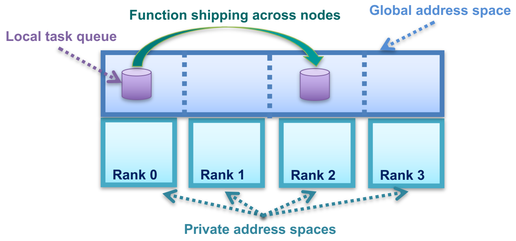
# UPC++ and GASNet: PGAS Support for Exascale Apps and Runtimes (WBS 2.3.1.14)


Paul H. Hargrove (PI)

With staff members: Dan Bonachea, Amir Kamil, Colin A. MacLean, Damian Rouson, Daniel Waters


## UPC++ at Lawrence Berkeley National Lab (upcxx.lbl.gov)

- UPC++ is a C++ PGAS library
  - Lightweight, asynchronous, one-sided communication (RMA)
  - Asynchronous remote procedure call (RPC)
  - Data transfers may be non-contiguous
  - Futures manage asynchrony, enable communication overlap
  - Collectives, teams, remote atomic updates
  - Provides building blocks to construct irregular data structures
- Latest software release: March 2022
  - Runs on systems from laptops to supercomputers
- Easy on-ramp and integration
  - Enables incremental development
  - Selectively replace performance-critical sections with UPC++
  - Interoperable with MPI, OpenMP, CUDA, etc.




**Integration efforts with ExaBiome (WBS 2.2.4.04)**

- MetaHipMer 2 (MHM2) demonstrated on OLCF Crusher, March 2022
- MHM2 is a pure UPC++ code
  - A rewrite of the original MHM application which used UPC and MPI
- UPC++'s RPC is a better fit to the problem than previous alternatives
- The rewrite reduced code size by roughly ¾
- Lower memory requirements and up to 6x better performance
- Produced record-breaking metagenome assembly on OLCF Summit

**Integration efforts with ExaGraph (WBS 2.2.6.07)**

- With PNNL team, have developed two UPC++ versions of a graph-matching problem from their IPDPS'19 paper
  - RMA version uses Puts to communicate among processes
  - RPC version uses asynchronous remote procedure calls to run logic on remote parts of the graph. Approx. 100 LoC reduction.
- Results on NERSC Cori Haswell (3.6B-edge Friendster):
  - Both UPC++ versions competitive with (or better than) best MPI versions up to at least 4,096 processes

**Integration efforts with NWChemEx (WBS 2.2.1.02)**

- Ported TAMM code base from Global Arrays/MPI to UPC++
  - TAMM implements distributed in-memory data store and compute for NWChemEx
  - UPC++ performance comparable to GA code (+10-15% run time)
- Current work-in-progress includes
  - Merging UPC++ communication into the main TAMM repository
  - Evaluating use of `upcxx::dist_array` in TAMM
  - Work toward UPC++ RPC in dynamically loaded libraries

## Case 1: Easy Distributed Hash-Table via Function Shipping and Futures

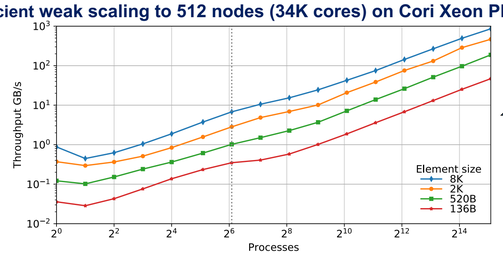
- **Distributed hash-table design is based on function shipping**
  - RPC inserts the key metadata at the target
  - Once the RPC completes, an attached callback issues a one-sided RMA Put (rput) to store the value data

```
// C++ global variables correspond to rank-local state
std::unordered_map<uint64_t, global_ptr<char>> local_map;
// insert a key-value pair and return a future
future<> dht_insert(uint64_t key, char *val, size_t sz) {
 future<global_ptr<char>> fut =
     rpc(key % rank_n(),              // RPC obtains location for the data
         [key,sz]() -> global_ptr<char> {    // lambda invoked by RPC
           global_ptr<char> gptr = new_array<char>(sz);
           local_map[key] = gptr;              // insert in local map
           return gptr;
         });
 return fut.then(                     // callback executes when RPC completes
       [val,sz](global_ptr<char> loc) -> future<> {
           return rput(val, loc, sz); });    // RMA Put the value payload
}
```

For details see [IPDPS'19]

- **Benefits:**
  - Use of **RPC** simplifies distributed data-structure design
    - Argument passing, remote queue management and progress engine are factored out of the application code
  - Asynchronous execution enables overlap

**Efficient weak scaling to 512 nodes (34K cores) on Cori Xeon Phi**

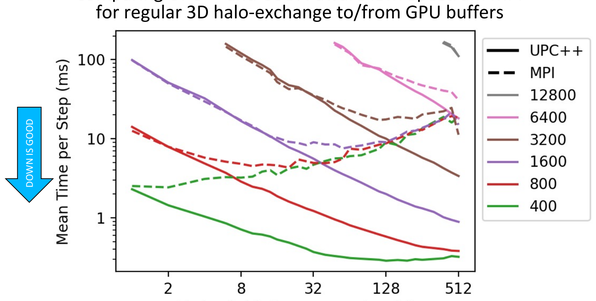



## Case 2: A 3D Halo-Exchange Example Using GPUs on OLCF Summit

- Began with a Kokkos heat-conduction tutorial example
  - Kokkos-based compute on the GPU
  - Regular 3D halo-exchange communication to/from GPU memory
- Communication converted from use of MPI message passing to UPC++ RMA to demonstrate UPC++/Kokkos interoperability
  - Use of UPC++ memory kinds for GPU memory management
  - Use of `upcxx::copy` for one-sided RMA data movement and for remote notification
- Despite no changes to the computation, we saw performance differences as a result of changing the programming model
- This figure shows strong scaling on OLCF Summit
  - Using six processes/node and one GPU/process
  - Six problem sizes and node counts from 1 to 512
  - Runs with UPC++ and CUDA-aware IBM Spectrum MPI
- UPC++ consistently meets or beats the performance of MPI

For additional details see [PAW-ATM'21]

Strong scaling of a Kokkos-based heat-conduction example, comparing UPC++ and CUDA-aware IBM Spectrum MPI for regular 3D halo-exchange to/from GPU buffers




Colors indicate problem size (side length of the computed cube)
Solid and dashed line styles indicate the UPC++ and MPI versions respectively


References:

- IPDPS'19: doi.org/10.25344/S4V88H
- LCPC'18: doi.org/10.25344/S4QP4W
- PAW-ATM'21: doi.org/10.25344/S4630V



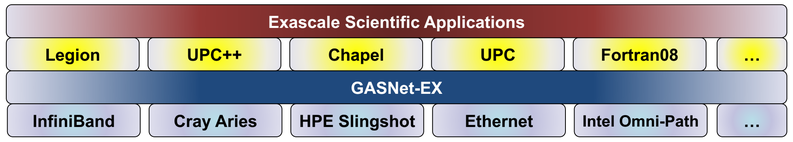

## GASNet-EX at Lawrence Berkeley National Lab (gasnet.lbl.gov)

GASNet-EX: communications middleware to support exascale clients

- One-sided communication – Remote Memory Access (RMA)
- Active Messages (AMs) – a form of remote procedure call
- Implemented over native APIs of all networks of interest to DOE
- Provides communication for several programming models including:
  - UPC++ (see left half of this poster)
  - Legion (WBS 2.3.1.08)
  - Chapel (from HPE, non-ECP)
- Backwards compatibility for the dozens of GASNet-1 clients

Major features of GASNet-EX developed under ECP funding:

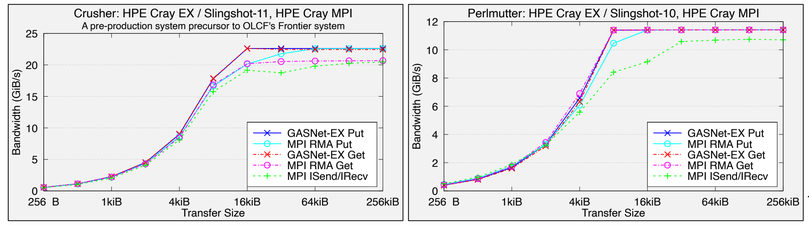
- "Immediate mode" injection to avoid stalls due to back-pressure
- Explicit handling of local completion (source buffer lifetime)
- Enhanced AM interfaces to reduce buffer copies between layers
- Vector-Index-Strided for non-contiguous point-to-point RMA
- Remote Atomics, implemented with NIC offload where available
- Subset teams and non-blocking collectives
- RMA directly to/from device memory on supported hardware
  - Includes Nvidia and AMD GPUs



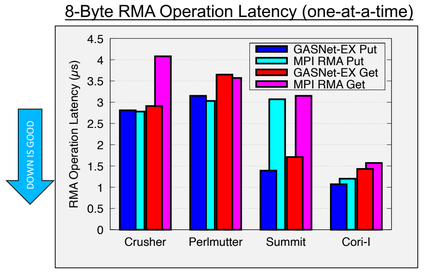

## GASNet-EX RMA Performance versus MPI RMA and Isend/Irecv

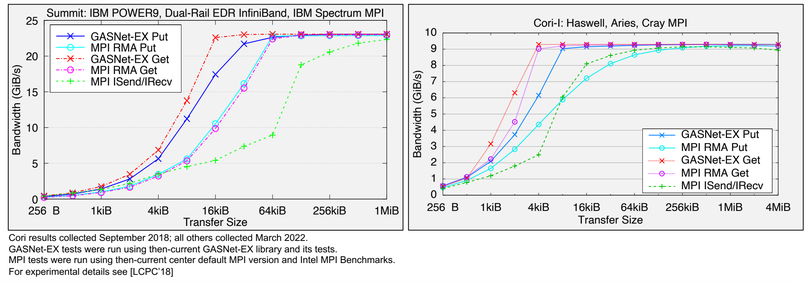
- Four distinct network hardware types
- The performance of GASNet-EX matches or exceeds that of MPI RMA and message-passing:
  - 8-byte Put latency up to 55% better
  - 8-byte Get latency up to 45% better
  - Better flood bandwidth efficiency: often reaching same or better peak at ½ or ¼ the transfer size

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




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

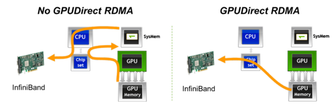


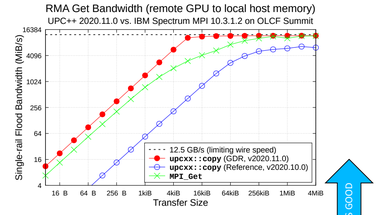

Cori results collected September 2018; all others collected March 2022.
GASNet-EX tests were run using then-current GASNet-EX library and its tests.
MPI tests were run using then-current center default MPI version and Intel MPI Benchmarks.
For experimental details see [LCPC'18]

## Support for RMA targeting GPU Memory – UPC++ and Legion/Realm Benchmarks

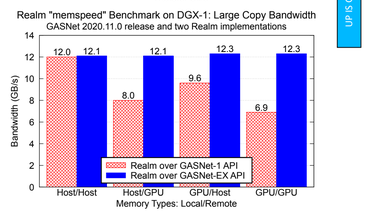
- Support for **Nvidia GPUDirect RDMA** (Nov. 2020)
  - Removes host CPU and memory bottlenecks from RMA transfers to/from GPU memory (see diagram ➔)
  - Works with Nvidia GPUs + Mellanox NICs




- Support for **AMD ROCmRDMA** (Sep. 2021)
  - Same benefits with AMD GPUs + Mellanox NICs
  - Demonstrated over HPE Slingshot-10 on OLCF's Spock
- Support for HPE Slingshot-11 and for Intel GPUs are each the subject of future work
- Comparisons of UPC++ to MPI RMA in CUDA-Aware IBM Spectrum MPI show UPC++ saturating more quickly to the peak (top-right plot)




- Realm is the low-level runtime for the Legion Programming System (WBS 2.3.1.08)
  - Communications services originally implemented over GASNet-1
  - New communications backend (Dec 2020) embraces capabilities specific to GASNet-EX
  - Most notable new capabilities are support for GPU RMA and for the HPE Slingshot network
  - Also leverages Immediate, NPAM, and local completion events for AM
- Some performance benefits of using GASNet-EX's Nvidia GPU support in Realm:
  - Large GPU memory xfers: same bandwidth as host memory (bottom-right plot)
  - Small GPU memory xfers: 2.2x to 3.0x latency improvement





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.