NETWORK-BASED COMPUTING LABORATORY

A Comprehensive Performance Evaluation of OpenSHMEM Libraries on InfiniBand Clusters

Jithin Jose, Jie Zhang, Akshay Venkatesh,
Sreeram Potluri, and Dhabaleswar K. Panda

Network-Based Computing Laboratory
Department of Computer Science and Engineering
The Ohio State University, USA

OHIO STATE

Outline

- Introduction and Motivation
- OpenSHMEM Communication Operations
- Evaluation Methodology
- Experimental Evaluations
- Overview of Results
- Conclusion

Introduction

- Partitioned Global Address Space (PGAS) models – increased interest in community
- Offer betters programmability; data driven Irregular applications can be easily expressed in PGAS models
- OpenSHMEM (http://openshmem.org/)
  - Library based PGAS model
  - Open specification to standardize the SHMEM (SHared MEMory) model
  - Simple model; allows existing vendor specific SHMEM codes to be made platform-independent with minimal effort

OpenSHMEM Implementations over InfiniBand

- OpenSHMEM Reference Implementation (UH-SHMEM)
  - Reference Implementation from University of Houston
  - www.openshmem.org
  - Uses GASNet as underlying runtime
- Scalable-SHMEM (Scalable-SHMEM)
  - OpenSHMEM implementation from Mellanox
  - www.mellanox.com/page/scalablehpc_overview
- OpenMPI-SHMEM (OMPI-SHMEM)
  - OpenSHMEM implementation from OpenMPI group
  - www.open-mpi.org
- MVAPICH2-X SHMEM (MV2X-SHMEM)
  - OpenSHMEM implementation over MVAPICH2-X Unified Runtime
  - Based on MVAPICH2
  - www.mvapich.cse.ohio-state.edu

Motivation

- Multi-core processors and high-performance interconnects have been driving the growth of HPC systems
- Several multi-core aware designs have been presented for implementing OpenSHMEM operations on InfiniBand clusters
- It is important for application developers to understand the performance of the various implementations, and choose the one that is right for their application and system
- Need a systematic performance comparison and an analysis of different implementations, to make this choice

Contributions

- Presents a detailed comparison of the performance of different OpenSHMEM implementations using point-to-point and collective micro-benchmarks
- Detailed analysis of performance trends observed in different OpenSHMEM implementations
- Performance analysis of application kernels, showing how users can draw a correlation between the micro-benchmark results and application performance

Outline

- Introduction and Motivation
- OpenSHMEM Communication Operations
- Evaluation Methodology
- Experimental Evaluations
- Overview of Results
- Conclusion

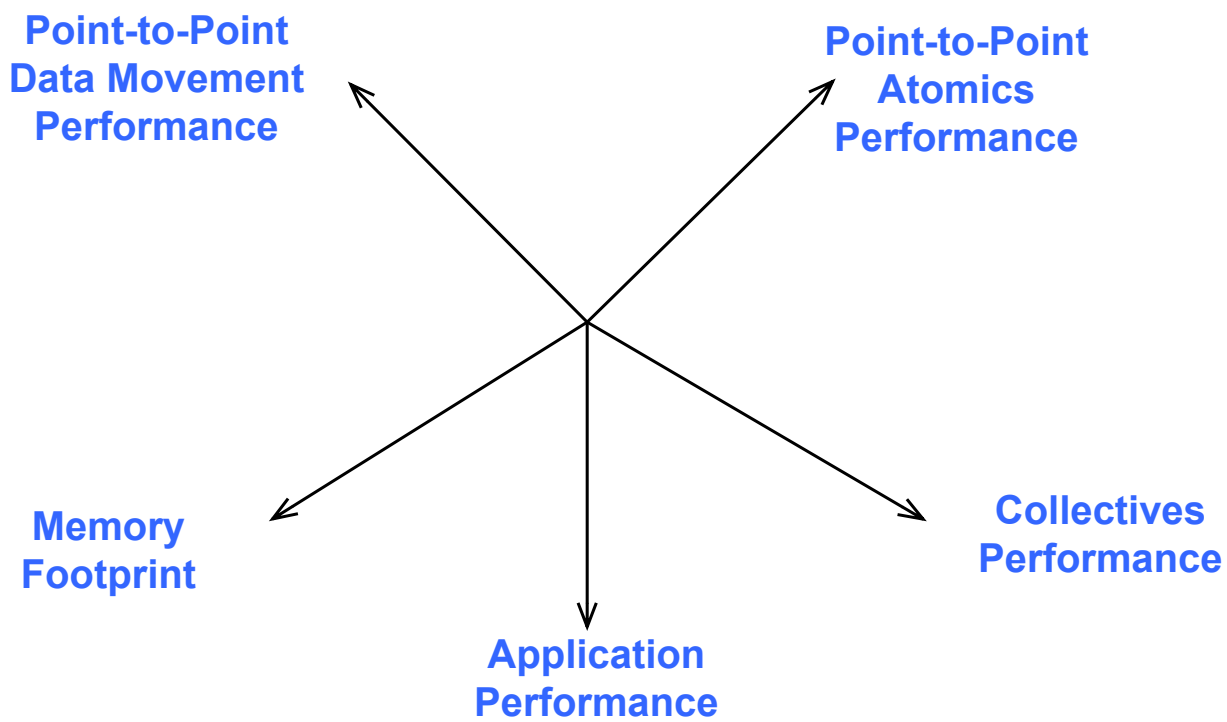
OpenSHMEM Communication Operations

- Data Movement Operations
  - Point-to-point data movement
  - shmem put, shmem get
- Atomic Operations
  - Updates remote memory atomically
  - shmem_swap, shmem_cswap
  - shmem_fadd, shmem_finc
  - shmem_add, shmem_inc
- Collective Operations
  - Participating PEs can be dynamically defined
  - shmem_broadcast, shmem_barrier, shmem_collect, shmem_fcollect, and shmem_reduce

Outline

- Introduction and Motivation
- Communication Operations
- Evaluation Methodology
- Experimental Evaluations
- Overview of Results
- Conclusion

Evaluation Methodology



- Five pronged approach
  - Performance of point-to-point data movement, point-to-point atomics, collectives operations
  - Memory Scalability
  - Application Performance

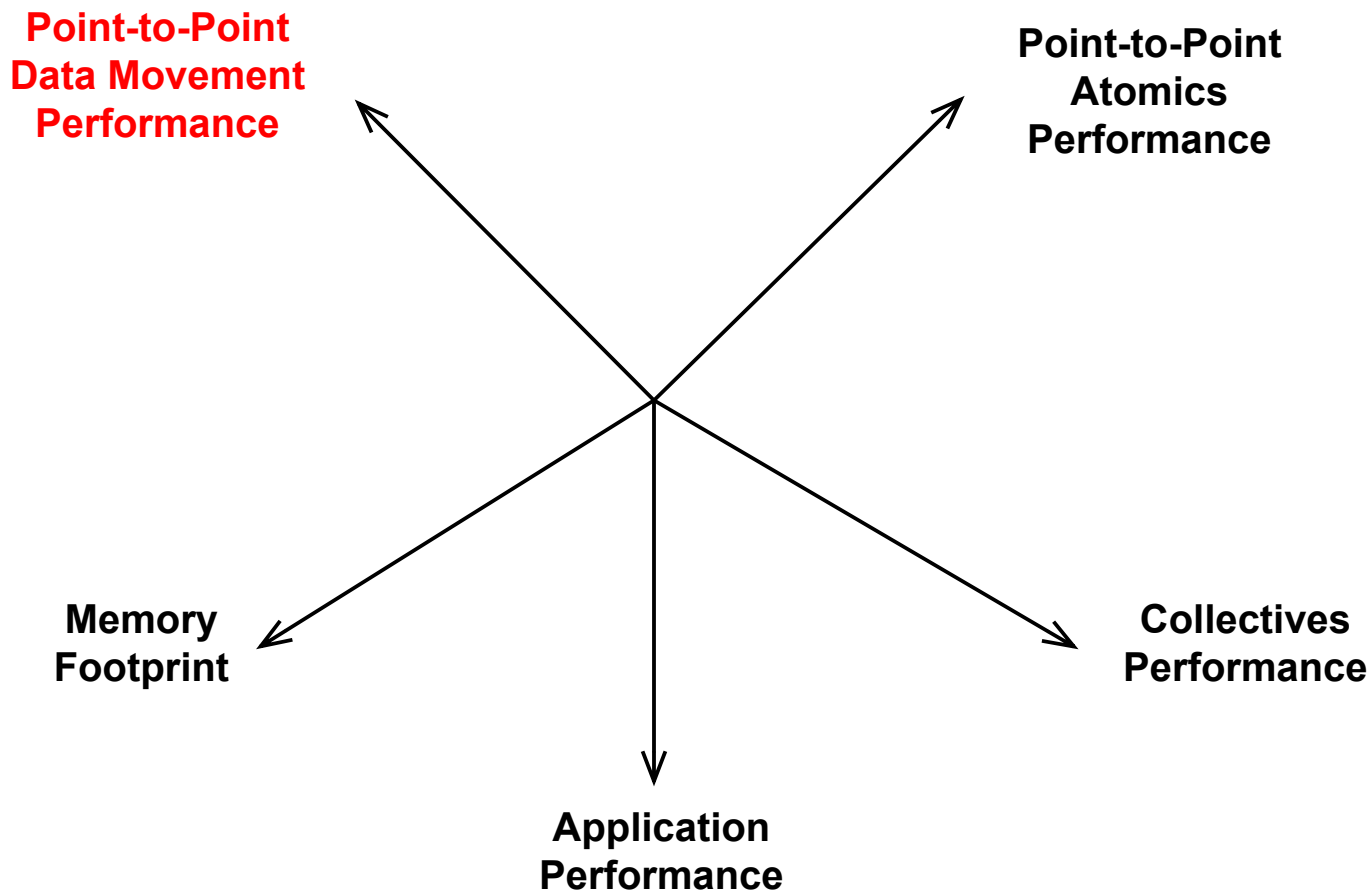
Outline

- Introduction and Motivation
- Communication Operations
- Evaluation Methodology
- Experimental Evaluations
- Overview of Results
- Conclusion

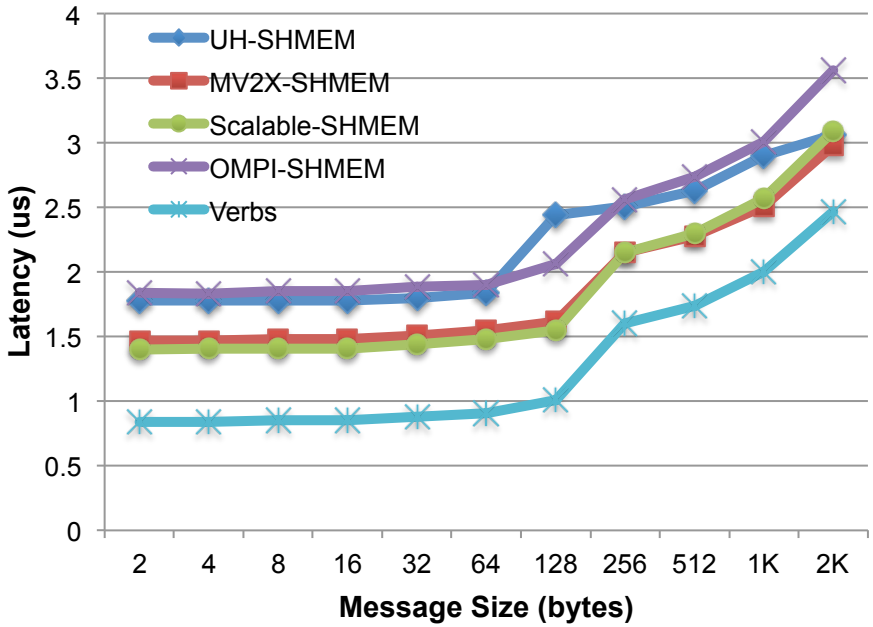
Experiment Setup

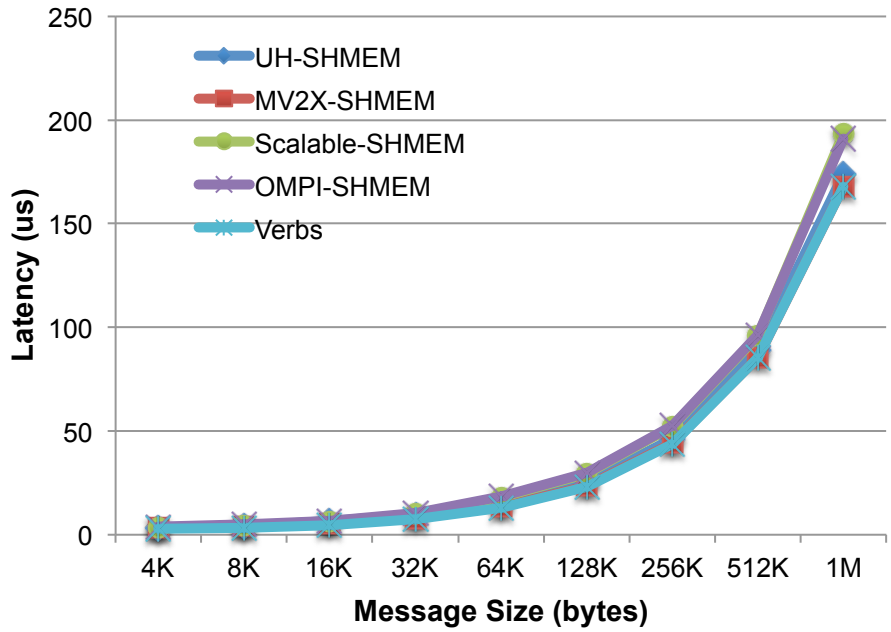
- ClusterA: TACC Stampede
  - Intel Sandybridge series of processors using Xeon dual 8 core sockets (2.70GHz) with 32GB RAM
  - Each node is equipped with FDR ConnectX HCAs (54 Gbps data rate) with PCI-Ex Gen3 interfaces
- Cluster B: OSU Cluster
  - Xeon Dual quad-core processor (2.67GHz) with 12GB RAM
  - Each node is equipped with QDR ConnectX HCAs (32Gbps data rate) with PCI-Ex Gen2 interfaces
  - Mellanox FCA
- Software Stacks
  - MVAPICH2-X OpenSHMEM (v2.0b)
  - UH-SHMEM (v1.0f), UH-SHMEM (v1.0d) over GASNet (v1.20.2)
  - OMPI-SHMEM (trunk version, Feb 15, 2014)
  - Scalable-SHMEM (v2.2)
- Benchmark Suite
  - OSU Benchmarks (v4.2)

Experimental Evaluations



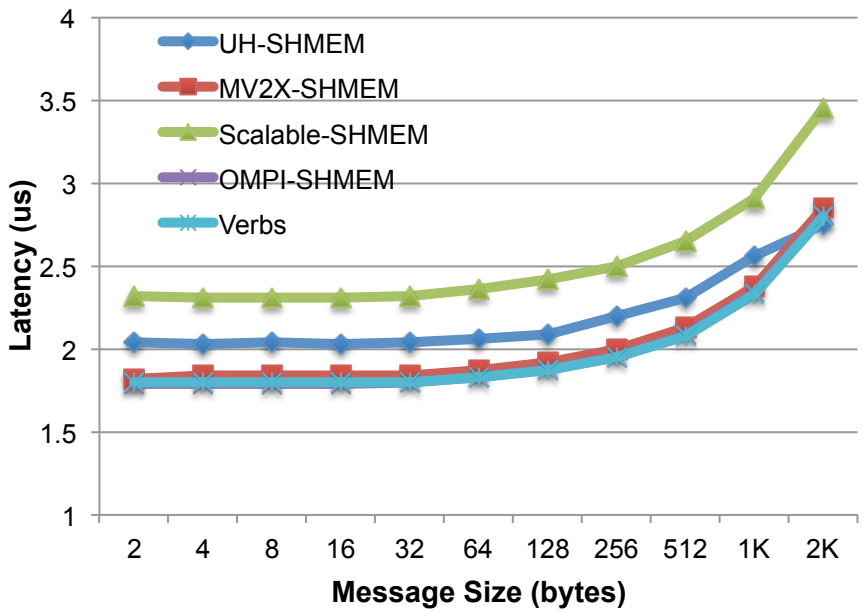
shmem_putmem Performance

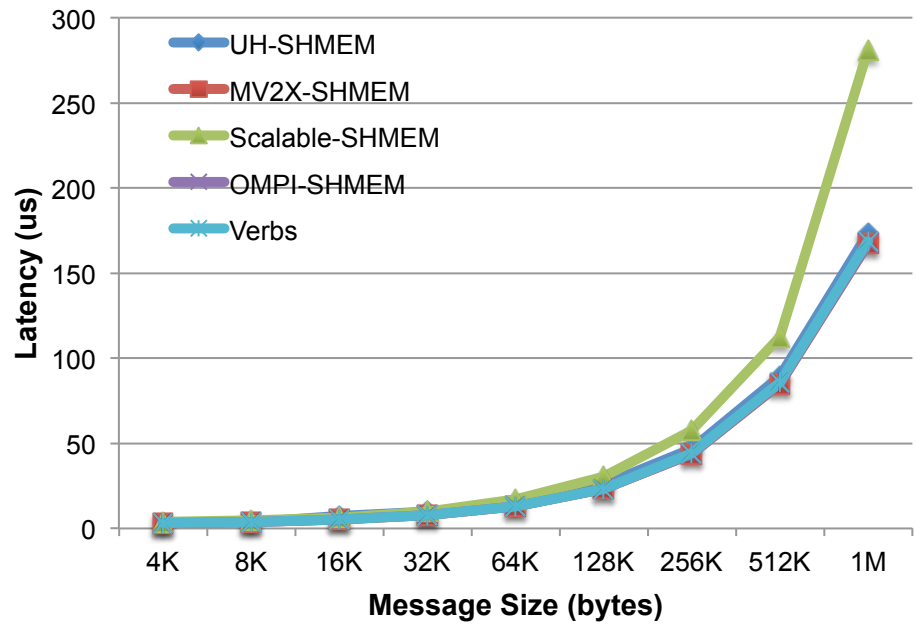




- Benchmark: Processing Element (PE) 0 issues shmem_putmem to write data at PE 1 and then calls shmem quiet; PE 1 waits on shmem_barrier
- Comparison with InfiniBand verbs level RDMA write
- Latency for 4-byte put operation (us): UH-SHMEM - 1.78, MV2X-SHMEM - 1.47, Scalable-SHMEM - 1.41, and OMPI-SHMEM - 1.83
- Verbs level write latency – 0.84 us

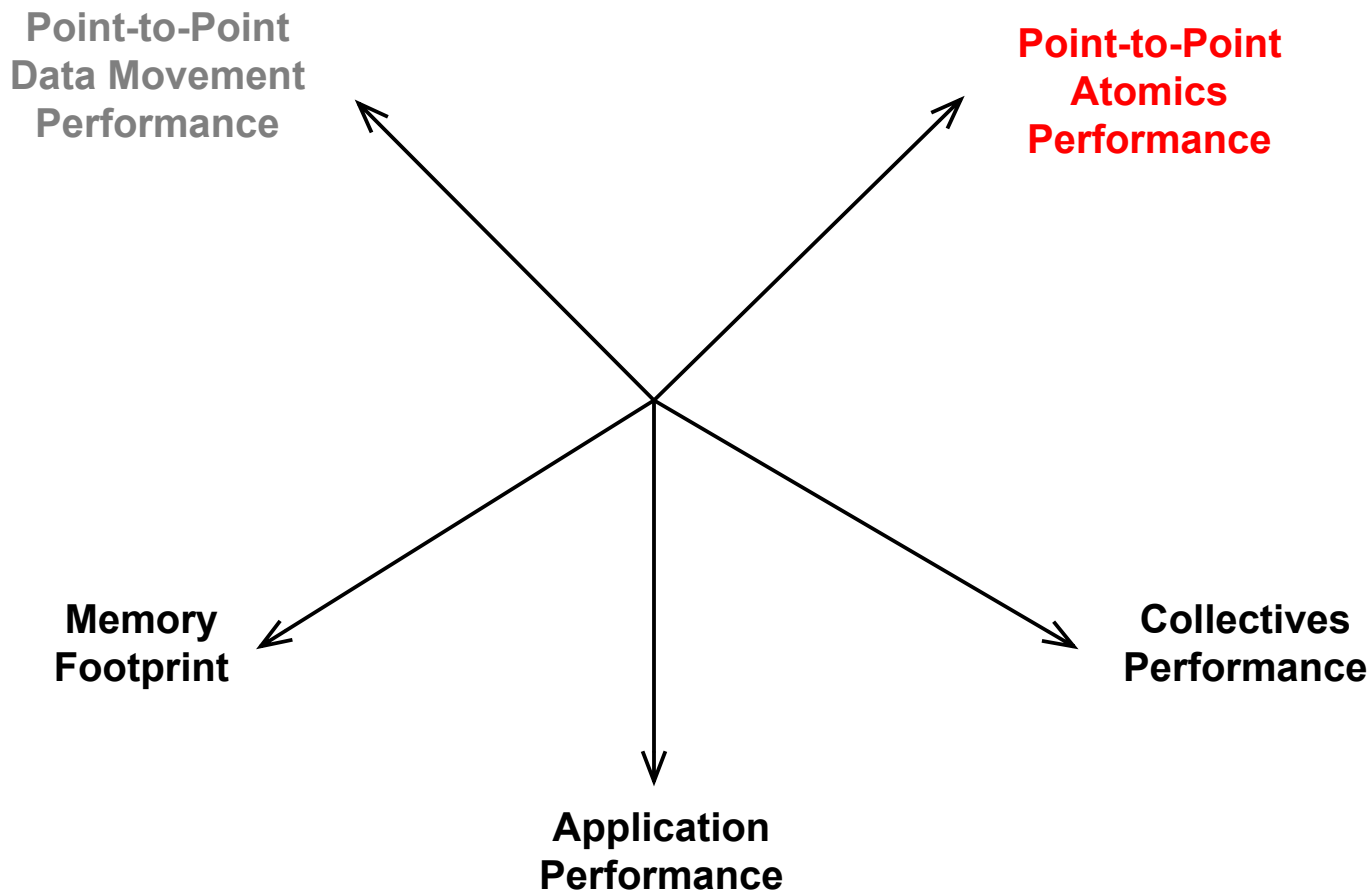
shmem_get Performance



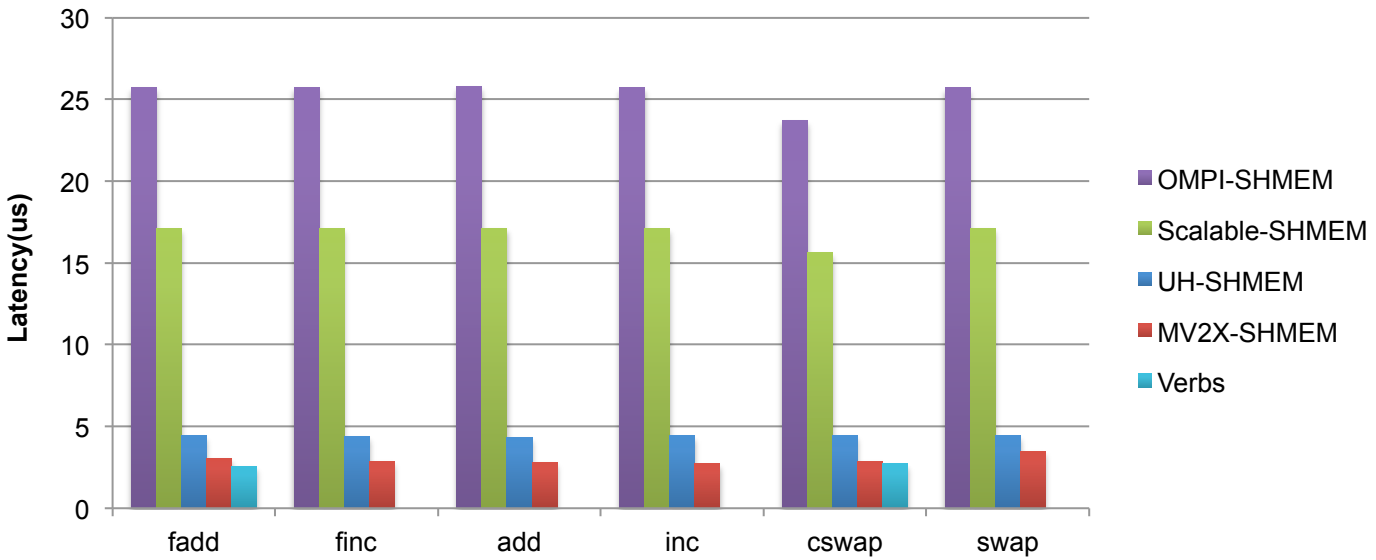


- Benchmark: PE0 issues shmem_getmem to fetch data at PE 1; PE 1 waits on shmem_barrier
- Comparison with InfiniBand verbs level RDMA read
- Latency for 4-byte put operation (us): UH-SHMEM – 2.04, MV2X-SHMEM – 1.84, Scalable-SHMEM – 2.31, and OMPI-SHMEM – 1.79
- Verbs level read latency – 1.74 us

Experimental Evaluations

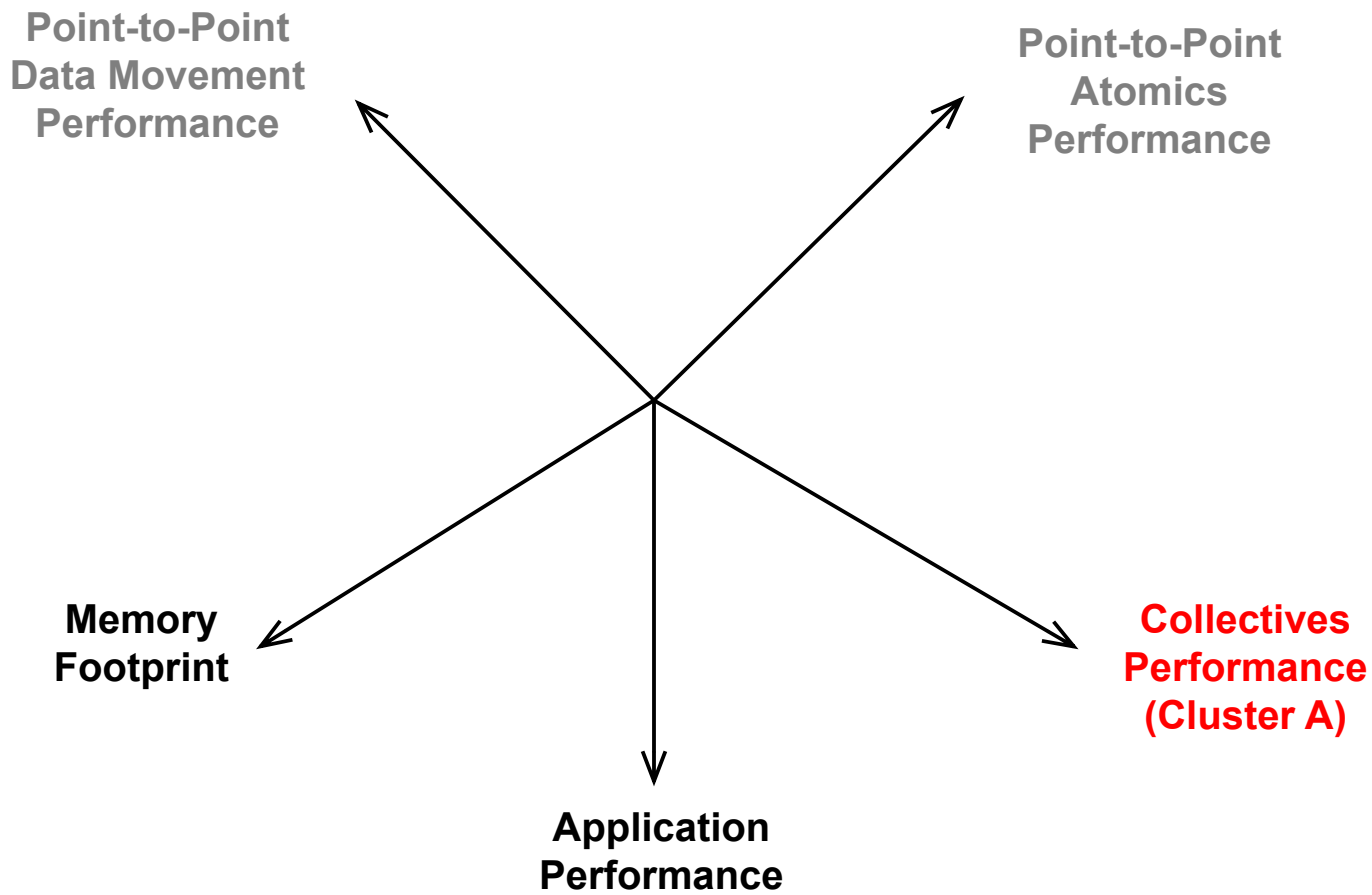


Atomics Performance

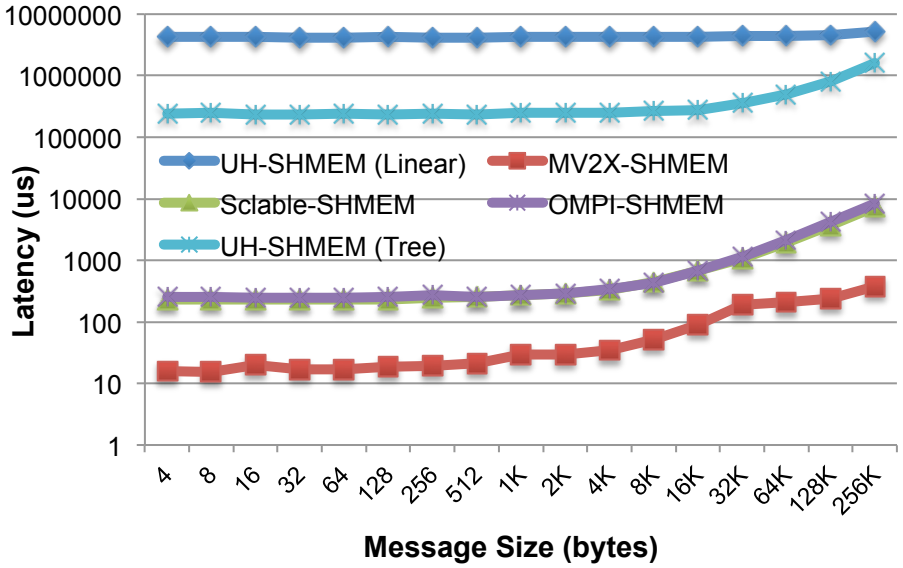


- Benchmark: PE0 issues atomic operation to PE 1; PE 1 waits on shmem_barrier
- Comparison with InfiniBand verbs level fetch-add and compare-swap
- Latency fetch-add operation (us): UH-SHMEM – 4.52, MV2X-SHMEM – 3.04, Scalable-SHMEM – 17.11, and OMPI-SHMEM – 25.74
- Verbs level latencies (us): fetch-add – 2.54, compare-swap – 2.71

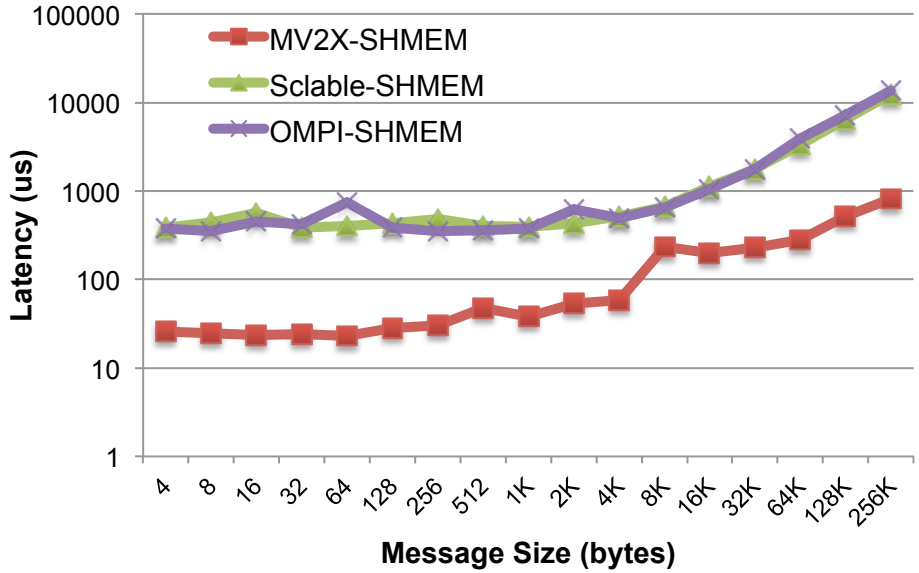
Experimental Evaluations



shmem_reduce Performance



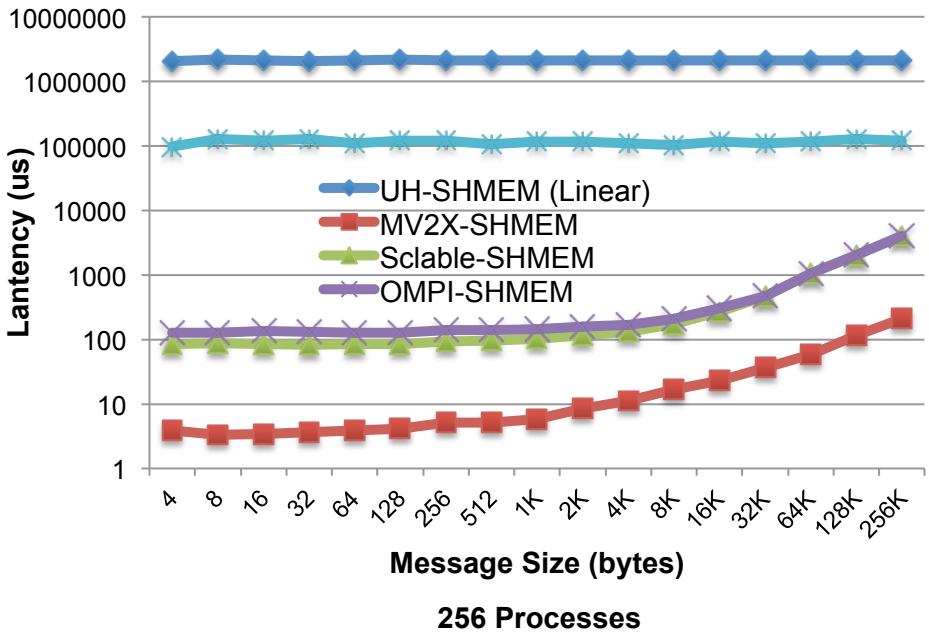
256 Processes



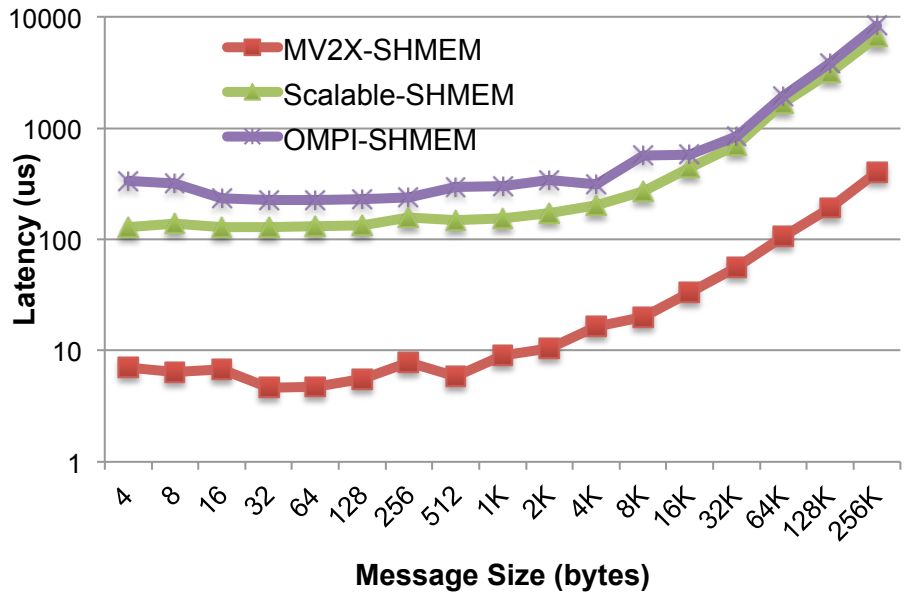
2,048 Processes

- Benchmark: measures average latency of collective operation across N processes, for various message lengths
- Tree and Linear versions available for UH-SHMEM
- Latency for 4-byte operation, with 2,048 processes (us):
  - MV2X-SHMEM – 25, Scalable-SHMEM – 240, and OMPI-SHMEM – 376

shmem_broadcast Performance



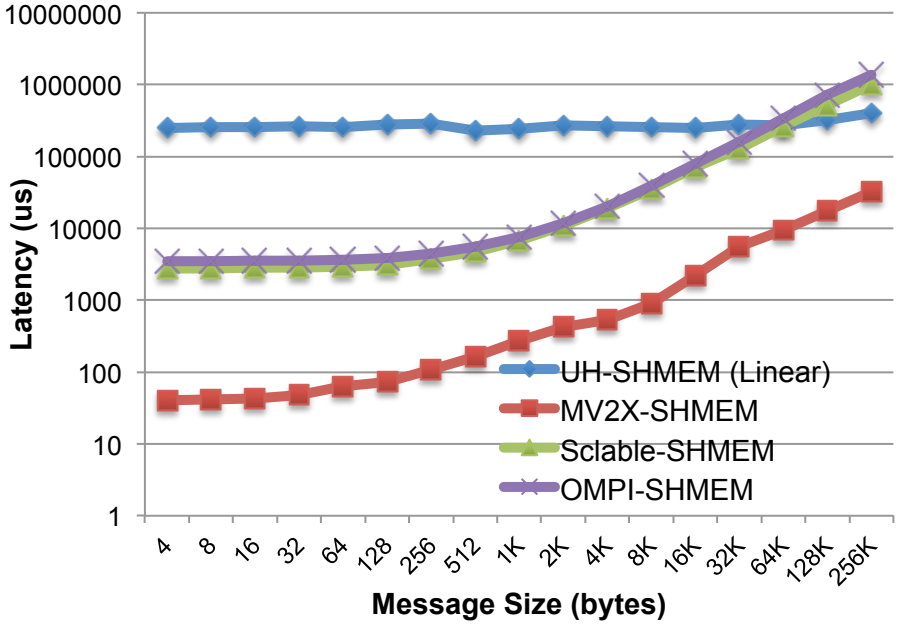
256 Processes



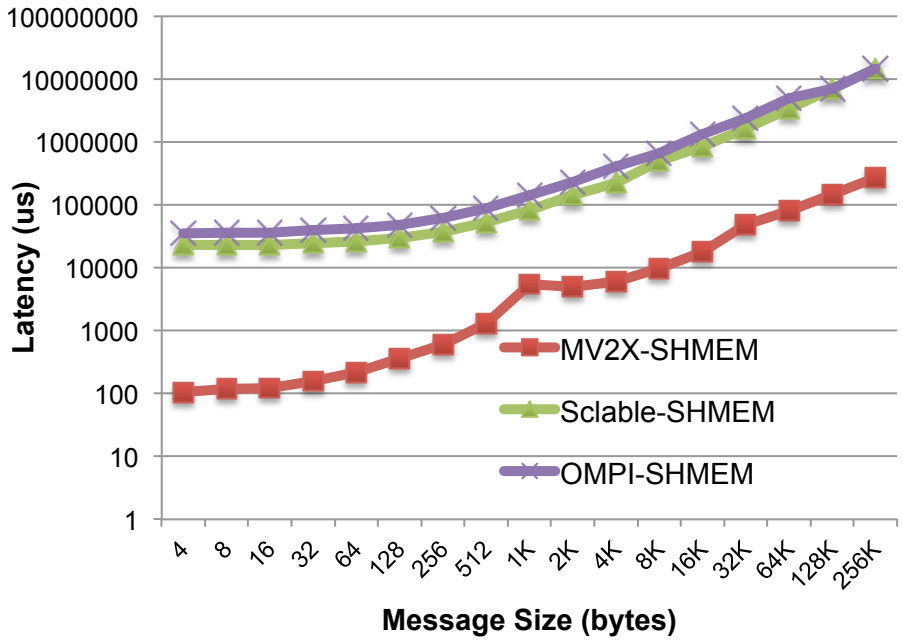
2,048 Processes

- Latency for 4-byte operation, with 2,048 processes (us):
  - MV2X-SHMEM – 7, Scalable-SHMEM – 129, and OMPI-SHMEM – 336
- Lower latencies with MV2-X SHMEM, similar performance for Scalable-SHMEM and OMPI-SHMEM

shmem_collect Performance



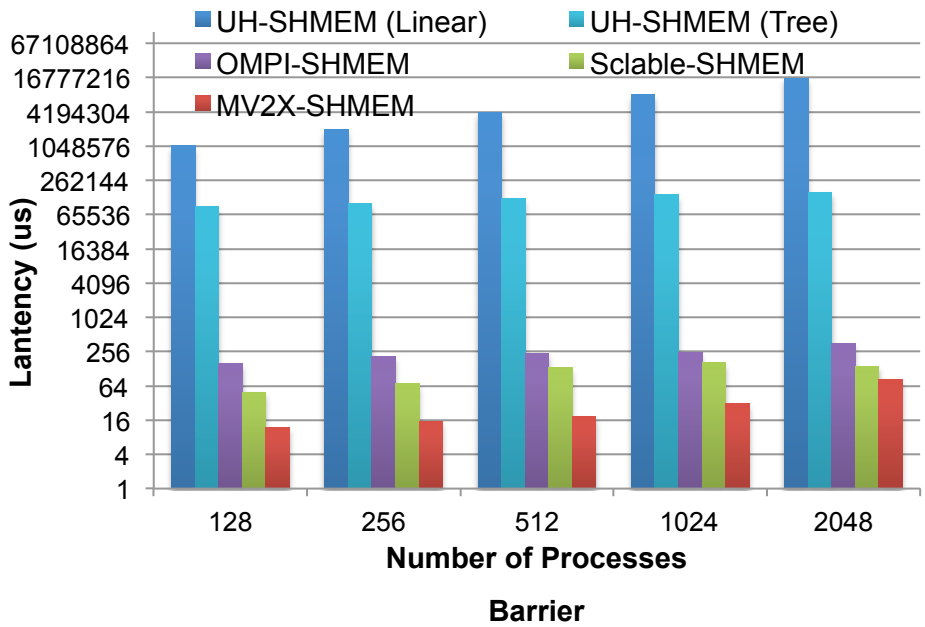
256 Processes

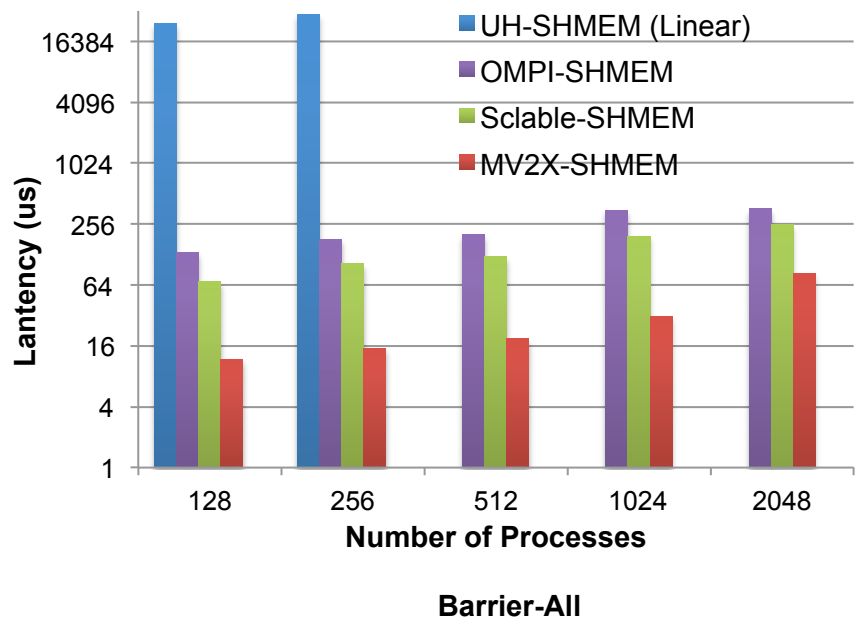


2,048 Processes

- Latency for 4-byte operation, with 2,048 processes (us):
  - MV2X-SHMEM – 105, Scalable-SHMEM – 22599, and OMPI-SHMEM – 35054
- Lower latencies with MV2-X SHMEM, similar performance for Scalable-SHMEM and OMPI-SHMEM

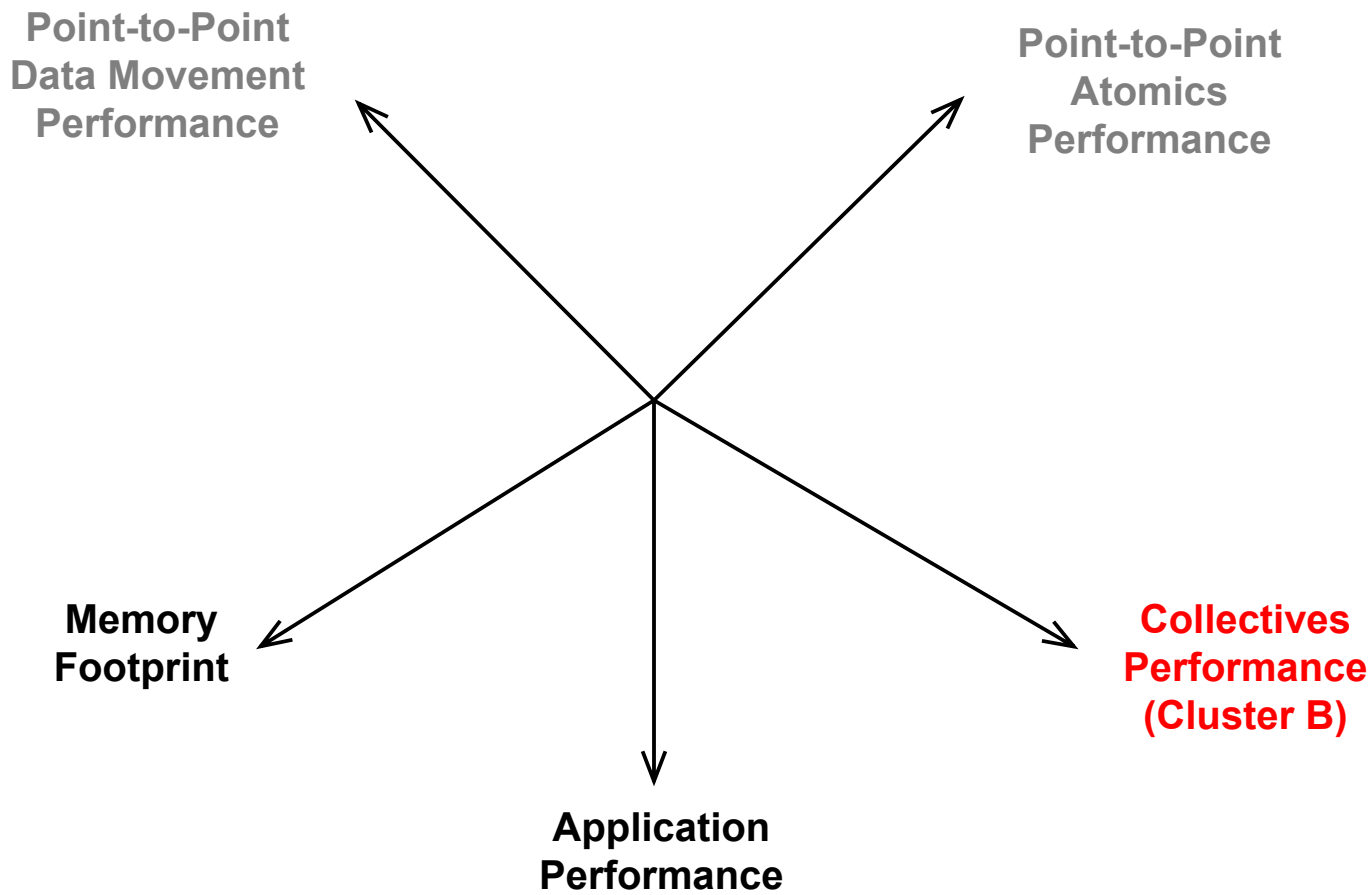
shmem_barrier Performance



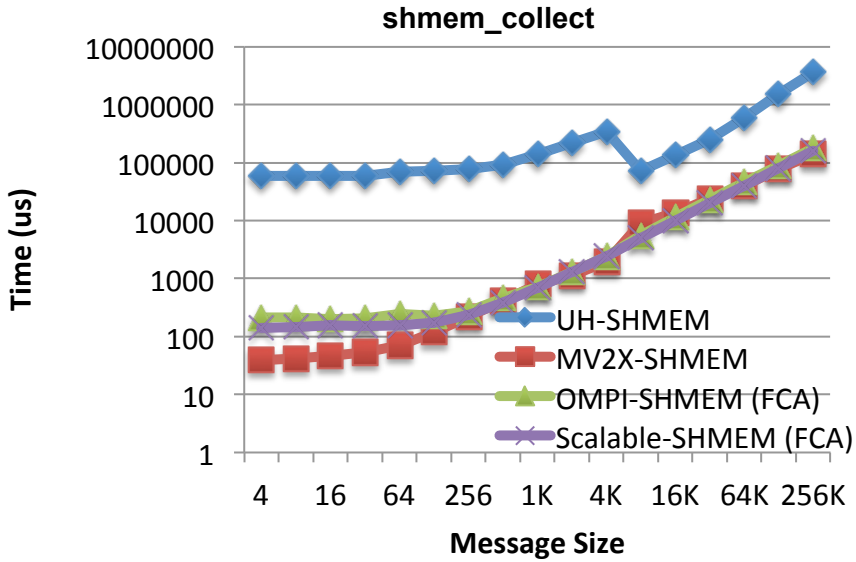


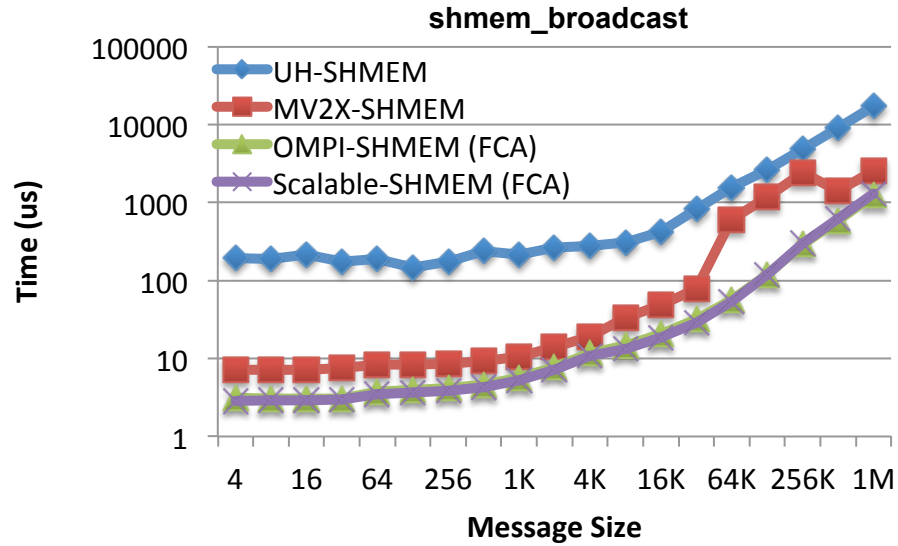
- Dynamic group creation for shmem_barrier; all PEs participate in shmem_barrier_all
- Barrier operation latency, with 2,048 processes (us):
  - MV2X-SHMEM – 83, Scalable-SHMEM – 141, and OMPI-SHMEM – 357
- Barrier-All operation latency, with 2,048 processes (us):
  - MV2X-SHMEM – 83, Scalable-SHMEM – 250, and OMPI-SHMEM – 361

Experimental Evaluations

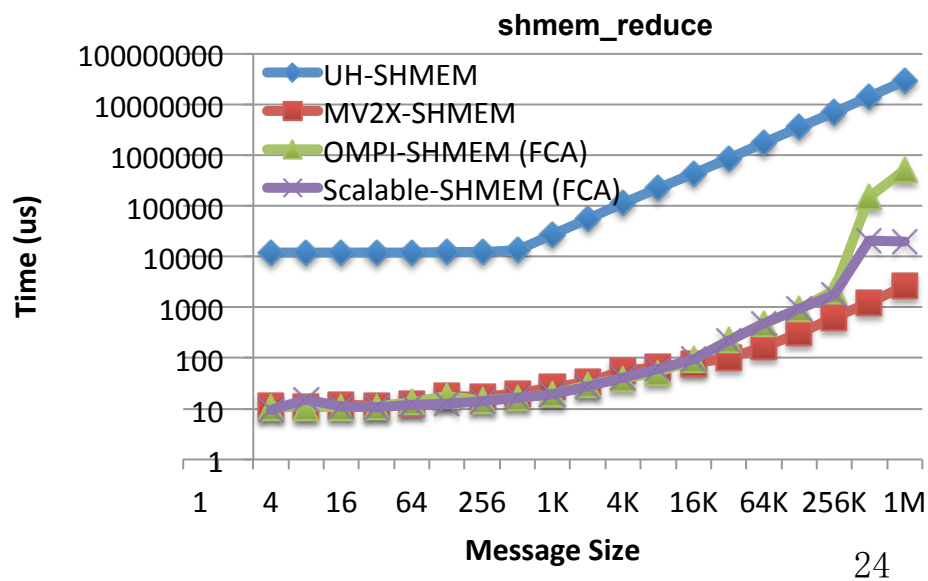


Collective Performance on Cluster B

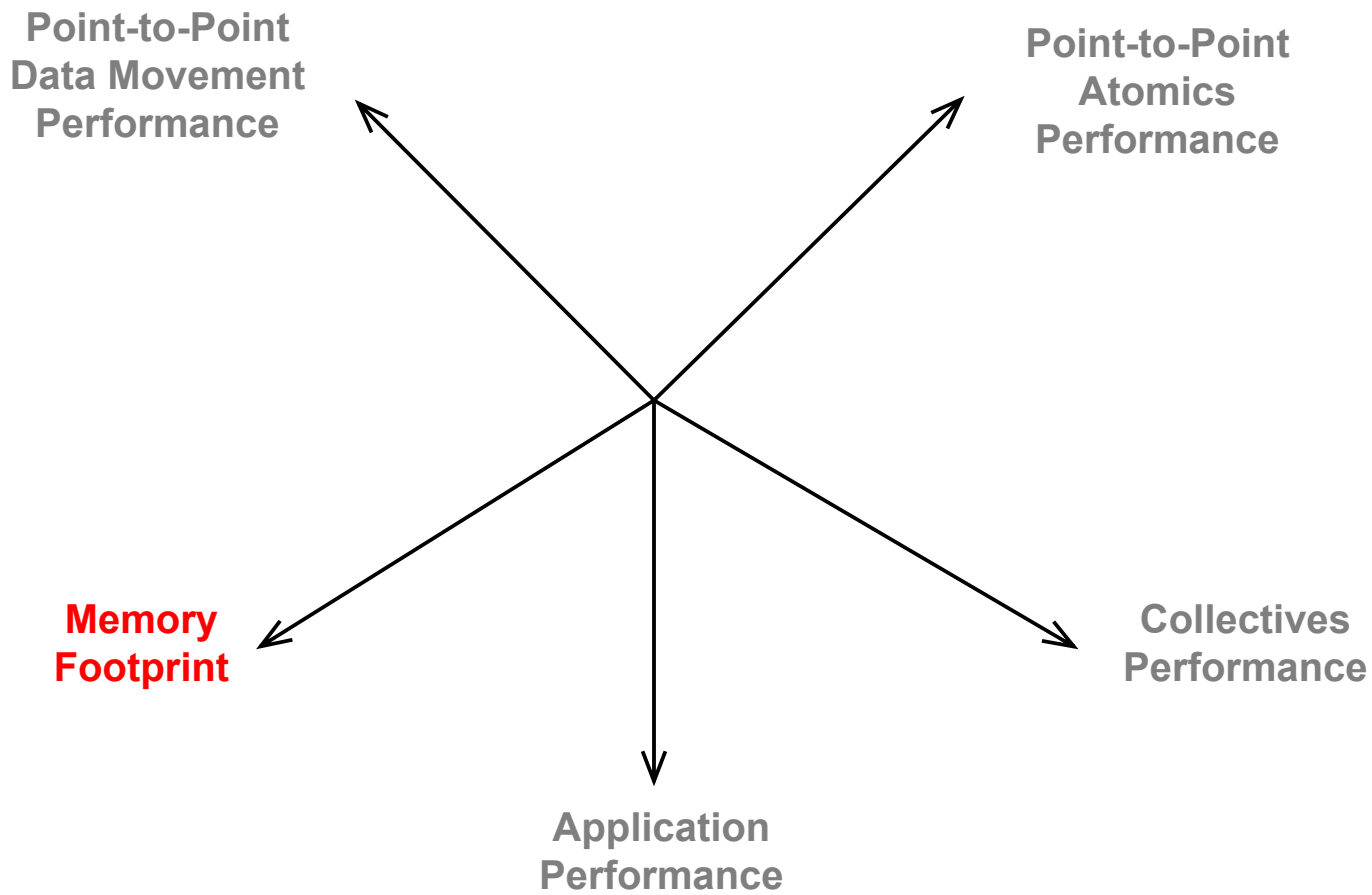




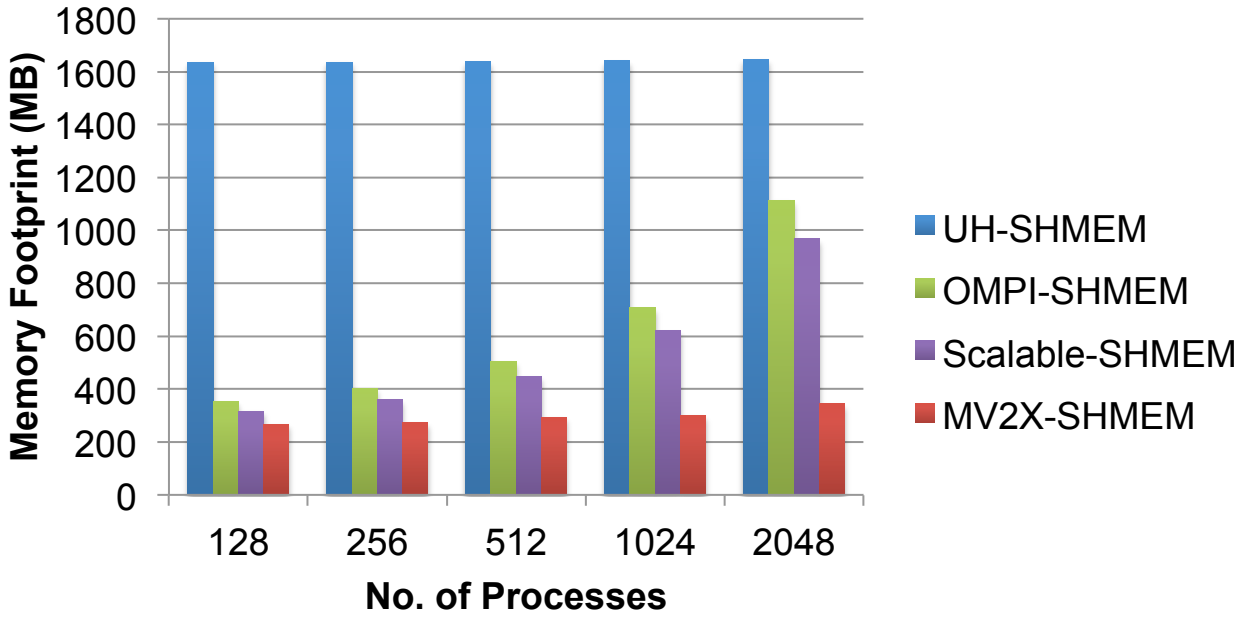
- Improved performance for OMPI-SHMEM and Scalable-SHMEM with FCA
- 8-byte reduce operation with 512 processes (us): MV2X-SHMEM – 10.9, OMPI-SHMEM – 10.79, Scalable-SHMEM – 11.97



Experimental Evaluations

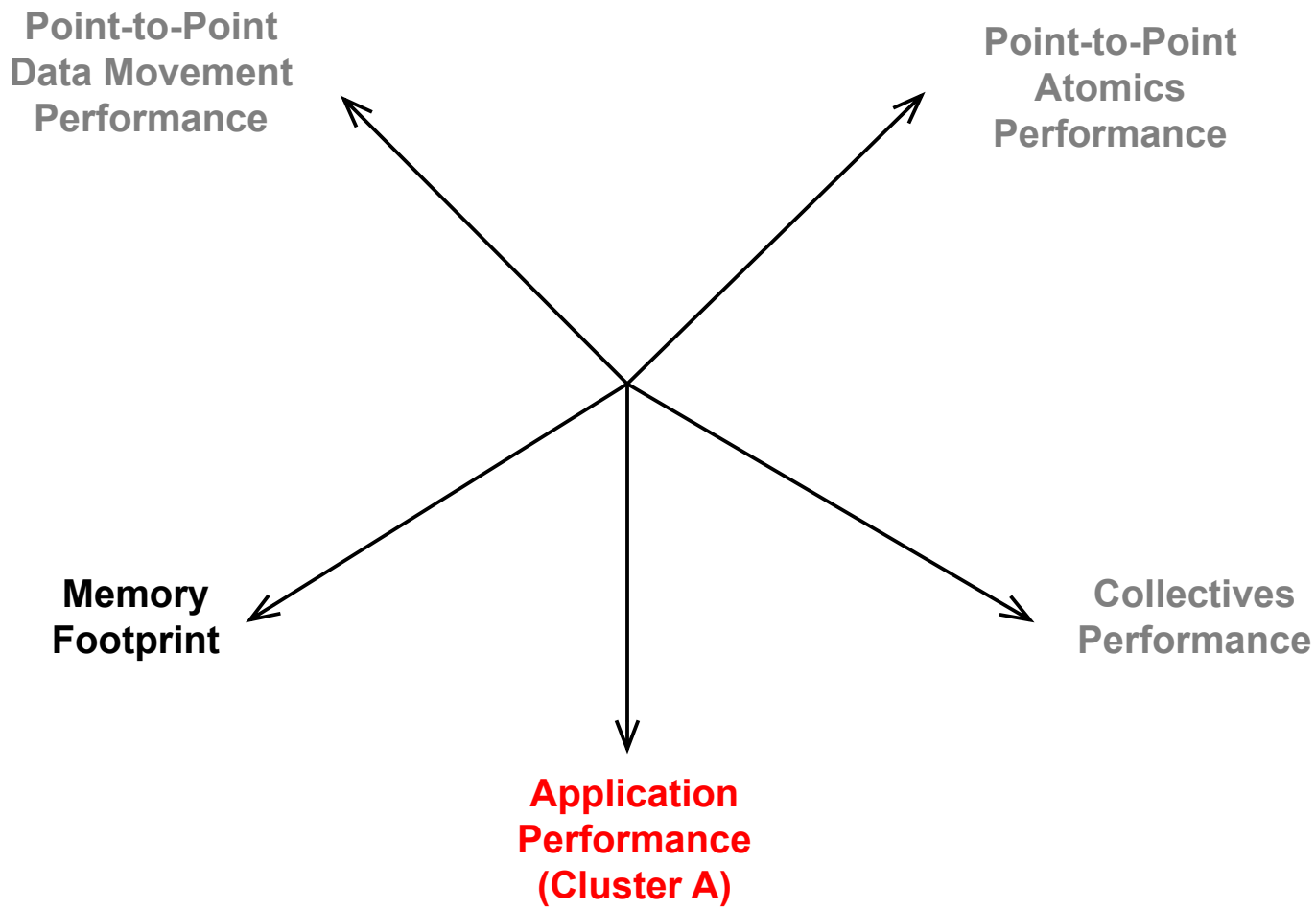


Memory Scalability

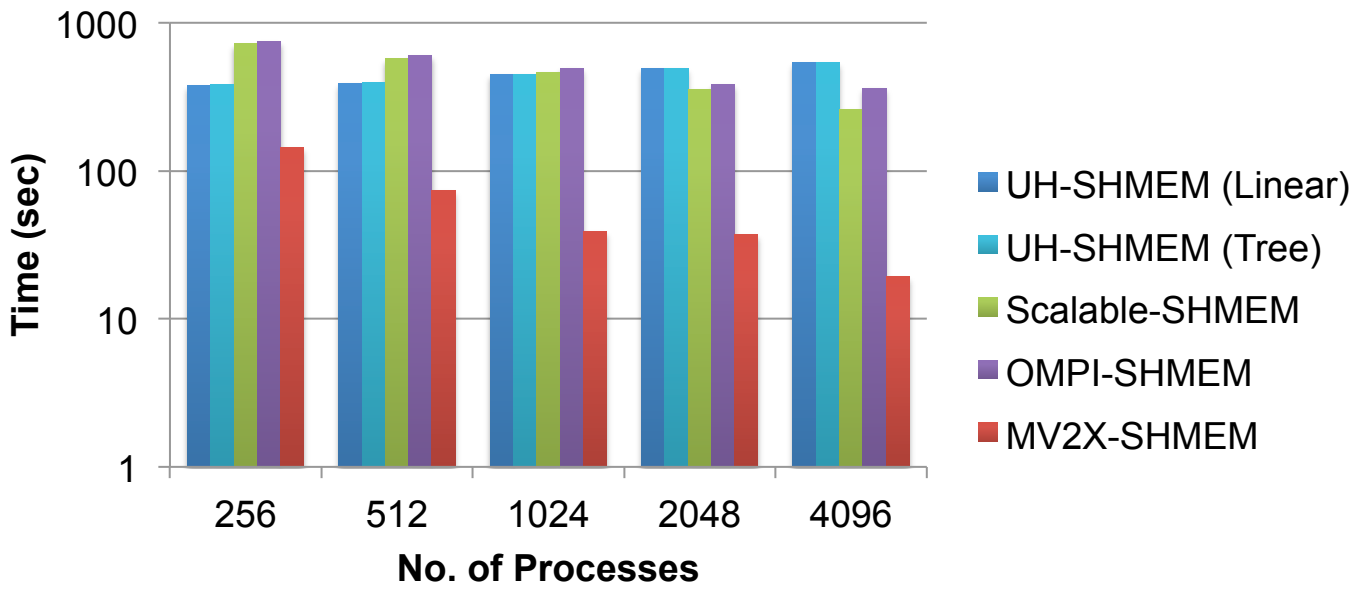


- Memory Footprint: memory consumption of a process
  - Lower memory footprint desirable for better scalability
- Evaluation using hello-world program, with shmem_barrier_all and shmem_collect calls, and shared heap size of 256M
- Memory footprint at 4,096 processes (MB): UH-SHMEM – 1646, OMPI-SHMEM – 1111, Scalable-SHMEM – 967, and MV2X-SHMEM – 344

Experimental Evaluations

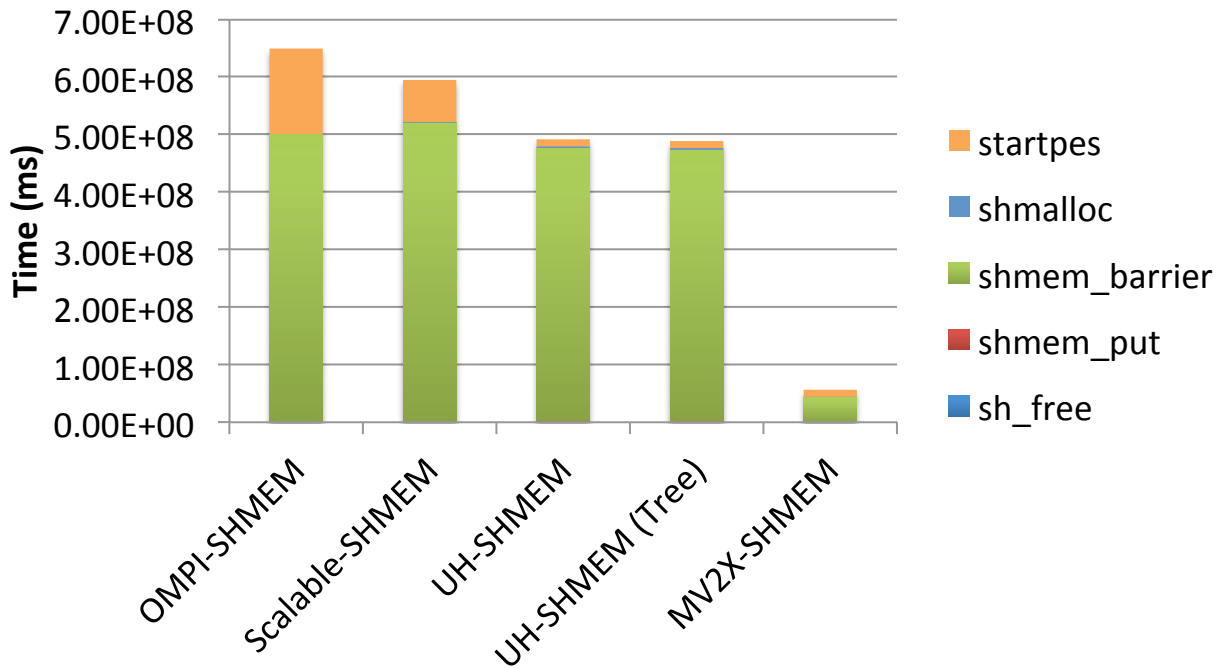


Heat Image Kernel



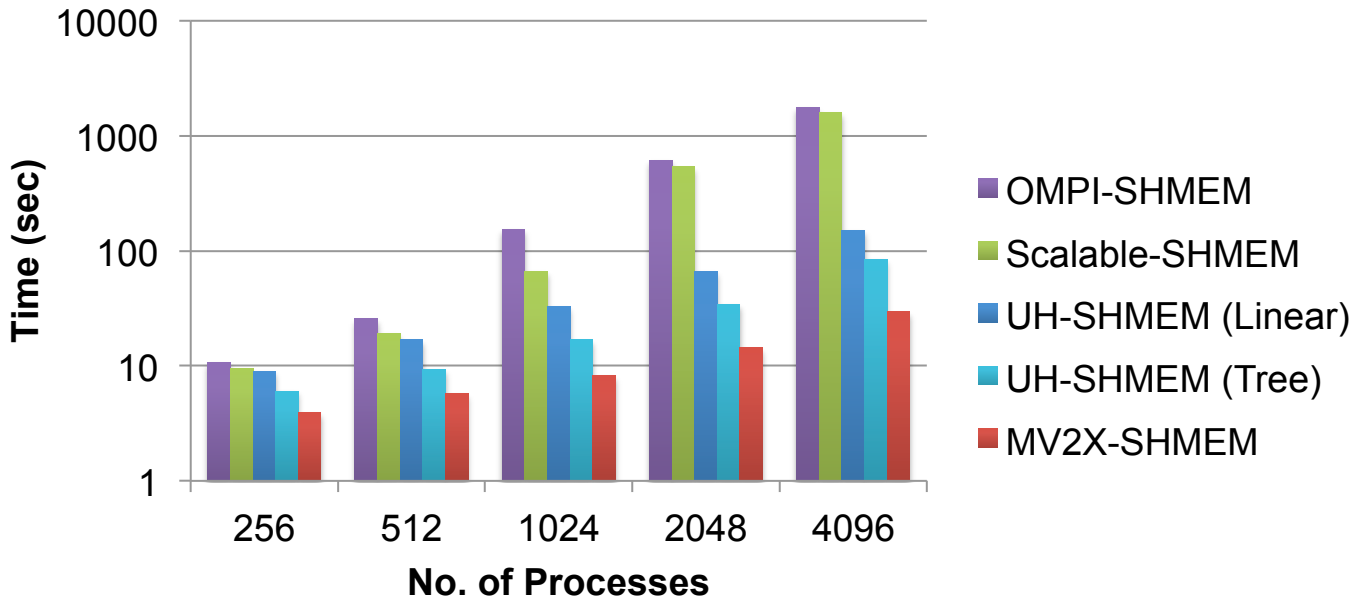
- Solves the heat conduction task based on row-based distribution
- Data transfer across the matrix rows/columns (shmem put) and synchronization operations (shmem_barrier_all)
- Execution time at 4,096 processes for input size of 32K x 32K (sec):
  - UH-SHMEM-Linear – 538, UH-SHMEM-Tree – 537, MV2X-SHMEM – 19, Scalable-SHMEM – 257, and OMPI-SHMEM – 361

Heat Image Kernel – Profiling Results



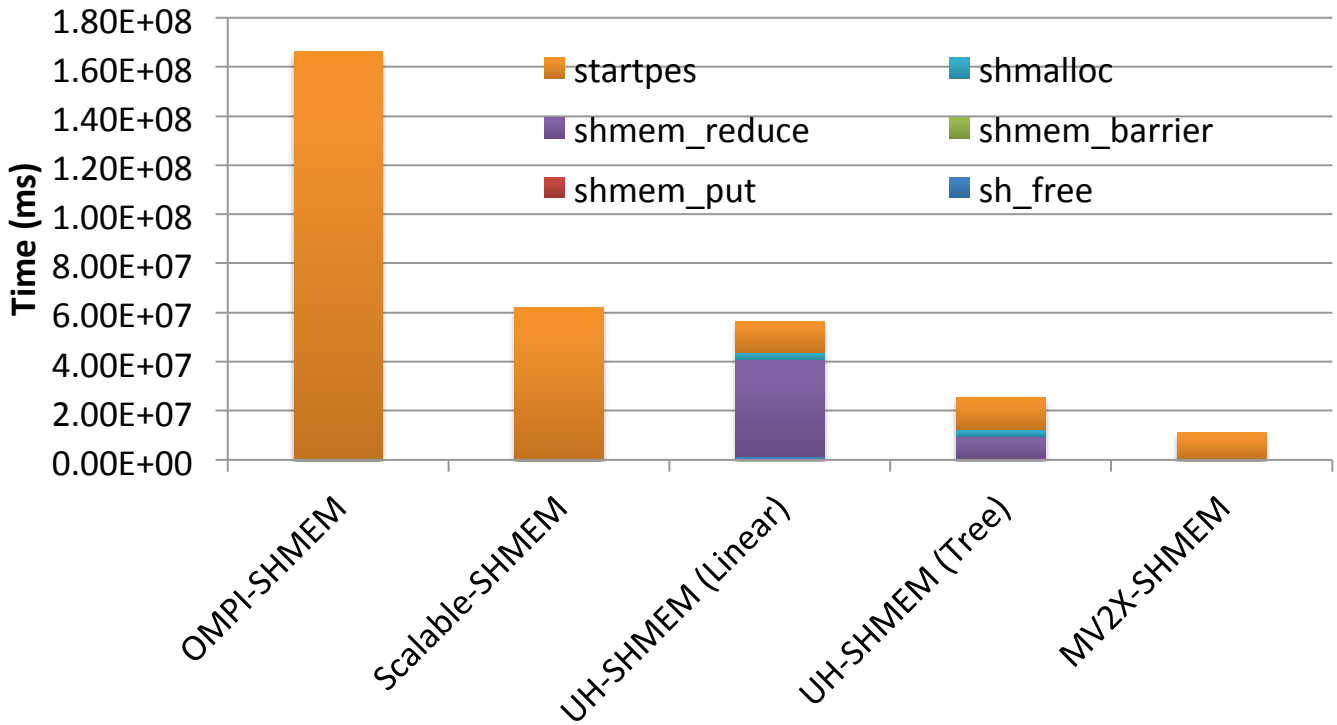
- Profiling results at 1,024 processes, using HPC-Toolkit
- Most time spent in barrier operation(s)
- Similar trends as the collective micro-benchmark results

DAXPY Kernel



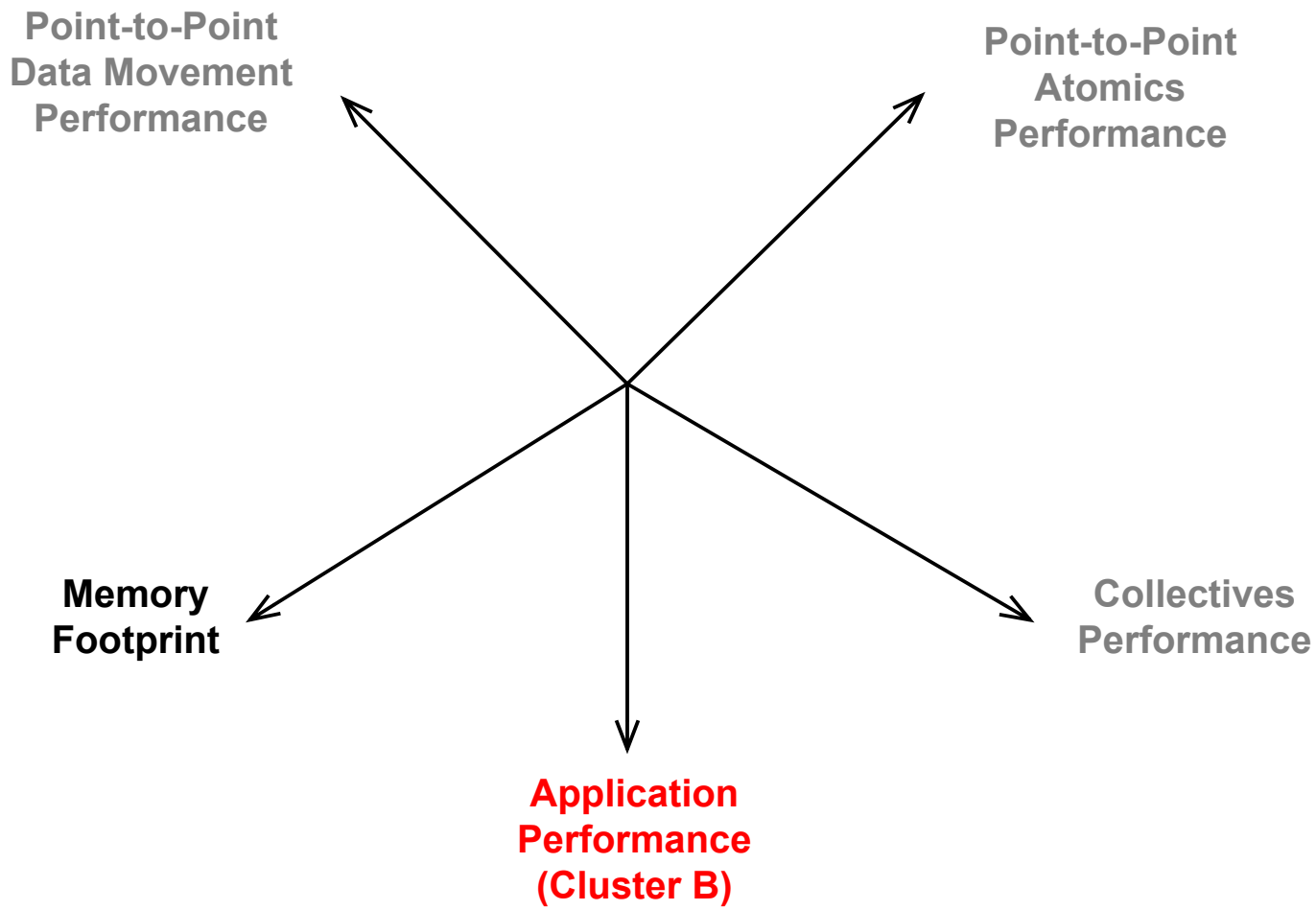
- Data transfer using shmem_put operation, synchronization using shmem_barrier_all, and reduction using shmem_reduce
- Execution time reported includes initialization time
- Execution time at 4,096 processes (sec):
  - UH-SHMEM-Linear – 151, UH-SHMEM-Tree – 83, MV2X-SHMEM – 29, Scalable-SHMEM – 1594, and OMPI-SHMEM – 1776

DAXPY Kernel – Profiling Results

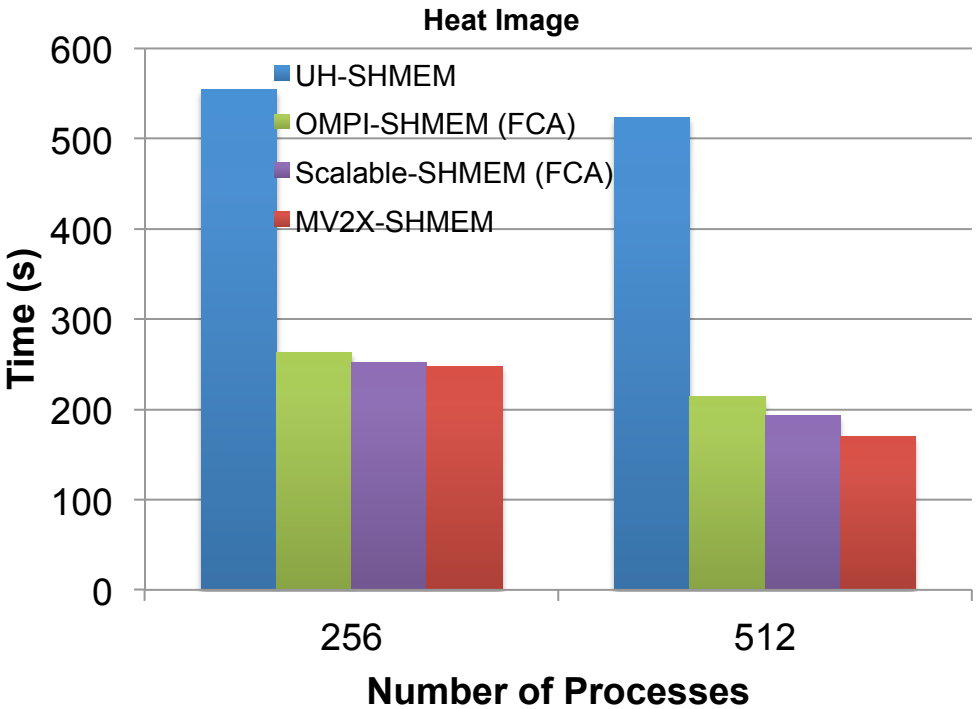


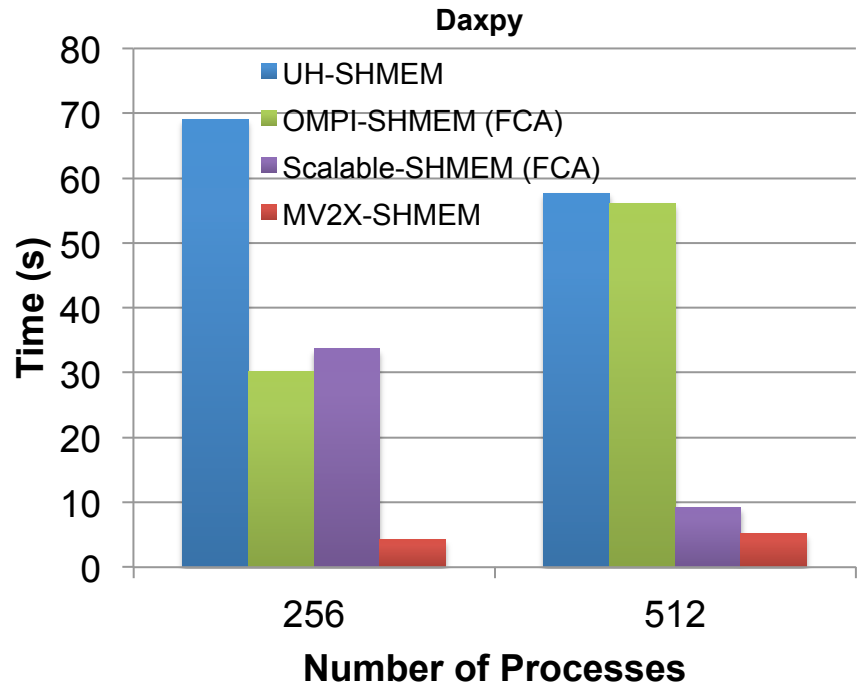
- Profiling results at 1,024 processes
- Profiling indicates most time spent in initialization
- shmem_reduce results similar to the micro-benchmark results

Experimental Evaluations



Application Results on Cluster B



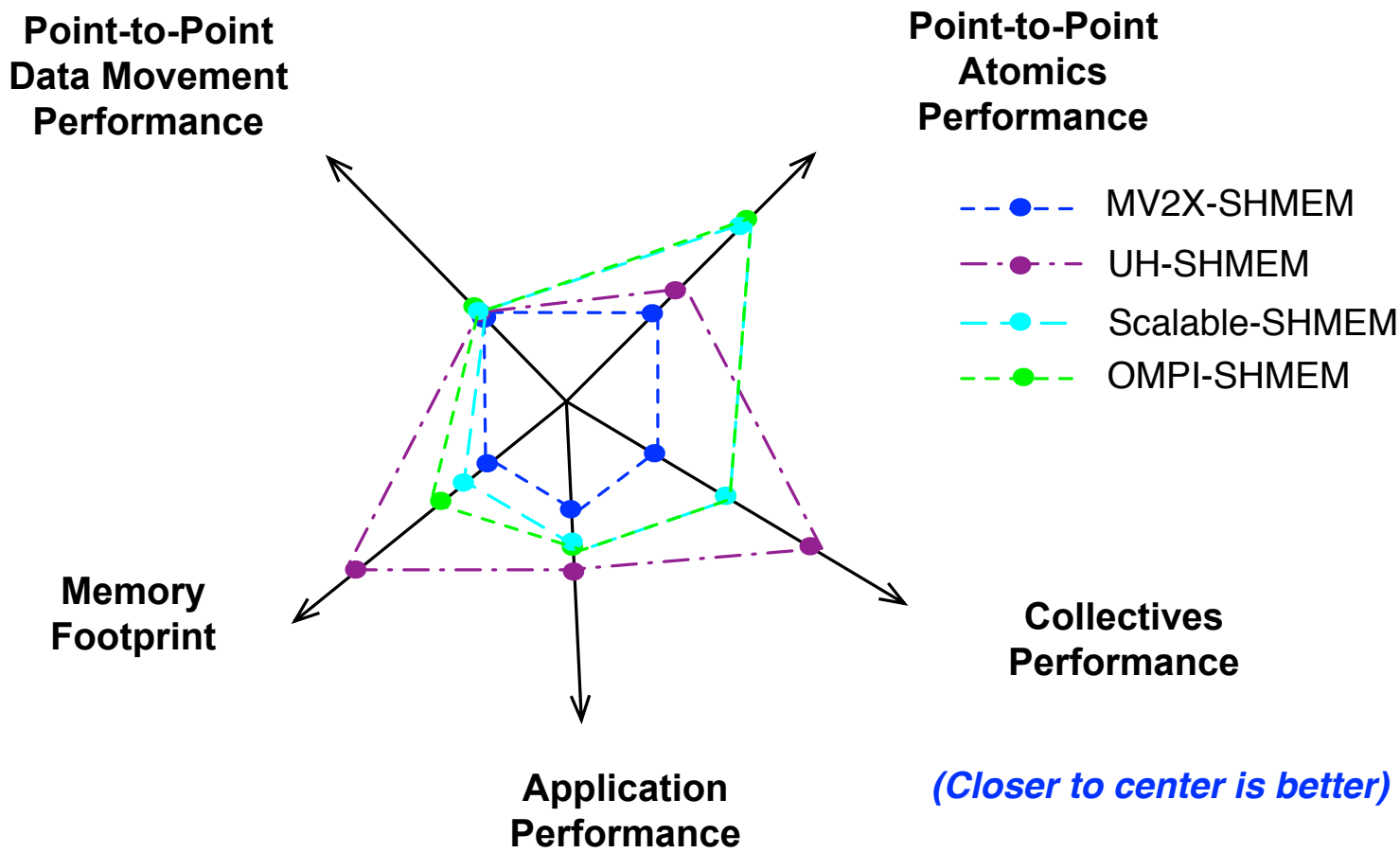


- Improved performance for OMPI-SHMEM and Scalable-SHMEM with FCA
- Execution time for 2DHeat Image at 512 processes (sec):
  - UH-SHMEM – 523, OMPI-SHMEM – 214, Scalable-SHMEM – 193, MV2X-SHMEM – 169
- Execution time for DAXPY at 512 processes (sec):
  - UH-SHMEM – 57, OMPI-SHMEM – 56, Scalable-SHMEM – 9.2, MV2X-SHMEM – 5.2

Outline

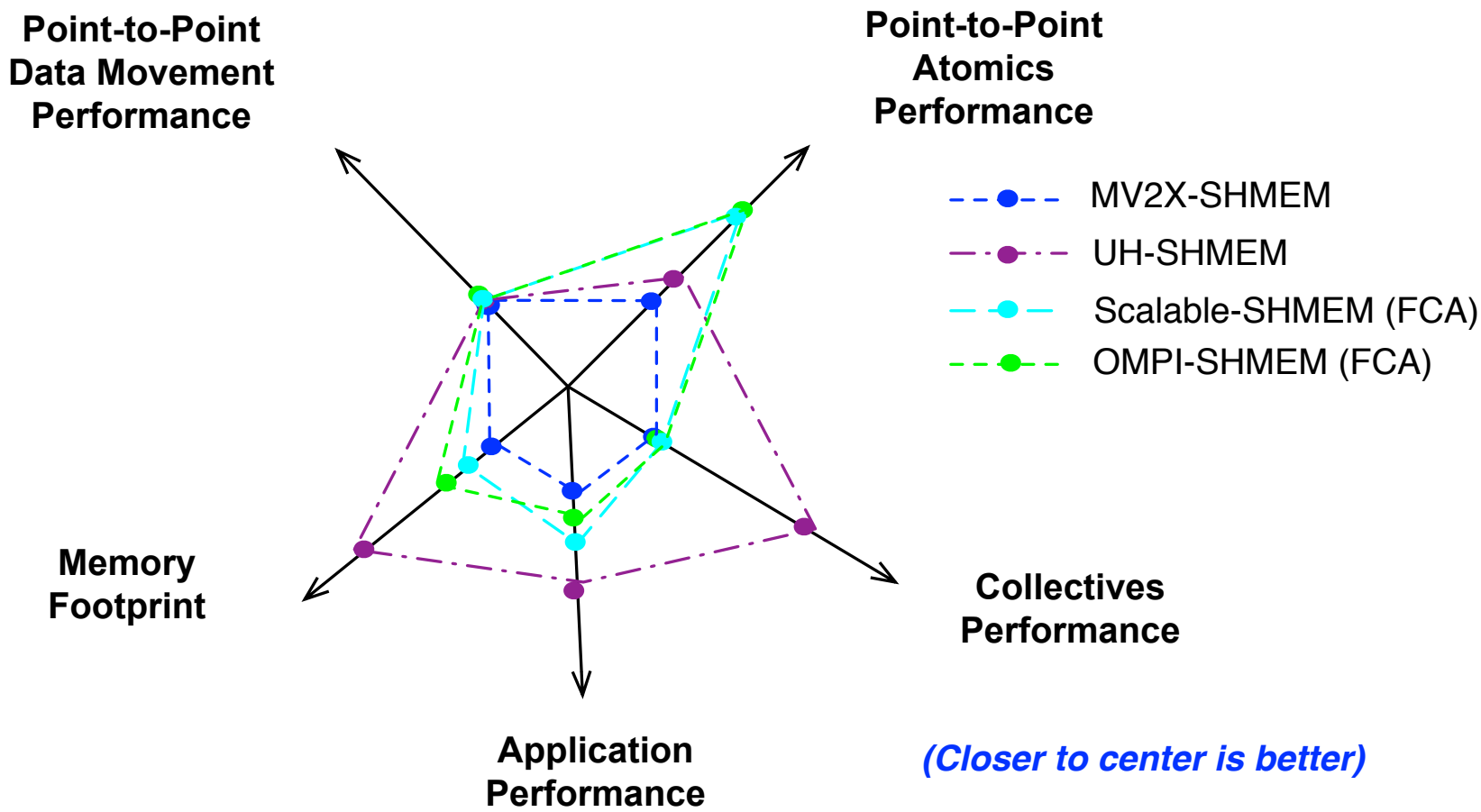
- Introduction and Motivation
- Communication Operations
- Evaluation Methodology
- Experimental Evaluations
- Overview of Results
- Conclusion

Overview of Results (Cluster A)



(Closer to center is better)

Overview of Results (Cluster B)



(Closer to center is better)

Outline

- Introduction and Motivation
- Communication Operations
- Evaluation Methodology
- Experimental Evaluations
- Overview of Results
- Conclusion

Conclusion

- Presented a detailed evaluation of different OpenSHMEM libraries over InfiniBand network
  - MV2X-SHMEM, OMPI-SHMEM, Scalable-SHMEM, UH-SHMEM
- Detailed evaluation of various communication routines
  - Point-to-Point, Atomic, and Collective Operations
- Memory Scalability Evaluations
- Application Evaluations
  - HeatImage and DAXPY Kernels
  - Analysis using split-up performance evaluation
- Correlation between micro-benchmark and application performance

Thank You!

{jose, zhanjie, akshay, potluri, panda}@cse.ohio-state.edu

Network-Based Computing Laboratory

http://nowlab.cse.ohio-state.edu/

MVAPICH Web Page

http://mvapich.cse.ohio-state.edu/