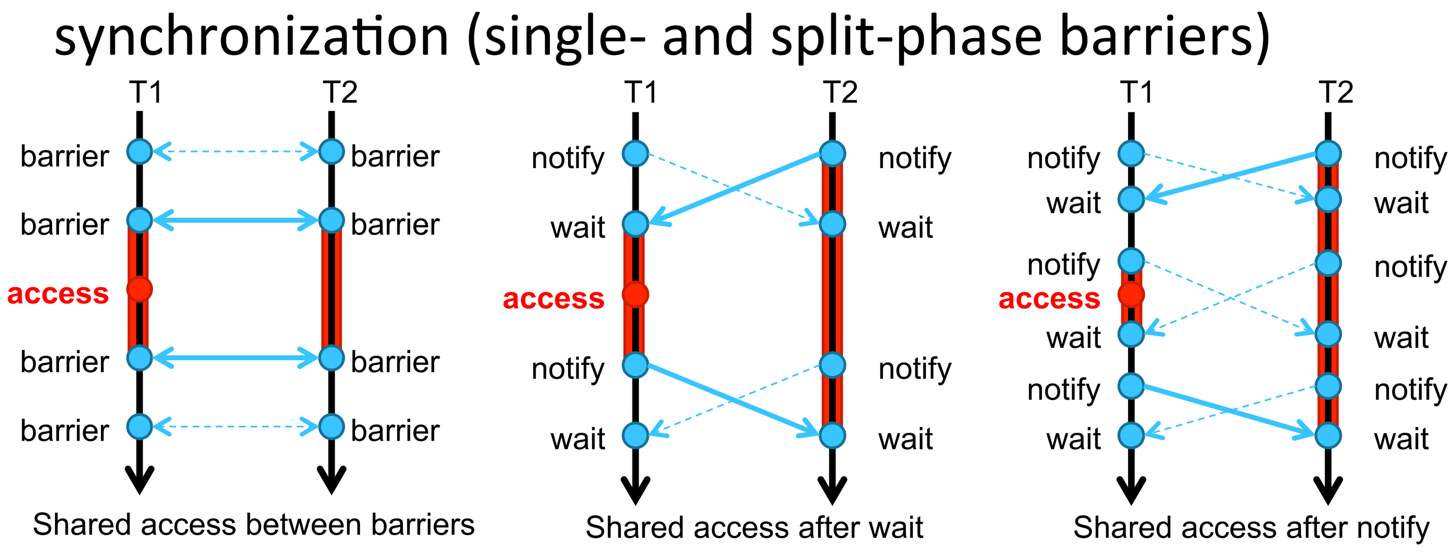
PGAS Research at Berkeley

http://upc.lbl.gov

Scalable Data Race Detection

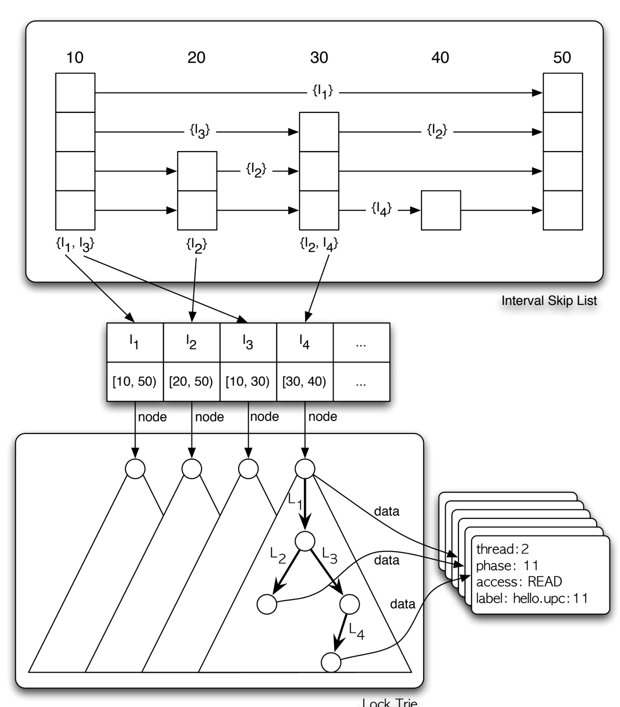
- Active Testing: Leverage program analysis to make testing quickly find real concurrency bugs
  - Phase 1: Use imprecise static or dynamic program analysis to find abstract states where a potential concurrency bug can happen (Race Detector)
  - Phase 2: Directed testing based on the abstract states obtained from phase 1 (Race Tester)
- THRILLE – THRead Interposition Library and Lightweight Extensions: Active testing framework for UPC
- Implementation of race detector and tester for programs written in shared memory style
  - Supports shared memory accesses using shared pointers and bulk transfers (`upc_memcpy`)
  - Tracks fine-grained synchronization (locks) and bulk synchronization (single- and split-phase barriers)



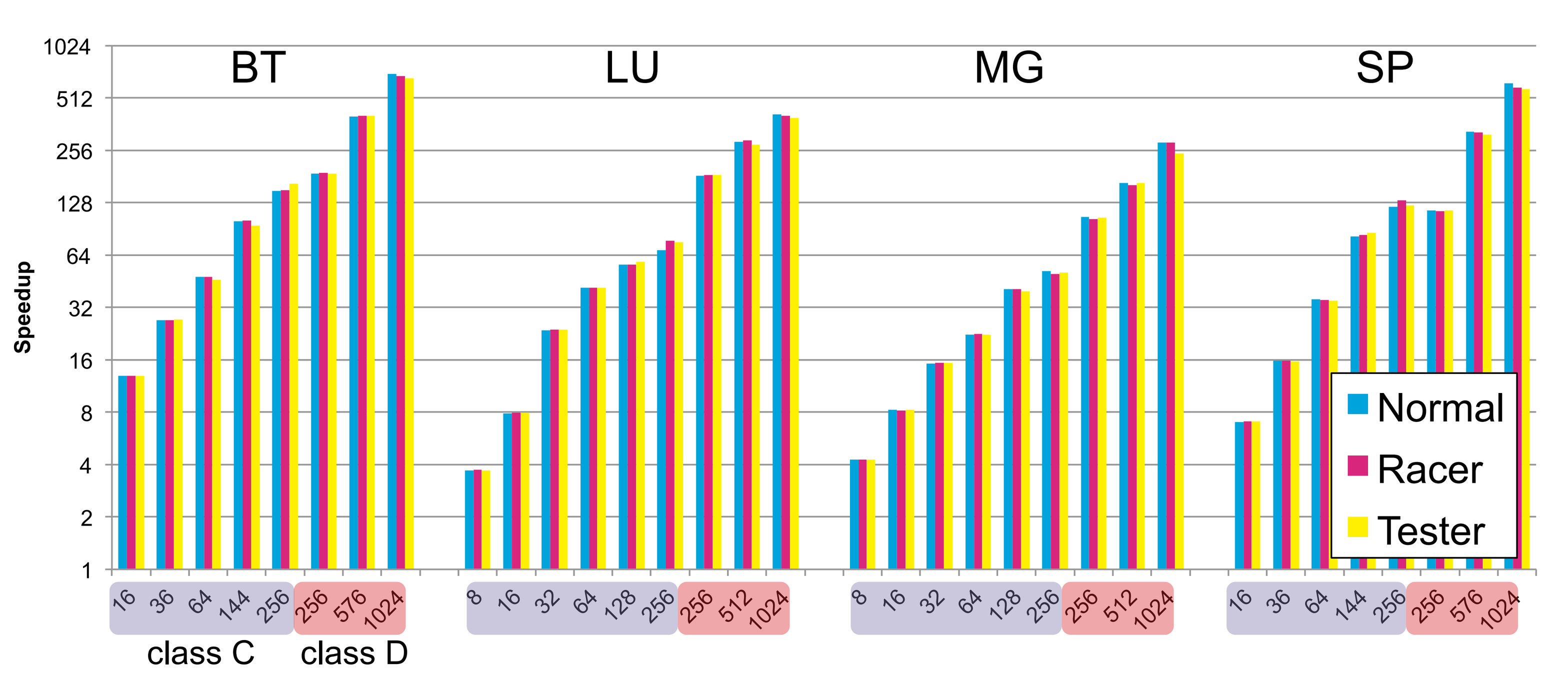
- All bugs found using 4 cores on workstation

Benchmark	LoC	Runtime	ThrilleRacer		ThrilleTester	
			Overhead	Pot. race	Overhead	Conf. race
guppie	227	2.094s	12%	2	1.7%	2
knapsack	191	2.099s	14.9%	2	1.8%	2
laplace	123	2.101s	16.3%	0	-	-
psearch	777	2.982s	1.8%	3	3.8%	2
FT 2.3	2306	8.711s	6.1%	2	4.8%	2
CG 2.4	1939	3.812s	0.5%	0	-	-
IS 2.4	1449	3.073s	1.1%	0	-	-
MG 2.4	2314	4.895s	3.1%	2	1.2%	2
BT 3.3	9626	48.78s	0.5%	8	0.8%	0

- Optimizations for scalability
  - Efficient data structures for *memory ranges*
    - Interval Skiplists
    - Lock Tries



  - Minimize Communication
    - Prune all but *weakest* shared access information
    - Coalesce queries to barrier boundaries
  - Sampling with Exponential Backoff
- Scalability results on large cluster (up to 1024 cores)



- Maximum 8% slowdown at 8K cores
  - Franklin Cray XT4 Supercomputer at NERSC
  - Quad-core 2.3GHz CPU and 8GB RAM per node
  - Portals interconnect
- Download available at http://upc.lbl.gov/thrille.shtml

UPC Collectives Library 2.0

- Goals
  - Improve usability and enable performance optimization over the previous UPC 1.2 collective specification
  - Chart a path towards MPI interoperability
- Library-Only Approach
  - Collectives operate on multi-valued shared objects
  - Ordering of non-blocking collectives
    - Execution order is defined by collective start
    - Allow users to wait for completion in arbitrary order
- Features
  - Equivalents to some popular MPI collectives
    - Teams for collectives (similar to MPI communicators)
    - Allreduce, Reduce and Reduce-scatter
    - "v" versions of collectives (e.g., allgatherv and alltoalv)
  - Non-blocking collectives (with and without waiting handles)
  - Simplification of synchronization flags and input buffers
- Implementation
  - Blocking collectives mapped well to MPI collectives
  - Berkeley and IBM will provide reference implementations based on GASNet and PAMI, respectively
  - Applications being developed with the new collectives
    - Communication-avoiding 2.5-D matrix multiplication and Choleskey Factorization (team broadcast and team reduce)
    - Multi-dimensional FFT (team alltoall)
    - Breath-First Search (Graph500) (allgatherv and alltoallv)

`Example of the new collectives interface (broadcast):`

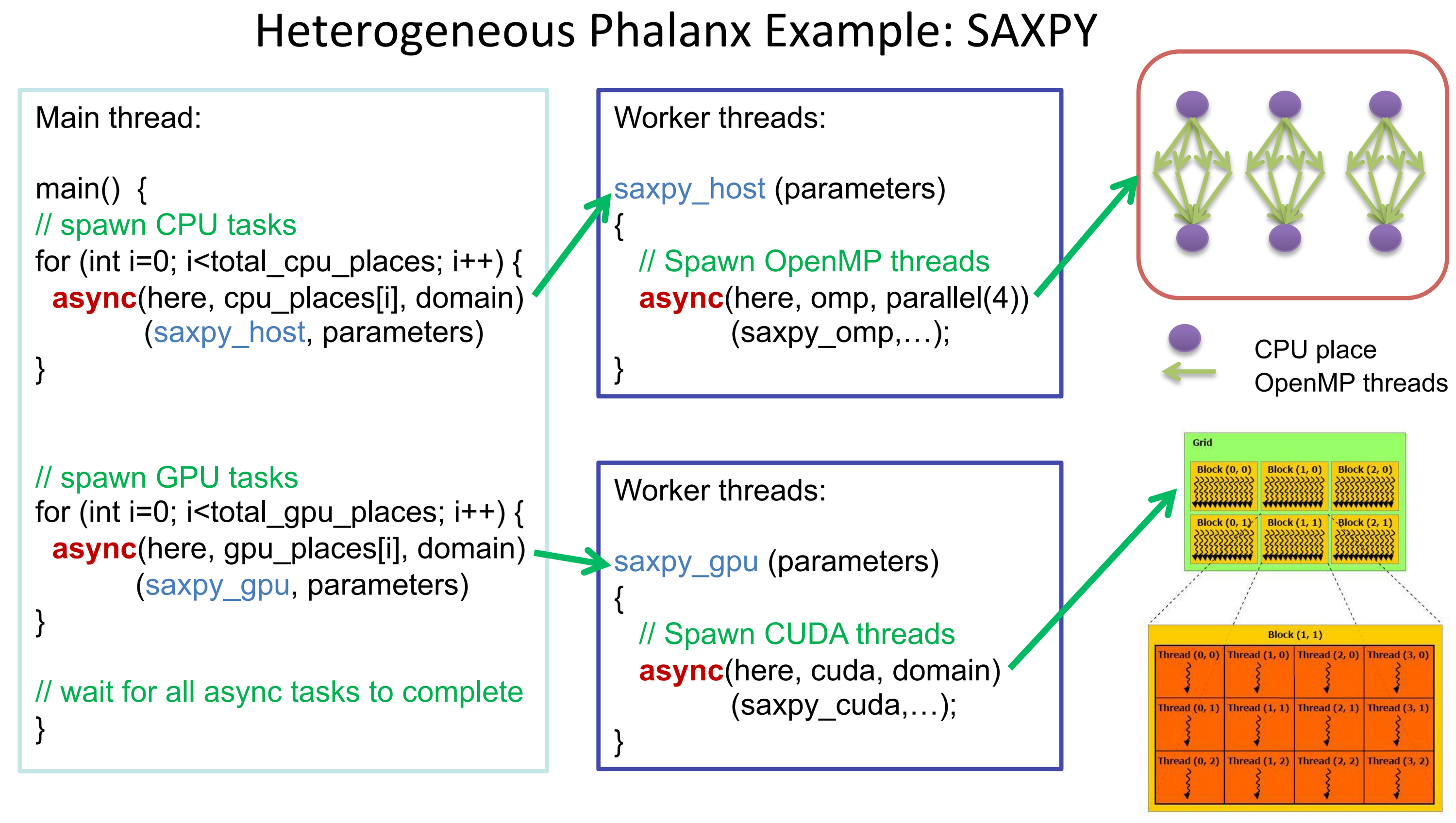
```
upccoll_return_t
upccoll_bcast (shared void * sendbuf, size_t sendcount, upccoll_dtype_t sendtype,
               shared void * recvbuf, size_t recvcount, upccoll_dtype_t recvtype,
               int root, upccoll_team_t team, upccoll_flag_t flags,
               upccoll_handle_t * handle);
```

For the complete UPC collectives 2.0 API and detailed discussion, see "UPC Collectives Library 2.0" by George Almasi, Paul Hargrove, Gabriel Tanase, Yili Zheng in the Fifth Conference on Partitioned Global Address Space Programming Models (PGAS11).

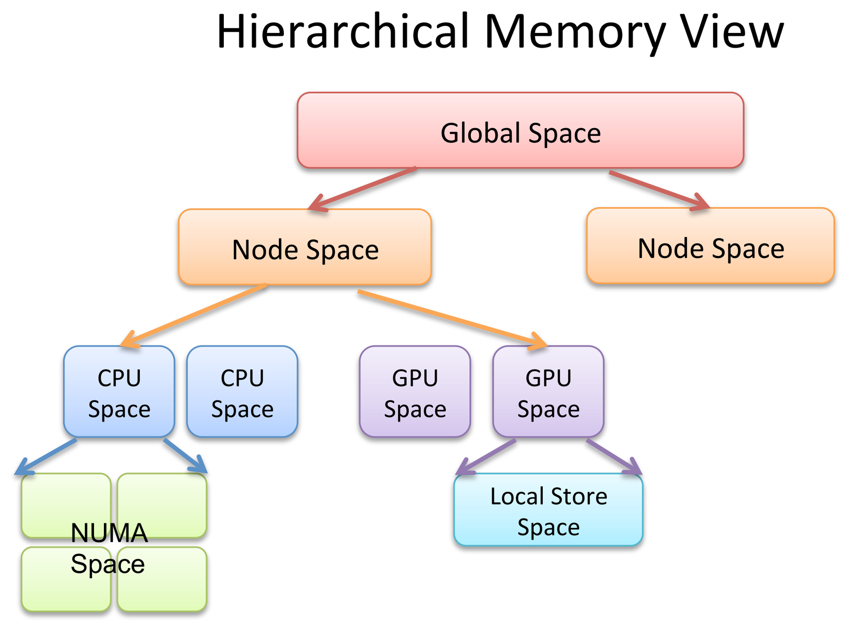
PGAS for Heterogeneous Computing

- DARPA UHPC Echelon Project
  - Two orders of magnitude increase in application execution energy efficiency over today's CPU systems
  - Improve programmer productivity
  - Resilient hardware and reliable software
  - Multi-institution team led by NVIDIA
- Phalanx Programming System for Echelon
  - Global address space programming model with dynamic asynchronous task execution
  - Prototyped in C++ with CUDA and GASNet
  - Run on current heterogeneous GPU clusters

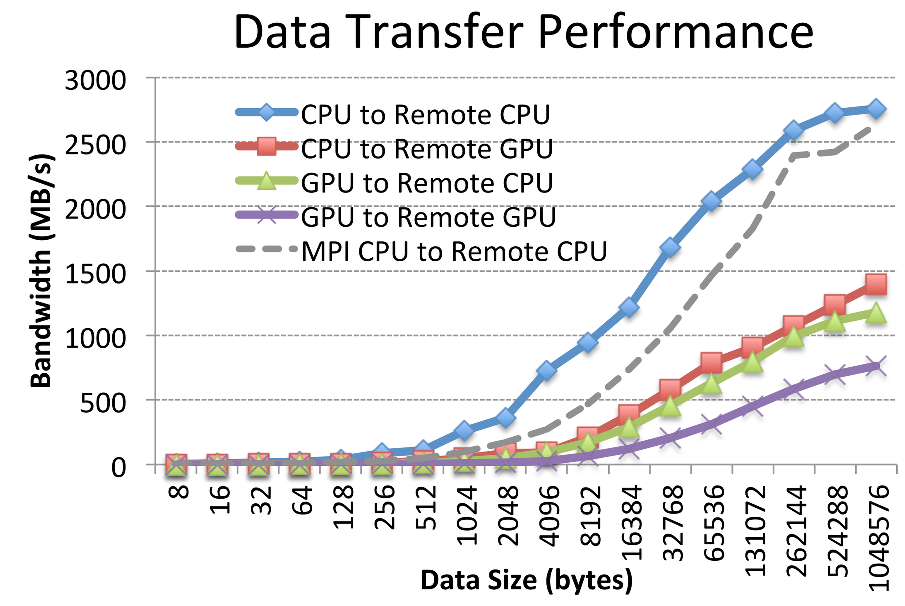
Heterogeneous Phalanx Example: SAXPY



Hierarchical Memory View



Data Transfer Performance



Intel Xeon 5530 with QDR Infiniband and NVIDIA Fermi GPU

This research was supported in part by the Office of Science of the U.S. Department of Energy under contract DE-FC03-01ER25509 and by the DARPA contract HR0011-10-9-0008. 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.