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

- UPC++ is a C++11+ PGI library
- Lightweight, asynchronous, one-sided communication
- Asynchronous remote function execution (function shipping)
- Data transfers may be non-contiguous
- Functions manage asynchronous, 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 applets 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

Greetings: local_map

// c++ globals variables correspond to rank-local state and un-shared memory
struct ikey { int k, v; } local_map[4];

// insert a key/value pair and return a future
extern struct ikey * local_map_insert(unsigned k, int v)
{
    ikey * x = new ikey {
        .k = k, .v = v
    };
    // add x to local_map
    return x;
}

// RPC obtain location for the data (key, val) // local_map: ikey[]
// returns the index into which rank k was inserted
int local_map_search(unsigned k)
{
    for (ikey * x = local_map; x; ++x) {
        if (x->k == k)
            return x->v;
    }
    return -1;
}

// callback executes when RPC completes 
// return v[global_ptr] 
return local_map[local_map_search(key)].v;

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

- GASNet-EX: communications middleware to support exascale clients
  - One-sided communication: Remote Memory Access (RMA)
  - Active Messages: remote procedure call
  - Feature-rich RMA: multiple APIs for all networks of interest to DOE
- GASNet-EX is an evolution of GASNet-1 for exascale
  - Retains GASNet-1’s (N+1) scalability (unlimited APIs)
  - Provides near-native performance across all architectural RMA APIs
  - Improves performance of asynchronous PGAS models
  - GASNet-EX RMA versus MPI-3 RMA and MPI-2 RMA

GASNet-EX augments and enhances GASNet-1

- Enhancements address needs of modern asynchronous PGAS models
- Interfaces adapted for improved scalability
- Features critical to PGAS 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 flushing)
- New APIs, e.g. to reduce buffer copies between layers
- Vector-foreign RMA for non-contiguous point-to-point RMA
- Remote Atomics, implemented with NIC offload where available
- Teams of non-blocking collectives

Future enhancements may include:
- Dependent operations to control ordering of in-flight operations
- Offset-based addressing
- Multiple enumerate APIs, e.g. to enhance multithreading support
- Support for “out-of-segment” remote addresses
- Communication directly to/from device memory (e.g. GPU/Direct)

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.

Impact:
- On average, symPACK delivers a 2.6x speedup over a state-of-the-art sparse symmetric solver
- UPC++’s one-sided pull strategy avoids the need for (and cost of) unneeded messages in MPI
- On Extend-Add, the increased overlap exposed by UPC++ yields up to a 1.6x speedup over MPI collective and 3.1x over MPI message-passing implementations.

Example of UPC++ interface usage: RMA Put

// interface to remote memory with put function
void * gex_PUT_U64(gex_q_t, gasnet_handle_t, uint64_t, size_t flags);

void * gex_PUT_BLOB(gex_q_t, gasnet_handle_t, void *, size_t, size_t flags);

Vector-indexed-Blended (VIS) Interfaces for Non-Contiguous RMA

- Enables more flexible mode selection
- Provides asynchronous operations
- Selects optimal behavior (e.g. immediate mode and offset-based addressing)
- Enables asynchronous operation without intrusive library operations

Examples of VIS interface function pairs:

- gexᚱ');</ref>