# Characterizing One-/Two-sided Designs in OpenSHMEM Collectives


UNIVERSITY OF CALIFORNIA MERCED

PADSYS Lab

**Yuke Li[1], Yanfei Guo[2], and Xiaoyi Lu[1]**

[1]University of California Merced, {yli304, xiaoyi.lu}@ucmerced.edu

[2]Argonne National Laboratory, yguo@anl.gov

Argonne NATIONAL LABORATORY


## Introduction and Motivation

- OpenSHMEM is a widely used PGAS programming model in the HPC area
- The *team* concept and team-based collective communication in the latest OpenSHMEM Specification v1.5 are similar to the communicator and collective communication in MPI

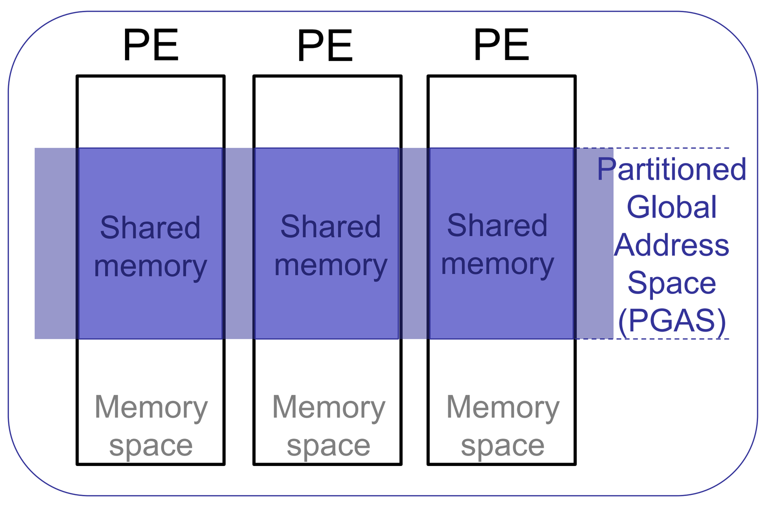


Fig. 1: An OpenSHMEM team

- The different design approaches (the native one-sided and the MPI-based two-sided communication) can lead to different performance characteristics on HPC clusters (even for the same collective routine)
- We characterize two aspects that can influence the performance
  - Synchronization methods
  - Collective algorithms
- We compare the native one-sided design and the MPI-based two-sided design in collectives using benchmarks, and show how big the performance difference between them

## Experiment Setup

- **Library Selection:**
  - ❖ One-sided design
    - Sandia OpenSHMEM (SOS) [1]
  - ❖ Two-sided design
    - OpenSHMEM over MPI (OSHMPI) [2][3]
- **Platform:**
  - ❖ Bebop HPC cluster [4]. Each node has 36-core Intel Broadwell CPU. The nodes are connected with Omni-Path Fabric
- **MPI Implementation:** MPICH [5][6]
- **Benchmark:** OSU Micro-Benchmarks [7]

## Conclusion

- The performance of OpenSHMEM varies a lot (up to 10X) with several factors
- The comparison exposes the performance characteristics of different collective communication designs in OpenSHMEM
- We believe the performance characterizations can give the community some insights for future research avenues

## Designs of OpenSHMEM Libraries

- **Sandia OpenSHMEM (SOS)**
  - ❖ Native one-sided design -- a source PE *puts* data into (or *gets* data from) the shared global memory of the target PE
  - ❖ The synchronization of the one-sided design does not require the target PE to actively acknowledge data receiving to the source PE
  - ❖ SOS collectives natively implement some algorithms – Tree, Ring, Round-Robin algorithms
- **OpenSHMEM over MPI (OSHMPI)**
  - ❖ MPI-based two-sided design -- a source PE *sends* data, which matches with a *recv* called by a target PE
  - ❖ Less flexible synchronization. It needs *send and recv* coordination between the source and the target PEs
  - ❖ OSHMPI collectives essentially call MPI collectives. Taking MPICH as an example, it provides rich collective algorithms – Tree, Ring, Bruck, Recursive Doubling, etc. The algorithm can be chosen at runtime by scheduling

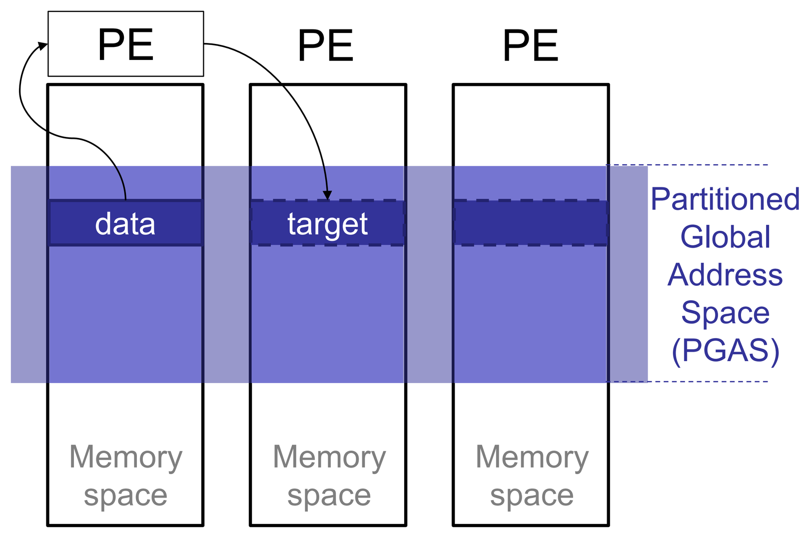


Fig. 2: One-sided communication design -- A PE *puts* data directly to target shared global memory of the other PE

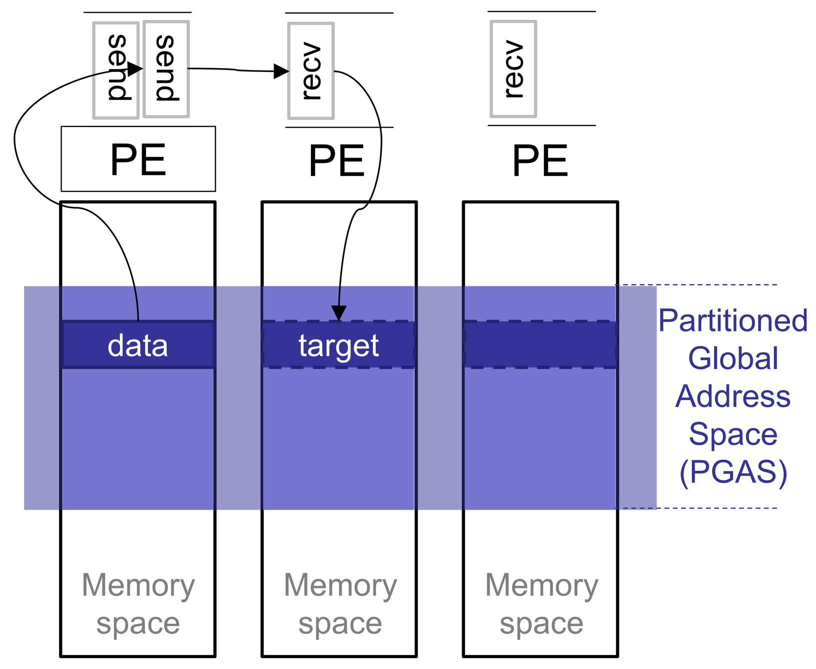


Fig. 3: Two-sided communication design -- A *send* called by source PE must match with a *recv* called by target PE

## Performance Characterization

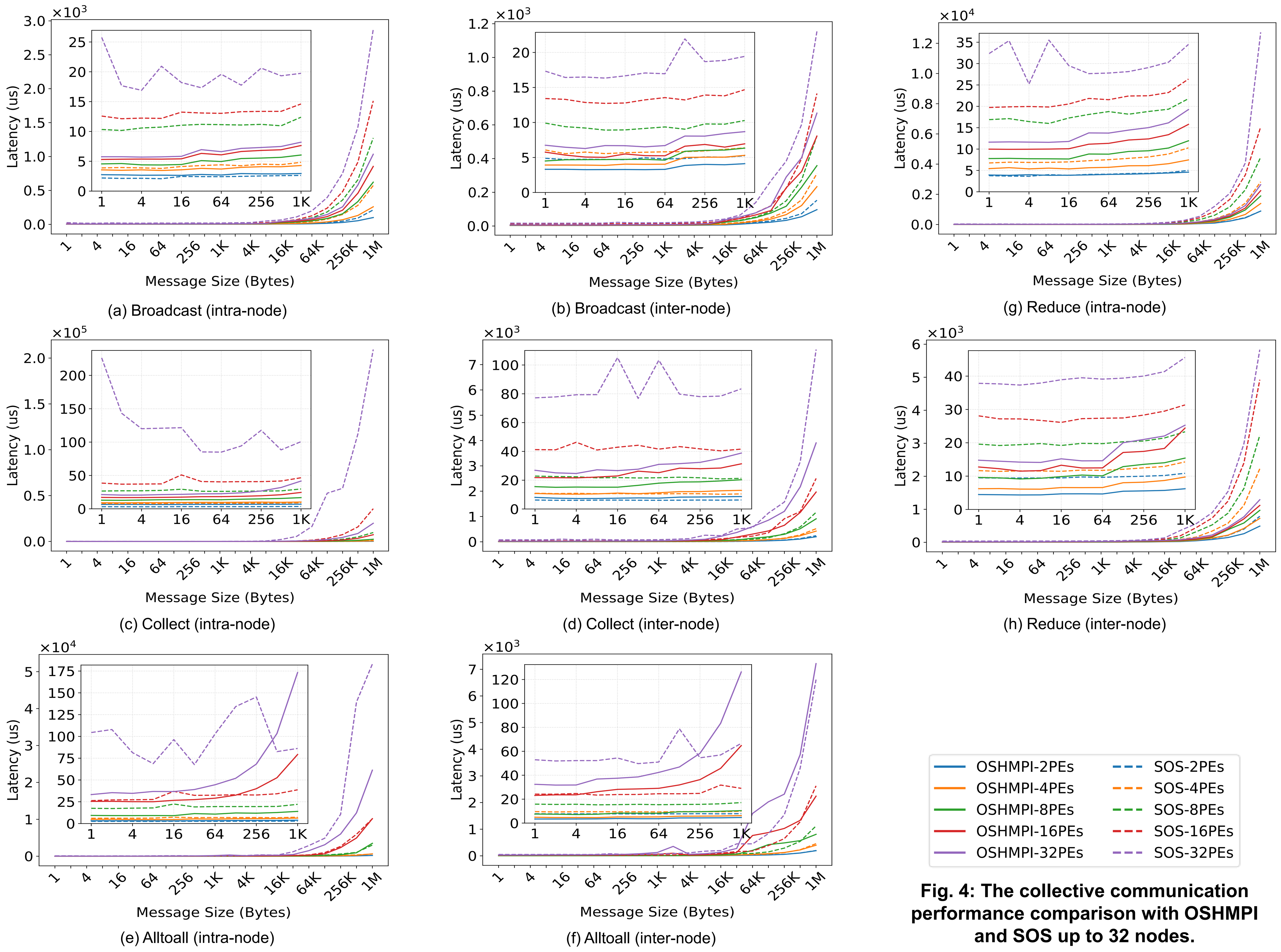


Fig. 4: The collective communication performance comparison with OSHMPI and SOS up to 32 nodes.

| Key Findings | Reason Explanations |
|---|---|
| OSHMPI (two-sided) is faster than SOS (one-sided) in most cases. | The existing two-sided design (e.g., OSHMPI) can inherit the advantages in well-optimized MPI engine, while the one-sided design (e.g., SOS) still needs optimized implementations. |
| SOS shows lower or comparable latency in certain cases:<br>- Communication with a small number of PEs like 2, for *collect*, intra-node *broadcast*, *alltoall*, and *reduce* collectives.<br>- The *alltoall* collective for medium messages in intra-node communication and medium-large messages in inter-node communication with more PEs, like 16 and 32. | With a small number of PEs, basic point-to-point primitive performance and synchronization method dominate the performance. SOS's native one-sided design can achieve better overlapping with simpler synchronizations. Besides, SOS also has some specific optimizations on Omni-Path Fabric. For the *alltoall* collective, MPI internal routine needs further optimization or tuning for these particular settings. |
| SOS shows unstable performance in the case of intra-node communication with 32 PEs. | SOS uses extra helper threads to progress the internal communication tasks and thus it is oversubscribed. |

## References


[1] S. Corporation, "Sandia-openshmem/sos." [Online]. Available: https://github.com/Sandia-OpenSHMEM/SOS

[2] M. Si, H. Fu, J. R. Hammond, and P. Balaji, "Openshmem over mpi as a performance contender: Thorough analysis and optimizations," in OpenSHMEM and Related Technologies. OpenSHMEM in the Era of Exascale and Smart Networks, S. Poole, O. Hernandez, M. Baker, and T. Curtis, Eds. Cham: Springer International Publishing, 2022, pp. 39–60.

[3] A. N. Laboratory, "Oshmpi: Openshmem implementation over mpi." [Online]. Available: https://github.com/pmodels/oshmpi

[4] "Bebop." [Online]. Available: https://www.lcrc.anl.gov/systems/resources/bebop/

[5] A. N. Laboratory, "Mpich: High-performance portable mpi." [Online]. Available: https://www.mpich.org/

[6] "Official mpich repository." [Online]. Available: https://github.com/pmodels/mpich

[7] D. K. Panda, H. Subramoni, C.-H. Chu, and M. Bayatpour, "The mvapich project: Transforming research into high-performance mpi library for hpc community," Journal of Computational Science, vol. 52, p. 101208, 2021, case Studies in Translational Computer Science. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877750320305093



**This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration, and by the U.S. Department of Energy, Office of Science, under Contract DE-AC02-06CH11357. We gratefully acknowledge the computing resources provided on Bebop (and/or Swing and/or Blues), a high-performance computing cluster operated by the Laboratory Computing Resource Center at Argonne National Laboratory. This work was supported in part by the NSF research grant OAC #2321123 and a DOE research grant DE-SC0024207.**