# UPC AT SCALE

Rajesh Nishtala, *Yili Zheng*, Paul Hargrove, Katherine Yelick

Lawrence Berkeley National Lab

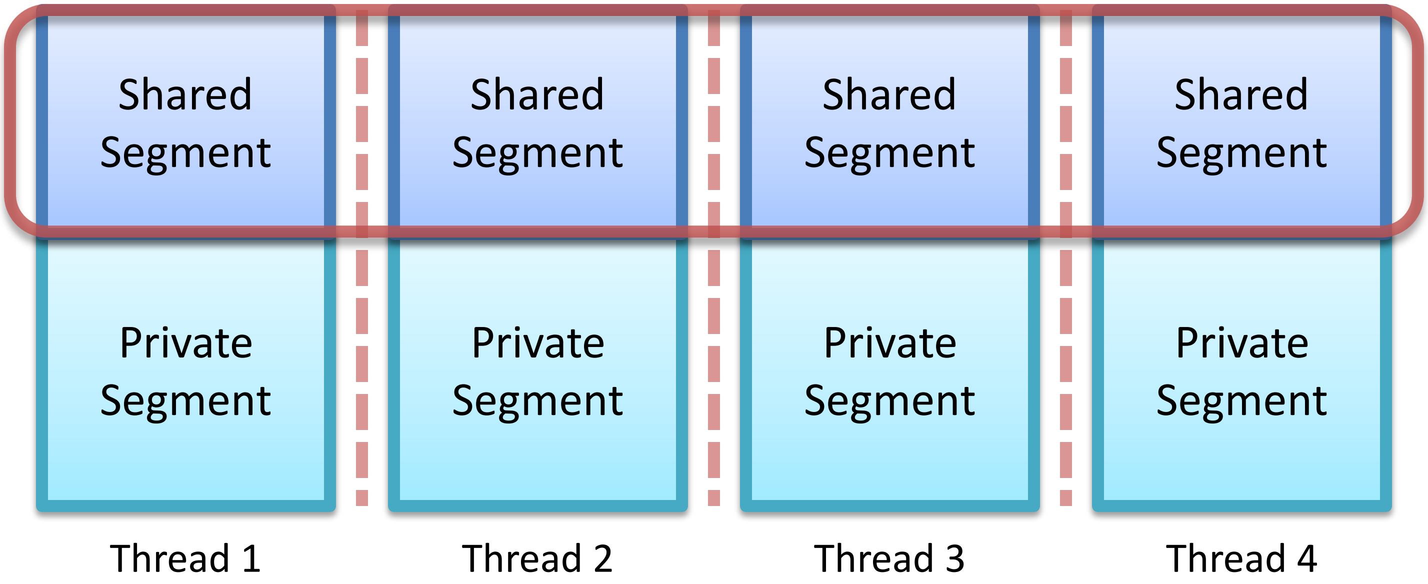
# Berkeley UPC Group

- PI: Katherine Yelick
- Group members: Filip Blagojevic, Dan Bonachea, Paul Hargrove, Costin Iancu, Seung-Jai Min, Yili Zheng
- Former members: Christian Bell, Wei Chen, Jason Duell, Parry Husbands, Rajesh Nishtala , Mike Welcome
- A joint project of LBNL and UC Berkeley

# Outline

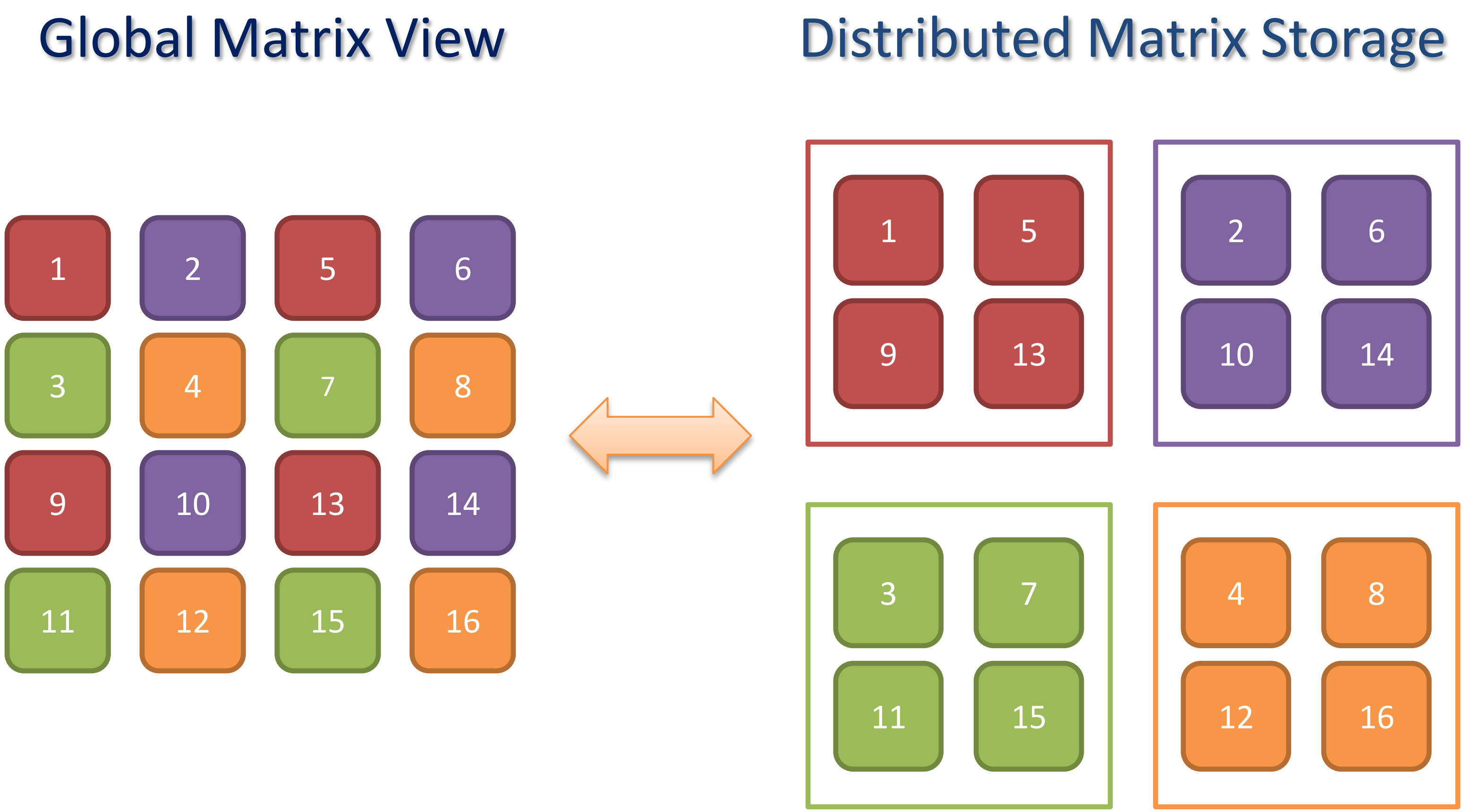
- Partitioned Global Address Space Programming Model
- Berkeley UPC and GASNet
- One-sided communication and Active Messages
- Collective Communication
- Benchmarks

# Partitioned Global Address Space



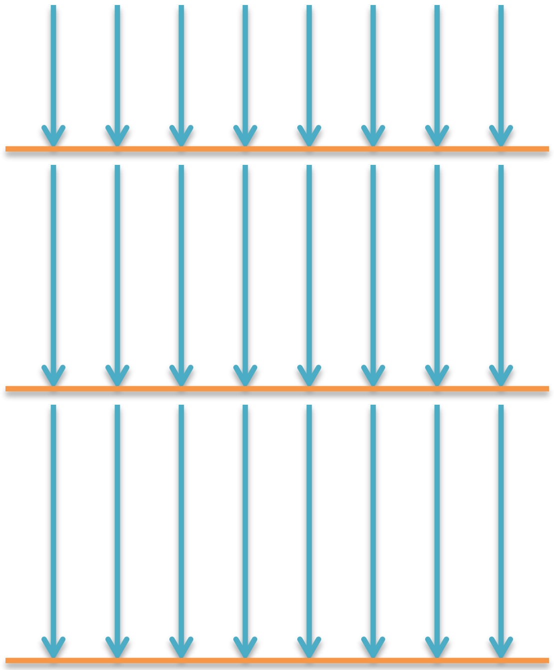

- Global data view abstraction for productivity
- Vertical partitions among threads for locality control
- Horizontal partitions between shared and private segments for data placement optimizations
- Friendly to non-coherent cache architecture

# PGAS Example: Global Matrix Distribution



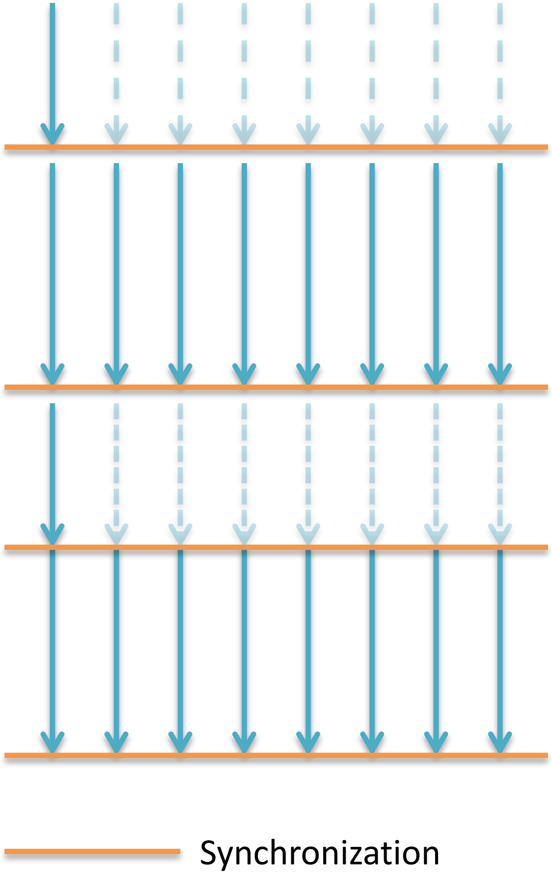
# UPC Programming Models

**SPMD**



**Bulk Synchronous Parallel with Computation and Communication Overlaps**

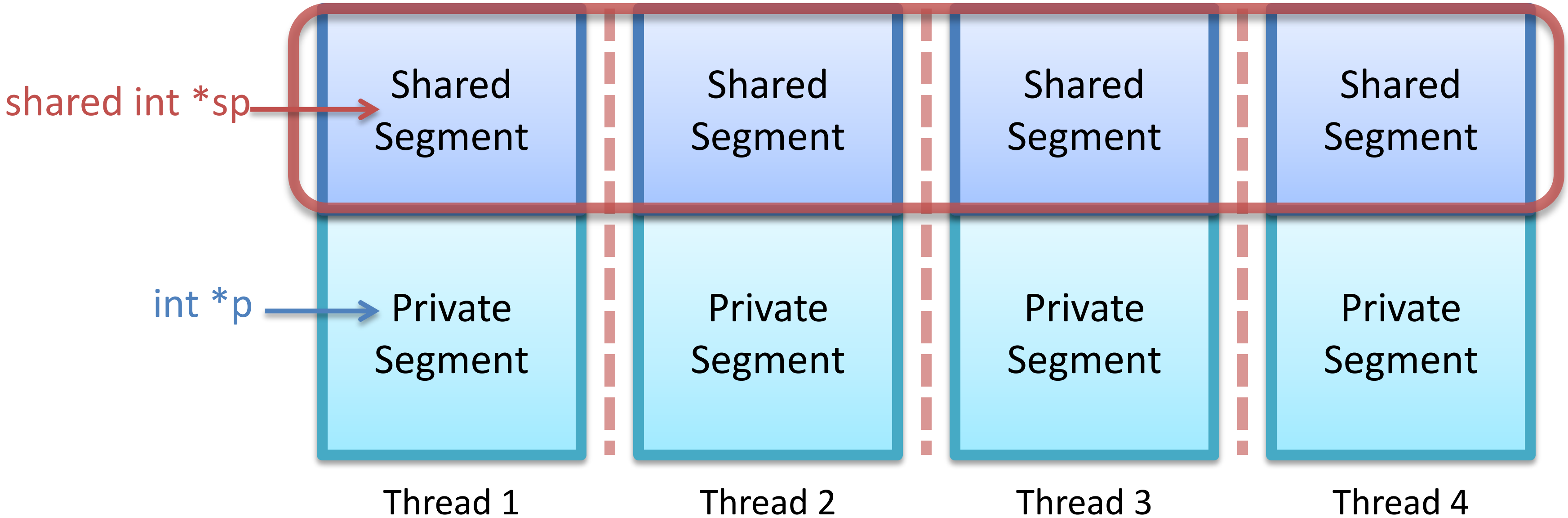
**Fork-Join**



# UPC Overview

- PGAS dialect of ISO C99
- Distributed shared arrays
- Dynamic shared-memory allocation
- One-sided shared-memory communication
- Synchronization: barriers, locks, memory fences
- Collective communication library
- Parallel I/O library

# UPC PGAS Example

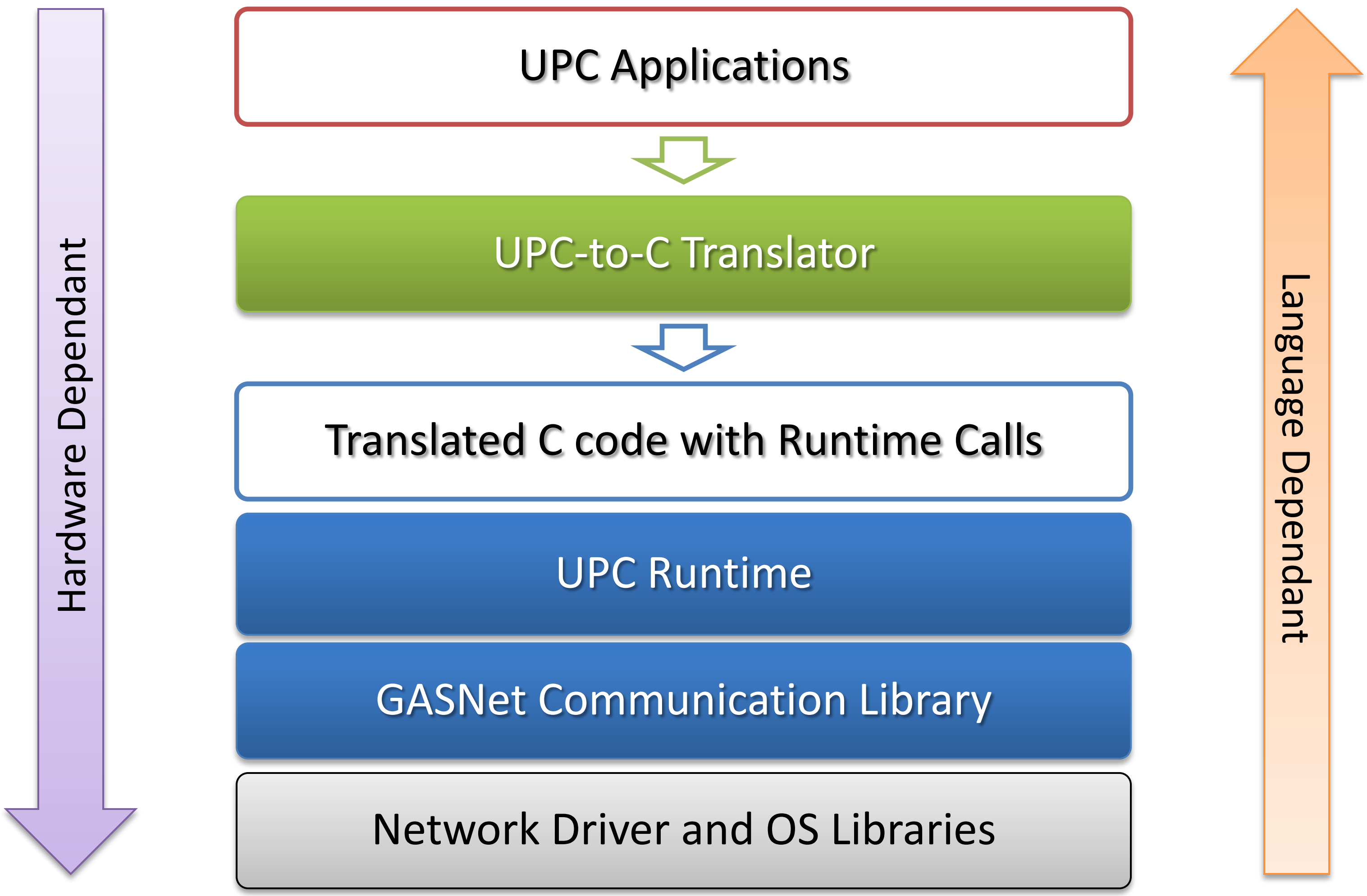



| | | | | |
|---|---|---|---|---|
| Standard C | p = malloc(4) | *p STOP | *p STOP | *p STOP |
| UPC | sp = upc_alloc(4) | *sp ✓ | *sp ✓ | *sp ✓ |

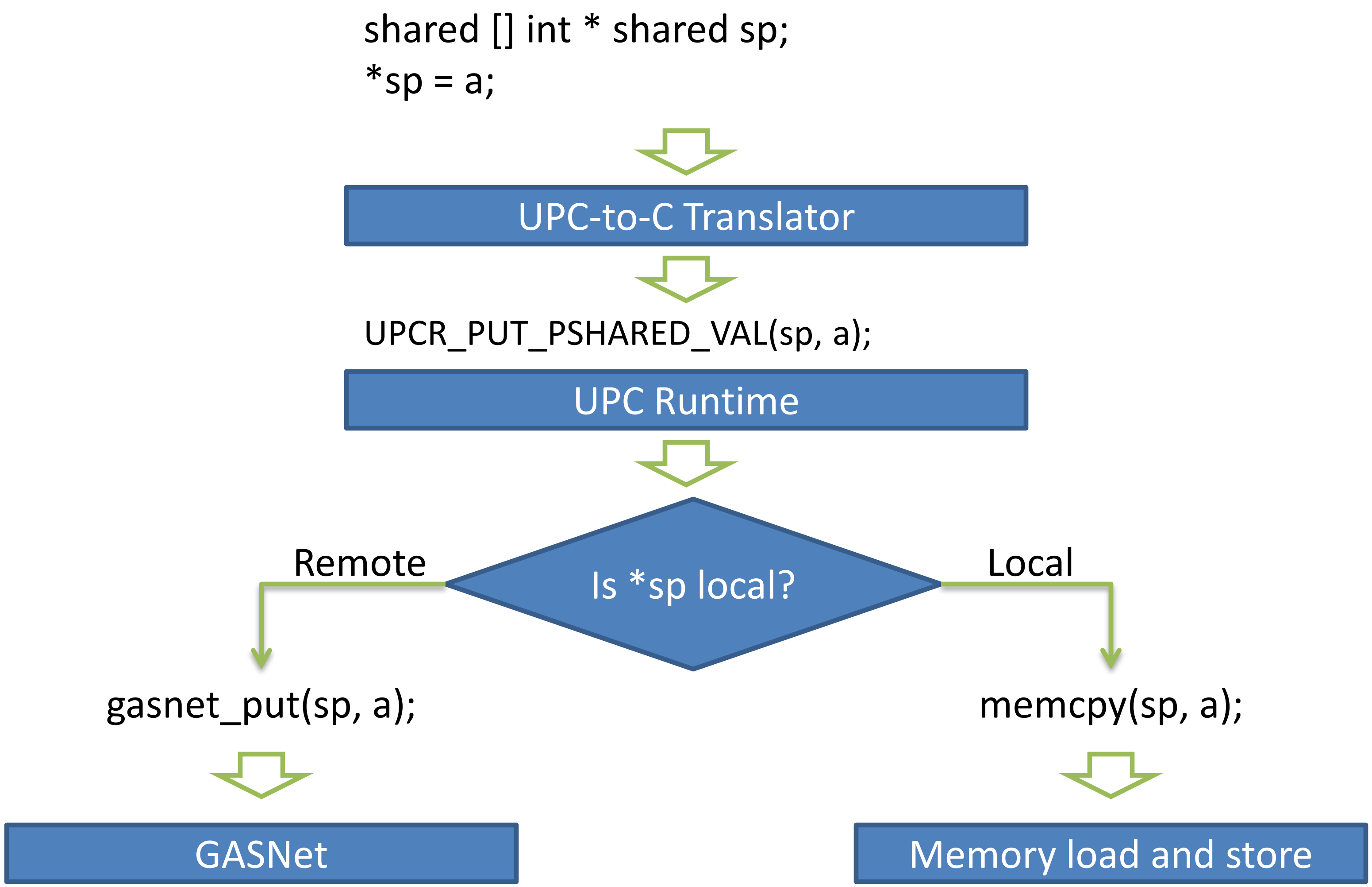
# Outline

- Partitioned Global Address Space Programming Model
- Berkeley UPC and GASNet
- One-sided communication and Active Messages
- Collective Communication
- Benchmarks

# Berkeley UPC Software Stack



# Translation and Call Graph Example



# UPC Compiler Implementation

- Source-to-source translator based on the Open64 compiler infrastructure
  - Portable: work with most popular back-end compilers; support remote translation
  - High performance: leverage existing Open64 program analysis and optimizations
- UPC-specific Optimizations
  - Message vectorization
  - Message strip-mining
  - Overlapping communication
  - Data reshaping

See Berkeley UPC Publications (http://upc.lbl.gov/publications/#compiler ) for further information on compiler analysis and optimizations.

# UPC Runtime Implementation

- Modular design with a well-defined API
  - Support multiple front-end compilers
  - Enable runtime optimizations
- Light-weight implementation
- Efficient shared-memory management
- Fast intra-node communication via hardware shared-memory
  - Pthreads
  - Processes with POSIX shared-memory

# GASNet Implementation

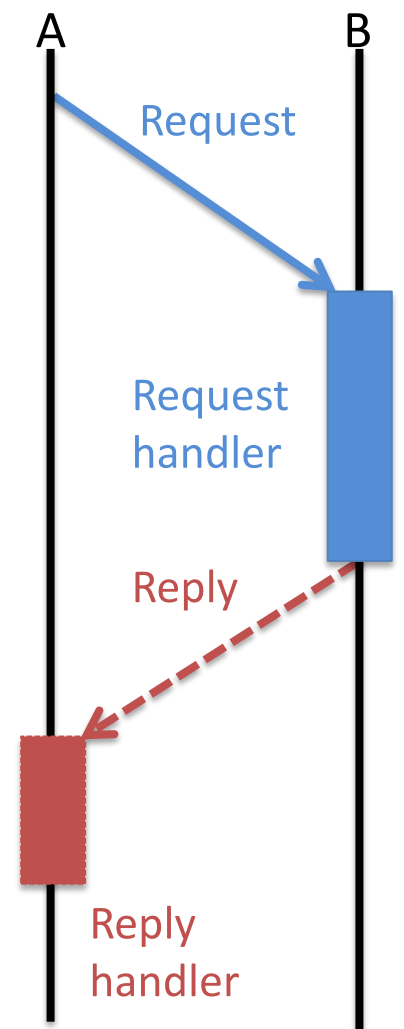
- Core API
  - Active Messages
- Extended API
  - Non-Blocking One-sided Communication
  - Collective Communication
  - Point-to-Point Synchronizations
  - Vector, Indexed, Stride Data Transfer
- Portable tools
  - timers, memory barriers, atomic ops and portable data types

# Outline

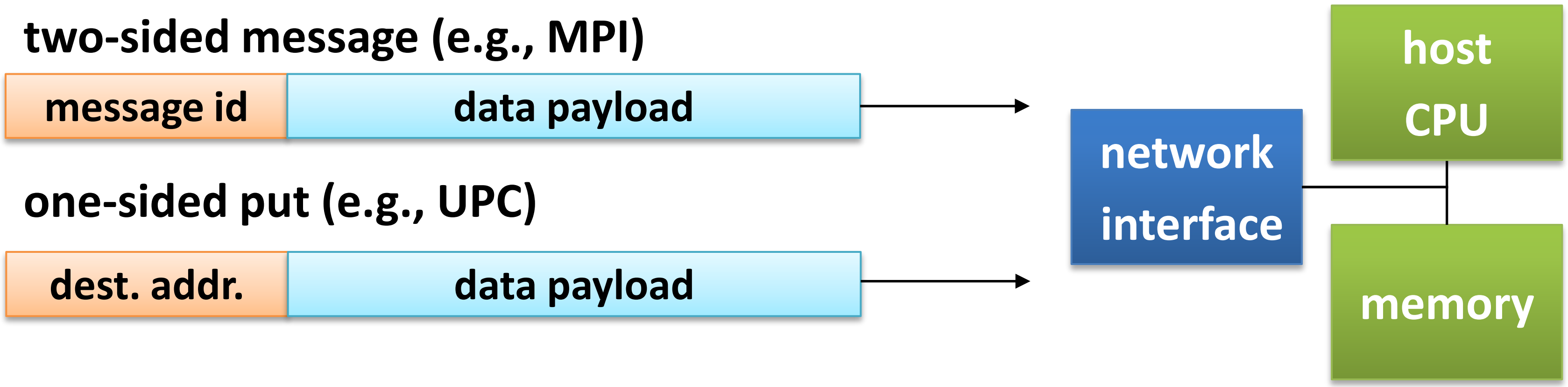
- Partitioned Global Address Space Programming Model
- Berkeley UPC and GASNet
- One-sided communication and Active Messages
- Collective Communication
- Benchmarks

# Active Messages

- Active messages = Data + Action
- Key enabling technology for both one-sided and two-sided communications
  - Software implementation of Put/Get
  - Eager and Rendezvous protocols
- Remote Procedural Calls
  - Facilitate "owner-computes"
  - Spawn asynchronous tasks

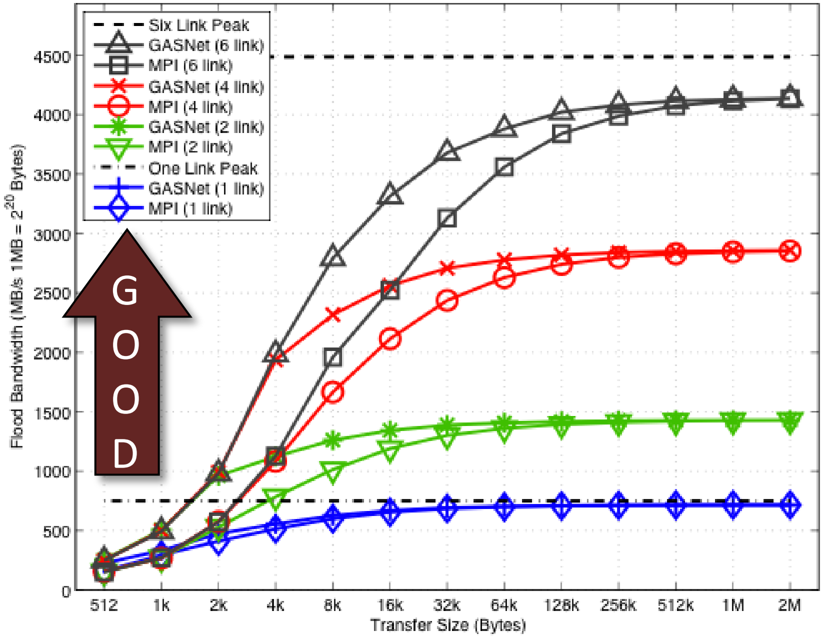


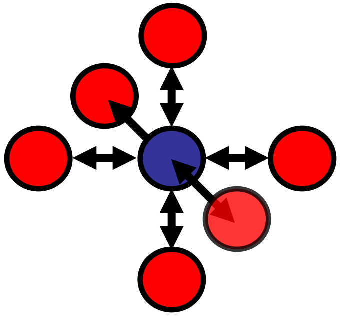
# One-Sided vs. Two-Sided Messaging




- Two-sided messaging
  - Message does not contain information about the final destination; need to look it up on the target node
  - Point-to-point synchronization implied with all transfers
- One-sided messaging
  - Message contains information about the final destination
  - Decouple synchronization from data movement

# GASNet Bandwidth on BlueGene/P



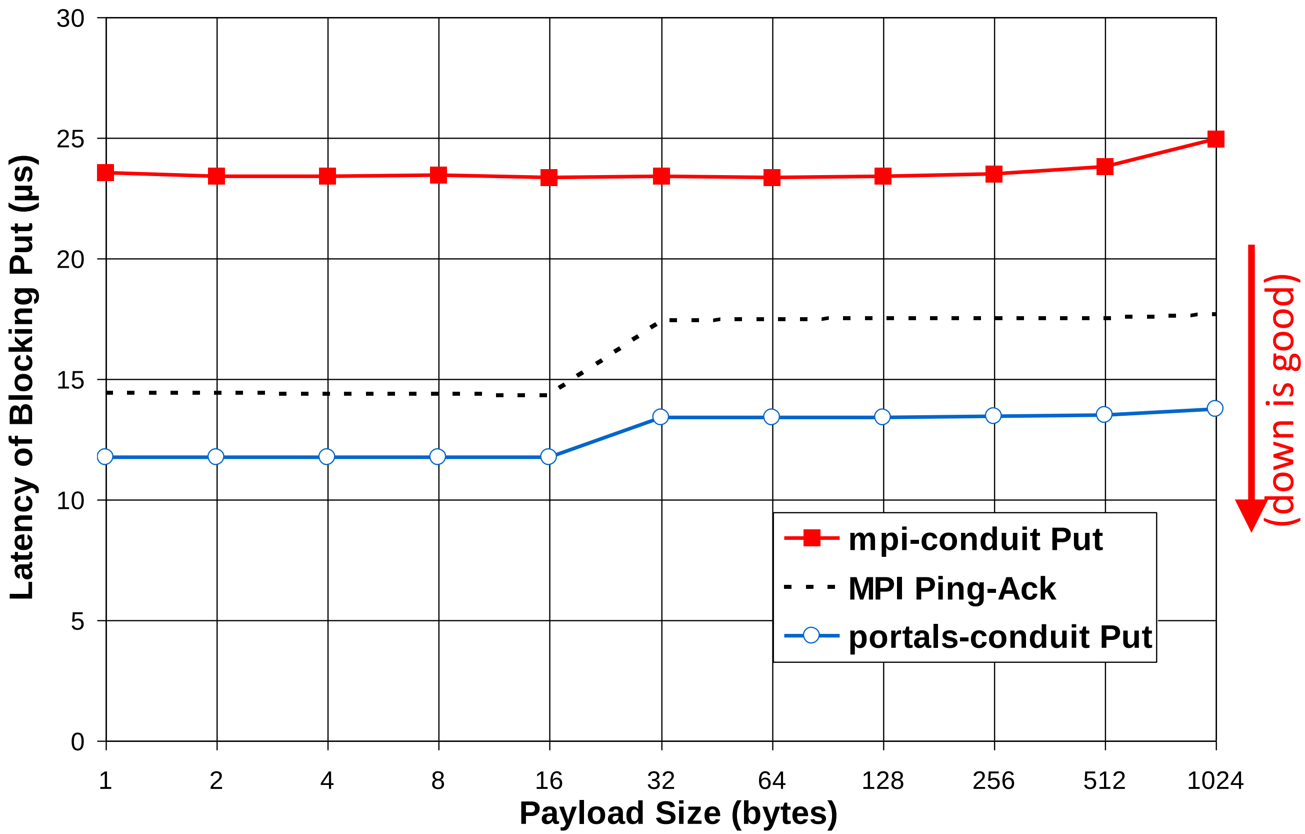



* Kumar et. al showed the maximum achievable bandwidth for DCMF transfers is 748 MB/s per link so we use this as our peak bandwidth
See "The deep computing messaging framework: generalized scalable message passing on the blue gene/P supercomputer", Kumar et al. ICS08

- Torus network
  - Each node has six 850MB/s* bidirectional links
  - Vary number of links from 1 to 6
- Consecutive non-blocking puts on the links (round-robin)
- Similar bandwidth for large-size messages
- GASNet outperforms MPI for mid-size messages
  - Lower software overhead
  - More overlapping

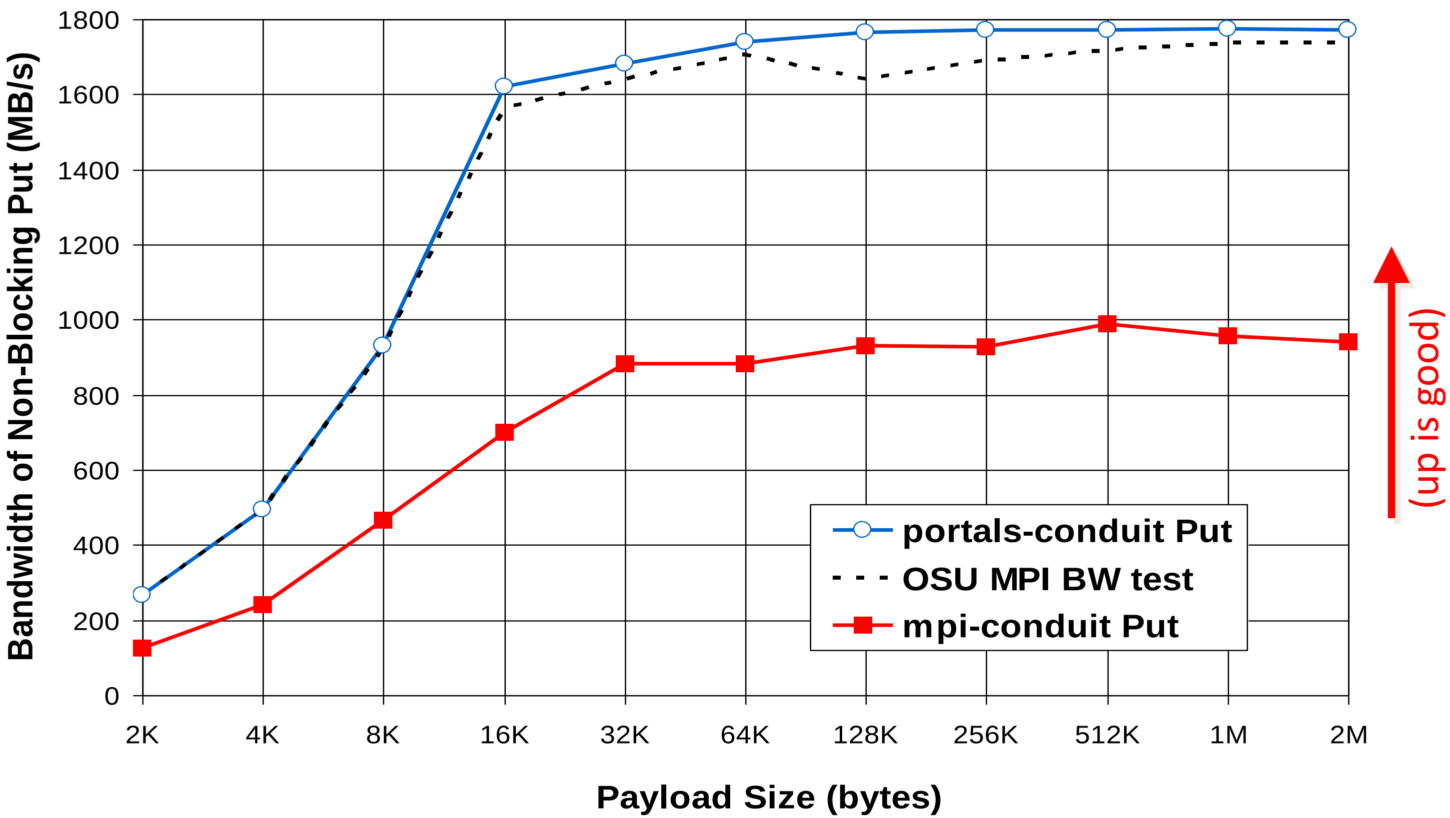
See "Scaling Communication Intensive Applications on BlueGene/P Using One-Sided Communication and Overlap", Rajesh Nishtala, Paul Hargrove, Dan Bonachea, and Katherine Yelick, *IPDPS 2009*

# GASNet Latency on Cray XT4



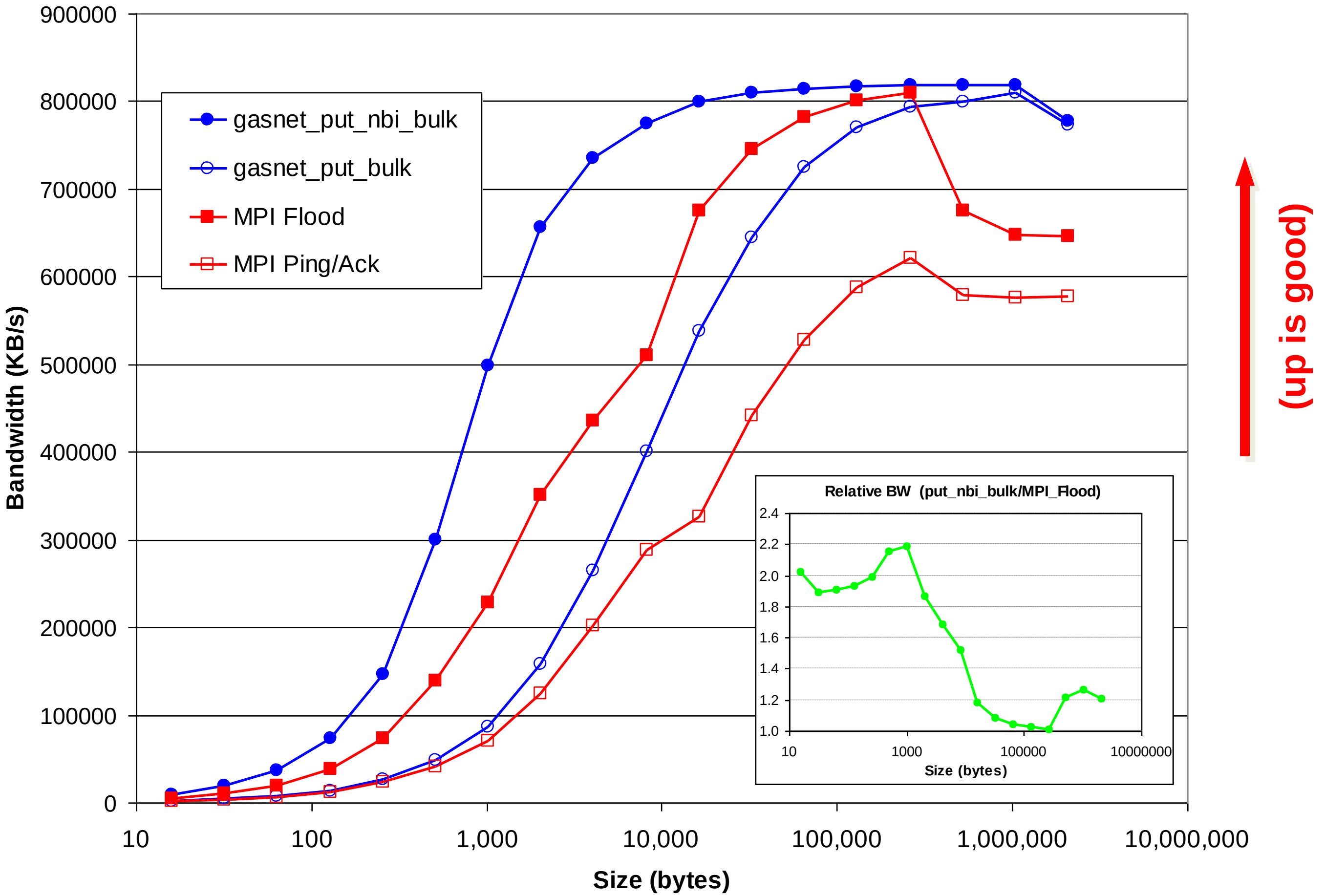

Slide source: Porting GASNet to Portals: Partitioned Global Address Space (PGAS) Language Support for the Cray XT, Dan Bonachea, Paul Hargrove, Michael Welcome, Katherine Yelick, CUG 2009

# GASNet Bandwidth on Cray XT4




Slide source: Porting GASNet to Portals: Partitioned Global Address Space (PGAS) Language Support for the Cray XT, Dan Bonachea, Paul Hargrove, Michael Welcome, Katherine Yelick, CUG 2009

# GASNet vs. MPI on InfiniBand (Jul '05)

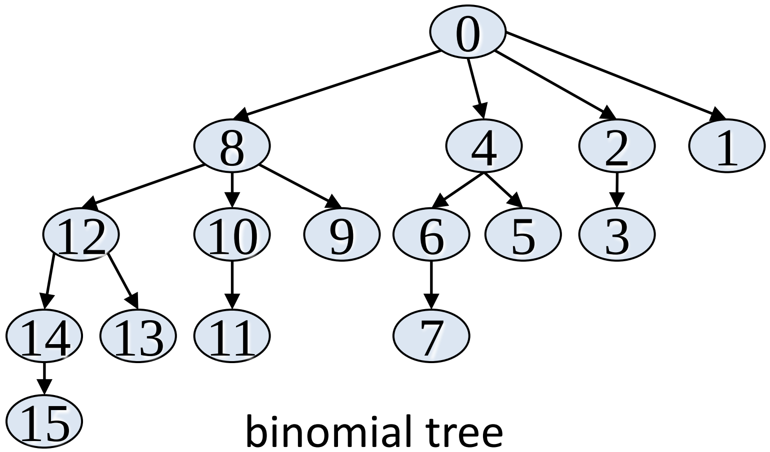



Slide source: Experiences Implementing Partitioned Global Address Space (PGAS) Languages on InfiniBand, Paul Hargrove et al

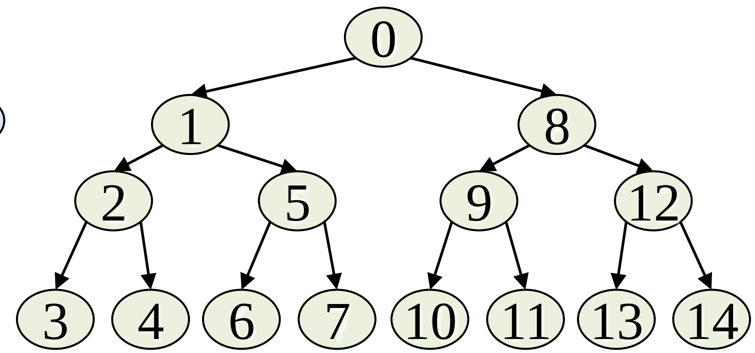
# Outline

- Partitioned Global Address Space Programming Model
- Berkeley UPC and GASNet
- One-sided communication and Active Messages
- Collective Communication
- Benchmarks

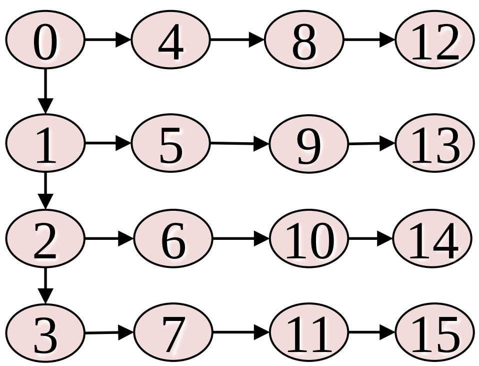
# Collective Communication Topologies



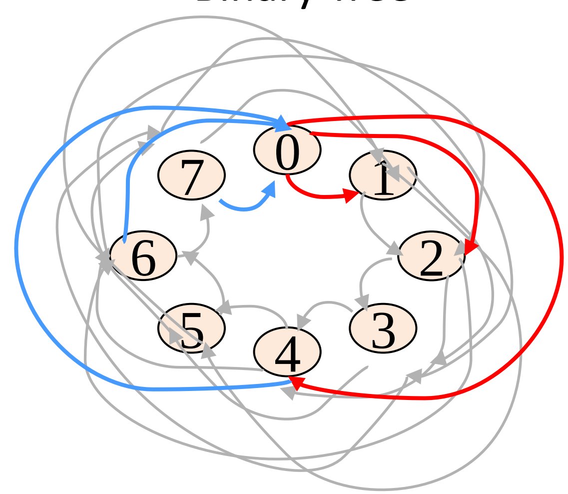

binomial tree




Binary Tree

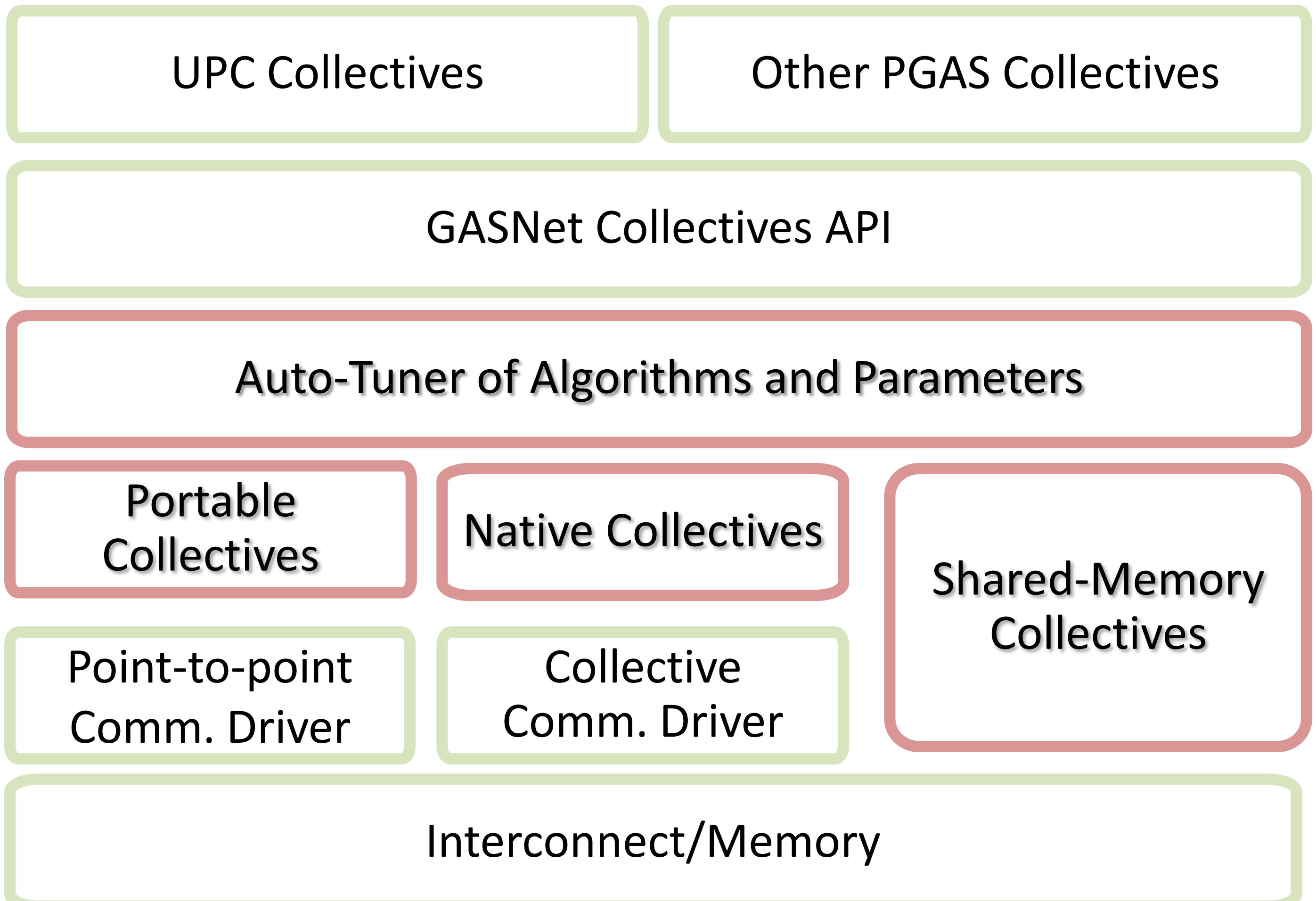



Fork Tree




Radix 2 Dissemination

# GASNet Collectives Organization



# Auto-tuning Collective Communication

## Offline tuning

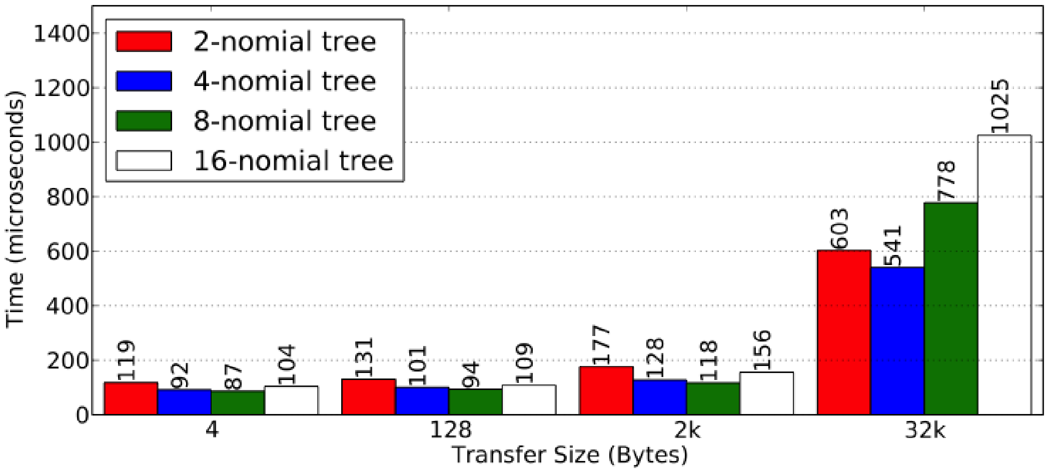
- Optimize for platform common characteristics
- Minimize runtime tuning overhead

## Online tuning

- Optimize for application runtime characteristics
- Refine offline tuning results

| Performance Influencing Factors | Performance Tuning Space |
|---|---|
| Hardware<br>▪ CPU<br>▪ Memory system<br>▪ Interconnect<br>Software<br>▪ Application<br>▪ System software<br>Execution<br>▪ Process/thread layout<br>▪ Input data set<br>▪ System workload | Algorithm selection<br>▪ Eager vs. rendezvous<br>▪ Put vs. get<br>▪ Collection of well-known algorithms<br>Communication topology<br>▪ Tree type<br>▪ Tree fan-out<br>Implementation-specific parameters<br>▪ Pipelining depth<br>▪ Dissemination radix |

# Broadcast

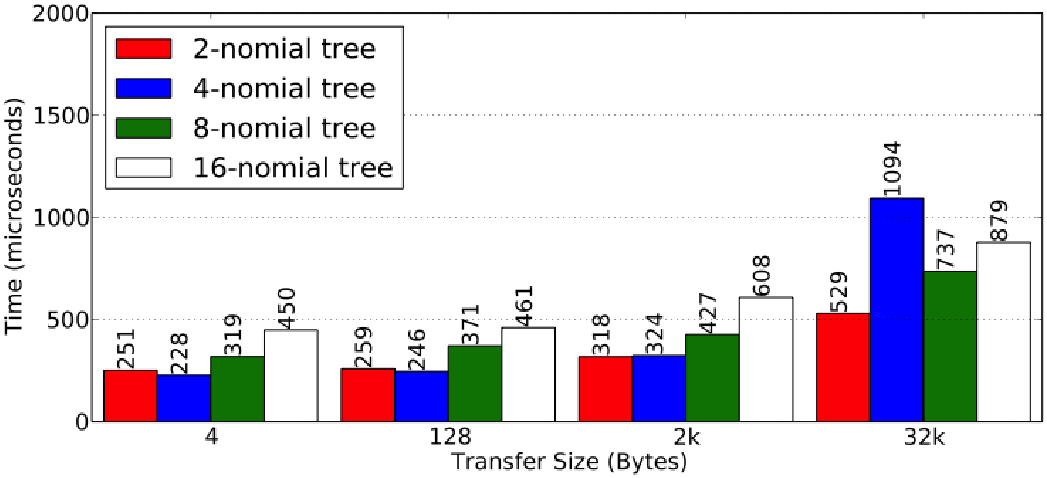



Broadcast on Sun Constellation (1024 cores)

- 4-nomial is consistently a “good” performer
- 8-nomial is best at < 2k bytes

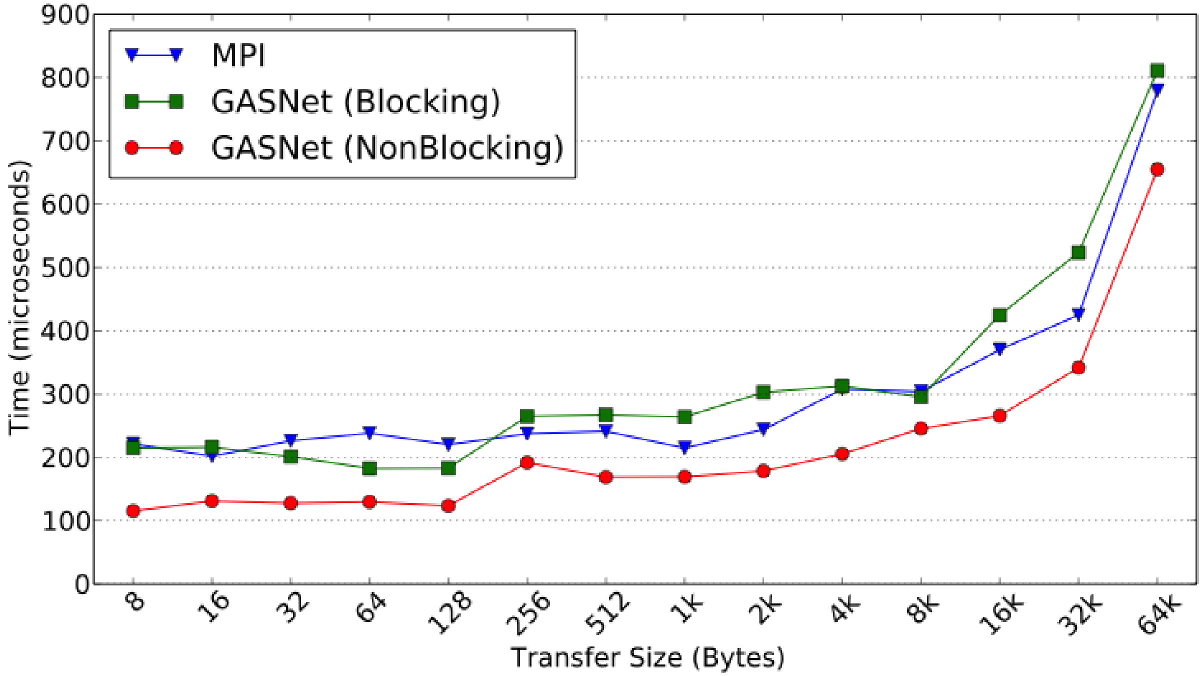
Broadcast on Cray XT4 (2048 cores)

- 4-nomial is best < 2k
- choosing 4-nomial at 32k leads to 2x degradation in performance



# Nonblocking Broadcast

- Benchmark overlaps collectives with each other
  - Collectives pipelined so that the network resources are more effectively used
  - 100-200 microsecond difference
  - We show later how this can be incorporated into a real application
  - All collectives built as state machines

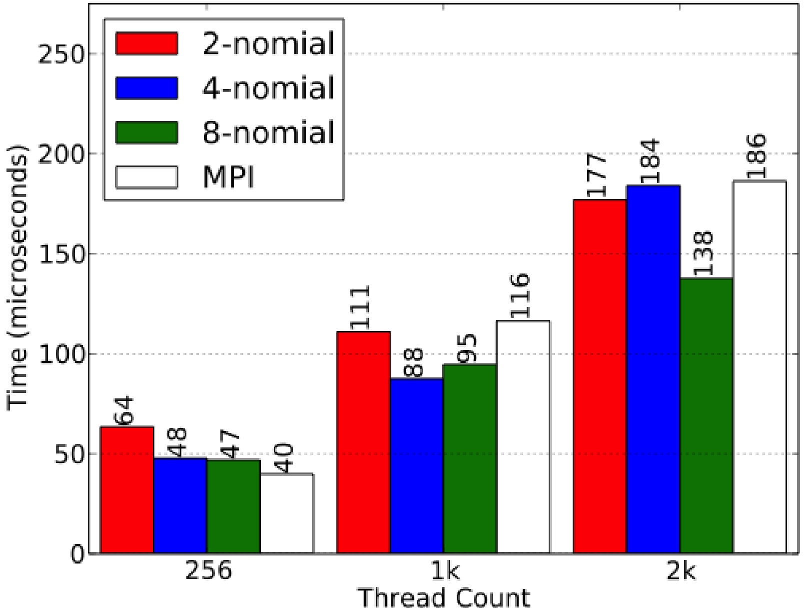


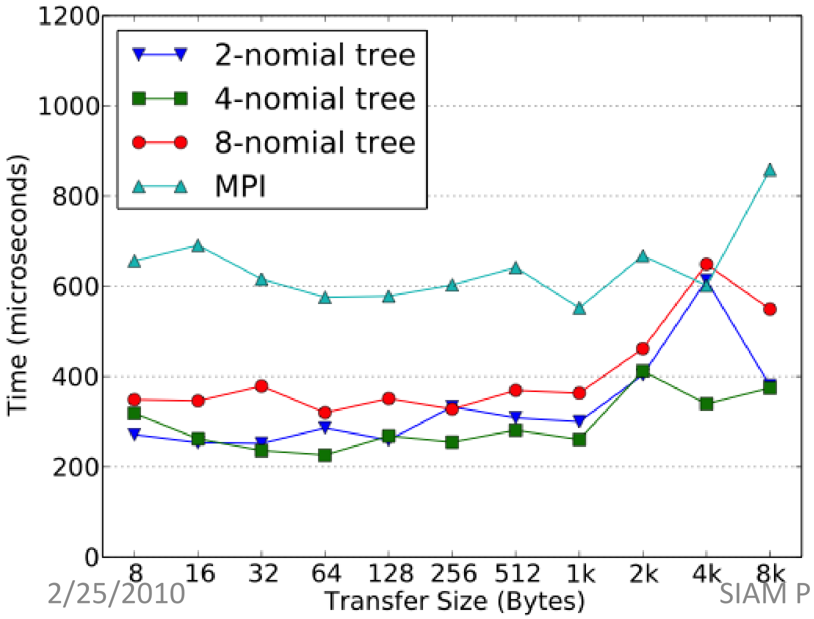

Cray XT4 Nonblocking Broadcast Performance (1024 Cores)

# Reduce

8-byte Reduce on Sun Constellation

- 8-nomial tree delivers best or close to optimal performance
- GASNet outperforms vendor-MPI by 18% at 1k cores and 25% at 2k cores



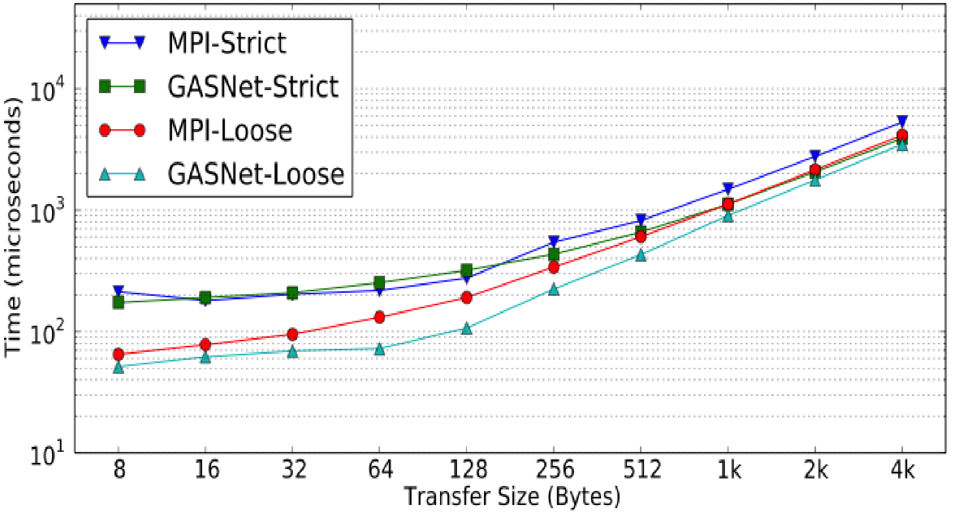

Reduce on Cray XT4 (2048 cores)

- 4-nomial consistently gives a good algorithm
- Average of 25% better performance over 8-nomial
- GASNet out performs MPI by > factor of 2x in most cases

# Scatter/Gather

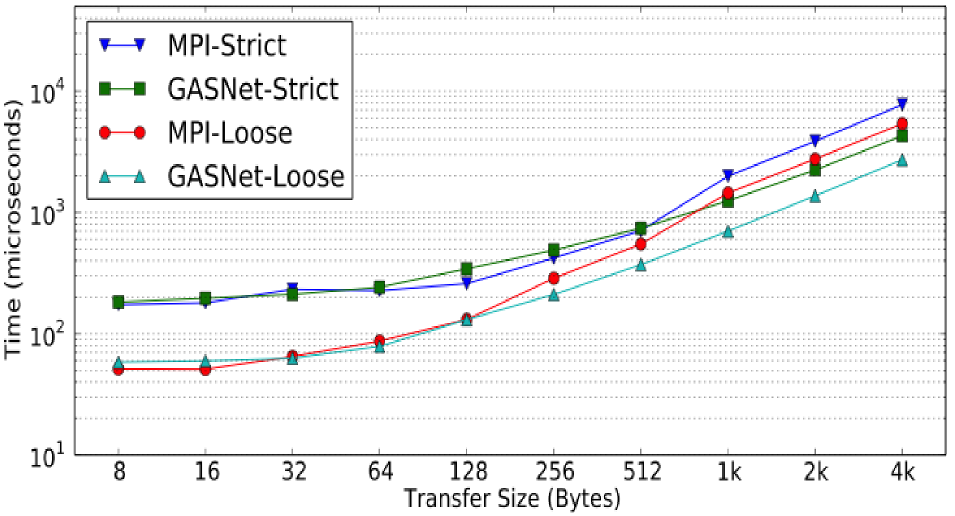
Scatter on 1536 cores of Cray XT5

- Loose synch. offers 4x performance improvement at low sizes
- Difference decreases at higher message sizes
- GASNet is able to deliver better performance for both modes compared to vendor MPI library




Gather on 1536 cores of Cray XT5

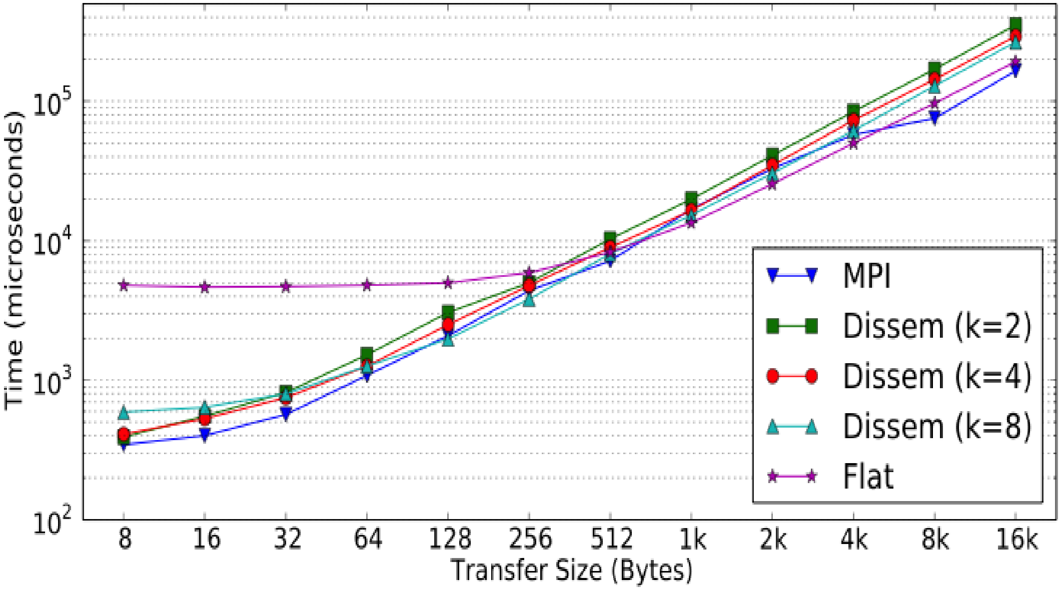
- Similar results as Scatter
- Looser synchronization continues to deliver good performance upto 4k bytes
- GASNet is able to consistently outperform vendor MPI library



# Exchange (Alltoall)

- Dissemination algorithm by Bruck et al. (1997)
  - Send the data multiple times through the network before it reaches the final destination
  - Uses less messages at the cost of more bandwidth
- Highlights a tradeoff between algorithmic choice
  - Intuition suggests there is a crossover point between the algorithms
- Finding the best algorithm is a tuning question that we will address in the automatic tuner section

- Penalty for picking bad algorithm is high
  - Radix-2 is best at 8 bytes but worst at 16k bytes
  - Flat algorithm becomes the best between 512 and 1k byte exchange
    - order of magnitude worse at 8 bytes
    - 28% (~73 ms) faster at 16 Kbytes

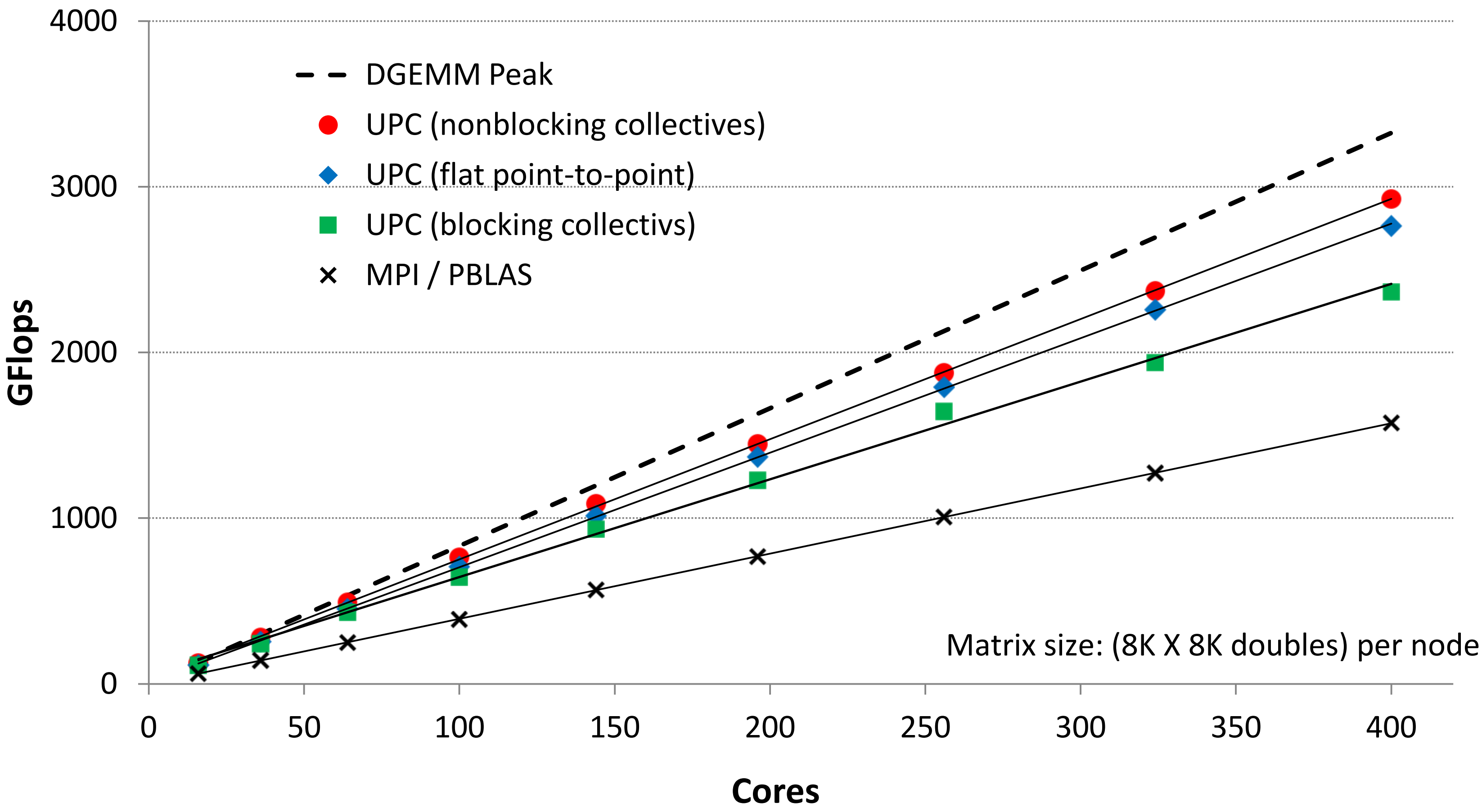



Exchange on Sun Constellation (256 cores)

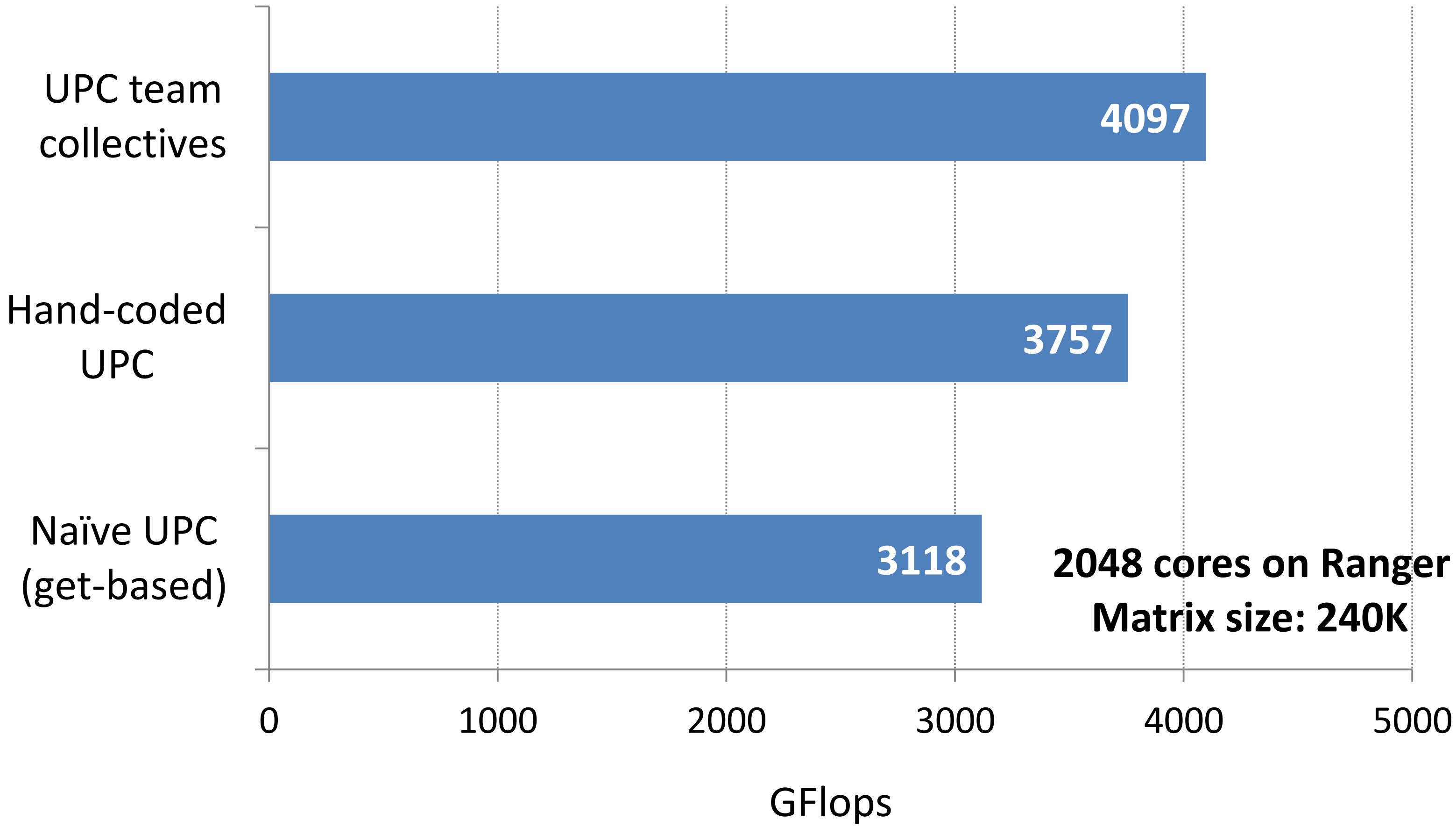
# Outline

- Partitioned Global Address Space Programming Model
- Berkeley UPC and GASNet
- One-sided communication and Active Messages
- Collective Communication
- Benchmarks

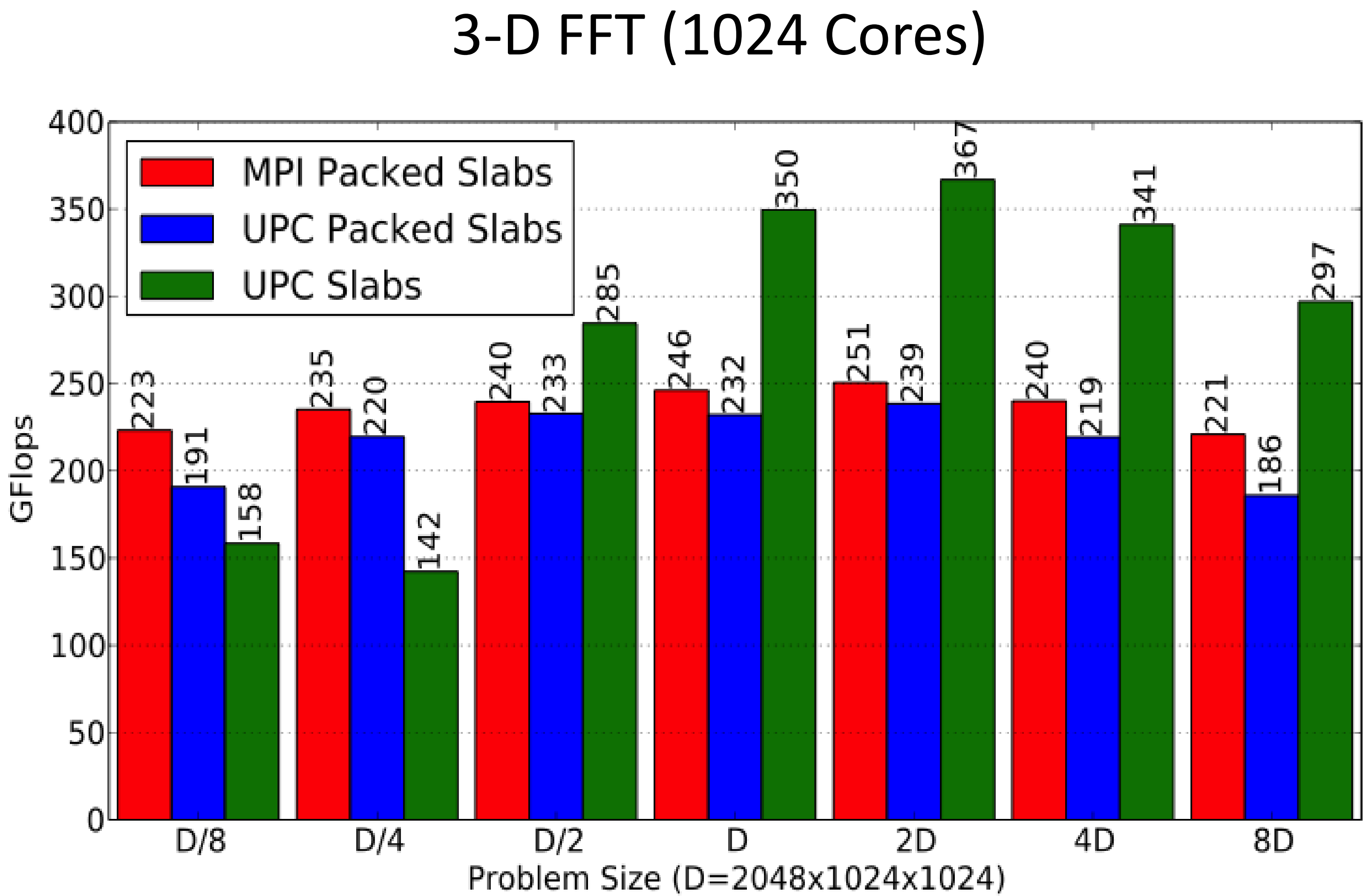
# Matrix-Multiplication on Cray XT4

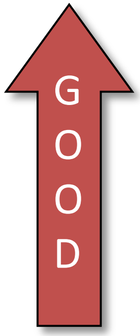


# Choleskey Factorization on Sun Constellation (Infiniband)



# FFT Performance on Cray XT4

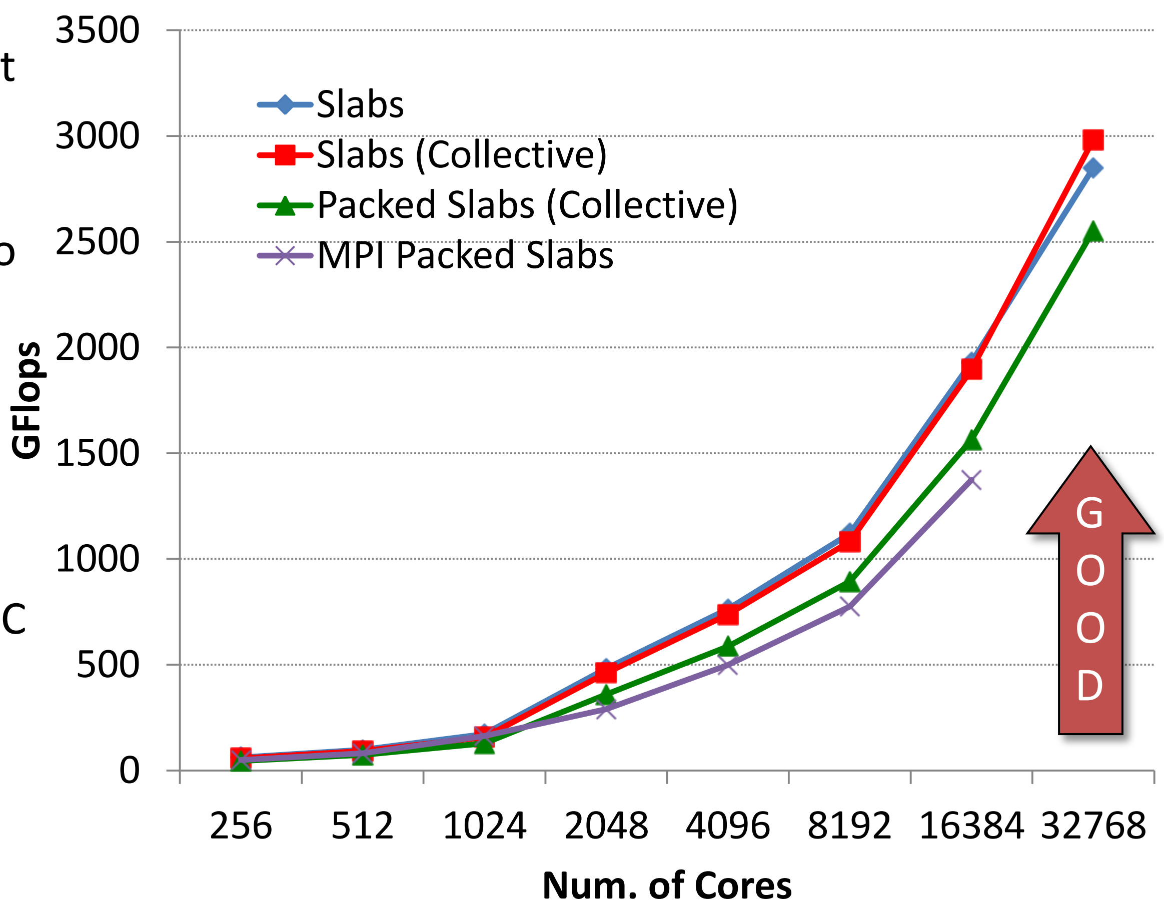


# FFT Performance on BlueGene/P

- PGAS implementations consistently outperform MPI
- Leveraging communication and computation overlaps yields best performance
  - More collectives in flight and more communication leads to better performance
  - At 32k cores, overlap algorithms yield 17% improvement in overall application time
- Numbers are getting close to HPC record
  - Future work to try to beat the record

HPC Challenge Peak as of July 09 is ~4.5 TFlops on 128k Cores



# Summary

- Demonstrated scalability to tens of thousands of cores
- Global address space improves productivity
- Data partitioning enables performance optimizations
- Interoperable with other programming models and languages including MPI, FORTRAN, C++
- Growing UPC community with actively developed and maintained software implementations
  - Berkeley UPC and GASNet: http://upc.lbl.gov
  - Other UPC compilers: Cray UPC, GCC UPC, HP UPC, IBM UPC, MTU UPC