GASNet-EX Memory Kinds: Support for Device Memory in PGAS Programming Models (Extended Poster Abstract)

Paul H. Hargrove, Dan Bonachea, Colin A. MacLean, Daniel Waters
Applied Mathematics and Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA, USA
pagoda@lbl.gov

1 INTRODUCTION

Lawrence Berkeley National Lab is developing a programming system to support HPC application development using the Partitioned Global Address Space (PGAS) model. This work includes two major components: UPC++ (a C++ template library) and GASNet-EX (a portable, high-performance communication library). This poster describes recent advances in GASNet-EX to efficiently implement Remote Memory Access (RMA) operations to and from memory on accelerator devices such as GPUs. Performance is illustrated via benchmark results from UPC++ and the Legion programming system, both using GASNet-EX as their communications library.

2 BACKGROUND

GASNet-EX [5, 13] is a lightweight communications middleware layer designed to support exascale clients, and is implemented over the native 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 [9]. It features one-sided communication via Remote Memory Access (RMA), remote procedure calls via Active Messages (AMs), remote atomic operations, and non-blocking collectives.

GASNet-EX is an evolution of GASNet-1 [4] and includes a backwards-compatibility layer to enable incremental migration of current GASNet-1 client software. Compared to GASNet-1, GASNet-EX provides enhancements needed for modern asynchronous PGAS models including adjusted interfaces for improved scalability, reduced CPU and memory overheads, and improved support for aggressive multi-threading [14]. GASNet has many important clients, including: UPC++ [21], the Legion programming system [3], HPE's Chapel language [7], the OpenSHMEM reference implementation [20], the Omni Xcalable Compiler [18], and many UPC [8, 15, 16] and CAF/Fortran [10–12] compilers. Of these, UPC++, Legion, Chapel and the Berkeley UPC Runtime have been updated to become GASNet-EX clients. Some of these clients are informing the direction of GASNet-EX development: features critical to UPC++ are being co-designed, and the GASNet-EX design is influenced by input from the Legion and Chapel teams.

Some API enhancements made in GASNet-EX (and detailed in [5]) include: endpoint naming using `(team,rank)` (for improved composability), "immediate mode" injection (to avoid stalls due to backpressure), explicit handling of local-completion (for improved buffer lifetime), "Negotiated-Payload" AM (to reduce buffer copying between layers), atomic operations in distributed memory (implemented using NIC offload where available), non-contiguous point-to-point RMA APIs, non-blocking collectives, multiple endpoints and segments, and support for communication to and from device memory (such as in a GPU). This poster describes this last item, support for communication involving device memory, which is known as "Memory Kinds" in GASNet-EX.

3 MEMORY KINDS

In GASNet-1, each process had a single communications endpoint with an optional remote-access memory segment established at initialization. Recent API enhancements, introduced in GASNet-EX in late 2020, add the capability for a GASNet-EX client to create multiple endpoints, each with an optional remote-access memory segment. Furthermore, this recent work introduces the concept of a memory kind which is an abstraction of memories with different properties and mechanisms for access[1].

Use of memory kinds by a client informs GASNet-EX that a given segment is in the memory of device of a given type, which ensures that appropriate access methods are used for communication. The current GASNet-EX release includes memory kinds support for Mellanox network hardware with GPUs from Nvidia and AMD[2]. Such a pairing of network and GPU can utilize the technology known as "GPUDirect RDMA" (GDR) to enable the network adapter to directly access the GPU memory (such as for RMA puts and gets) without the need to use the CPU or host memory to stage the transfer through any intermediate buffers. This zero-copy capability yields significant acceleration of eligible transfers. The poster describes these API extensions and evaluates the performance benefit, relative both to mechanisms used prior to memory kinds and to CUDA-enabled MPI (also using GDR).

4 BENCHMARK HIGHLIGHTS

To evaluate the performance of the GASNet-EX Memory Kinds implementation, the poster presents results of multiple microbenchmarks and one application kernel. This section presents some highlights selected from among those results.

4.1 UPC++

UPC++ [1, 2, 6] is a C++ library developed by the same team as GASNet-EX to provide high-level productivity abstractions appropriate for PGAS applications programming such as: remote procedure call, locality-aware APIs for user-defined distributed objects, and robust support for asynchronous execution to hide communication costs. UPC++ implements one-sided communication as a thin wrapper over GASNet-EX, delivering efficient performance.

UPC++ has its own "memory kinds" abstraction, which includes a global pointer class that enables the `upcxx::copy` function to express transfers between any combination of local and remote

SC21, November 14–19, 2021, St. Louis, MO, USA

https://doi.org/10.25344/S4P306

[1] For instance, it is not possible in general to use `memcpy()` to access device memory
[2] Support for other network and GPU vendors is planned as future work.

shared memory whether residing in host or device memory. The specification and implementation of memory kinds in UPC++ preceded the development of the corresponding support in GASNet-EX. Older UPC++ releases staged device memory transfers through host memory, whereas more recent releases utilize GASNet-EX memory kinds. Among other results shown on the poster, Fig. 1 shows the bandwidth of `upcxx::copy` for one particular transfer at various sizes. The data was collected on OLCF's Summit [19] supercomputer and includes series for UPC++ with both the older implementation of memory kinds that staged through host memory and the new zero-copy GDR implementation, as well as an equivalent MPI benchmark using IBM Spectrum MPI. The results demonstrate that GASNet-EX memory kinds enable substantial improvement in the performance of `upcxx::copy`, taking it from substantially underperforming relative to MPI, to delivering comparable or superior performance.

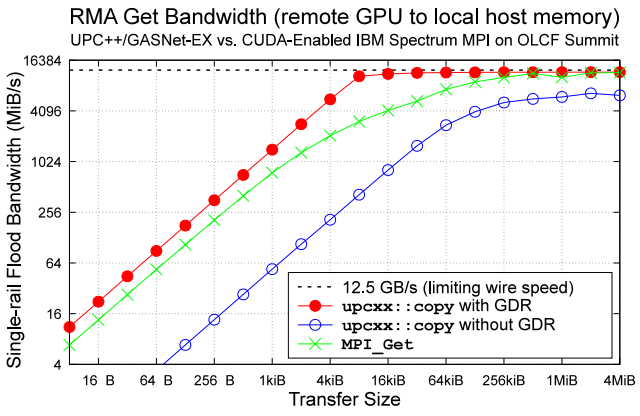


Figure 1: Performance comparison for GPU to host memory transfers in UPC++/GASNet-EX and MPI

4.2 Legion

The authors of Legion [3] characterize it as "a data-centric programming model for writing high-performance applications for distributed heterogeneous architectures" [17]. With its focus on heterogeneous systems, communication targeting GPU memory is a key part of Legion's Realm runtime system, making GASNet-EX Memory Kinds an important feature.

Legion version 20.12.0 retains its GASNet-1 backend while introducing a new communications backend utilizing the GASNet-EX APIs. Where the former explicitly stages GPU memory transfers through the host memory segment, the latter uses a GPU memory segment to enable RMA operations which target GPU memory without any staging. Among additional details given on the poster, Fig. 2 illustrates the performance improvement observed by switching from the GASNet-1 to GASNet-EX backend using the *same* GASNet library release[3]. These results show up to a 78% bandwidth improvement for transfers between a local and remote GPU.

[3]This is possible because GASNet-EX retains API compatibility with GASNet-1.

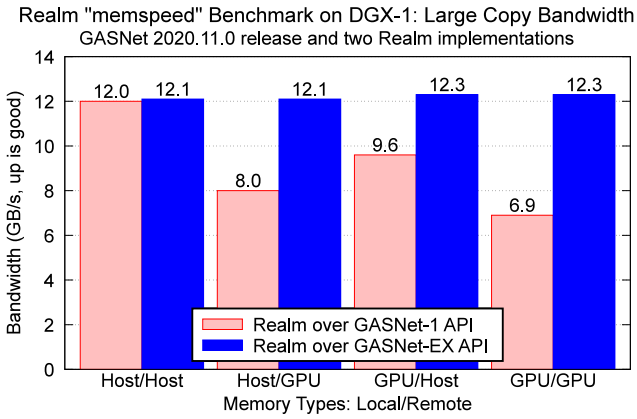


Figure 2: Legion `memspeed` microbenchmark "large copy bandwidth" performance for four different transfer patterns involving local and remote host and GPU buffers.

4.3 Kokkos Heat Conduction Example

The third benchmarking study shown on the poster is a Kokkos tutorial example, which solves the heat equation in three-dimensions using GPUs for the computation. This study compares performance of the original MPI example and a port to UPC++, and finds the latter performs as well or better on a wide range of problem sizes. Of particular interest is the finding, illustrated in Fig. 3, that the per-timestep latency for the UPC++ version is very uniform in contrast to a very high variability for MPI.

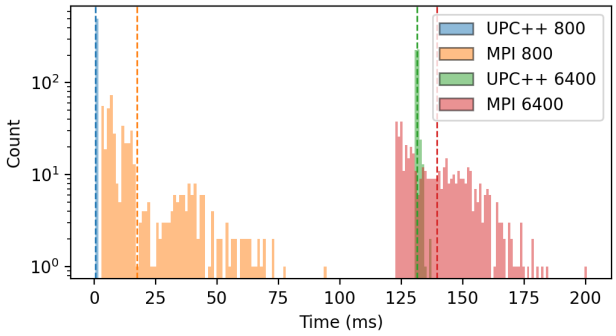


Figure 3: Histogram of latency to complete two time steps (sliding window) of the UPC++/GASNet-EX and MPI heat conduction simulations for two representative problem sizes. A dotted vertical line marks the median of each histogram.

5 CONCLUSIONS

GASNet-EX leverages hardware support to portably and efficiently implement Active Messages and Remote Memory Access (RMA). The recent addition of support for offloaded communication to and from GPU memory helps to improve and extend the role of PGAS programming models on modern heterogeneous systems.

ACKNOWLEDGMENTS

The authors gratefully acknowledge Sean Treichler of the Legion development team for collecting the raw data presented in Fig. 2.

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 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.

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.

We gratefully acknowledge the computing resources provided and operated by the Joint Laboratory for System Evaluation (JLSE) at Argonne National Laboratory.

REFERENCES

[1] John Bachan, Scott B. Baden, Dan Bonachea, Max Grossman, Paul H. Hargrove, Steven Hofmeyr, Mathias Jacquelin, Amir Kamil, Brian van Straalen, and Daniel Waters. 2021. *UPC++ v1.0 Programmer's Guide, Revision 2021.9.0.* Technical Report LBNL-2001424. Lawrence Berkeley National Laboratory. doi:10.25344/S4SW2T

[2] John Bachan, Scott B. Baden, Steven Hofmeyr, Mathias Jacquelin, Amir Kamil, Dan Bonachea, Paul H. Hargrove, and Hadia Ahmed. 2019. UPC++: A High-Performance Communication Framework for Asynchronous Computation. In *Proceedings of the 33rd IEEE International Parallel & Distributed Processing Symposium (IPDPS).* 11 pages. doi:10.25344/S4V88H

[3] Michael Bauer, Sean Treichler, Elliott Slaughter, and Alex Aiken. 2012. Legion: expressing locality and independence with logical regions. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '12).* doi:10.1109/SC.2012.71

[4] Dan Bonachea and Paul H. Hargrove. 2017. *GASNet Specification, v1.8.1.* Technical Report LBNL-2001064. Lawrence Berkeley National Laboratory. doi:10.2172/1398512

[5] Dan Bonachea and Paul H. Hargrove. 2018. GASNet-EX: A High-Performance, Portable Communication Library for Exascale. In *Languages and Compilers for Parallel Computing (LCPC'18).* doi:10.25344/S4QP4W

[6] Dan Bonachea and Amir Kamil. 2021. *UPC++ v1.0 Specification, Revision 2021.9.0.* Technical Report LBNL-2001425. Lawrence Berkeley National Laboratory. doi:10.25344/S4XK53

[7] Bradford L. Chamberlain, David Callahan, and Hans P. Zima. 2007. Parallel Programmability and the Chapel Language. In *International Journal of High Performance Computing Applications (IJHPCA)*, Vol. 21. 291–312.

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

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

[10] Y. Dotsenko, C. Coarfa, and J. Mellor-Crummey. 2004. A Multi-platform Co-Array Fortran Compiler. In *Proc. 13th International Conference on Parallel Architecture and Compilation Techniques (PACT).* doi:10.1109/PACT.2004.1342539

[11] Deepak Eachempati, Hyoung Joon Jun, and Barbara Chapman. 2010. 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)).* ACM, Article 13, 8 pages. doi:10.1145/2020373.2020386

[12] Alessandro Fanfarillo, Tobias Burnus, Valeria Cardellini, Salvatore Filippone, Dan Nagle, and Damian Rouson. 2014. OpenCoarrays: Open-source Transport Layers Supporting Coarray Fortran Compilers. In *Proceedings of the 8th International Conference on Partitioned Global Address Space Programming Models (PGAS '14).* ACM, New York, NY, USA, Article 4, 11 pages. doi:10.1145/2676870.2676876

[13] GASNet. home page. https://gasnet.lbl.gov.

[14] Paul H. Hargrove and Dan Bonachea. 2018. GASNet-EX Performance Improvements Due to Specialization for the Cray Aries Network. In *2018 IEEE/ACM Parallel Applications Workshop, Alternatives To MPI (PAW-ATM).* 23–33. doi:10.1109/PAW-ATM.2018.00008

[15] Intrepid Technology, Inc. Clang UPC Compiler. https://clangupc.github.io.

[16] Intrepid Technology, Inc. GCC/UPC Compiler. https://www.gccupc.org.

[17] Legion Programming System. home page. http://legion.stanford.edu/.

[18] Hitoshi Murai, Masahiro Nakao, Hidetoshi Iwashita, and Mitsuhisa Sato. 2017. Preliminary Performance Evaluation of Coarray-based Implementation of Fiber Miniapp Suite Using XcalableMP PGAS Language. In *Proceedings of the Second Annual PGAS Applications Workshop (PAW17).* ACM, Article 1, 7 pages. doi:10.1145/3144779.3144780

[19] Oak Ridge National Laboratory Leadership Computing Facility (ORNL/OLCF). Summit. https://olcf.ornl.gov/olcf-resources/compute-systems/summit/.

[20] Swaroop Pophale, Ramachandra Nanjegowda, Tony Curtis, Barbara Chapman, Haoqiang Jin, Stephen Poole, and Jeffery Kuehn. 2012. OpenSHMEM Performance and Potential: A NPB Experimental Study. In *Proceedings of the 6th Conference on Partitioned Global Address Space Programming Models (PGAS'12).* https://www.osti.gov/biblio/1055092

[21] UPC++. home page. https://upcxx.lbl.gov.

Artifact Description Appendix for SC21 poster: "GASNet-EX Memory Kinds: Support for Device Memory in PGAS Programming Models"

Paul H. Hargrove, Dan Bonachea, Colin A. MacLean, Daniel Waters
Applied Mathematics and Computational Research Division,
Lawrence Berkeley National Laboratory, Berkeley, CA, USA
pagoda@lbl.gov

This poster features data from multiple systems and benchmarks. This document provides the available/relevant requested information for each set of experiments plotted on the poster. Due to publication deadlines, it was not possible to collect data for all experiments using the most recent software versions.

Panel "GASNet-EX Host Memory RMA Performance versus MPI RMA and Isend/Irecv"

Because this panel is used to establish the baseline/background for the new work, the majority of its plots are reproduced with permission from our prior publication at LCPC'18 (https://doi.org/10.25344/S4QP4W). Section 3 of that publication provides information on the platforms used and benchmarks run.

The results for Summit (one group of bars in the latency plot and one entire bandwidth plot) are new, since the LCPC'18 paper predates public availability of Summit. Details of Summit at the time the data was collected are as follows, with all other details of the benchmarks run remaining unchanged from the LCPC'18 paper.

- "Summit" (see https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/)
- Relevant computer node hardware
  - IBM Power System AC922 node
  - 2x IBM POWER9 CPUs
  - 6x NVIDIA Volta V100s
  - Mellanox EDR 100G InfiniBand (dual-rail, ConnectX-5 HCAs)
- Relevant software versions
  - Red Hat Enterprise Linux Server 7.6
  - Linux 4.14.0-115.6.1.el7a.ppc64le kernel
  - IBM XL C/C++ for Linux, Version 16.1.1.3
  - IBM Spectrum MPI 10.3.0.0
  - Intel MPI Microbenchmarks 2019.2

Panel "UPC++ Microbenchmark Results with GPU Memory"

These results are from runs, on Summit, of "cuda_microbenchmark" in the UPC++ distribution and "osu_get_bw" from the OSU suite of MPI micro-benchmarks.

- "Summit" (see https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/)
- Relevant compute node hardware
  - IBM Power System AC922 node
  - 2x IBM POWER9 CPUs
  - 6x NVIDIA Volta V100s
  - Mellanox EDR 100G InfiniBand (dual-rail, ConnectX-5 HCAs)
- Relevant software versions
  - Red Hat Enterprise Linux Server 7.6
  - Linux 4.14.0-115.6.1.el7a.ppc64le kernel
  - GNU gcc/g++ compilers, version 6.4.0
  - IBM Spectrum MPI 10.3.1.2
  - UPC++ 2020.11.0
  - CUDA 10.1.243
  - OSU Micro-Benchmarks 5.6.3
- Commands used to launch benchmarks on two nodes with 1 process and 1 GPU per node:

```
jsrun --smpiargs=-gpu -g1 -r1 -p2 ./cuda_microbenchmark –t 100 –w 100 -sg
jsrun --smpiargs=-gpu -g1 -r1 -p2 ./osu_get_bw –i 100 –d cuda D H
```

Panel "Legion Microbenchmark Results with GPU Memory"

This panel's figures are the authors' presentation of raw data provided by Sean Treichler of the Legion development team at Nvidia, who provided only the following information: "All runs performed on same pair of DGX-1, using only 1 GPU (V100), 1 NIC (CX-6), and 1 NUMA domain per node". Software versions used include "GASNet-2020.11.0-memory_kinds_prototype" (available from gasnet.lbl.gov) and the developer's version of Legion's librealm which preceded their 20.12.0 release.

Panel "UPC++ Application Kernel Performance"

These results are from runs, on Summit, of a Kokkos tutorial example as described on the poster and its references.

- "Summit" (see https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/)
- Relevant compute node hardware
  - IBM Power System AC922 node
  - 2x IBM POWER9 CPUs
  - 6x NVIDIA Volta V100s
  - Mellanox EDR 100G InfiniBand (dual-rail, ConnectX-5 HCAs)
- Relevant software versions
  - Red Hat Enterprise Linux Server 7.6
  - Linux 4.14.0-115.6.1.el7a.ppc64le kernel
  - GNU gcc/g++ compilers, version 8.1.1
  - IBM Spectrum MPI 10.3.1.2
  - UPC++ 2021.3.0
  - CUDA 10.1.243
  - Kokkos 3.4.0

GASNet-EX Memory Kinds: Support for Device Memory in PGAS Programming Models

Paul H. Hargrove, Dan Bonachea, Colin A. MacLean, Daniel Waters

Applied Mathematics and Computational Research Division, Lawrence Berkeley National Laboratory

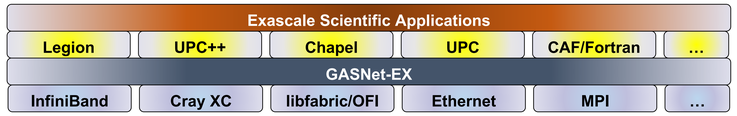
https://gasnet.lbl.gov

pagoda@lbl.gov

Click Here for Narration Video!

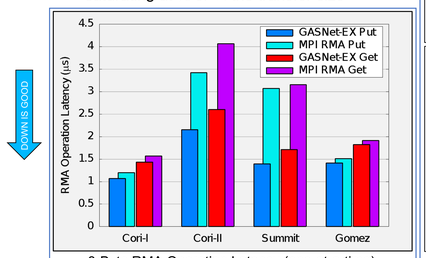
GASNet-EX Background

- GASNet-EX is communications middleware to support exascale clients
  - Widely adopted for implementation of Partitioned Global Address Space (PGAS) programming models
  - One-sided communication – Remote Memory Access (RMA)
  - Active Messages (AMs) - remote procedure call
  - Implemented over native APIs of all networks of interest to DOE
- GASNet-EX is an evolution of GASNet-1 for exascale
  - Retains GASNet-1's wide portability (laptops to supercomputers)
  - Focus remains on one-sided RMA and Active Messages
  - Reduces CPU and memory overheads
  - Improves support for aggressive multi-threading
  - Adds support for device memory (GPUs)

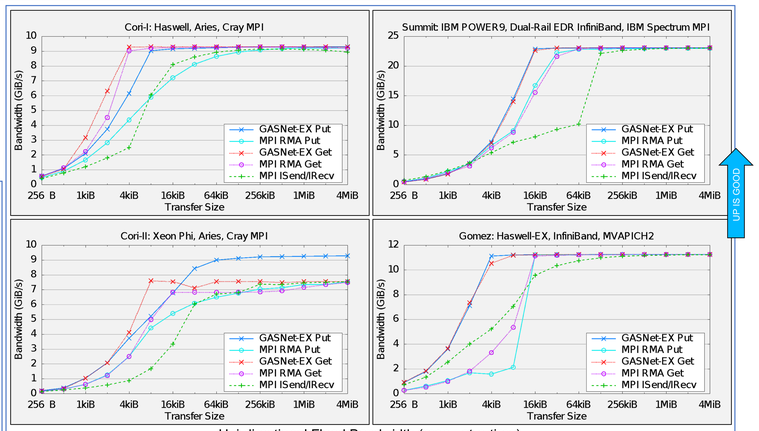


GASNet-EX Host Memory RMA Performance versus MPI RMA and Isend/Irecv

- Three different MPI implementations
- Two distinct network hardware types
- On four systems the performance of GASNet-EX matches or exceeds that of MPI RMA and message-passing:
  - 8-byte Put latency 6% to 55% better
  - 8-byte Get latency 5% to 45% better
  - Better flood bandwidth efficiency, typically saturating at ½ or ¼ the transfer size



8-Byte RMA Operation Latency (one-at-a-time)



Uni-directional Flood Bandwidth (many-at-a-time)

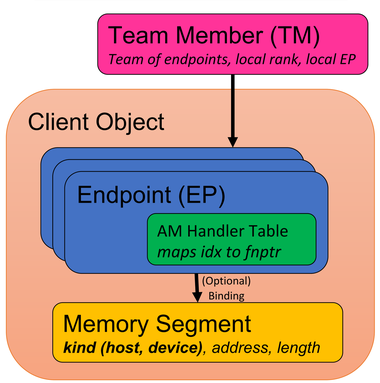
GASNet-EX results from v2018.9.0
MPI results from Intel MPI Benchmarks v2018.1

For more details see Languages and Compilers for Parallel Computing (LCPC'18).
https://doi.org/10.25344/S4QP4W

GASNet-EX Memory Kinds

- Performance for host memory RMA (see panel above) is largely due GASNet-EX's ability to efficiently utilize Remote Direct Memory Access (RDMA) capabilities in modern network hardware
  - No remote CPU involvement required for network adapter to access application memory
  - Simple and efficient implementation due to good semantic fit to the PGAS model
- The goal: extend GASNet-EX RMA to memory in devices such as GPUs, in addition to host memory
  - Challenge 1: RDMA capability for GPU memory is needed for best performance
    - Solved by the hardware vendors via GPUDirect RDMA (see upper-right panel)
  - Challenge 2: Retaining efficiency in the GASNet-EX implementation
    - This is addressed by our design of a new "memory kinds" abstraction in GASNet-EX
  - Challenge 3: Design for extensibility to other byte- or block-addressable "memories"
    - Our design encompasses such possible kinds as FPGA and storage
- Our solution: extend the concept of the "remote-access segment"
  - The arguments to every RMA operation specify a local and remote communications endpoint
  - To be valid for RMA operations, an endpoint must have an associated "bound segment"
  - NEW:
    - A "kind" can be created for each device: e.g. "CUDA device 0", "HIP device 2"
    - Instead of just a single implicit segment, multiple explicit segments are supported
    - Creation of a segment names the kind, along with a size and optional address
- This design means every RMA operation knows the device(s) involved from the existing arguments
  - Device addresses identified *without* dynamic queries or interposing on memory management APIs
- Good match to UPC++ and Legion programming models which already track host vs. device memory

High-level Object Model



GASNet-EX Support for GPUDirect RDMA (GDR)

- Memory kinds concept expresses use of device memory in RMA endpoint arguments
  - Implementation can easily identify appropriate transfer mechanism, including hardware-assisted technologies such as GPUDirect RDMA (GDR)
- GASNex-EX supports GDR in recent releases
  - Removes host CPU and memory bottlenecks from one-sided transfers to/from GPU memory, achieving true zero-copy (see diagram at right →)
  - Currently supports Nvidia and AMD GPUs; and Mellanox network adapters
  - Other accelerators and networks are the subject of future work, including
    - Intel GPUs; HPE Slingshot-11 network

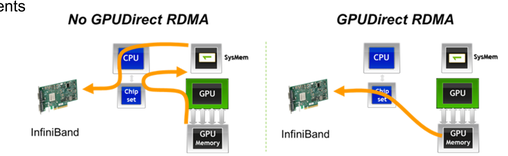
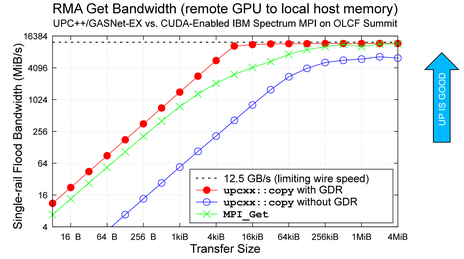


Image source: https://developer.nvidia.com/blog/introduction-cuda-aware-mpi/

UPC++ Microbenchmark Results with GPU Memory

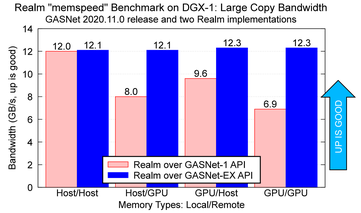
- UPC++ is our C++ productivity layer over GASNet-EX
  - Has its own analogous memory kinds concepts
  - Uses `upcxx::copy()` to express RMA between arbitrary shared memory
- Measurements of flood bandwidth of `upcxx::copy()` on OLCF's Summit
- Difference between UPC++ releases with and without GASNet-EX memory kinds shows benefit of GASNet-EX's new support for GDR
  - No longer staging through host memory (true zero-copy)
  - Large transfers: 2x better bandwidth
  - Small transfers: up to 30x better bandwidth
- Get operations to/from GPU memory now perform comparably to host memory
- Comparisons to MPI-3 RMA in CUDA-enabled IBM Spectrum MPI show UPC++ saturating to peak bandwidth at smaller transfer sizes

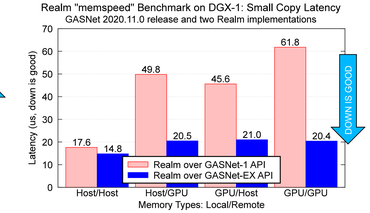


UPC++ results were collected using the version of the `cuda_benchmark` test that appears in the 2020.11.0 release.
MPI results are from `osu_get_bw` test in a CUDA-enabled build of OSU Micro-Benchmarks 5.6.3.
All tests were run between two nodes of OLCF Summit, over its EDR InfiniBand network.

Legion Microbenchmark Results with GPU Memory

- Legion characterizes itself as "a data-centric programming model for writing high-performance applications for distributed heterogeneous architectures''.
- Legion's Realm runtime provides communication services
  - Originally implemented over GASNet-1
  - Still works using legacy API support in current GASNet-EX
- Realm recently introduced a new GASNet-EX backend which embraces EX-specific capabilities
  - Leverages immediate, NPAM, and LC handles for AM
  - Leverages memory kinds for RMA, gaining GDR support
  - Multi-endpoint for RMA to/from additional host memory regions
- Figures at right show some performance benefits of using GDR
  - Large GPU transfers: bandwidth now matches host memory
  - Small GPU transfers: 2.2x to 3.0x latency improvement





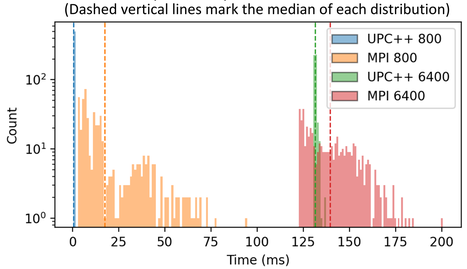
The authors gratefully acknowledge Sean Treichler of the Legion development team for collecting the data we present above.

UPC++ Application Kernel Performance

- Kokkos[1] is a popular on-node programming model providing C++ abstractions for use of multi-core and GPU-accelerated nodes with a single version of the application source code
- We converted a Kokkos tutorial example[2] from use of MPI message passing to UPC++ RMA
  - Use of UPC++ memory kinds for device memory management
  - Use of `upcxx::copy` for one-sided RMA data movement and remote notification
- Despite no changes to the computation, we observed unexpected performance differences
  - UPC++ RMA delivers better strong scaling (not pictured here) than MPI message passing
  - Even when performance is comparable, MPI version's performance has much larger variance
- This figure shows the relative performance and variance for two problem sizes and fixed resources
  - Histograms show the frequency of execution times, using a sliding window over two time steps to fairly account for MPI Recv and Send matching (this *reduces* MPI's measured variance)
  - UPC++ runs show lower (faster) median time step latency than the corresponding MPI runs
  - UPC++ runs show narrower (lower-variance) distributions than the corresponding MPI runs
- Future work planned to try to understand and eliminate MPI variance, and repeat the comparison
- More complete information will appear in a workshop talk on Friday morning:
  D. Waters, C. A. MacLean, D. Bonachea, P. H. Hargrove. **"Demonstrating UPC++/Kokkos Interoperability in a Heat Conduction Simulation (Extended Abstract)"**, In *2021 IEEE/ACM Parallel Applications Workshop, Alternatives To MPI+X (PAW-ATM)*, St. Louis, MO, Nov 2021. 5 pages. https://doi.org/10.25344/S4630V

[1] https://kokkos.org
[2] https://go.lbl.gov/paw21-kokkos-mpi-heat-conduction

Histogram of execution time for two time steps (sliding window) on 128 nodes of OLCF Summit, using 6 procs/node and 1 GPU/proc, for problem sizes 800^3 and 6400^3
(Dashed vertical lines mark the median of each distribution)



UPC++ results use UPC++ v2021.3.0 w/ GPUDirect RDMA
MPI results use CUDA-enabled IBM Spectrum MPI v10.3.1.2

https://doi.org/10.25344/S4P306

This research was supported in part 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 Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.
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.
We gratefully acknowledge the computing resources provided and operated by the Joint Laboratory for System Evaluation (JLSE) at Argonne National Laboratory.