GASNet-EX: A High-Performance, Portable Communication Library for Exascale

Dan Bonachea

gasnet-staff@lbl.gov   gasnet.lbl.gov

Joint work with Paul H. Hargrove
and the LBNL Pagoda Project (CRD/CLaSS)

Acknowledgements

This research was funded 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 **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.

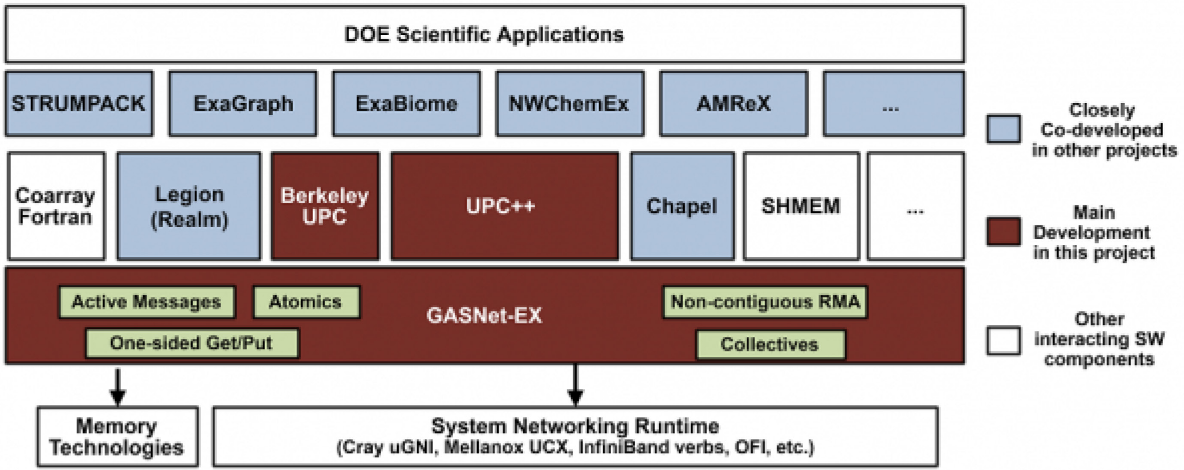
This research used resources of the **Argonne Leadership Computing Facility**, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

The Pagoda Project

https://crd.lbl.gov/pagoda

Support for lightweight communication for exascale applications, frameworks and runtimes

- **GASNet-EX** low-level layer that provides a network-independent interface suitable for Partitioned Global Address Space (PGAS) runtime developers
- **UPC++** C++ PGAS library for application, framework and library developers, a productivity layer over GASNet-EX



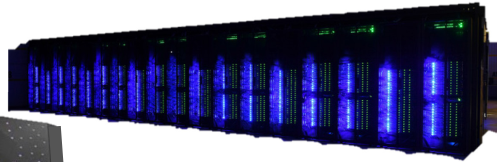
Motivating HPC system trends

The first exascale systems will appear soon

- Cores per node is growing (including GPU cores)
- Network transfers becoming smaller and more frequent
- Network latency is <u>not</u> improving

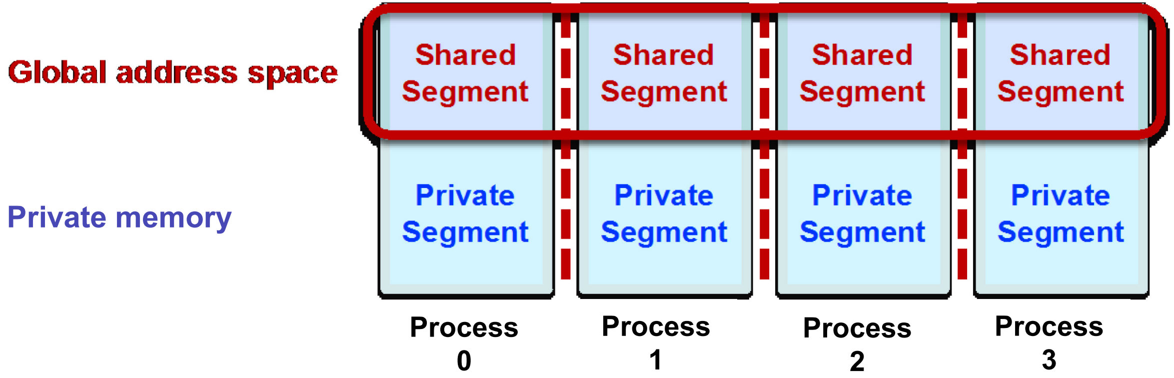
Need to reduce communication costs in software

- Minimize software overhead per transfer
- Overlap communication to hide latency
- Use simpler communications protocols (RDMA)



A Partitioned Global Address Space programming model

- Global Address Space
  - Processes may read and write *shared segments* of memory
  - Global address space = union of all the shared segments
- Partitioned
  - *Global pointers* to objects in shared memory have an affinity to a particular process
  - Locality explicitly managed by the programmer to optimize communication
  - In conventional shared memory programming, pointers do not encode affinity



The PGAS model

Partitioned **G**lobal **A**ddress **S**pace

- Support global memory, leveraging the network's RDMA capability
- Distinguish private and shared memory
- Separate synchronization from data movement

Languages that provide PGAS:

UPC, Titanium, Chapel, X10, Co-Array Fortran (Fortran 2008)

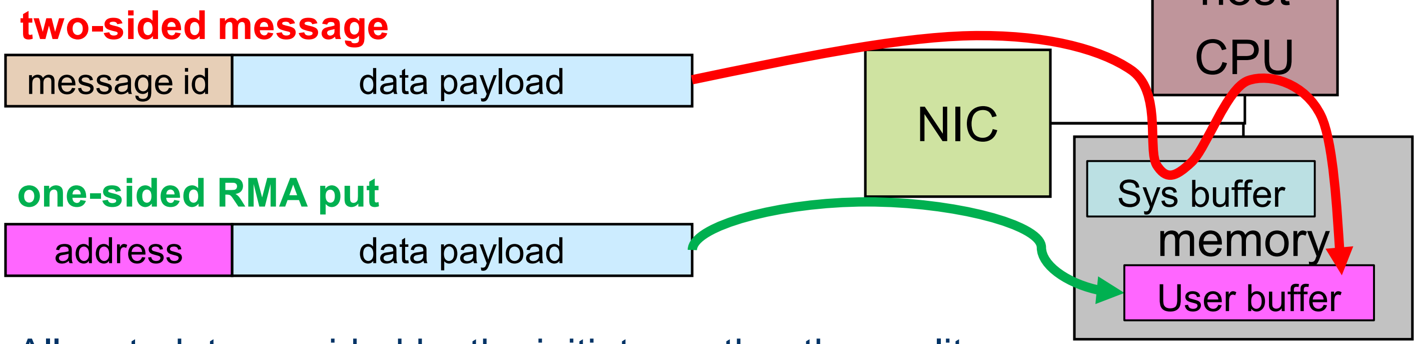
Libraries that provide PGAS:

UPC++, OpenSHMEM, Co-Array C++, Global Arrays, DASH, MPI-RMA

A key semantic property is support for one-sided RMA

Reducing communication overhead using one-sided RMA

- Idea: Let each process directly access another's memory via a global pointer
- Communication is **one-sided** : there is no "receive" operation
  - No need to match sends to receives
  - No unexpected messages
  - No need to guarantee message ordering



  - All metadata provided by the initiator, rather than split between sender and receiver
  - Supported in hardware through RDMA (Remote Direct Memory Access)
- Looks like shared memory: shared data structures with asynchronous access

GASNet-1: Historical Overview

- Started in 2002 to provide a portable network communication runtime for three PGAS languages:
  - Titanium, UPC and CAF
- Primary features:
  - Non-blocking RMA (one-sided Put and Get)
  - Active Messages (simplification of Berkeley AM-2)
- Motivated by semantic issues in (then current) MPI-2.0
  - https://doi.org/10.1504/IJHPCN.2004.007569

Int. J. High Performance Computing and Networking, Vol. 1, Nos. 1/2/3, 2004 91

Problems with using MPI 1.1 and 2.0 as compilation targets for parallel language implementations

Dan Bonachea* and Jason Duell
Computer Science Division,
University of California at Berkeley,
Berkeley, California, USA

https://gasnet.lbl.gov

GASNet: Adoption and Portability

Client runtimes

LBNL UPC++
Berkeley UPC
GCC/UPC
Clang UPC
Chapel (Cray/HPE)
Legion (Stanford/NVIDIA/…)
Titanium
Rice Co-Array Fortran
OpenUH Co-Array Fortran
OpenCoarrays in GCC Fortran
OpenSHMEM reference impl.
Omni XcalableMP
PARADISE++ Devastator
At least 6 others known to us

Network conduits

OpenFabrics Verbs (InfiniBand)
Mellanox MXM and VAPI (InfiniBand)
Cray uGNI (Gemini and **Aries**)
Intel PSM2 (OmniPath)
IBM PAMI (BG/Q and others)
UDP (any TCP/IP network)
MPI 1.1 or newer
IBM DCMF (BG/P)
IBM LAPI (Colony and Federation)
Cray Portals3 (Seastar)
SHMEM (Cray X1 and SGI Altix)
Quadric elan3/4 (QsNet I/II)
OFI / libfabric (many)
UCX (many)
Myricom GM (Myrinet)
Dolphin SISCI
Sandia Portals4
Shared memory (no network)

Supported platforms

– Over 10 compiler families, 15 operating systems and dozens of architectures

* These lists and counts include both current and past support

https://gasnet.lbl.gov

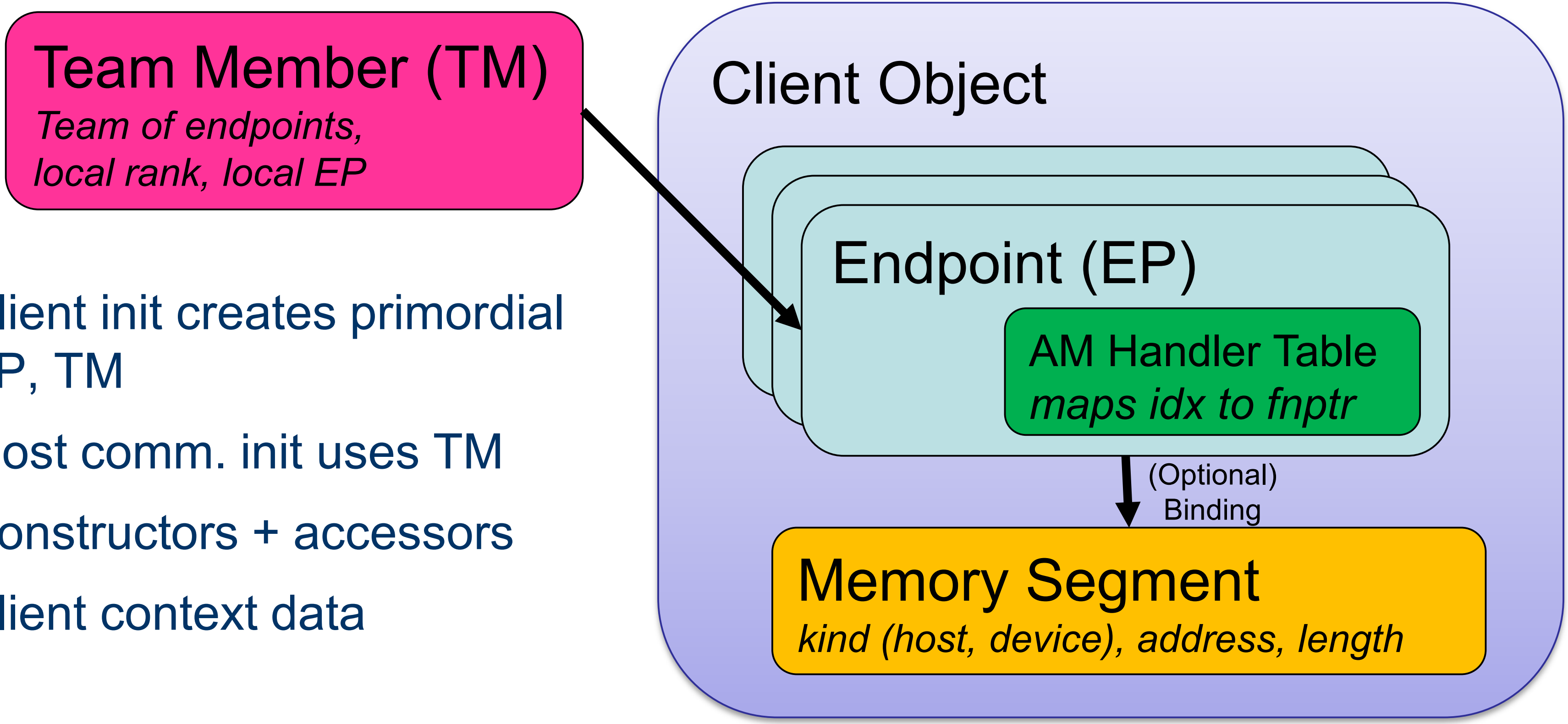
BERKELEY LAB

GASNet-EX: Overview

- GASNet-EX is the next generation of GASNet
  - Addressing needs of newer programming models such as UPC++, Legion and Chapel
  - Incorporates 20 years of lessons learned and focuses on the challenges of emerging exascale systems
  - Provides backward compatibility for GASNet-1 clients
- Motivating goals include
  - Support more client asynchrony
  - Enable more client adaptation
  - Improve memory footprint
  - Improve threading support
  - Support offload to network h/w
  - Support multi-client applications
  - Support for device memory

https://gasnet.lbl.gov

GASNet-EX High-level Object Model



- Client init creates primordial EP, TM
- Most comm. init uses TM
- Constructors + accessors
- Client context data

Overview of GASNet-EX Improvements

GASNet-1:

```
gasnet_handle_t
gasnet_put_nb(gasnet_node_t node, void *dest_addr,
              void *src_addr, size_t nbytes);
```

GASNet-EX:

```
gex_Event_t
gex_RMA_PutNB(gex_TM_t tm, gex_Rank_t rank, gex_Addr_t dest_addr,
              void *src_addr, size_t nbytes,
              gex_Event_t *lc_opt, gex_Flags_t flags);
```

A representative example: non-blocking RMA Put

Changes between these two (in red on following slides) illustrate some of the most meaningful changes made in the GASNet-EX design.

They provide the means to address several goals.

Overview of GASNet-EX Improvements

GASNet-1:

```
gasnet_handle_t
gasnet_put_nb(gasnet_node_t node, void *dest_addr,
              void *src_addr, size_t nbytes);
```

GASNet-EX:

```
gex_Event_t
gex_RMA_PutNB(gex_TM_t tm, gex_Rank_t rank, gex_Addr_t dest_addr,
              void *src_addr, size_t nbytes,
              gex_Event_t *lc_opt, gex_Flags_t flags);
```

Destination memory space

GASNet-1: an integer `node` identifier to name a process

GASNet-EX: a `(team,rank)` pair to name an "Endpoint"

- "Team" is an ordered sets of Endpoints (also used in collectives)
- Multiple Endpoints for multi-threading and access to device memory
- Multiple Client runtimes for hybrid applications

Overview of GASNet-EX Improvements

GASNet-1:

```
gasnet_handle_t
gasnet_put_nb(gasnet_node_t node, void *dest_addr,
              void *src_addr, size_t nbytes);
```

GASNet-EX:

```
gex_Event_t
gex_RMA_PutNB(gex_TM_t tm, gex_Rank_t rank, gex_Addr_t dest_addr,
              void *src_addr, size_t nbytes,
              gex_Event_t *lc_opt, gex_Flags_t flags);
```

Destination Address

GASNet-1: a remote virtual address

GASNet-EX: a remote virtual address or an *offset*

- Offsets can improve scalability of clients using symmetric heaps
- Used with multiple endpoints it provides addressing flexibility, which can be useful for device memory

Overview of GASNet-EX Improvements

GASNet-1:
```
gasnet_handle_t
gasnet_put_nb(gasnet_node_t node, void *dest_addr,
              void *src_addr, size_t nbytes);
```

GASNet-EX:
```
gex_Event_t
gex_RMA_PutNB(gex_TM_t tm, gex_Rank_t rank, gex_Addr_t dest_addr,
              void *src_addr, size_t nbytes,
              gex_Event_t *lc_opt, gex_Flags_t flags);
```

Return Type

GASNet-1: “handle” to test for operation completion

- Thread-specific (only the issuing thread can test/wait for completion)

GASNet-EX: “Event” generalizes handle in two directions

- Not thread-specific (for progress threads, continuation passing, etc.)
- Supports multiple sub-events (e.g. local completion on later slide)

Overview of GASNet-EX Improvements

GASNet-1:

```
gasnet_handle_t
gasnet_put_nb(gasnet_node_t node, void *dest_addr,
              void *src_addr, size_t nbytes);
```

GASNet-EX:

```
gex_Event_t
gex_RMA_PutNB(gex_TM_t tm, gex_Rank_t rank, gex_Addr_t dest_addr,
              void *src_addr, size_t nbytes,
              gex_Event_t *lc_opt, gex_Flags_t flags);
```

Local Completion (when local source buffer may be overwritten)

GASNet-1: ...`put_nb()` vs. ...`put_nb_bulk()`

- Local completion can occur separately from remote completion
- Option to conflate it with either injection or remote completion

GASNet-EX: `lc_opt` selects a local completion behavior

- Both GASNet-1 options, plus an Event the client can test/wait

Overview of GASNet-EX Improvements

GASNet-1:
```
gasnet_handle_t
gasnet_put_nb(gasnet_node_t node, void *dest_addr,
              void *src_addr, size_t nbytes);
```

GASNet-EX:
```
gex_Event_t
gex_RMA_PutNB(gex_TM_t tm, gex_Rank_t rank, gex_Addr_t dest_addr,
              void *src_addr, size_t nbytes,
              gex_Event_t *lc_opt, gex_Flags_t flags);
```

Per-operation Flags

GASNet-EX: introduces extensibility modifiers

- *Require* non-default behaviors, such as offset-based addressing
- *Allow* optional behaviors, such as “Immediate Mode” (later slide)
- *Assert* properties which may eliminate more costly dynamic checks

GASNet-1: has no direct equivalent

Selected GASNet-EX Improvements

- Several new EX features are already delivering benefits
- This section reports on some of these
  - Local Completion Control
  - Immediate-mode Communication Injection
  - Negotiated-payload Active Messages
  - Remote Atomics
  - Device Memory RMA offload
- Results collected on Cray XC40 and Summit IB systems
- More details in LCPC’18:
  - Bonachea, Hargrove. "GASNet-EX: A High-Performance, Portable Communication Library for Exascale", https://doi.org/10.25344/S4QP4W

BERKELEY LAB

Local Completion Control

- GASNet-EX introduces means for client to test (or wait) for local completion *between* injection and completion of a non-blocking Put
  - At granularity of single operations or specified groups
- Exposes greater opportunity for fine-grained overlap
  - Enable separate tracking of local vs remote completion
  - Improves client control over buffer lifetime
    - Facilitates recycling local buffers sooner

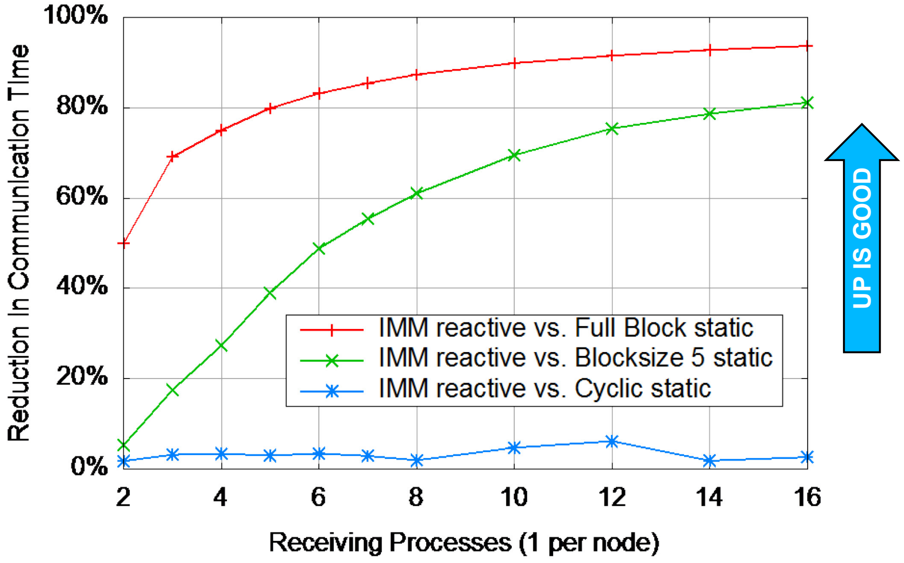
Immediate-mode Communication Injection

- Lack of resources can stall communication injection
  - Such backpressure may be path-specific
- New feature allows client adaptation to such a scenario
  - E.g. work-stealing could select a different victim
- Immediate-mode is a flag which permits (does not require) implementation to return *without* performing communication, in the presence of backpressure
- Enables client to avoid stalls in low-resource conditions

Immediate-mode Communication Injection

- Figure illustrates performance on a benchmark modeling AM communication with inattentive peers
- Shows reduction in time to complete communication using a "reactive" immediate-mode approach
- The series compare reactive to three distinct static schedules
- Best case is 93% reduction

Reduced communication delays using immediate-mode Active Messages



ARGONNE NATIONAL LABORATORY

GASNet-EX

BERKELEY LAB

Remote Atomic Memory Operations

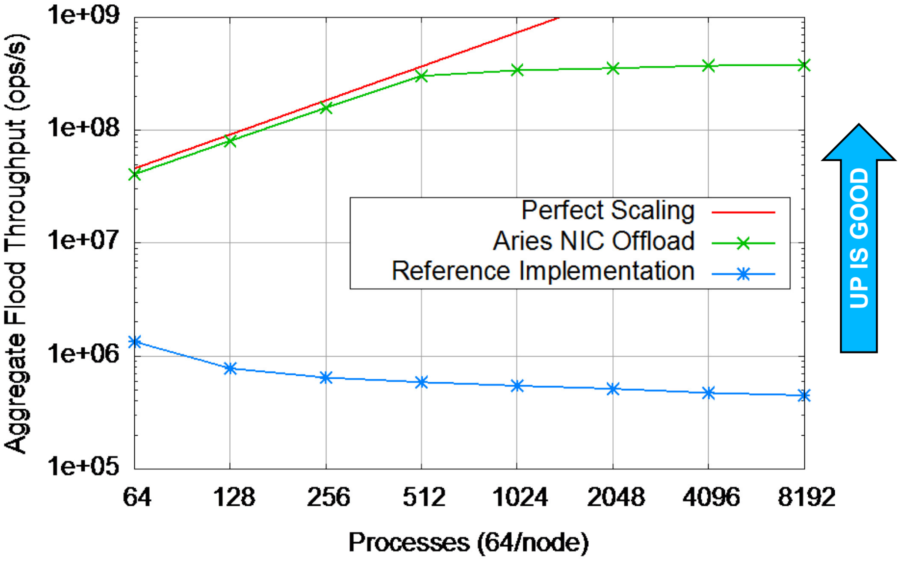
- "Remote Atomics" is a new family of RMA interfaces
  - Analogous to MPI accumulate operations
- Interface designed with NIC hardware offload in mind
- Uses the "atomics domain" concept
  - Introduced in UPC 1.3 spec
  - Enables efficient offload, even in the presence of concurrent updates to the same location using multiple distinct operations

Remote Atomic Memory Operations

- Offload reduces latency of fetch-and-add by **70%** relative to generic AM-based reference

- Figure shows aggregate throughput of a “hot-spot” test of fetch-and-add (all to one)
- Green series shows robust scaling to saturation when offloaded to the Aries NIC

Scaling of a remote atomics “hot-spot” test in the Cray Aries network



Negotiated-Payload Active Messages (NPAM)

- "Negotiated-Payload" is a new family of AM interfaces
  - Splits AM injection into distinct Prepare and Commit phases
  - Client and GASNet can negotiate the buffer size and ownership
- Main use case:
  - Client can assemble payload directly into outgoing buffer
  - Removes critical-path payload **`memcpy`** for some patterns

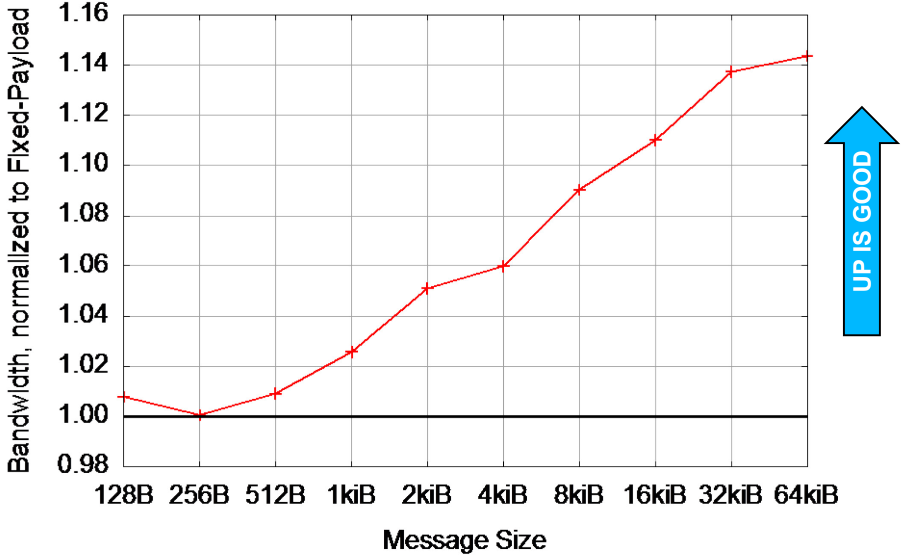
```
// Fixed-Payload code, for which most conduits require a memcpy to an internal buffer:
assemble_payload(client_buf, len);  // writes client-owned memory
gex_AM_RequestMedium1(team, rank, handler, client_buf, len, GEX_EVENT_NOW, flags, arg);

// Negotiated-Payload avoids the memcpy via payload assembly into a GASNet-owned buffer:
gex_AM_SrcDesc_t sd = gex_AM_PrepareRequestMedium(team, rank, NULL, len, len, NULL, flags, 1);
assemble_payload(gex_AM_SrcDescAddr(sd), len);  // writes GASNet-owned memory
gex_AM_CommitRequestMedium1(sd, handler, len, arg);
```

Negotiated-Payload Active Messages

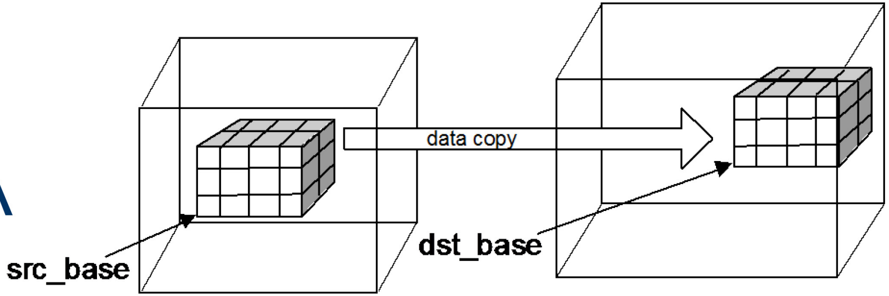
- Figure shows an AM ping-pong bandwidth benchmark using the memcpy-removal pattern on the previous slide
- Normalized to the Fixed-Payload performance
- Shows NPAM implementation for Cray Aries network delivering up to a 14% improvement

Aries-conduit NPAM speedup on a ping-pong test with dynamically-generated payload



ARGONNE NATIONAL LABORATORY GASNet-EX BERKELEY LAB

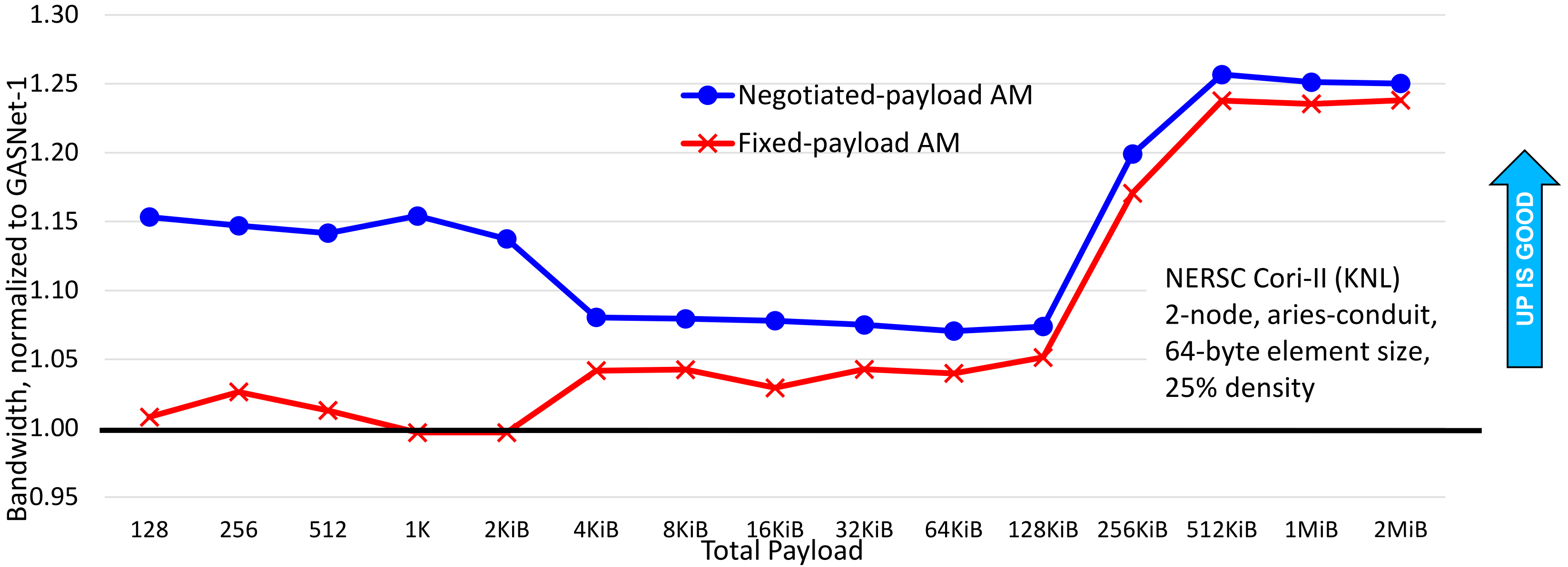
Non-Contiguous RMA



- GASNet APIs for sparse RMA Vector-Indexed-Strided
  - Express non-blocking Put and Get of non-contiguous data
  - Names reflects the three metadata formats
    - Different trade-offs between size and generality
  - EX provides some improvements to this GASNet-1 extension
- Implementation uses Active Messages, when appropriate, for pack/unpack of data
  - Benefits from reimplementation using Negotiated-Payload AM

Non-Contiguous RMA

Improved Inter-node Strided Put performance, relative to GASNet-1

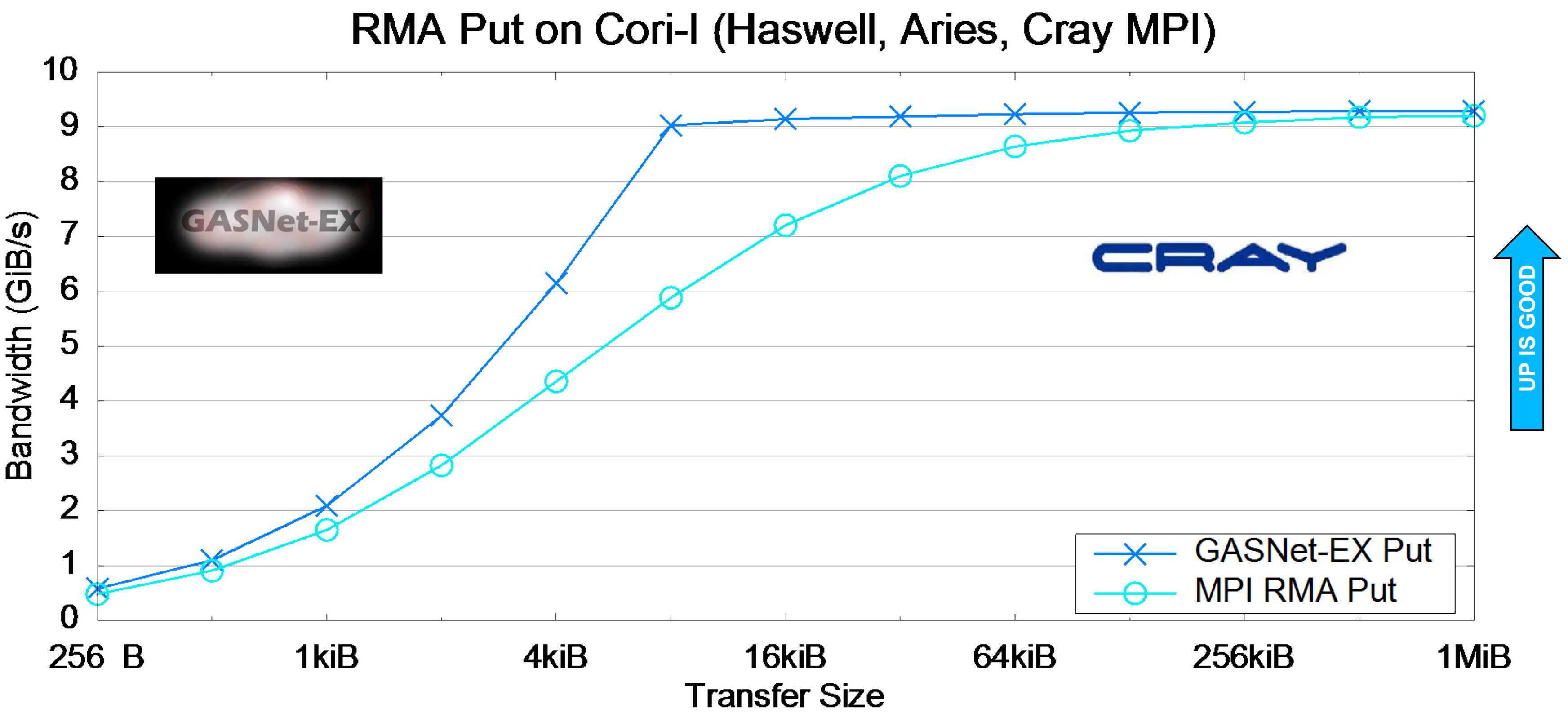


- Red series shows performance using Fixed-Payload AM
- Blue series shows performance using Negotiated-Payload AM
- Both normalized to GASNet-1

RMA Bandwidth Microbenchmarks

- Measured unidirectional flood bandwidth and RMA latency between two nodes, one process per node.
- Intel MPI Benchmarks v2018.1 to measure MPI-3 RMA
  - IMB-RMA test, Unidir_put and Unidir_get subtests
  - "Aggregate" result category reports bandwidth of
    - Series of many **`MPI_Put`** (or **`Get`**) operations
    - A single final call to **`MPI_Win_flush`**
  - All within a passive-target access epoch established by a call to **`MPI_Win_lock(SHARED)`** *outside* the timed region
- GASNet-EX measures nearest semantic equivalent

RMA Bandwidth Microbenchmarks



NERSC BERKELEY LAB

RMA Bandwidth Microbenchmarks

- Showing results collected on four platforms
  - Three different MPI implementations (Cray, IBM and MVAPICH2)
  - Two distinct networks (Cray Aries and Mellanox EDR InfiniBand)
  - Three CPU families (Xeon Haswell, Xeon Phi, and POWER9)
  - Complete details are given in:

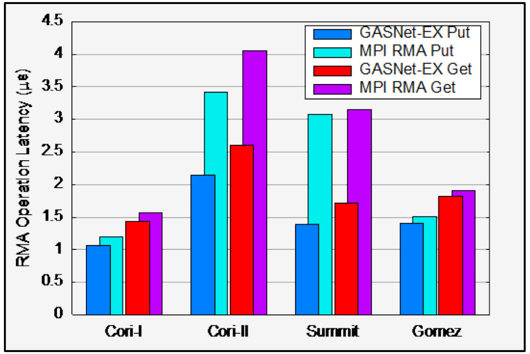
  LCPC’18: Bonachea, Hargrove. "GASNet-EX: A High-Performance, Portable Communication Library for Exascale", https://doi.org/10.25344/S4QP4W

- Results are collected in “out of the box” configurations
  - Used center’s defaults on the three production systems
  - No non-default tuning knobs used to improve performance

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

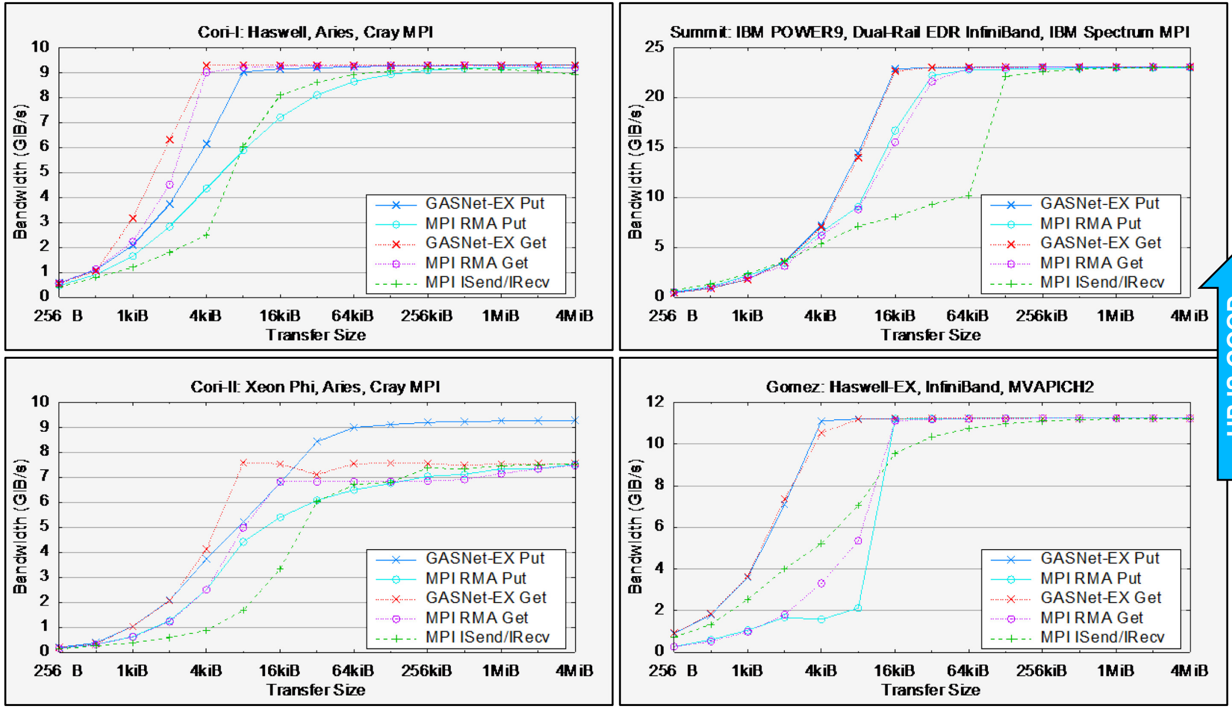
- Three different MPI implementations
- Two distinct network hardware types
- On four systems the performance of GASNet-EX matches or exceeds that of MPI RMA and message-passing:
  - 8-byte Put latency 6% to 55% better
  - 8-byte Get latency 5% to 45% better
  - Better flood bandwidth efficiency, typically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

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



DOWN IS GOOD

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



UP IS GOOD

GASNet-EX results from v2018.9.0 and v2019.6.0. MPI results from Intel MPI Benchmarks v2018.1.
For more details see LNCS 11882: Proceedings of Languages and Compilers for Parallel Computing (LCPC).
doi: 10.1007/978-3-030-34627-0_11

More recent results on Summit here replace the paper's results from the older Summitdev.

GASNet-EX

IBM Spectrum MPI

MVAPICH

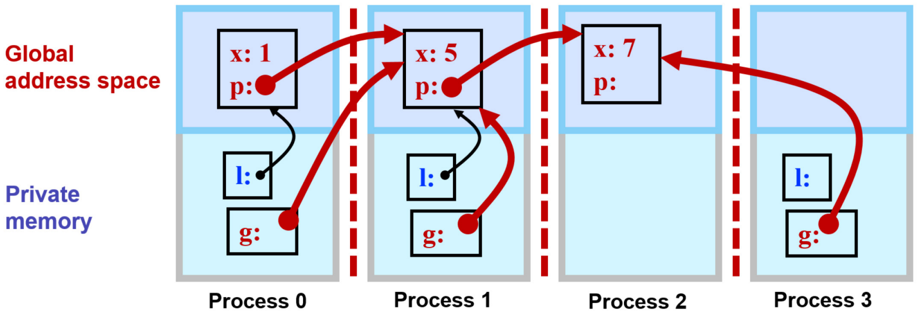
BERKELEY LAB

Overview of UPC++ Features

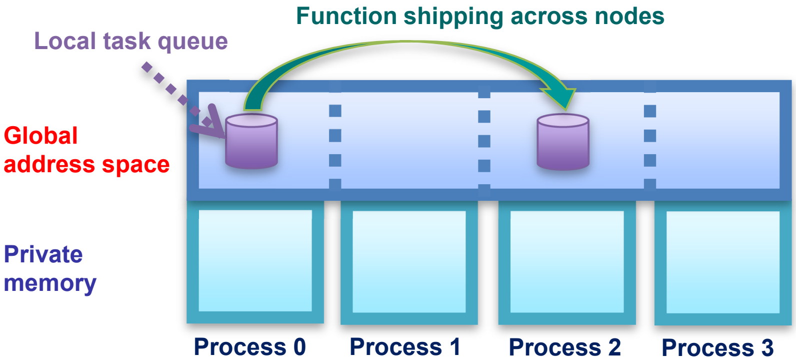
- One-sided RMA and Global Pointers
- Remote Procedure Calls (RPC)
- Future-based asynchrony, continuations
- Personas (multi-threading)
- Remote atomics, distributed objects
- Hierarchical shared mem, node-level bypass
- Teams and collectives
- Serialization, non-contiguous transfers
- Memory Kinds for GPU support
- Interoperability with other models

Details: https://upcxx.lbl.gov

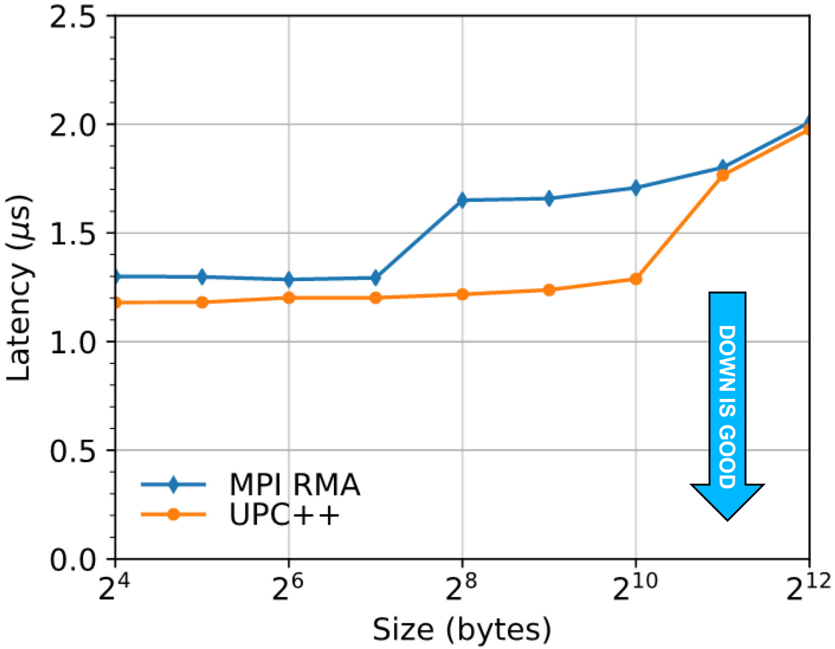
RMA and Global Pointers



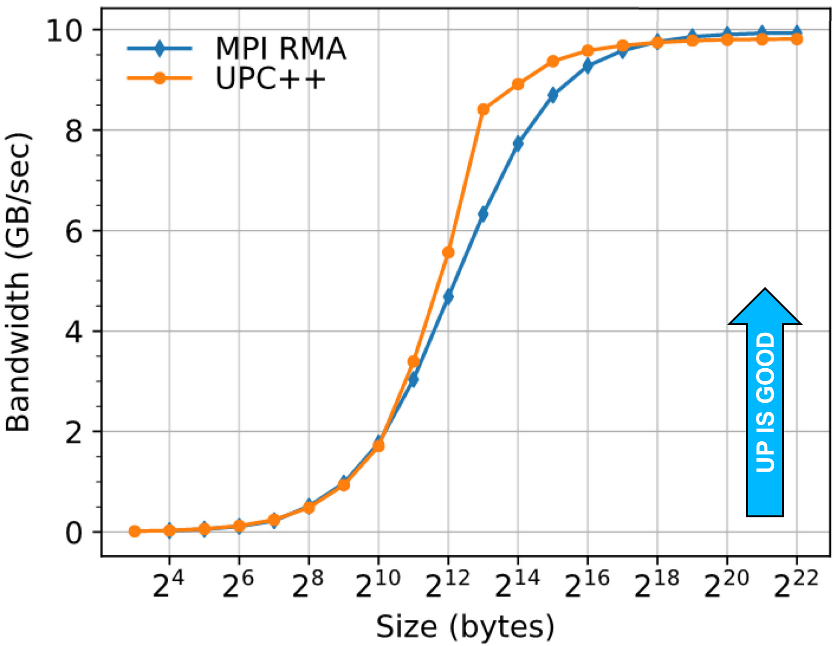
Remote Procedure Calls



UPC++ RMA Performance over GASNet-EX



Round-trip Put Latency
(lower is better)



Unidirectional Flood Put Bandwidth
(higher is better)

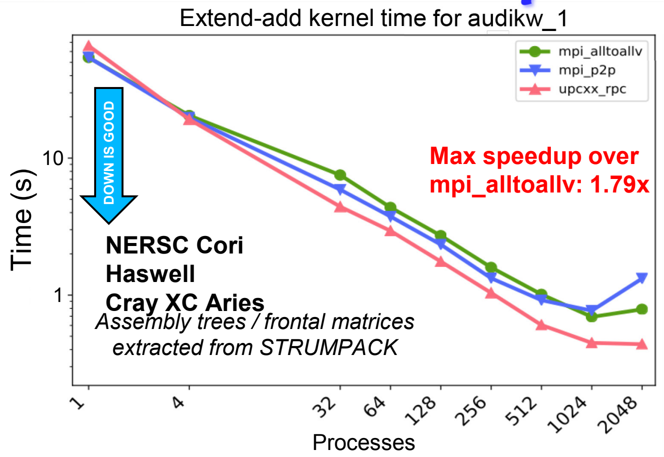
Data collected on Cori Haswell
MPI-RMA results from Cray MPI and Intel MPI benchmarks
Details: https://doi.org/10.25344/S4V88H

Sparse Matrix Solvers in UPC++

Sparse multifrontal direct linear solver: **Extend-add** kernel

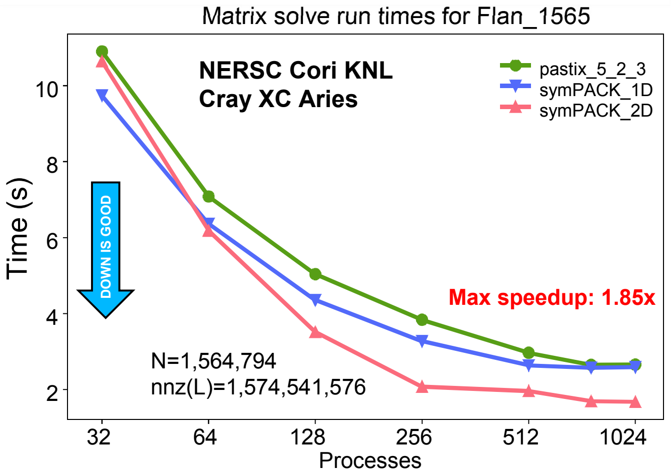
- Important building block for **multifrontal sparse solvers**
- 1.63x/1.79x speedup over best MPI version on cori KNL/Haswell
- Details in IPDPS'19 paper:

  Bachan, Baden, Hofmeyr, Jacquelin, Kamil, Bonachea, Hargrove, Ahmed. "UPC++: A High-Performance Communication Framework for Asynchronous Computation", https://doi.org/10.25344/S4V88H



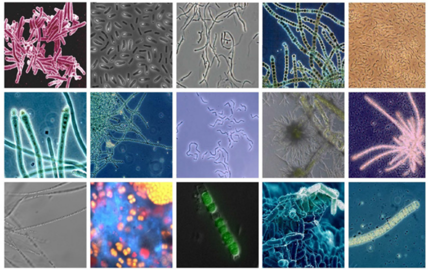
symPACK: a solver for sparse symmetric matrices

- Implemented entirely in UPC++
- UPC++ enables exploiting a finer-granularity task graph
  - RPC naturally expresses a pull-based model for matrix blocks
  - Simplifies resource management and improves strong scalability
- Competitive with state-of-the-art solvers
- Details: https://upcxx.lbl.gov/sympack



Work and results by Mathias Jacquelin, funded by SciDAC CompCat and FASTMath

UPC++ Applications: ExaBiome



ExaBiome: Exascale Solutions to Microbiome Analysis (LBNL, LANL, JGI)

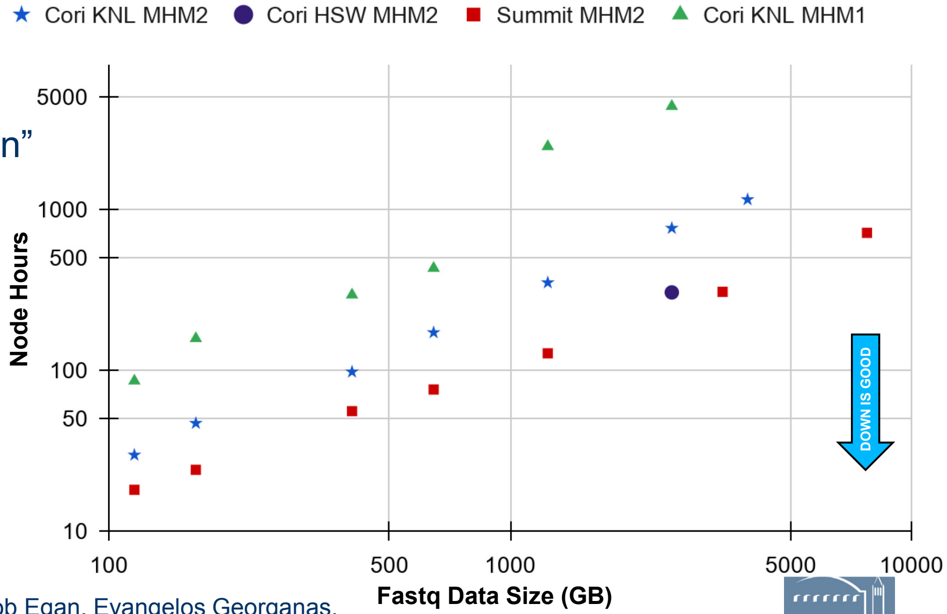
- MetaHipMer is a distributed-memory metagenome assembler

MetaHipMer v2 (MHM2) rewritten *entirely* in UPC++ (released Sept 2020)

- previous versions used UPC, UPC++ and MPI, followed by just UPC and UPC++
- **4x reduction in code size** (60 kLOC ➔ 15 kLOC)
- Runs **2x to 10x faster** than MHM1
  and uses **2x less memory**
- “pure UPC++ is more portable and easier to install/run”
- Supports GPUs on Summit/Cori for alignment kernel

Enabling **new science in genomic analysis**

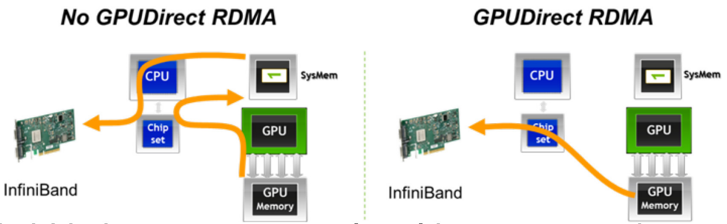
- The 7.7TB assembly on Summit is by far the largest metagenome ever assembled
- Projected lower-bound for 50TB dataset is 4650 node hours on Summit
  - i.e. full Summit machine for one hour



Steve Hofmeyr, Rob Egan, Evangelos Georganas,
leads on MetaHipMer software

JGI
Los Alamos NATIONAL LABORATORY EST.1943
BERKELEY LAB

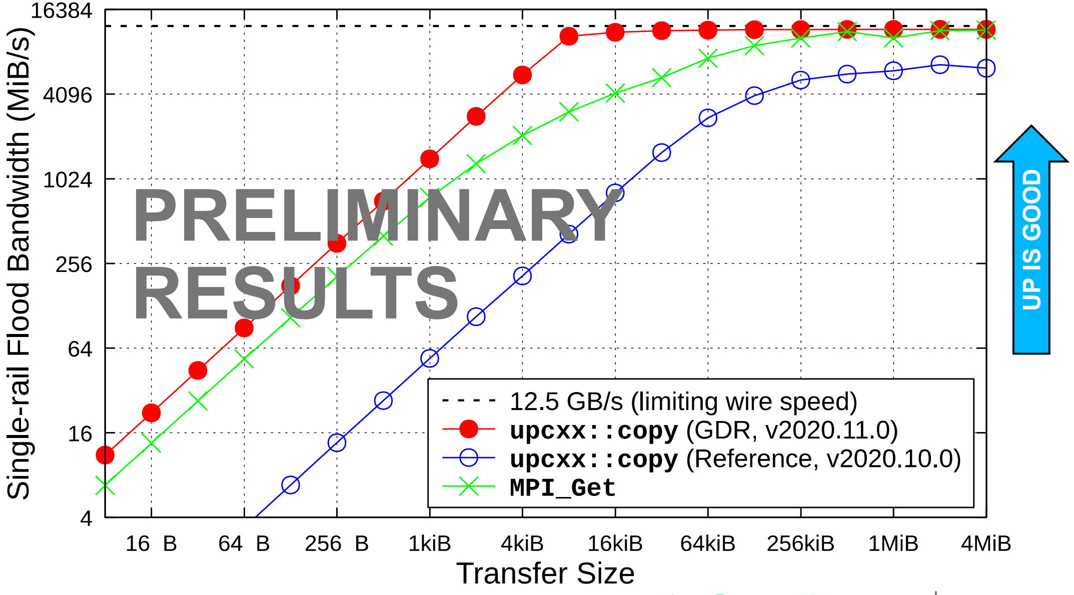
RMA to/from GPU Memory



- Measurements of flood bandwidth of **`upcxx::copy()`** on Summit
- Difference between the two most recent releases shows benefit of GASNet-EX's new support for GPUDirect RDMA (GDR)
  - No longer staging through host memory
  - Large xfers: 2x better bandwidth
  - Small xfers: up to 30x better bandwidth
- Get operations to/from GPU memory now perform comparably to host memory
- Preliminary comparisons to MPI-3 RMA in GDR-enabled IBM MPI show UPC++ saturating more quickly to the peak

RMA Get Bandwidth (remote GPU to local host memory)

UPC++ 2020.11.0 vs. IBM Spectrum MPI 10.3.1.2 on OLCF Summit



UPC++ results were collecting 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.

GASNet-EX in Legion Programming System

Realm is Legion's low-level runtime, providing comm. services

- Originally implemented over GASNet-1
- Still works using legacy API support in current GASNet-EX

Realm introduced a new GASNet-EX backend (Dec 2020)

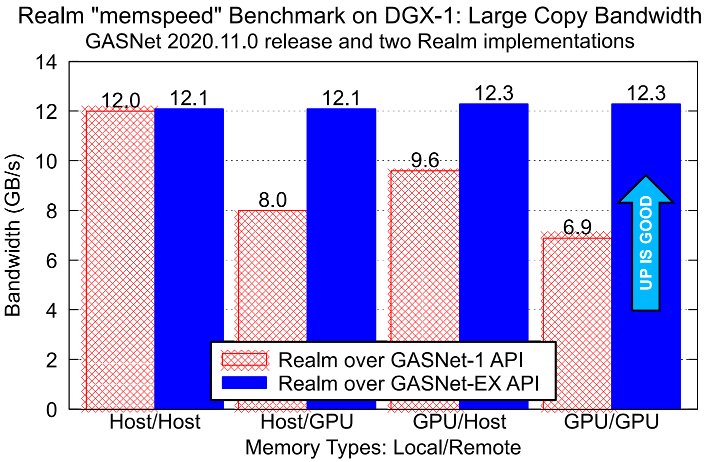
- Embraces EX-specific capabilities
- Leverages Immediate, NPAM, and LC handles for AM
- Most notable new capability is recently released GDR support

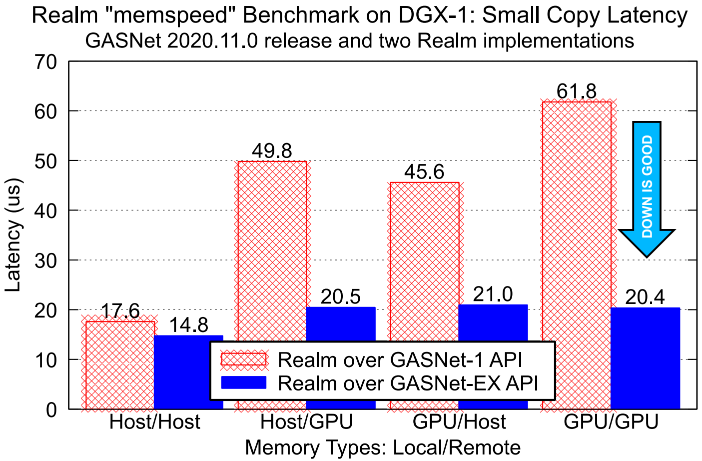
Figures illustrate some performance benefits of using new GDR support in GASNet-EX:

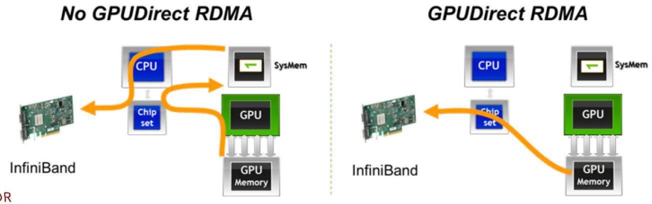
- Large GPU memory xfers: same bandwidth as host mem
- Small GPU memory xfers: 2.2x to 3.0x latency improvement

Multi-endpoint allows RDMA for *all* Realm-allocated host buffers

- Avoids copies needed with the GASNet-1 API







Conclusions

- GASNet-EX is the next generation of GASNet, addressing needs of newer programming models
  - Asynchrony, adaptivity, threading, scalability, device memory, ...
- In production use by UPC++, Legion, Chapel and others
- Provides backward compatibility for GASNet-1 clients
- Benefits of new features are already measurable
- Delivers RMA performance competitive with MPI-RMA

THANK YOU!

gasnet-staff@lbl.gov

gasnet.lbl.gov

upcxx.lbl.gov

LCPC'18: Bonachea, Hargrove.
"GASNet-EX: A High-Performance, Portable Communication Library for Exascale",
https://doi.org/10.25344/S4QP4W

IPDPS'19: Bachan, Baden, Hofmeyr, Jacquelin, Kamil, Bonachea, Hargrove, Ahmed.
"UPC++: A High-Performance Communication Framework for Asynchronous Computation",
https://doi.org/10.25344/S4V88H