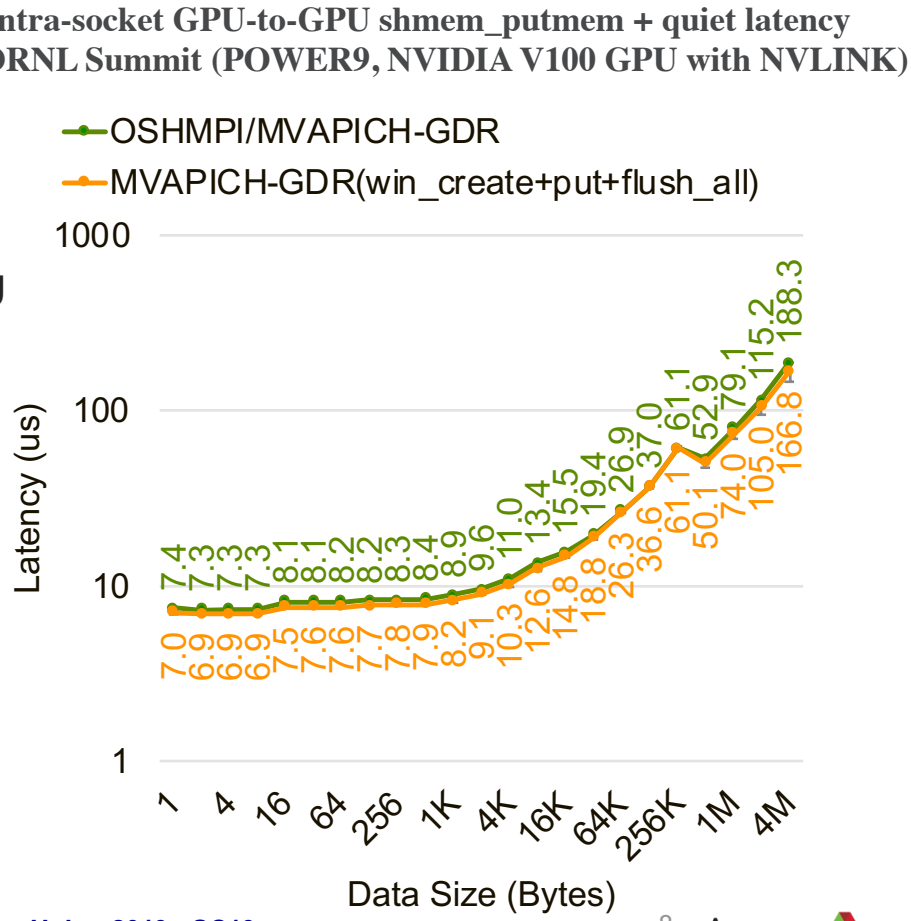




Initial Investigation of OpenSHMEM over GPU-aware MPI

- GPU-aware communication
  - **CPU-initialized** or GPU-initialized
  - Data transfer between GPUs, GPU-host buffers
  - GPUDirect support for GPU data transfer
    - IPC for intranode GPU-to-GPU
    - GPUDirect RDMA for internode GPU-to-GPU
- OpenSHMEM over GPU-aware MPI
  - OSHMPI/RMA/**MVAPICH-GDR**
  - Enable GPU symmetric heap in OSHMPI through extended API

```
dest_on_gpu = shmemx_cuda_malloc(size);
...
shmem_putmem(dest_on_gpu, source_on_gpu,
                          nelems, pe);
shmem_quiet();
...
shmemx_cuda_free(dest_on_gpu);
...
```



Summary

- OSHMPI performance analysis and optimizations
  - Focused on essential RMA operations (optimizations are also valid for AMO)
  - Instruction-count level analysis explains the root cause of performance gap with native OpenSHMEM implementations
  - Optimized OSHMPI/MPICH can deliver similar performance as that of native implementations (~5% gap with SOS)
- OpenSHMEM over GPU-aware MPI
  - Initial investigation on ORNL Summit showed that OpenSHMEM can support direct GPU-GPU communication via MPI RMA with very low development cost
- Ongoing / next step:
  - Performance analysis and optimizations for OpenSHMEM collectives and atomics
  - OpenSHMEM 1.5 support (e.g., team, wait_{until|test}_{all|any|some}, nonblocking AMO...)