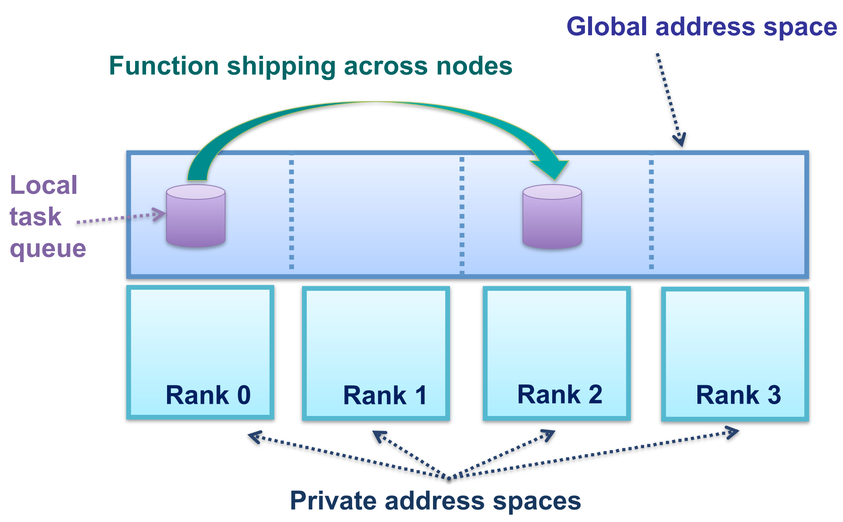
UPC++ and GASNet: PGAS Support for Exascale Apps and Runtimes

Scott B. Baden (PI) and Paul Hargrove (co-PI) 2.3.1.14

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

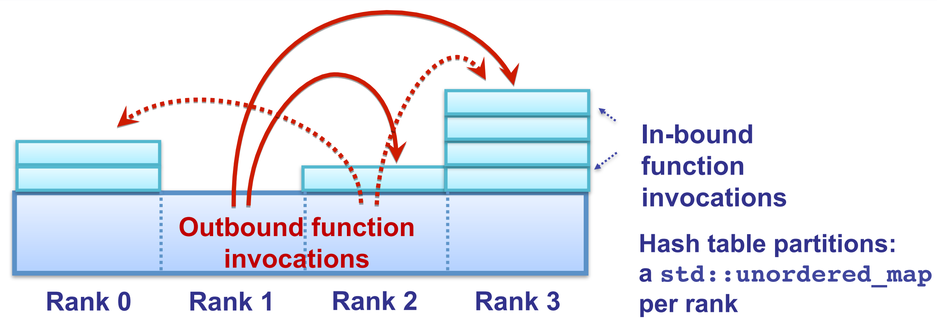


- UPC++ is a C++11 library
  - Lightweight, asynchronous, PGAS one-sided communication
  - Asynchronous remote function execution (function shipping)
  - Data transfers may be non-contiguous
  - Futures manage asynchrony, enable communication overlap
  - Collectives, teams, remote atomic updates
  - 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: Jan 2018
- Runs on systems from laptops to supercomputers

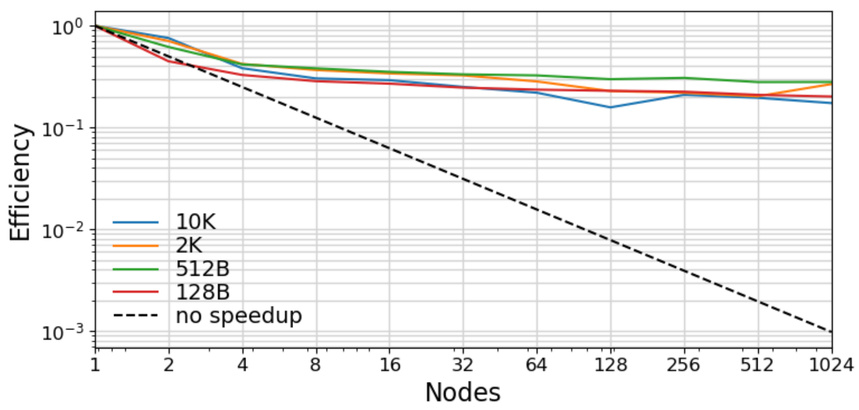
Case 1: Easy distributed hash-table via function shipping and futures

- **Function shipping** via **RPC** simplifies distributed data-structure design
  - RPC inserts the key meta data at the target
  - Once the RPC completes, a callback attached to the RPC uses a one sided rput to store the associated 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<> { // lambda RMA put
                 return rput(val, loc, sz); });
}
```



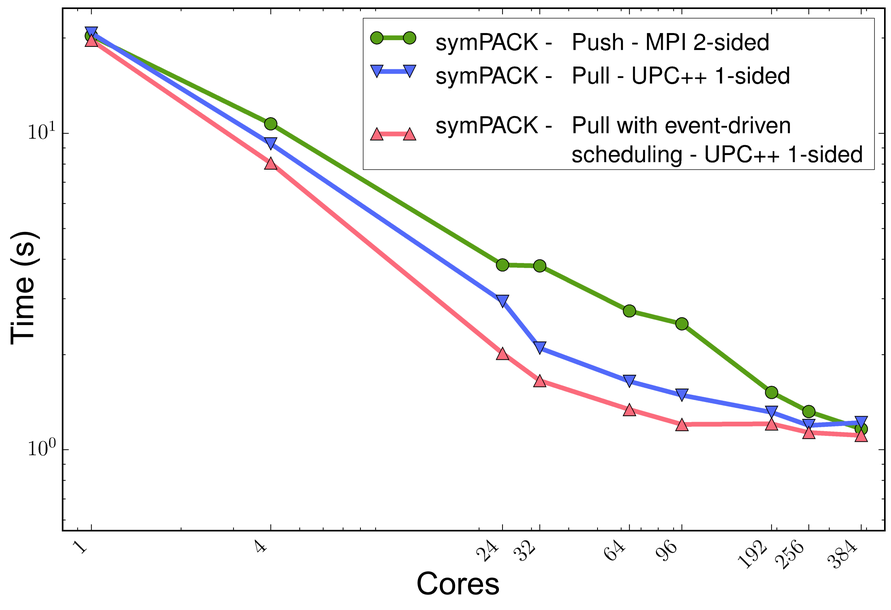
- Efficient weak scaling from 4 nodes onwards (Cori KNL)



Case 2: symPACK: UPC++ asynchronous task-based sparse Cholesky solver

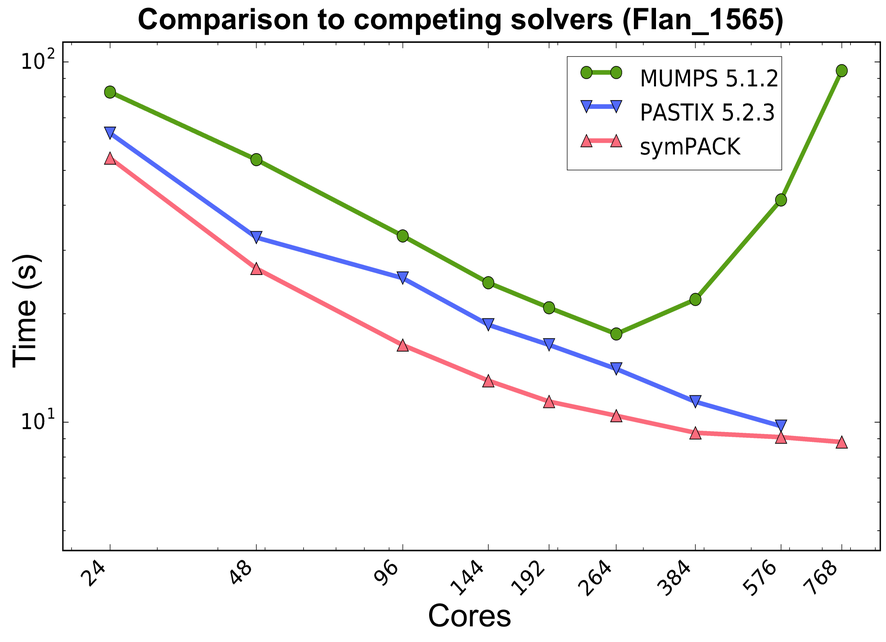
- **Application:** ***symPACK***, a parallel direct linear solver for sparse symmetric matrices
- **Challenges:** Sparse matrix factorizations have low computational intensity and irregular communication patterns
- **Solution***:* UPC++ **function shipping** enables an efficient pull communication strategy and event-driven scheduling
- **Impact***:* on average*,* ***symPACK*** delivers a ×2.65 speedup over the best state-of-the-art sparse symmetric solver (Results on Edison) UPC++'s one-sided pull strategy avoids the need for (and cost of) unexpected messages in MPI

Impact of communication strategy (boneS10)



Push – MPI 2-sided communication
Pull – UPC++: RPC + RMA Get when ready
2 variants: with and without event driven scheduling

Comparison to competing solvers (Flan_1565)



Strong scaling of symmetric solvers
(factorization time only)