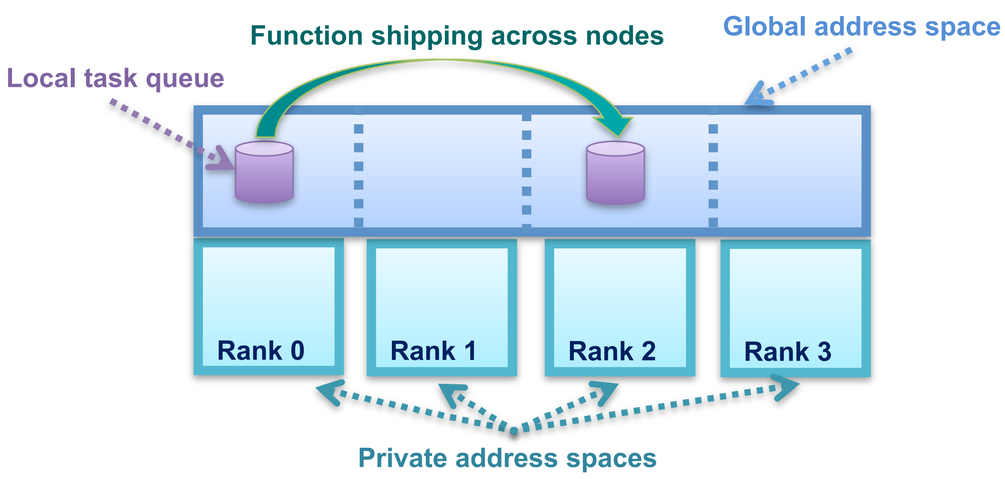
UPC++: Asynchronous RMA and RPC Communication for Exascale Applications

WBS 2.3.1.14

Team Members: Amir Kamil, John Bachan, Dan Bonachea, Paul H. Hargrove, Erich Strohmaier and Daniel Waters

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

- UPC++ is a C++11 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: September 2019
  - 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)**
  - ExaBiome's HipMer 1.2.1 release (September 2019)
    - The k-mer counting step rewritten from MPI to UPC++
    - UPC++ RPC is a better fit to the problem
    - Reduces code size by roughly ½
      - More readable and maintainable
    - Lower memory requirements and better scaling
  - Current work-in-progress
    - Previous UPC stages of HipMer pipeline rewritten in UPC++
    - Approximately 85% reduction in code size
    - ExaBiome team's initial results:
      - Comparable genome assembly results
      - Lower memory requirements and better performance
- **Integration efforts with ExaGraph (WBS 2.2.6.07)**
  - Worked with PNNL team to develop 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 execute logic on remote parts of the graph
  - Initial 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
    - At 4,096 processes: UPC++ RMA version is 4.3x faster than best MPI-3 (RMA+neighborhood collective) version

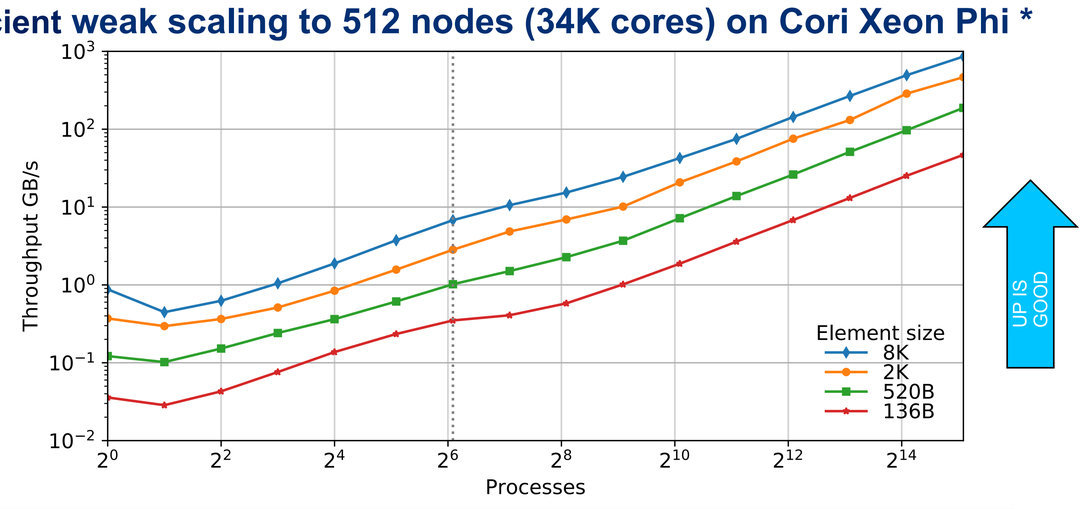
Case 1: Easy Distributed Hash-Table via Function Shipping and Futures

- **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
}
```

- **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

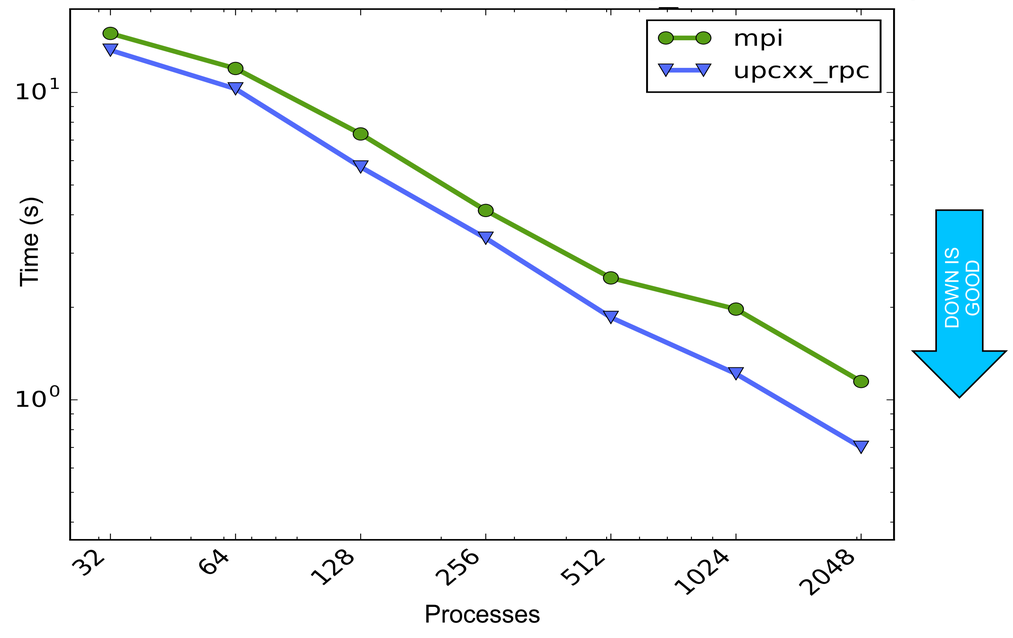
Efficient weak scaling to 512 nodes (34K cores) on Cori Xeon Phi *



Case 2: Asynchronous Sparse Matrix Solvers

- **A time consuming operation in multifrontal sparse solvers:**
  - ***Extend-add:*** update a distributed sparse matrix, scattering the packed data source
- **Challenge:**
  - This operation has low computational intensity and exhibits irregular communication patterns
- **Solution**:
  - UPC++ **function shipping** via RPC enables efficient communication and asynchrony, increasing overlap and improving performance of ***Extend-add***
- **Impact:**
  - UPC++ enhances overlap in ***Extend-add***, yielding up to a 1.63x speedup over MPI collective and 3.11x over MPI message-passing implementations. The green line in the figure corresponds to the fastest of these two variants.

Strong scaling comparison of the UPC++ implementation of Extend-add using RPC and an MPI variant for the audikw_1 matrix on NERSC Cori Xeon Phi (using 64 cores/node) *



*** For more details see IPDPS'19. https://doi.org/10.25344/S4V88H**

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.