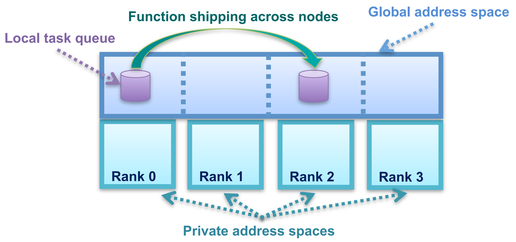
UPC++ and GASNet-EX: PGAS Support for Exascale Apps and Runtimes

Scott B. Baden (PI), Paul H. Hargrove (co-PI),
Hadia Ahmed, John Bachan, Dan Bonachea, Steve Hofmeyr, Mathias Jacquelin, Amir Kamil, Brian van Straalen

UPC++ at Lawrence Berkeley National Lab (http://upcxx.lbl.gov)

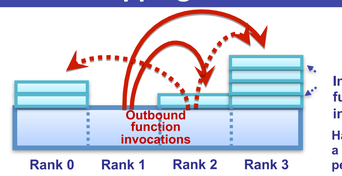


- UPC++ is a C++11 PGAS library
  - Lightweight, asynchronous, one-sided communication
  - Asynchronous remote function execution (function shipping)
  - Data transfers may be non-contiguous
  - Futures manage asynchrony, enable communication overlap
  - Collectives, teams, remote atomic updates
  - Distributed irregular data structures
- Easy on-ramp and integration
  - Interoperable with MPI+OpenMP/CUDA etc.
  - Enables incremental development
  - Replace performance-critical sections with lightweight PGAS
- Latest software release: September 2018
  - Runs on systems from laptops to supercomputers

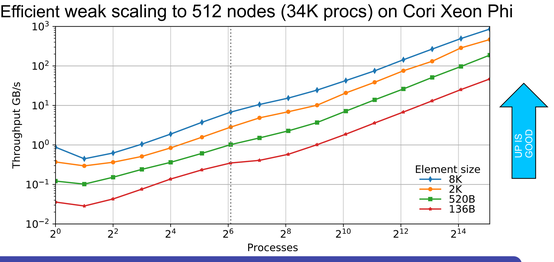
Case 1: Easy Distributed Hash-Table via Function Shipping and Futures

- **Function shipping** via **RPC** simplifies distributed data-structure design
  - RPC inserts the key metadata at the target
  - Once the RPC completes, an attached callback issues a one-sided rput to store the value data
- **Benefits:**
  - Key insertion and storage allocation handled at the target
  - Asynchronous execution enables communication-computation overlap

```
// 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) {
  return 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;
             }).then(                // callback executes when RPC completes
               [val,sz](global_ptr<char> loc) -> future<> {
                 return rput(val, loc, sz); }); // RMA put the value payload
}
```

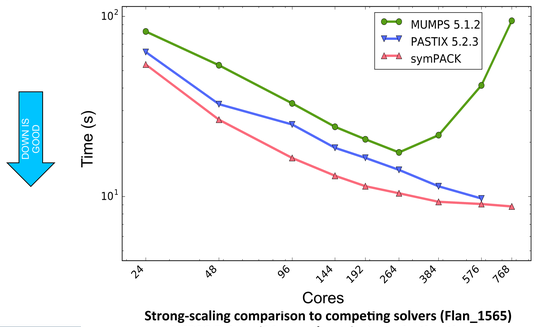


Efficient weak scaling to 512 nodes (34K procs) on Cori Xeon Phi



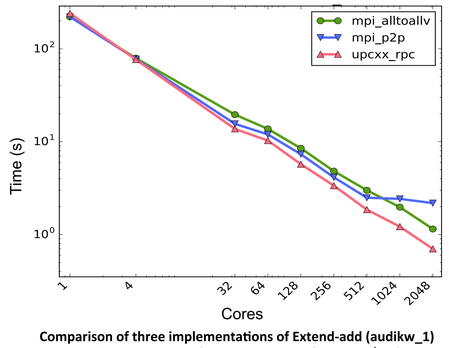
Case 2: Asynchronous Sparse Matrix Solvers

- **Solvers:**
  - ***symPACK***, a direct linear solver for sparse symmetric matrices
  - ***Extend-add*** proxy application, a critical step of multifrontal solvers
- **Challenges:**
  - Sparse matrix factorization has low computational intensity and irregular communication
- **Solution**:
  - UPC++ **function shipping** enables efficient pull communication and event-driven scheduling in ***symPACK***, and better overlap and performance in the ***Extend-Add*** operation.



Strong-scaling comparison to competing solvers (Flan_1565) on NERSC Edison (24 cores/node) - factorization time only

- **Impact:**
  - On average, ***symPACK*** delivers a 2.65x speedup over a state-of-the-art sparse symmetric solver
  - UPC++'s one-sided pull strategy avoids the need for (and cost of) unexpected messages in MPI.
  - On ***Extend-add***, the increased overlap exposed by UPC++ yields up to a 1.63x speedup over MPI collective and 3.11x over MPI message-passing implementations.



Comparison of three implementations of Extend-add (audikw_1) on NERSC Cori Xeon Phi – using 64 out of 68 cores/node

GASNet-EX at Lawrence Berkeley National Lab (http://gasnet.lbl.gov)

- GASNet-EX: communications middleware to support exascale clients
  - One-sided communication – Remote Memory Access (RMA)
  - Active Messages - remote procedure call
  - Implemented over the native APIs for 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)
  - Provides backwards compatibility with GASNet-1 clients
  - Focus remains on one-sided RMA and Active Messages
  - Reduces CPU and memory overheads
  - Improves many-core and multi-threading support
- GASNet-1 clients include:
  - Multiple UPC and CAF/Fortran08 compilers
  - Stanford's Legion Programming System
  - Cray Chapel Language
  - OpenSHMEM Reference Implementation
  - Omni XcalableMP Compiler
- GASNet-EX clients include:
  - UPC++, Legion, and Cray Chapel
  - PaRSEC and ExaBiome (exploring)
- GASNet-EX augments and enhances GASNet-1
  - Enhancements address needs of modern asynchronous PGAS models
  - Interfaces adjusted for improved scalability
  - Features critical to UPC++ are being co-designed
  - Using input from Legion and Cray Chapel, who are adopting the new APIs
- Current enhancements:
  - "Immediate mode" injection to avoid stalls due to back-pressure
  - Explicit handling of local-completion (source buffer lifetime)
  - New AM interfaces, e.g. to reduce buffer copies between layers
  - Vector-Index-Strided for non-contiguous point-to-point RMA
  - Remote Atomics, implemented with NIC offload where available
  - Teams and non-blocking collectives
- Future enhancements may include:
  - Dependent operations to control ordering of in-flight operations
  - Offset-based addressing
  - Multiple endpoints/segments, e.g. to enhance multithreading support
  - Support for "out-of-segment" remote addresses
  - Communication directly to/from device memory (e.g. GPUDirect)

Highlights from Recent Work

Example of EX interface updates: RMA Put

- 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);
```

- **`gex_Event_t`** return type introduces events to generalize GASNet handles.
- **`tm`** argument adds team (ordered sets of ranks), into which **`rank`** indexes.
- **`gex_Addr_t`** type will enable offset-based addressing via same interface.
- **`lc_opt`** argument introduces explicit control over local completion, generalizing the bulk/non-bulk interfaces of GASNet-1.
- **`flags`** argument provides extensibility. For instance to:
  - Select optional behaviors (e.g., immediate mode and offset-based addressing)
  - Provide assertions regarding arguments (e.g., to streamline the operation)

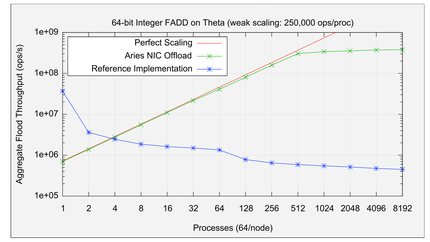
Vector-Indexed-Strided (VIS) Interfaces for Non-Contiguous RMA

- Formalizes and generalizes an unofficial extension to GASNet-1
- Three metadata formats for different scenarios
  - Vector: fully general array of iovec-like (address, length) pairs
  - Indexed: array of addresses and a single length
  - Strided: arbitrary rectangular sections of dense multi-dimensional arrays
    - GASNet-EX adds transposition and reflection capabilities
- This table shows the speed-up resulting from use of aggressive pack/unpack optimizations. Each datum is the mean of the bandwidth ratios for a sweep over a range of transfers representing reasonable inputs to the VIS functions.

SYSTEM	NETWORK	INDEXED		STRIDED		VECTOR	
		GET	PUT	GET	PUT	GET	PUT
Cori-I	Cray Aries	11.68 ×	10.06 ×	12.55 ×	12.63 ×	8.83 ×	7.69 ×
Theta	Cray Aries	10.03 ×	7.70 ×	11.10 ×	9.94 ×	7.13 ×	5.89 ×
Titan	Cray Gemini	7.33 ×	7.21 ×	8.09 ×	8.61 ×	5.33 ×	5.51 ×
SummitDev	Mellanox InfiniBand	5.45 ×	5.17 ×	5.67 ×	5.63 ×	4.29 ×	4.29 ×
Cetus	IBM BG/Q	2.66 ×	3.49 ×	4.01 ×	4.34 ×	2.10 ×	2.82 ×

Speed-up resulting from communication aggregation using VIS

Remote Atomics with Cray Aries NIC Offload



- Implements the Atomic Domains concept (first introduced by UPC 1.3)
  - Domains permit use of NIC offload even when not coherent with CPU
  - Domains are created collectively outside the critical path
  - A Domain has an associated data type and set of allowed operations
    - Domains select the best implementation for the data type and ops
    - e.g. use offload if and only if NIC implements **all** the requested ops
- Example: non-blocking atomic fetch-and-add (FADD) on unsigned 64-bit integer

```
gex_Event_t ev = // *result = ATOMICALLY( *target += addend )
gex_AD_OpNB_U64(domain, &result, target_rank, target_address,
                GEX_OP_FADD, addend, 0 /*unused op2*/, flags);
```

- **`flags`** includes optional behaviors and assertions, such as memory fences
- GASNet-EX provides a network-independent "reference implementation"
  - Uses Active Messages to perform operations using the target CPU
  - Uses GASNet-Tools for atomicity (inline assembly for numerous CPUs)
- Specialization for Cray Aries improves performance vs. reference implementation
  - Reduces latency of inter-node FADD from 4.9us to 2.8us
  - Greatly increases throughput under contention
- The figure above shows throughput of 1 to 8192 processes (64 per node) performing pipelined FADD of a central counter (measured on ALCF's Theta).

GASNet-EX RMA Flood Bandwidth versus MPI-3 RMA and Isend/Irecv

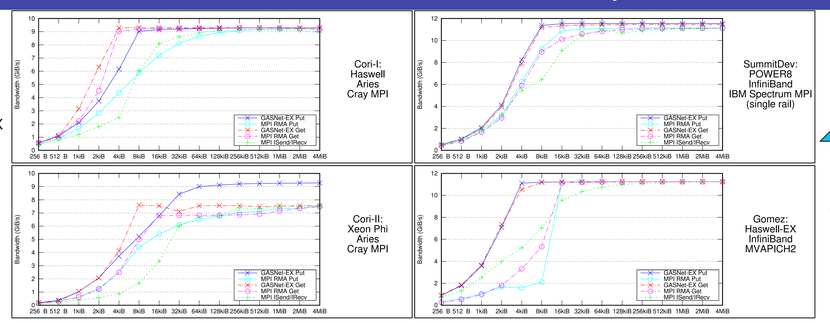
Flood Bandwidth Comparison

- GASNet-EX RMA versus MPI-3 RMA and message-passing
- Three different MPI implementations
- Two distinct network hardware types

On four systems the performance of GASNet-EX matches or exceeds that of MPI-3 RMA and message-passing.

For complete details see:
Bonachea, Dan and Hargrove, Paul H. (2018): **GASNet-EX: A High-Performance, Portable Communication Library for Exascale**.
To appear in: Languages and Compilers for Parallel Computing (LCPC'18).
`https://doi.org/10.25344/S4QP4W`

(Intel MPI Benchmarks v2018.1 and GASNet-EX v2018.9.0)



Uni-directional Flood Bandwidth across Transfer Sizes

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.