GASNet-EX RMA Communication Performance on Recent Supercomputing Systems

Paul H. Hargrove, Dan Bonachea
Computer Languages and Systems Software Group
Lawrence Berkeley National Laboratory
{PHHargrove, DOBonachea}@lbl.gov

***Abstract*—Partitioned Global Address Space (PGAS) programming models, typified by systems such as Unified Parallel C (UPC) and Fortran coarrays, expose one-sided Remote Memory Access (RMA) communication as a key building block for High Performance Computing (HPC) applications. Architectural trends in supercomputing make such programming models increasingly attractive, and newer, more sophisticated models such as UPC++, Legion and Chapel that rely upon similar communication paradigms are gaining popularity.**

GASNet-EX is a portable, open-source, high-performance communication library designed to efficiently support the networking requirements of PGAS runtime systems and other alternative models in emerging exascale machines. The library is an evolution of the popular GASNet communication system, building upon 20 years of lessons learned. We present microbenchmark results which demonstrate the RMA performance of GASNet-EX is competitive with MPI implementations on four recent, high-impact, production HPC systems. These results are an update relative to previously published results on older systems. The networks measured here are representative of hardware currently used in six of the top ten fastest supercomputers in the world, and all of the exascale systems on the U.S. DOE road map.

***Index Terms*—HPC, PGAS, RMA, Active Messages, Exascale Computing, Middleware**

I. Background

GASNet-EX is a language-independent, networking middleware layer that provides network-independent, high-performance communication primitives for High-Performance Computing (HPC). Unlike the dominant MPI communication standard, the GASNet-EX interface and implementation are designed specifically to meet the needs of alternative programming models on emerging exascale systems. GASNet-EX is implemented directly over the native/proprietary APIs of many networks, including all of those in use at the HPC centers of the U. S. Department of Energy's Office of Science [1]. GASNet-EX's interface is primarily intended as a compilation target and for use by runtime library writers (as opposed to domain scientists), and the primary goals are high performance, interface portability, and expressiveness. GASNet-EX provides communication services for many projects, including both programming models and other parallel libraries and frameworks. Examples of alternative HPC programming models using GASNet-EX include: UPC++ [2–4], the Legion programming system [5], HPE's Chapel language [6], the Omni Xcalable Compiler [7], and many UPC [8–10] and CAF/Fortran [11–14] compiler runtimes. GASNet-EX has also been adopted for communication services by a number of parallel libraries and frameworks, including [15, 16]. See [17] for full details on current client software, and Fig. 1 for an overview of the GASNet-EX software ecosystem.

GASNet-EX offers a variety of communication services to runtime clients, notably including: Remote Memory Access (RMA), Active Messages (AM), remote atomic memory operations and non-blocking collectives; see [18] for further details on GASNet-EX features. This paper focuses on the RMA performance of GASNet-EX, which is critical to the efficiency of many client applications. Specifically, we repeat the performance evaluation presented in our earlier paper [18] on more recent production HPC platforms that feature correspondingly newer network hardware and software stacks. As such, some descriptive portions of that paper are reproduced here with permission.

II. Experimental Systems

For our benchmarking efforts, we selected four high-impact production HPC systems at U.S. Department of Energy (DOE) computing centers. The networks in these systems are representative of hardware currently used in six of the top ten fastest supercomputers in the world, according to the June 2022 Top500 [19] list (current at the time of writing). Notably, the number one spot in that list is held by Frontier [20] with the same Slingshot-11 network that we evaluate on Perlmutter (see below), which will also appear in DOE's other two announced exascale systems: Aurora [21] and El Capitan [22].

Our measurements attempt to reproduce the experience of a non-expert end-user. Therefore, all systems were used as configured by the respective HPC centers, using default

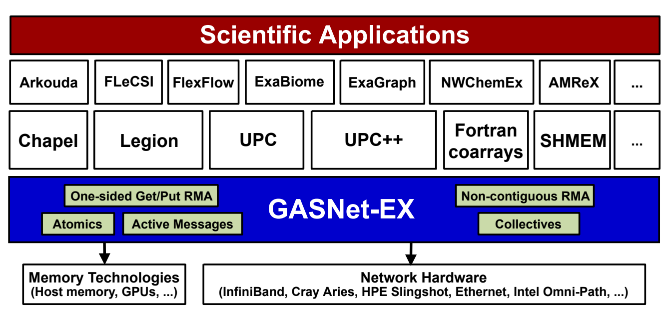

Fig. 1. GASNet-EX software ecosystem

 doi:10.25344/S40C7D

versions of installed environment modules for all software except for GASNet-EX and the microbenchmark codes.

Summit: The "Summit" [23] system at OLCF [24] consists of IBM AC922 nodes, each with two 22-core POWER9 CPUs and connected to a 100Gb/s EDR InfiniBand network by two Mellanox "ConnectX-5" HCAs, each with affinity to a single socket.

Cori Haswell: The Cray XC40 system at NERSC [25] known as "Cori" [26] offers nodes with two 16-core Intel Xeon E5-2698v3 "Haswell" processors and a Cray Aries [27, 28] network.

Perlmutter SS-10 / SS-11: The last two systems evaluated are partitions of the HPE Cray EX system at NERSC [25] known as "Perlmutter" [29]. Nodes in both partitions contain a single 64-core AMD EPYC 7763 "Milan" CPU and are connected to a 200Gb/s HPE Slingshot network, but they differ in the NICs which attach them to the network. The Slingshot-10 ("SS-10") nodes each hold two Mellanox "ConnectX-5" NICs operating at 100Gb/s, while the Slingshot-11 ("SS-11") nodes each hold four HPE-proprietary "Cassini" NICs operating at 200Gb/s[1].

As with all HPC-relevant network hardware, the systems evaluated provide Remote Direct Memory Access (RDMA) communication support; this allows GASNet-EX to offload RMA communication work to the network hardware on both sides, while minimizing software overheads associated with payload copying or CPU-driven communication. GASNet-EX's lightweight RMA operations are deliberately designed to streamline this mapping, avoiding semantics encumbrances that could incur impedance mismatches or other unwanted overheads.

When building software (including GASNet-EX and all microbenchmarks) we followed the instructions without the application of any expert knowledge. No configuration settings, environment variables, or similar means were used to tune the benchmark performance of GASNet-EX or MPI[2]. Each of the HPC centers provide multiple compiler family options in their production software environment. We selected each system's default version of the GNU compilers, even where a different compiler family was the default, because GNU is universally available and enjoys widespread compatibility with library clients.

We benchmarked GASNet-EX version `2022.3.0` using two tests selected from those provided with the source code distribution. For MPI benchmarking we measured the vendor-supplied default MPI implementation on each system, using the publicly available Intel MPI Benchmarks (IMB) [30] version `v2021.3` (the latest official benchmark release at the time of writing). For detailed experimental methodology, see the Artifact Description (AD) Appendix.

[1]Benchmarks on Perlmutter utilize only a single NIC per process, since neither GASNet-EX nor HPE Cray MPI currently support more.

[2]with one exception: On Summit we set environment variables to restrict both GASNet-EX and the MPI implementation to a single rail per process on the dual-rail network, to ensure a meaningful comparison. We recommend this configuration because it can yield significant latency improvements.

III. RMA Flood Bandwidth Benchmark Results

A "flood bandwidth" benchmark measures achievable bandwidth at a given transfer size by initiating a large number of non-blocking transfers and waiting for them all to fully complete. The reported metric is the total volume of data transferred, divided by the total elapsed time. We report uni-directional (one initiator to one target) flood bandwidths, where the passive target waits in an appropriate synchronization operation.

For GASNet-EX we used the `testlarge` microbenchmark to measure performance of the `gex_RMA_PutNBI` and `gex_RMA_GetNBI` functions, synchronized with a final `gex_NBI_Wait`. We measured flood bandwidth of the `MPI_Put` and `MPI_Get` functions using the "Aggregate" timings from, respectively, the `Unidir_put` and `Unidir_get` tests from the `IMB-RMA` suite (these tests measure the time to issue RMA and synchronize using `MPI_Win_flush`, within a passive-target access epoch established by a `MPI_Win_lock(SHARED)` call outside the timed region – see [30] for further details). The `testlarge` benchmark reports bandwidths in units of "MiB/s" (2^{20} bytes per second), whereas the `IMB` tests use "MB/s" (10^6 bytes per second). Both have been converted to "GiB/s" (2^{30} bytes per second) for the plots which follow. Although RMA and message passing are semantically different, for comparison purposes the plots also report uni-directional bandwidth of `MPI_Isend`/`MPI_Irecv`, from the "Aggregate" timings of the `Uniband` test from the `IMB-MPI1` suite.

All tests ran between two compute nodes, using a single process and single NIC on each. Data was collected from 16 distinct batch jobs, each running one instance of each GASNet-EX and MPI test back-to-back. Each data point plotted reports the maximum achieved bandwidth for that benchmark and transfer size. For RMA and message passing tests we used $10,000$ and 500 iterations, respectively.

In Fig. 2, "$\times$" markers denote GASNet-EX RMA, "$\circ$" markers denote MPI RMA, and "$+$" markers denote MPI message passing. RMA Put results are distinguished by the use of solid lines (in shades of blue), while RMA Get results use dot-dashed lines (in shades of red). Dashed lines (in green) are message-passing results. The horizontal axis (transfer size) is logarithmic, while the vertical axis (bandwidth) is linear.

We find the uni-directional flood bandwidth of GASNet-EX RMA operations is uniformly comparable to or better than the corresponding MPI RMA operations. GASNet-EX results are seen to rise more rapidly to the maximum bandwidth, exceeding 90% of saturation bandwidth at transfer sizes as small as 4KiB to 8KiB, for both Put and Get across all four systems. While both GASNet-EX and MPI RMA operations eventually reach each system's asymptotic saturation bandwidth, their approach to the maximum differs. On Summit and Cori Haswell, GASNet-EX reaches the maximum for significantly smaller transfers than for MPI; in other words, one can achieve the maximum bandwidth with a wider range of transfer sizes using GASNet-EX than using MPI. On

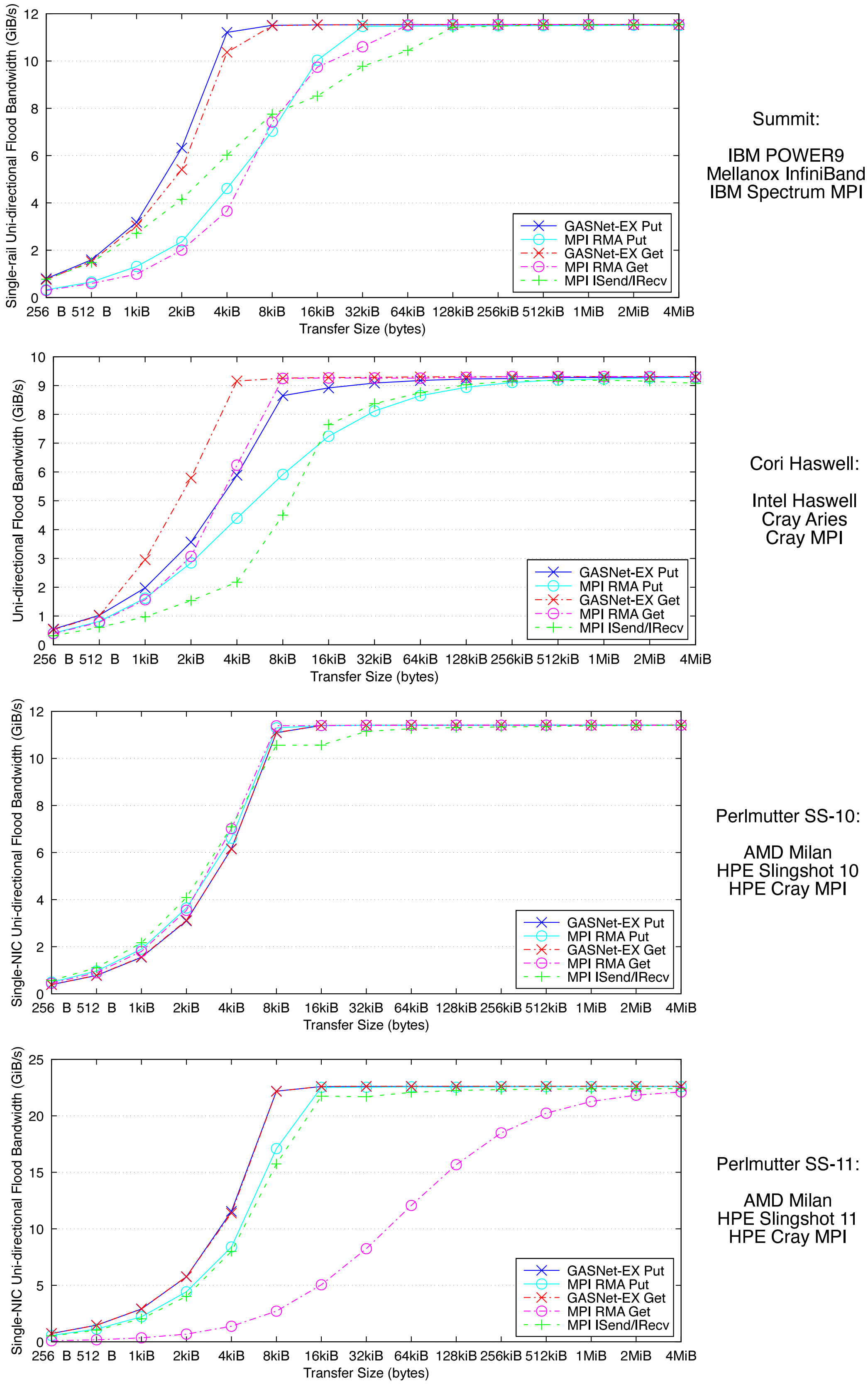

Fig. 2. Uni-directional flood bandwidth versus transfer size

TABLE 1
ROUND-TRIP LATENCY OF 8-BYTE RMA ACCESSES

System	8-Byte RMA Put Latency GASNet-EX	MPI RMA	Ratio	8-Byte RMA Get Latency GASNet-EX	MPI RMA	Ratio
Summit	1.47 μs	3.50 μs	2.38	1.78 μs	3.49 μs	1.96
Cori Haswell	1.07 μs	1.45 μs	1.36	1.44 μs	1.83 μs	1.27
Perlmutter SS-10	3.14 μs	6.48 μs	2.06	3.58 μs	6.98 μs	1.95
Perlmutter SS-11	2.72 μs	2.49 μs	0.92	2.79 μs	6.17 μs	2.21

Perlmutter SS-10, the RMA performance of the two libraries is nearly indistinguishable. In the case of Perlmutter's higher-bandwidth SS-11 NICs, GASNet-EX Puts reach the maximum at about half the transfer size required for MPI. For RMA Gets on SS-11, MPI's rise to the maximum bandwidth is anomalously slow relative to all other bandwidth data[3].

On all systems GASNet-EX RMA flood bandwidth performance is at least comparable to MPI message passing, with a significant advantage across a wide range of transfers on three of the four systems.

IV. RMA LATENCY BENCHMARK RESULTS

We next report on the round-trip latency of GASNet-EX and MPI RMA operations. These benchmarks report the mean time to fully complete a single RMA Put or Get operation, computed by timing a long sequence of blocking operations. For GASNet-EX we measured the `gex_RMA_PutBlocking` and `gex_RMA_GetBlocking` functions using the `testsmall` microbenchmark. For MPI benchmarking we report the "Non-aggregate" timings from the `Unidir_put` and `Unidir_get` tests from the `IMB-RMA` suite, which are semantically equivalent to the GASNet-EX test. These are the same tests used to measure MPI RMA flood bandwidth, but differ by executing a sequence of `MPI_Put` (or `MPI_Get`) calls alternating with calls to `MPI_Win_flush`, whereas the "Aggregate" timings used for bandwidth have only a *single* `MPI_Win_flush` at the end.

Data was collected from the same 16 batch jobs described for the flood bandwidth benchmark. Our latency results are summarized in Tbl. 1, which reports the minimum across benchmark runs for the average latency achieved using blocking RMA Put and Get with 8-byte payloads. Each row includes the ratio of the corresponding GASNet-EX and MPI results. This ratio is also representative of timings over power-of-two sizes from 4 bytes to 1024 bytes (not shown) for all four systems. However, as the transfer size grows the variable cost of data movement grows relative to other fixed costs of the communication, and the ratios therefore trend slowly nearer to 1.0. In nearly all cases measured, GASNet-EX demonstrated comparable or better small RMA latency relative to MPI, sometimes by more than a factor of two. The sole exception is on the SS-11 platform, where MPI's 8-byte Puts have roughly 8% lower latency than those of GASNet-EX.

[3]Diagnosing performance anomalies in HPE's implementation of MPI RMA is outside the scope of this paper, but the authors acknowledge the possibility that expert tuning of the sort we've expressly avoided might address this behavior.

V. CONCLUSION AND FUTURE WORK

We presented updated microbenchmark results demonstrating the RMA performance of GASNet-EX is competitive with several vendor MPI implementations on modern production HPC systems whose networks are representative of emerging exascale systems. We found that GASNet-EX RMA bandwidth outperformed the equivalent MPI RMA operations by up to 2.7x for Puts and up to 3.1x for Gets at certain transfer sizes, reaching saturation bandwidth at up to 8x smaller transfer sizes[4]. For small-transfer latency, GASNet-EX RMA outperformed MPI RMA by up to 2.38x.

The Slingshot results on Perlmutter utilize GASNet-EX's most recent ofi-conduit backend, which is currently labeled as "experimental" because it still remains largely untuned. On HPE Slingshot both ofi-conduit and Cray MPI communicate with the NIC via the underlying OFI libfabric layer, which also continues to evolve in both stability and performance. As such, we expect results from both libraries on Slingshot (especially for the SS-11 Cassini NIC) will continue to improve as both software stacks continue to mature. Future work includes tuning the performance of ofi-conduit and specializing its implementation to more effectively expose hardware capabilities.

GASNet-EX has recently extended its RMA interfaces to enable RMA for memory located on accelerator devices, such as the GPUs made by NVIDIA and AMD, which have become very popular in HPC systems and will provide the majority of computational power on exascale platforms [31]. Current and future work includes expanding this feature to support additional varieties of accelerators and leverage hardware offload capabilities on additional network fabrics.

ACKNOWLEDGMENTS

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.

This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

[4]The anomalously poor outlier behavior of MPI RMA Get on Perlmutter SS-11 has been excluded from this summary.

References

[1] *DOE Advanced Scientific Computing Research (ASCR) Facilities*, https://science.energy.gov/ascr/facilities.

[2] J. Bachan, S. B. Baden, S. Hofmeyr, M. Jacquelin, A. Kamil, D. Bonachea, P. H. Hargrove, and H. Ahmed, "UPC++: A High-Performance Communication Framework for Asynchronous Computation," in *Proceedings of the International Parallel & Distributed Processing Symposium (IPDPS)*, 2019, doi:10.25344/S4V88H.

[3] D. Bonachea and A. Kamil, "UPC++ v1.0 Specification, Revision 2022.3.0," Lawrence Berkeley National Laboratory, Tech. Rep. LBNL-2001452, March 2022, doi:10.25344/S4530J.

[4] J. Bachan, S. B. Baden, D. Bonachea, M. Grossman, P. H. Hargrove, S. Hofmeyr, M. Jacquelin, A. Kamil, B. van Straalen, and D. Waters, "UPC++ v1.0 Programmer's Guide, Revision 2022.3.0," Lawrence Berkeley National Laboratory, Tech. Rep. LBNL-2001453, March 2022, doi:10.25344/S41C7Q.

[5] M. Bauer, S. Treichler, E. Slaughter, and A. Aiken, "Legion: expressing locality and independence with logical regions," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'12)*, 2012, doi:10.1109/SC.2012.71.

[6] B. L. Chamberlain, D. Callahan, and H. P. Zima, "Parallel programmability and the Chapel language," in *International Journal of High Performance Computing Applications (IJHPCA)*, vol. 21, no. 3, August 2007, pp. 291–312, doi:10.1177/1094342007078442.

[7] H. Murai, M. Nakao, H. Iwashita, and M. Sato, "Preliminary Performance Evaluation of Coarray-based Implementation of Fiber Miniapp Suite Using XcalableMP PGAS Language," in *Proceedings of the Second Annual PGAS Applications Workshop (PAW'17)*, 2017, doi:10.1145/3144779.3144780.

[8] W. Chen, D. Bonachea, J. Duell, P. Husband, C. Iancu, and K. Yelick, "A Performance Analysis of the Berkeley UPC Compiler," in *Proceedings of the 17th International Conference on Supercomputing (ICS)*, June 2003, doi:10.1145/782814.782825.

[9] *GCC/UPC Compiler*, Intrepid Technology, Inc., https://github.com/Intrepid/GUPC.

[10] *Clang UPC Compiler*, http://clangupc.github.io.

[11] Y. Dotsenko, C. Coarfa, and J. Mellor-Crummey, "A Multi-platform Co-Array Fortran Compiler," in *Parallel Architecture and Compilation Techniques (PACT)*, 2004, doi:10.1109/PACT.2004.1342539.

[12] D. Eachempati, H. J. Jun, and B. Chapman, "An Open-source Compiler and Runtime Implementation for Coarray Fortran," in *Proceedings of the Fourth Conference on Partitioned Global Address Space Programming Models (PGAS'10)*, 2010, doi:10.1145/2020373.2020386.

[13] A. Fanfarillo, T. Burnus, V. Cardellini, S. Filippone, D. Nagle, and D. Rouson, "OpenCoarrays: Open-source Transport Layers Supporting Coarray Fortran Compilers," in *Partitioned Global Address Space Programming Models (PGAS)*, 2014, doi:10.1145/2676870.2676876.

[14] D. Rouson and D. Bonachea, "Caffeine: CoArray Fortran Framework of Efficient Interfaces to Network Environments," in *Proceedings of the Eighth Annual Workshop on the LLVM Compiler Infrastructure in HPC (LLVM-HPC2022)*, November 2022, doi:10.25344/S4459B.

[15] B. Brock, A. Buluç, and K. A. Yelick, "BCL: A cross-platform distributed container library," *Proceedings of the 48th International Conference on Parallel Processing (ICPP)*, 2019, doi:10.1145/3337821.3337912.

[16] C. Chan, B. Wang, J. Bachan, and J. Macfarlane, "Mobiliti: Scalable Transportation Simulation Using High-Performance Parallel Computing," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, Nov 2018, doi:10.1109/ITSC.2018.8569397.

[17] *GASNet*, http://gasnet.lbl.gov.

[18] D. Bonachea and P. H. Hargrove, "GASNet-EX: A High-Performance, Portable Communication Library for Exascale," in *Proceedings of Languages and Compilers for Parallel Computing (LCPC'18)*, ser. LNCS, vol. 11882. Springer, October 2018, doi:10.25344/S4QP4W.

[19] TOP500.org, "June 2022," https://www.top500.org/lists/top500/2022/06/.

[20] OLCF, "Frontier," https://www.olcf.ornl.gov/frontier/.

[21] ALCF, "Aurora," https://www.alcf.anl.gov/aurora.

[22] LLNL, "LLNL and HPE to partner with AMD on El Capitan, projected as world's fastest supercomputer," doi:10.25344/S4WK5S.

[23] OLCF, "Summit," https://olcf.ornl.gov/olcf-resources/compute-systems/summit/.

[24] *Oak Ridge National Laboratory Leadership Computing Facility (ORNL/OLCF)*, https://www.olcf.ornl.gov.

[25] *National Energy Research Scientific Computing Center (NERSC)*, https://www.nersc.gov.

[26] NERSC, "Cori Haswell Nodes," doi:10.25344/S4859K.

[27] B. Alverson, E. Froese, L. Kaplan, and D. Roweth, "Cray XC Series Network," Cray Inc., White Paper WP-Aries01-1112, November 2012, doi:10.25344/S4RW2H.

[28] P. H. Hargrove and D. Bonachea, "GASNet-EX performance improvements due to specialization for the Cray Aries network," in *2018 IEEE/ACM Parallel Applications Workshop, Alternatives To MPI (PAW-ATM)*, November 2018, pp. 23–33, doi:10.25344/S44S38.

[29] NERSC, "Perlmutter System Phase 1+2," August 2022, doi:10.25344/S4GC7R.

[30] Intel Corporation, "Introducing Intel® MPI Benchmarks," https://software.intel.com/en-us/articles/intel-mpi-benchmarks.

[31] P. H. Hargrove and D. Bonachea, "GASNet-EX Memory Kinds: Support for Device Memory in PGAS Programming Models," in *International Conference for High Performance Computing, Networking, Storage, and Analysis (SC21)*, November 2021, doi:10.25344/S4P306.

[32] *SC22 Reproducibility Initiative*, https://sc22.supercomputing.org/submit/reproducibility-initiative/.

Artifact Description (AD) Appendix

This appendix describes the methodology used for all experiments whose results are presented in this paper. This is in accordance with the SC22 Reproducibility Initiative [32].

The intent is to measure the peak communication performance of the GASNet-EX and MPI middleware libraries running natively on the network hardware of production supercomputing systems. As such, all experiments were run natively on compute-node hardware as standard user-mode processes, without any containerization or virtual-machine technology.

A. Software used to perform benchmarking

- GASNet `v2022.3.0` source code:
  https://gasnet.lbl.gov/EX/GASNet-2022.3.0.tar.gz
  - No external data required to initialize
  - Benchmarks are in the `tests` directory, written in C with direct calls to GASNet-EX
- Intel MPI Benchmarks `v2021.3` source code:
  https://github.com/intel/mpi-benchmarks/releases/tag/IMB-v2021.3
  - No external data required to initialize
  - Benchmarks are in the `src_cpp` directory, written in C++ with direct calls to MPI

B. Benchmark commands

The following command lines were used to launch the various benchmarks, where `[RUN]` is a placeholder for "`jsrun -p 2 -r 1`" on Summit, and for "`srun -c8 --cpu_bind=cores -n2 -N2`" on the other systems. In all cases, these commands ran within an exclusive batch allocation of two compute nodes. All tests ran between two compute nodes, using a single process and single NIC on each.

- Bandwidth tests:
  - `[RUN] testlarge -m -in 10000 4194304 B`
  - `[RUN] IMB-RMA -time 600 -iter_policy off -iter 10000 -msglog 4:22 Unidir_put`
  - `[RUN] IMB-RMA -time 600 -iter_policy off -iter 10000 -msglog 4:22 Unidir_get`
  - `[RUN] IMB-MPI1 -time 600 -iter_policy off -iter 500 -msglog 4:22 Uniband`
- Latency tests:
  - `[RUN] testsmall -m -in 1000000 4096 A`
  - `[RUN] IMB-RMA -time 600 -iter_policy off -iter 1000000 -msglog 2:12 Unidir_put`
  - `[RUN] IMB-RMA -time 600 -iter_policy off -iter 1000000 -msglog 2:12 Unidir_get`
  - `[RUN] IMB-MPI1 -time 600 -iter_policy off -iter 1000000 -msglog 2:12 PingPong`

Data was collected from 16 distinct batch jobs, each running one instance of each GASNet-EX and MPI test back-to-back. Each data point plotted reports the maximum average bandwidth (or minimum average latency) for that benchmark and transfer size.

The `testlarge` benchmark reports bandwidths in units of "MiB/s" (2^{20} bytes per second), whereas the `IMB` tests use "MB/s" (10^6 bytes per second). Both have been converted to "GiB/s" (2^{30} bytes per second) for the bandwidth plots.

C. Systems and their software environments

1) ***OLCF Summit***:

- Hardware in each IBM Power System AC922 node:
  - Dual-socket 22-core 3.07GHz IBM POWER9 CPUs
  - Dual-rail Mellanox EDR InfiniBand with "ConnectX-5 Ex" HCAs
  - 512 GB DDR4-2666 system memory
  - 6x NVIDIA Volta V100 GPUs (not used)
- Software environment used:
  - Red Hat Enterprise Linux release 8.2
  - Compute node kernel `4.18.0-193.46.1.el8_2.ppc64le`
  - GASNet-EX `v2022.3.0`, `ibv-conduit`
  - Relevant environment modules: (provided by the HPC center)
    - `gcc/9.1.0`
    - `spectrum-mpi/10.4.0.3-20210112`

*2) **Cori Haswell**:*

- Hardware in each Cray XC40 node:
  - Dual-socket 16-core 2.3GHz Intel Xeon E5-2698v3 “Haswell” CPUs
  - Cray Aries network with Dragonfly topology [27]
  - 128 GB DDR4-2133 memory
- Software environment used:
  - SUSE Linux Enterprise Server 15 SP2
  - Compute node kernel `5.3.18-24.46_6.0.29-cray_ari_c`
  - GASNet-EX `v2022.3.0`, `aries-conduit`
  - Relevant environment modules: (provided by the HPC center)
    * `PrgEnv-gnu/6.0.10`
    * `gcc/11.2.0`
    * `craype/2.7.10`
    * `cray-mpich/7.7.19`
    * `gni-headers/5.0.12.0-7.0.3.1_3.12__gd0d73fe.ari`
    * `craype-network-aries`
    * `craype-haswell`

*3) **Perlmutter SS-10**:*

- Hardware in each HPE Cray EX node:
  - Single-socket 64-core 2.45GHz AMD EPYC 7763 “Milan” CPU
  - 2x Mellanox “ConnectX-5” HCAs (100Gb/s each) connected via HPE Slingshot network
  - 256 GB DDR4-3200 system memory
  - 4x NVIDIA Ampere A100 GPUs (not used)
- Software environment used:
  - SUSE Linux Enterprise Server 15 SP3
  - Compute node kernel `5.3.18-150300.59.43_11.0.51-cray_shasta_c`
  - GASNet-EX `v2022.3.0`, `ofi-conduit`, with `verbs;ofi_rxm` (InfiniBand) libfabric provider
  - Relevant environment modules: (provided by the HPC center)
    * `PrgEnv-gnu/8.3.3`
    * `gcc/11.2.0`
    * `craype/2.7.16`
    * `cray-mpich/8.1.17`
    * `cray-pmi/6.1.3`
    * `libfabric/1.11.0.4.124`
    * `craype-network-ofi`
    * `craype-x86-milan`

*4) **Perlmutter SS-11**:*

- Hardware in each HPE Cray EX node:
  - Single-socket 64-core 2.45GHz AMD EPYC 7763 “Milan” CPU
  - 4x HPE “Cassini” NICs (200Gb/s each) connected via HPE Slingshot network
  - 256 GB DDR4-3200 system memory
  - 4x NVIDIA Ampere A100 GPUs (not used)
- Software environment used:
  - SUSE Linux Enterprise Server 15 SP3
  - Compute node kernel `5.3.18-150300.59.43_11.0.51-cray_shasta_c`
  - GASNet-EX `v2022.3.0`, `ofi-conduit`, with `cxi` (HPE Cassini) libfabric provider
  - Relevant environment modules: (provided by the HPC center)
    * `PrgEnv-gnu/8.3.3`
    * `gcc/11.2.0`
    * `craype/2.7.16`
    * `cray-mpich/8.1.17`
    * `cray-pmi/6.1.3`
    * `libfabric/1.15.0.0`
    * `craype-network-ofi`
    * `craype-x86-milan`