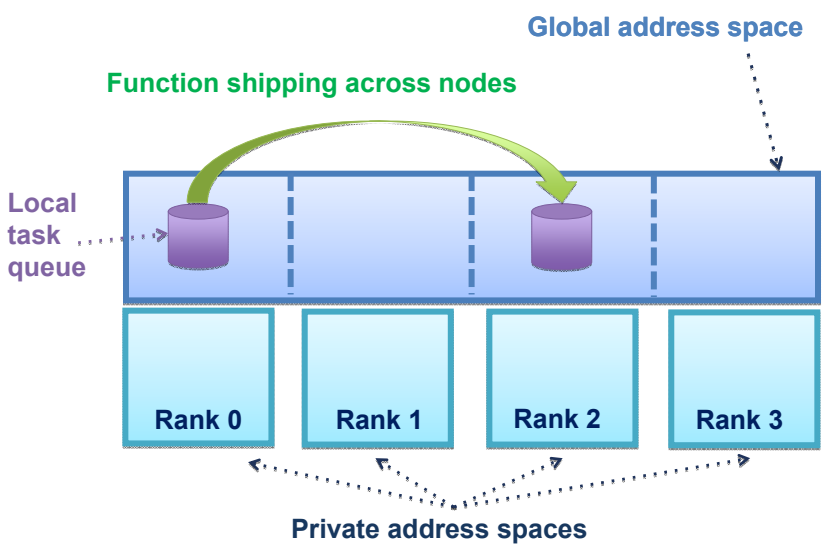
UPC++: a PGAS C++ Library

http://upcxx.lbl.gov

Execution model and PGAS interface

UPC++ provides PGAS-style lightweight one-sided communication and asynchronous task execution features to C++ applications



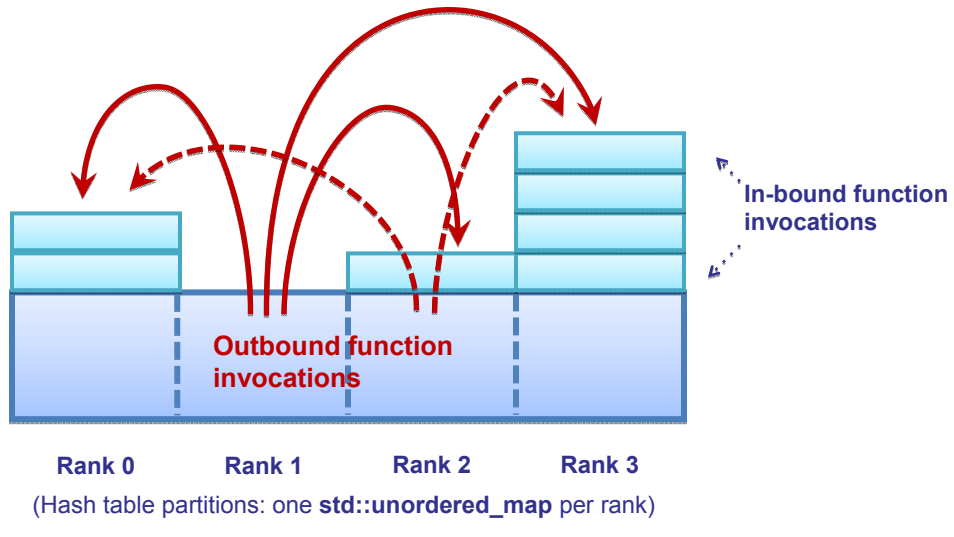
- Easy on-ramp for applications
  - A C++11 library
  - Compatible with existing MPI+OpenMP/CUDA code bases
- All data motion is asynchronous
  - Futures and continuations to manage overlap
  - RMA operations for direct access to remote shared data
  - Co-processor memory support
- Supports distributed irregular data structures used in adaptive mesh refinement, sparse solvers, graph algorithms
  - Remote Procedure Calls (RPC)
  - Distributed objects
  - Non-contiguous RMA communication
  - Remote atomics
- Teams and collectives

Easy distributed hash-table via remote procedure call and futures

- **Remote Procedure Calls** simplify distributed data-structure design.
  - Use ***rpc*** to ship updates to the key's owning rank.
  - One-sided nature avoids tedious work of declaring expected messages as is typical with two-sided messaging.
- **Futures** hide the latency of remote operations, naturally express overlap of independent operations.

```
// c++ "global" variables become rank-local state.
std::unordered_map<int, int> my_dht_local;

// owner does the work, result is a future<int>
upcxx::future<int> dht_fetch_inc (int key) {
  return upcxx::rpc(                          // `rpc` sends lambda to rank
    key % upcxx::rank_n(),                    // owner rank in key-to-rank partition
    [=]() { return my_dht_local[key]++; }     // [=] captures `key`, used remotely
  );
}
```



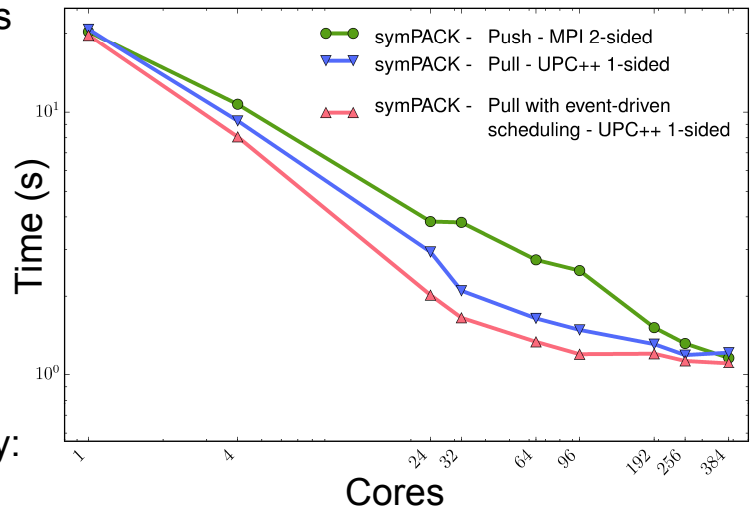
(Hash table partitions: one **std::unordered_map** per rank)

symPACK: UPC++ asynchronous task-based sparse symmetric solver

- **Application: *symPACK***, a sparse direct linear solver for symmetric matrices.
- **Challenges:** Sparse matrix factorizations have low computational intensity and irregular communication patterns.
- **Solution***:* UPC++ **rpcs** and **rgets** enable 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. UPC++ enables a one-sided pull strategy: RPC avoids the need (and cost) of scheduling messages in MPI, to avoid deadlocking on many small messages.

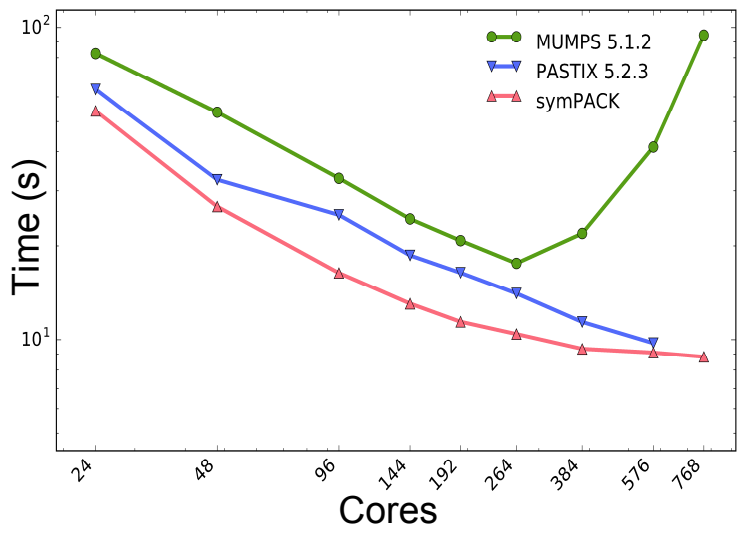
Strong Scaling on NERSC Edison - Cray XC30 (24 cores/node)

Impact of communication strategy (boneS10)



Push – MPI 2-sided communication
Pull – UPC++ 1-sided communication with/without event driven scheduling

Comparison to competing solvers (Flan_1565)



Strong scaling of symmetric solvers (factorization time only)

Partners and acknowledgements

Pagoda Team

Scott B. Baden, Paul H. Hargrove
John Bachan, Dan Bonachea, Steven Hofmeyr,
Mathias Jacquelin, Amir Kamil, Brian van Straalen

UPC++ is part of the
LBNL Pagoda Project
Funded by the DOE
Exascale Computing Project

Application Partners

- AMREx
- ExaBiome
- Sparse Solvers

This research was supported in part by the Office of Science of the U.S. Department of Energy under contract DE-AC02-05CH11231. This research used resources of the National Energy Research Scientific Computing Center supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-05CH11231.