UPC AT SCALE

Yili Zheng
Lawrence Berkeley National Laboratory

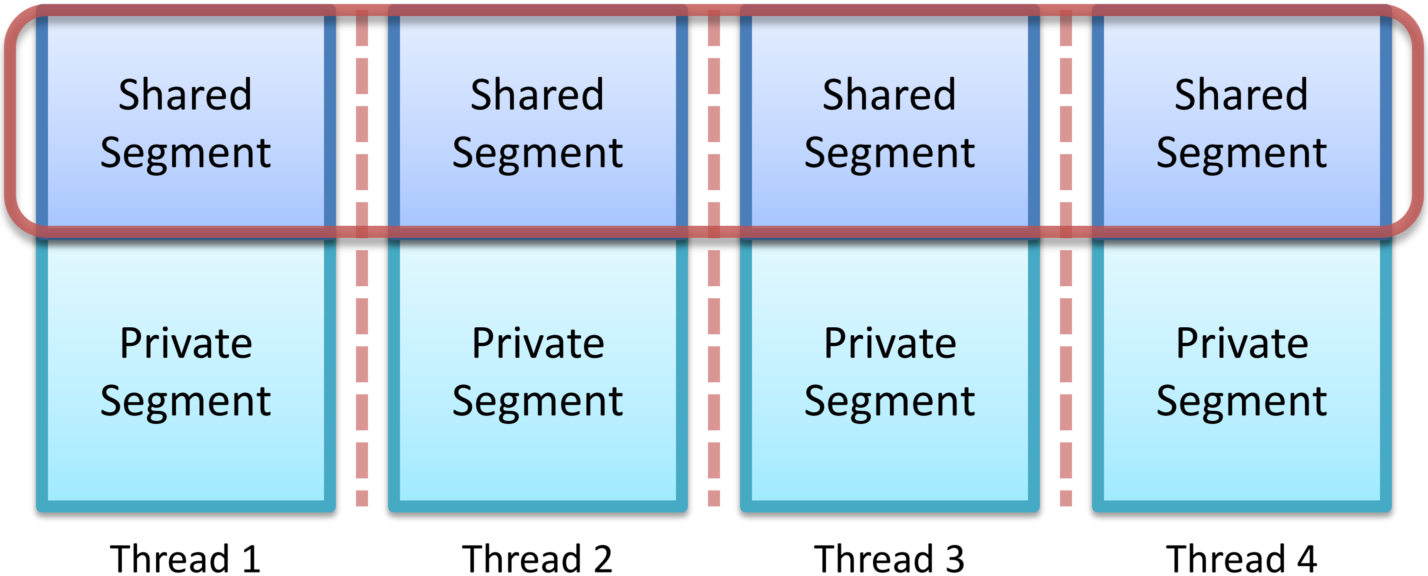
Berkeley UPC Team

- Project Lead: Katherine Yelick
- Team 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

Motivation

- Scalable systems have either distributed memory or shared memory without cache coherency
  - Clusters: Ethernet, Infiniband, CRAY XT, IBM BlueGene
  - Hybrid nodes: CPU + GPU or other kinds of accelerators
  - SoC: IBM Cell, Intel Single-chip Cloud Computer (SCC)
- Challenges of Message Passing programming models
  - Difficult data partitioning for irregular applications
  - Memory space starvation due to data replication
  - Performance overheads from two-sided communication semantics

Partitioned Global Address Space

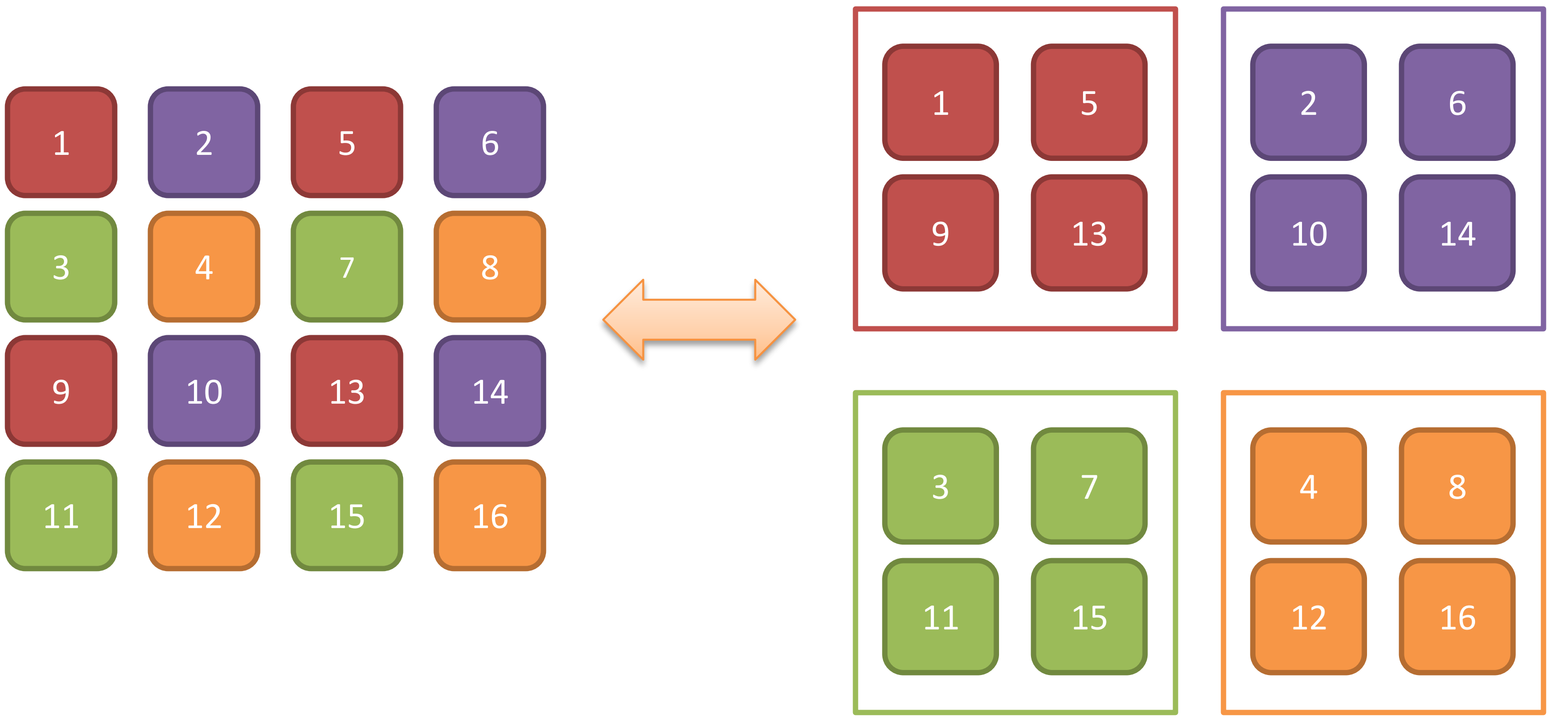


- Global data view abstraction for productivity
- Vertical partitions among threads for locality control
- Horizontal partitions between shared and private segments for data placement optimizations
- Friendly to non-cache-coherent architectures

PGAS Example: Global Matrix Distribution

Global Matrix View

Distributed Matrix Storage



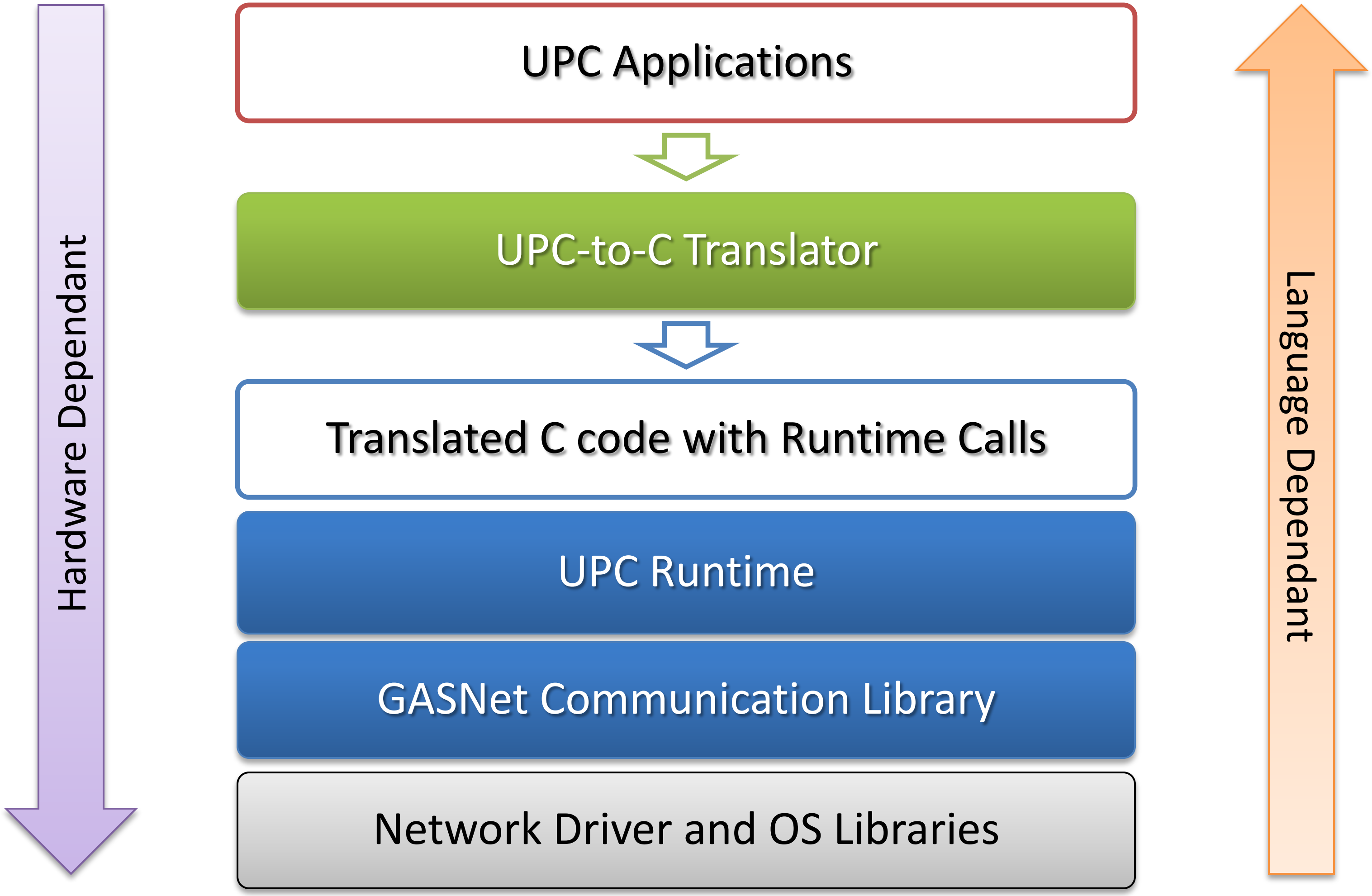
UPC Overview

- 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

Key Components for Scalability

- One-sided communication and active messages
- Efficient resource sharing for multi-core systems
- Non-blocking collective communication

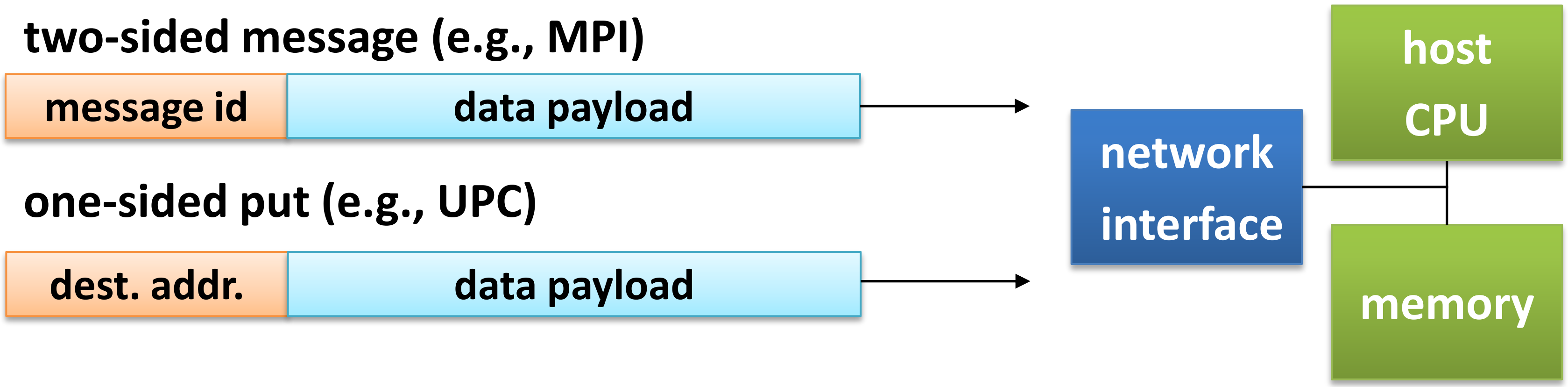
Berkeley UPC Software Stack



Berkeley UPC Features

- Data transfer for complex data types (vector, indexed, stride)
- Non-blocking memory copy
- Point-to-point synchronization
- Remote atomic operations
- Active Messages
- Extension to UPC collectives
- Portable timers

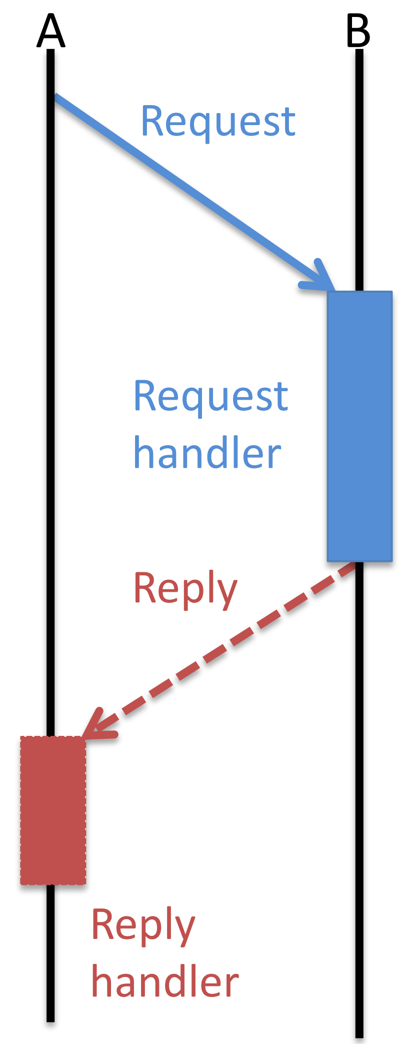
One-Sided vs. Two-Sided Messaging



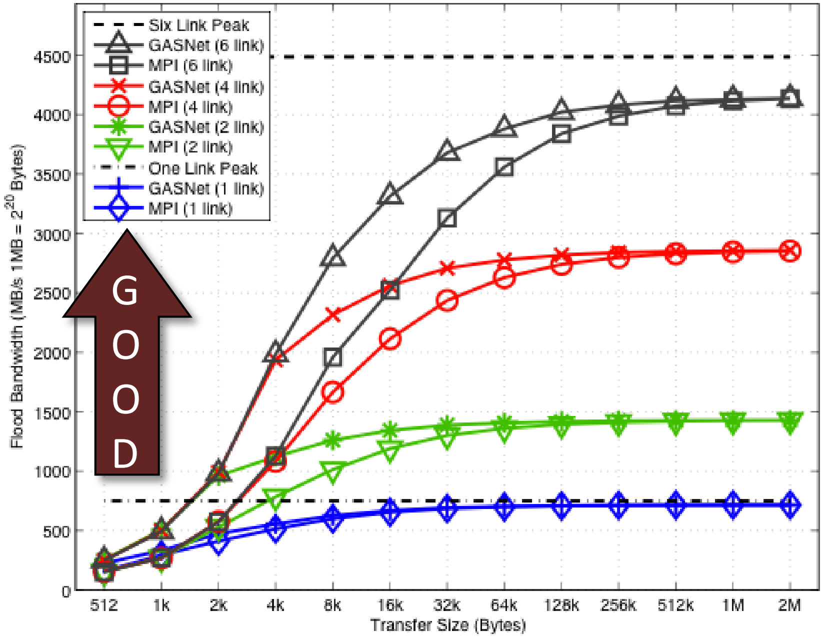
- Two-sided messaging
  - Message does not contain information about the final destination; need to look it up on the target node
  - Point-to-point synchronization implied with all transfers
- One-sided messaging
  - Message contains information about the final destination
  - Decouple synchronization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

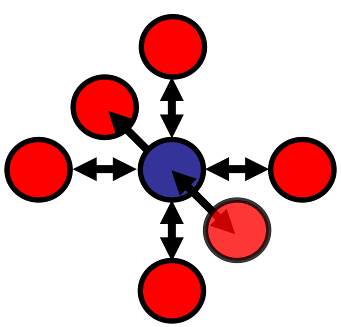
Active Messages

- 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



GASNet Bandwidth on BlueGene/P



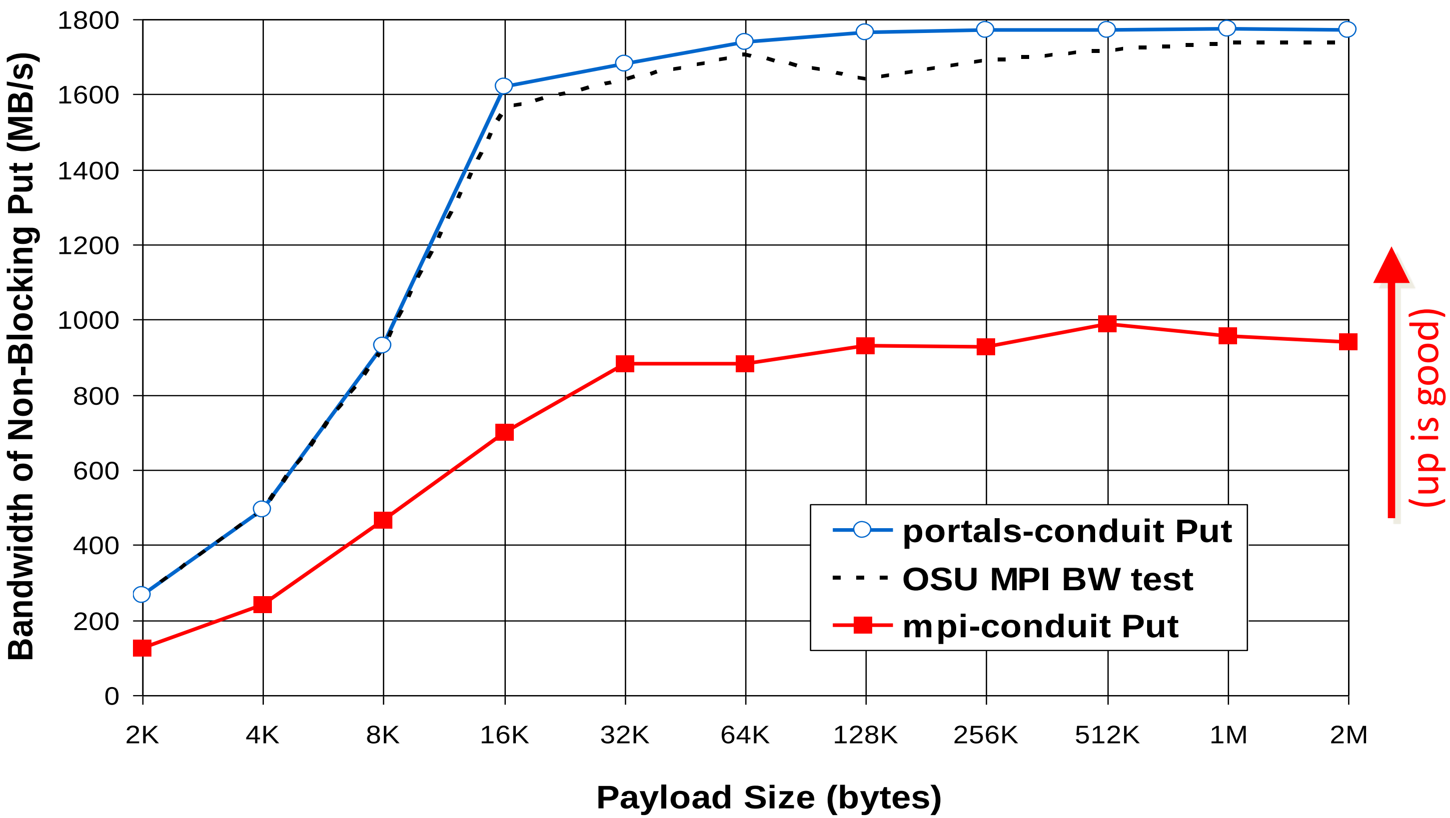


* Kumar et. al showed the maximum achievable bandwidth for DCMF transfers is 748 MB/s per link so we use this as our peak bandwidth
See "The deep computing messaging framework: generalized scalable message passing on the blue gene/P supercomputer", Kumar et al. ICS08

- 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

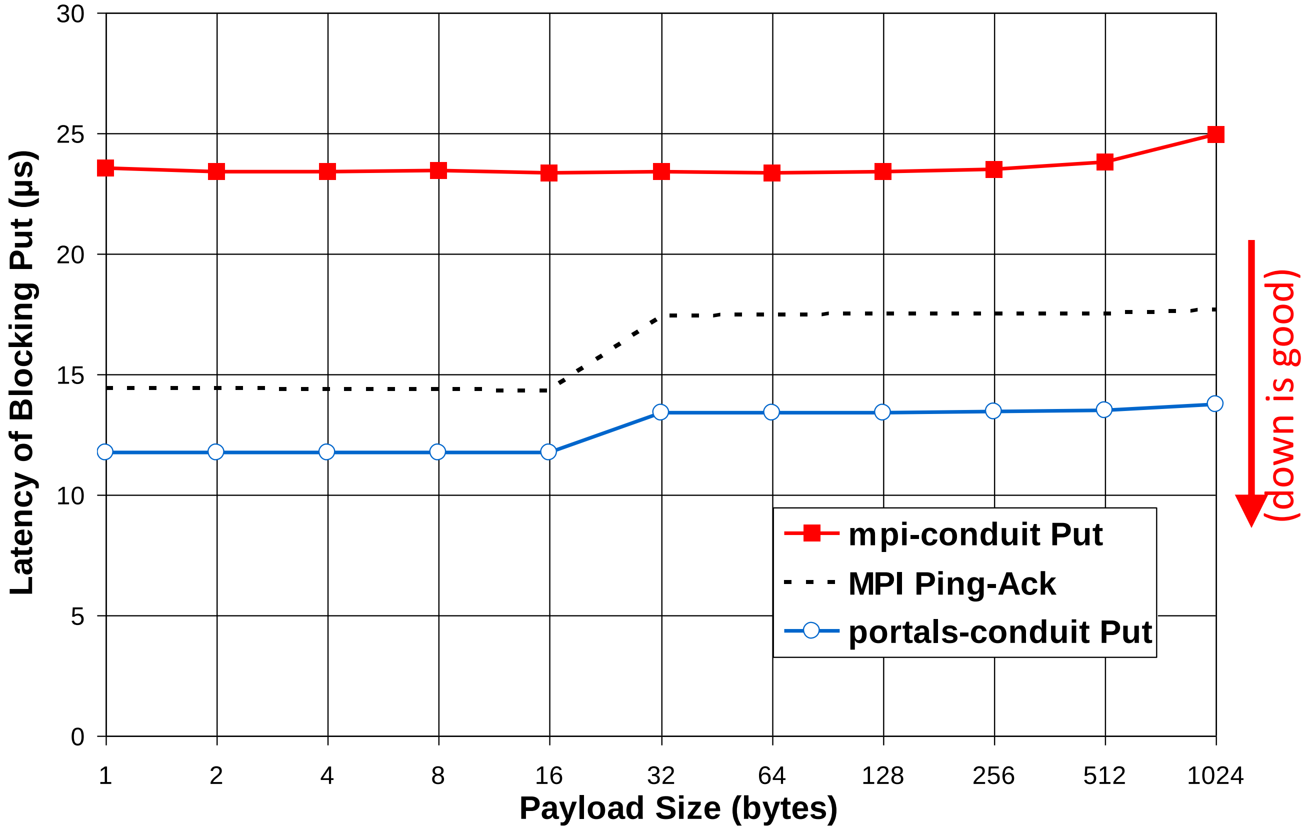
See "Scaling Communication Intensive Applications on BlueGene/P Using One-Sided Communication and Overlap", Rajesh Nishtala, Paul Hargrove, Dan Bonachea, and Katherine Yelick, *IPDPS 2009*

GASNet Bandwidth on Cray XT4



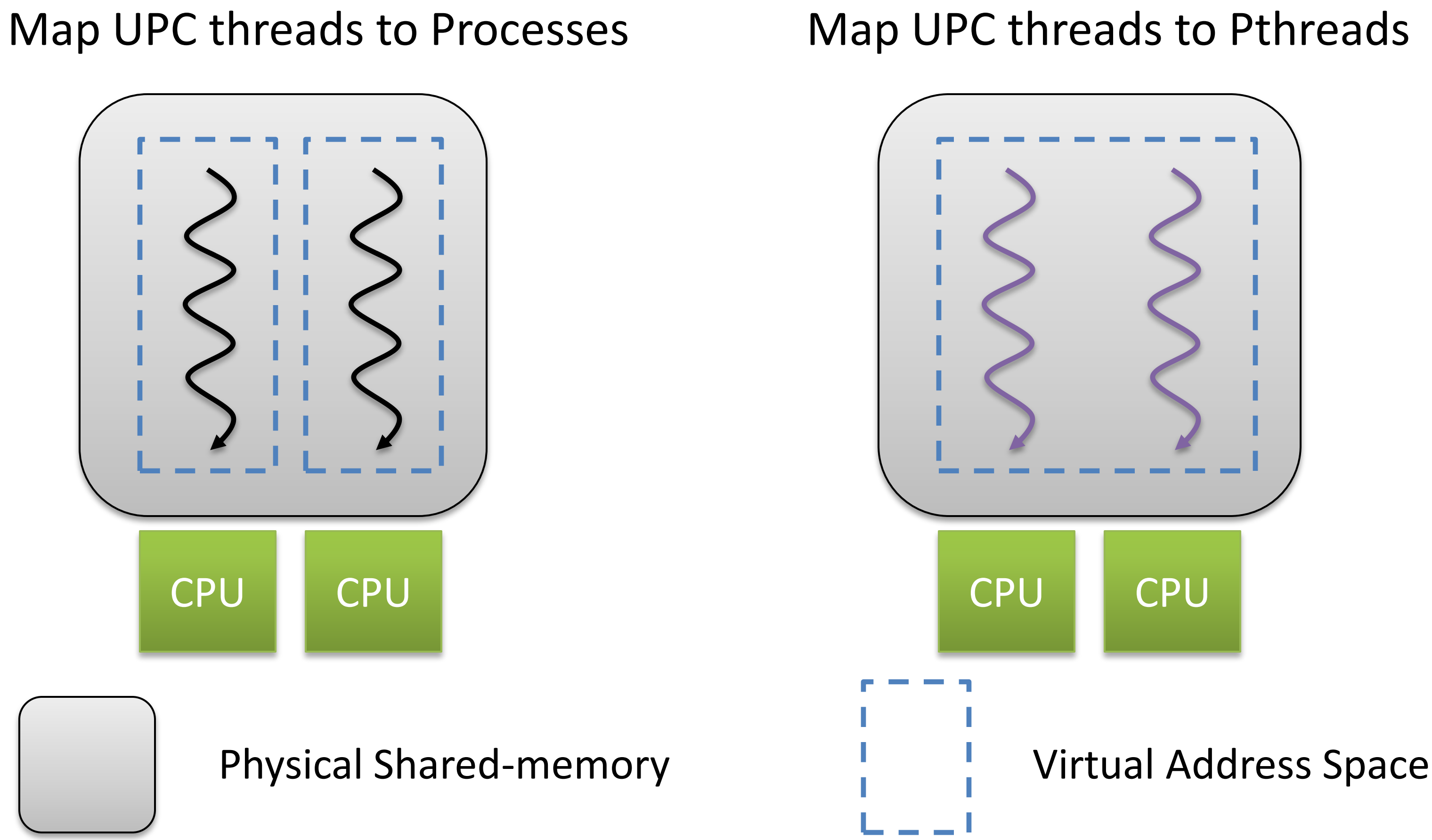
Slide source: Porting GASNet to Portals: Partitioned Global Address Space (PGAS) Language Support for the Cray XT, Dan Bonachea, Paul Hargrove, Michael Welcome, Katherine Yelick, CUG 2009

GASNet Latency on Cray XT4

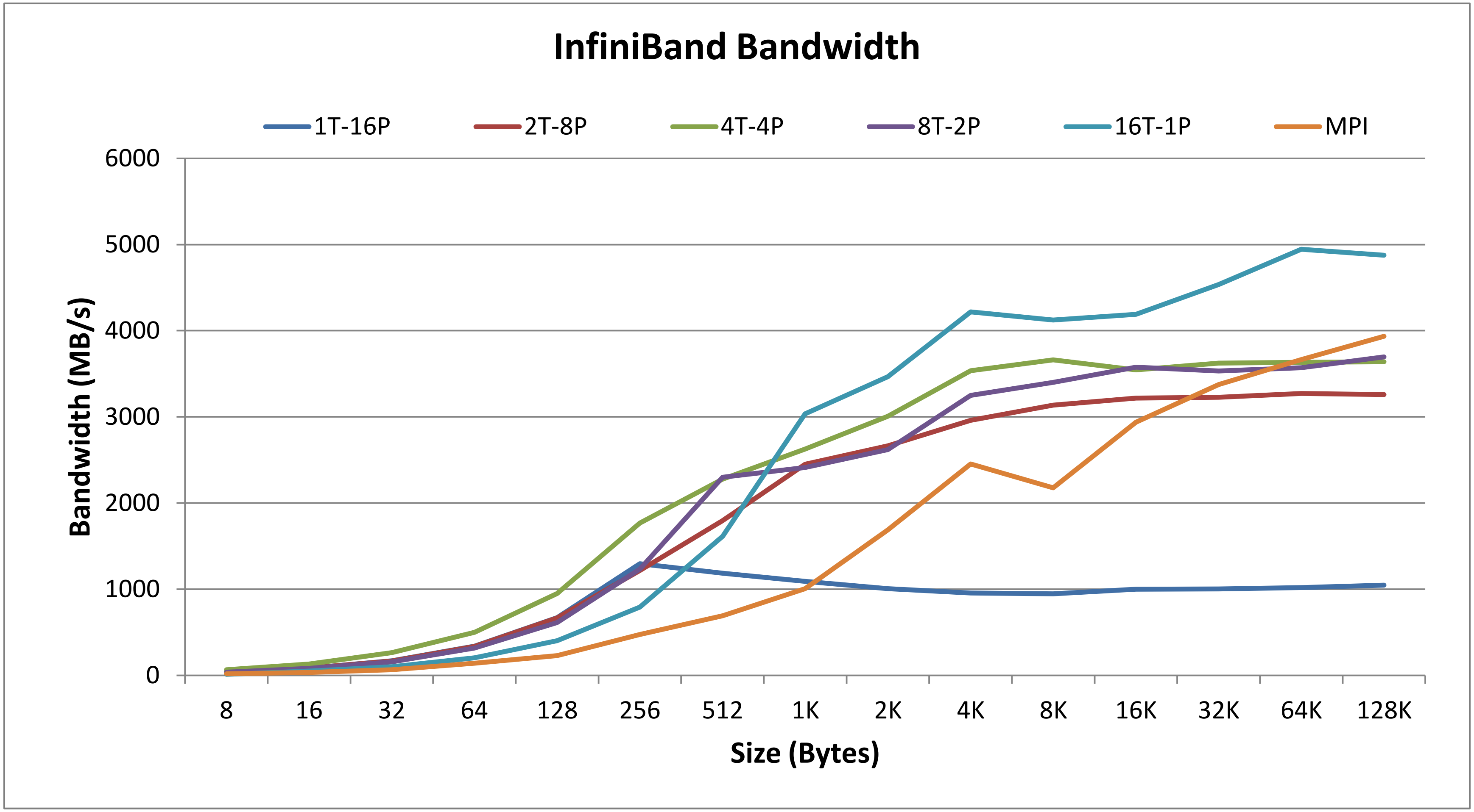


Slide source: Porting GASNet to Portals: Partitioned Global Address Space (PGAS) Language Support for the Cray XT, Dan Bonachea, Paul Hargrove, Michael Welcome, Katherine Yelick, CUG 2009

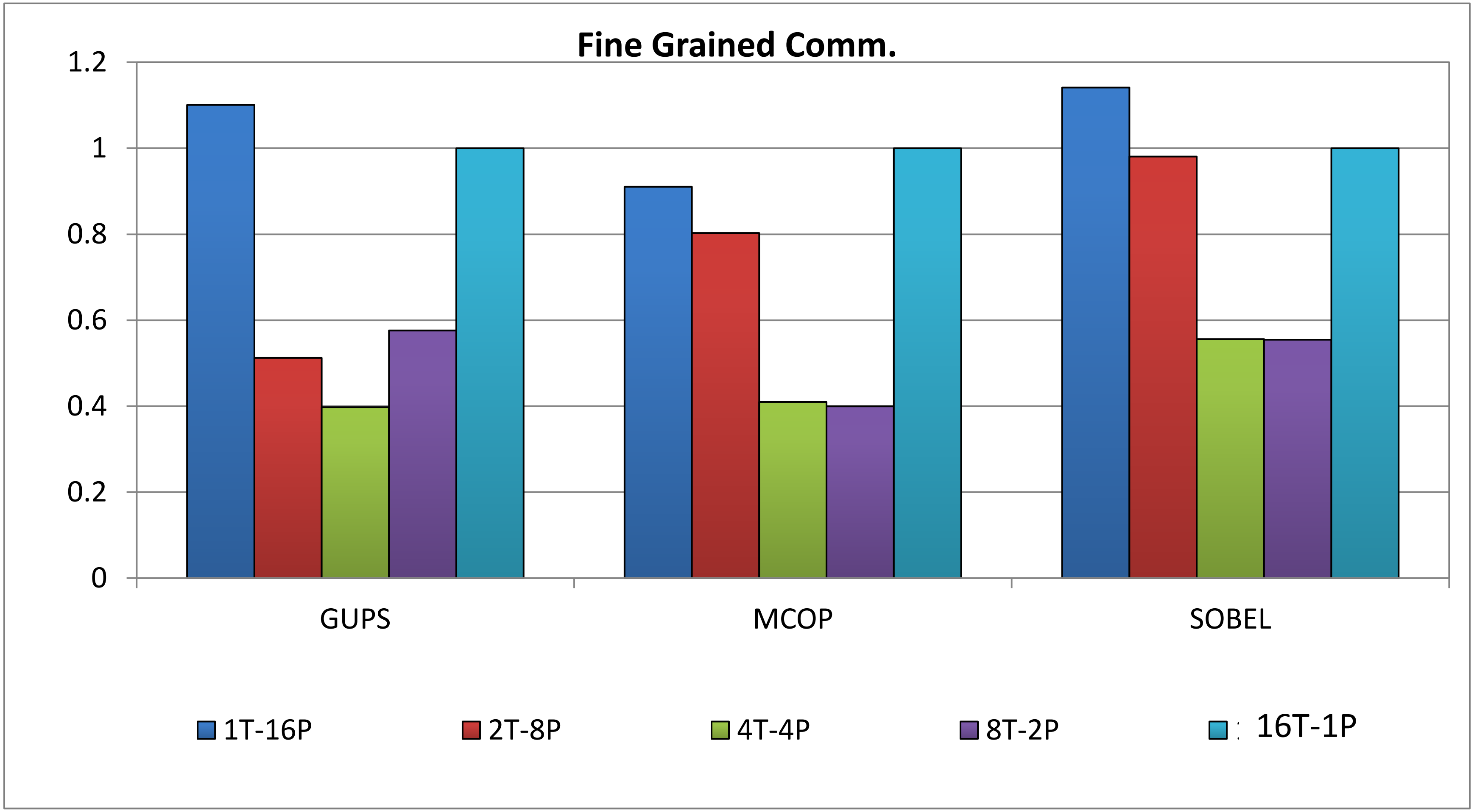
Execution Models on Multi-core – Process vs. Thread



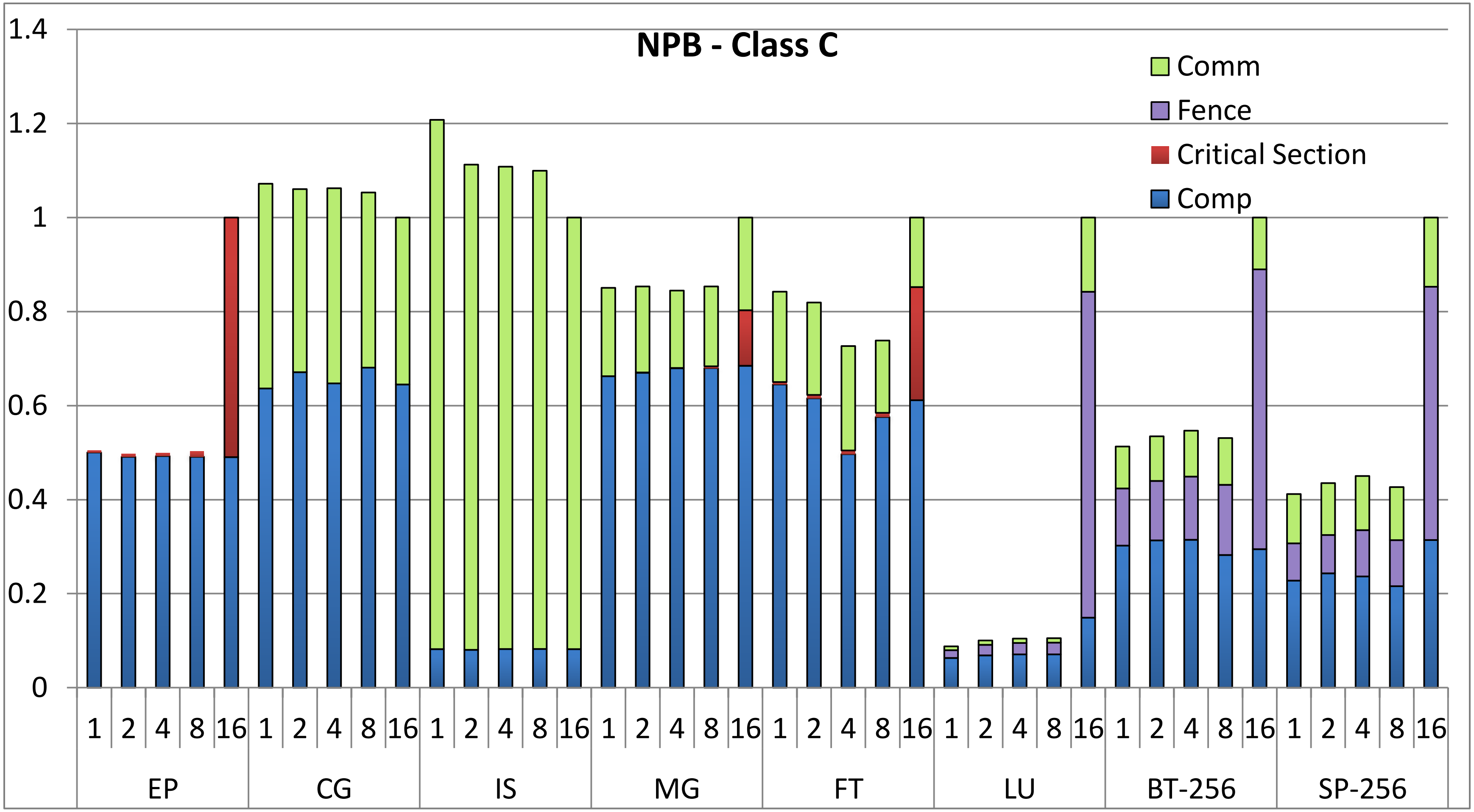
Point-to-Point Performance – Process vs. Thread



Application Performance – Process vs. Thread



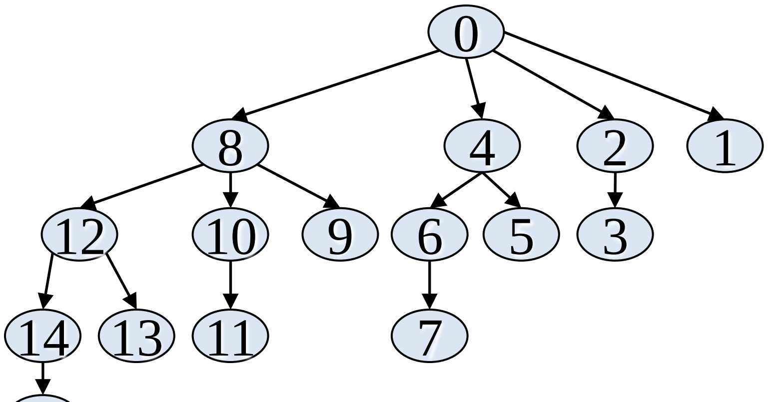
NAS Parallel Benchmarks – Process vs. Thread



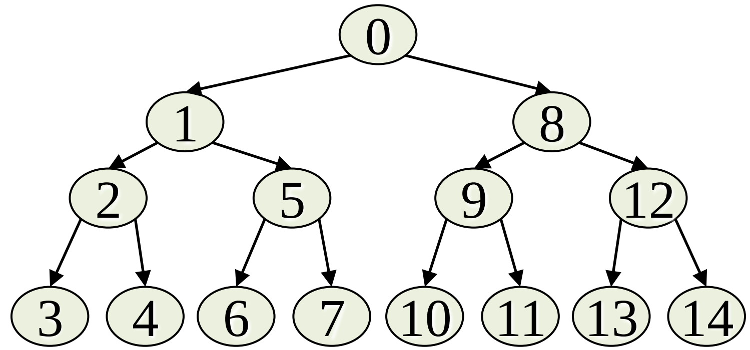
Collective Communication for PGAS

- Communication patterns similar to MPI: broadcast, reduce, gather, scatter and alltoall
- Global address space enables one-sided collectives
- Flexible synchronization modes provide more communication and computation overlapping opportunities

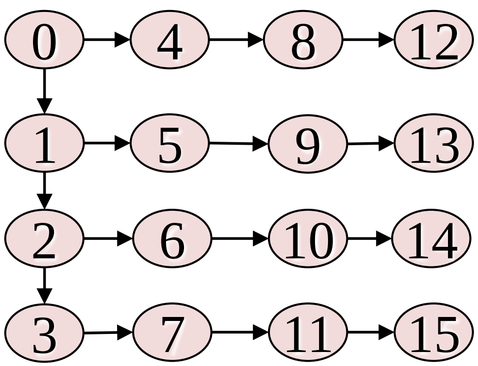
Collective Communication Topologies



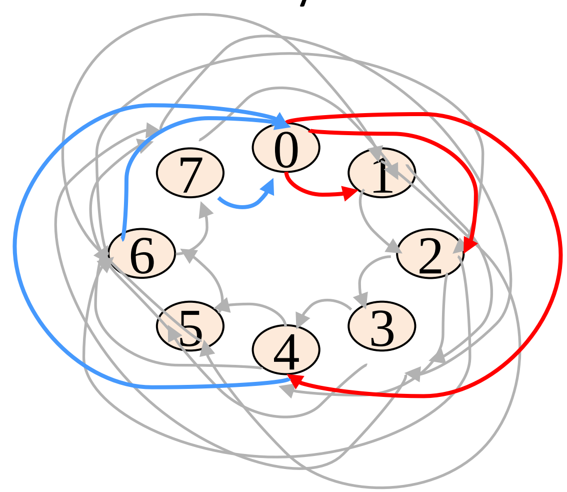
binomial tree



Binary Tree

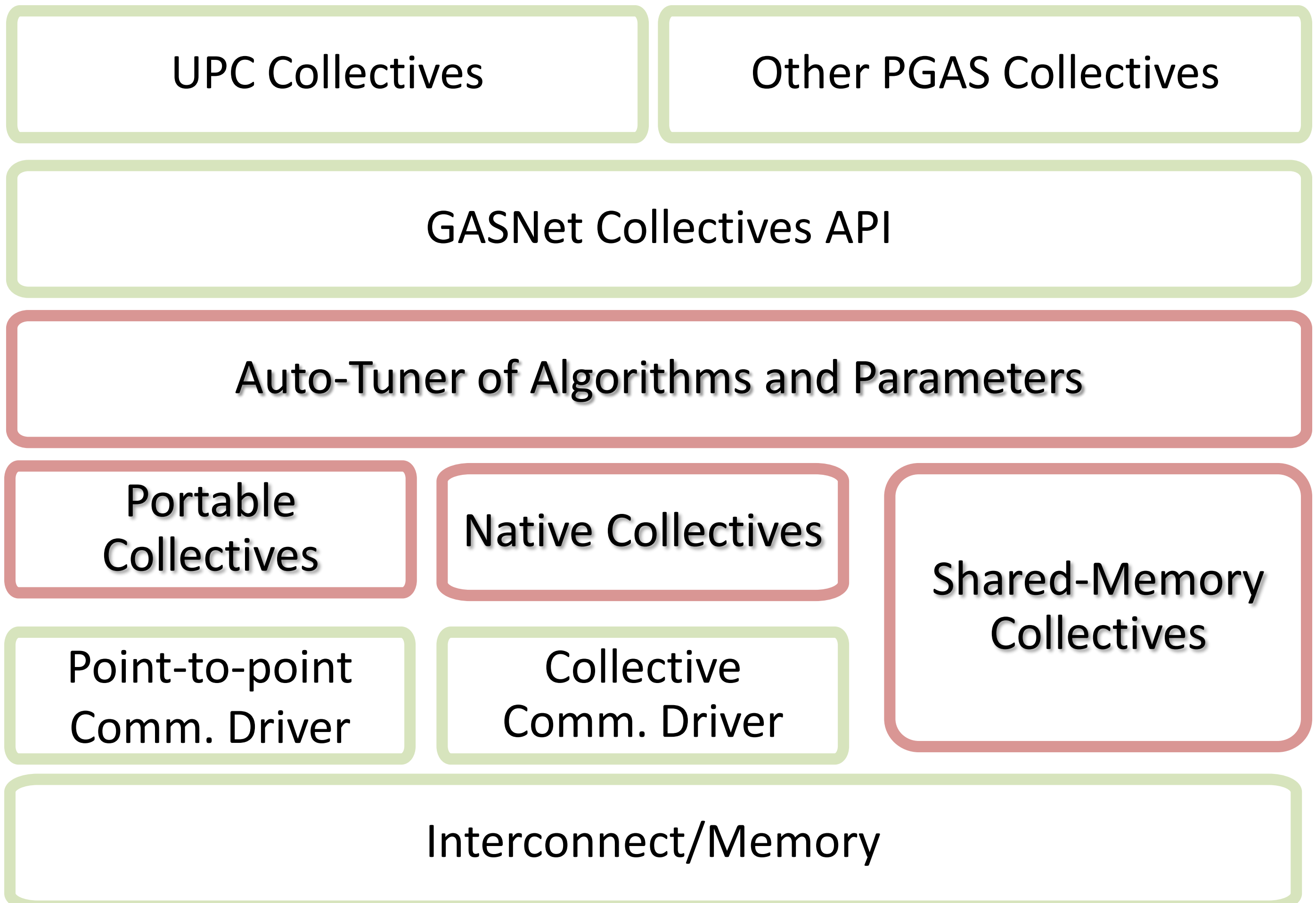


Fork Tree



Radix 2 Dissemination

GASNet Module Organization



Auto-tuning Collective Communication

Offline tuning

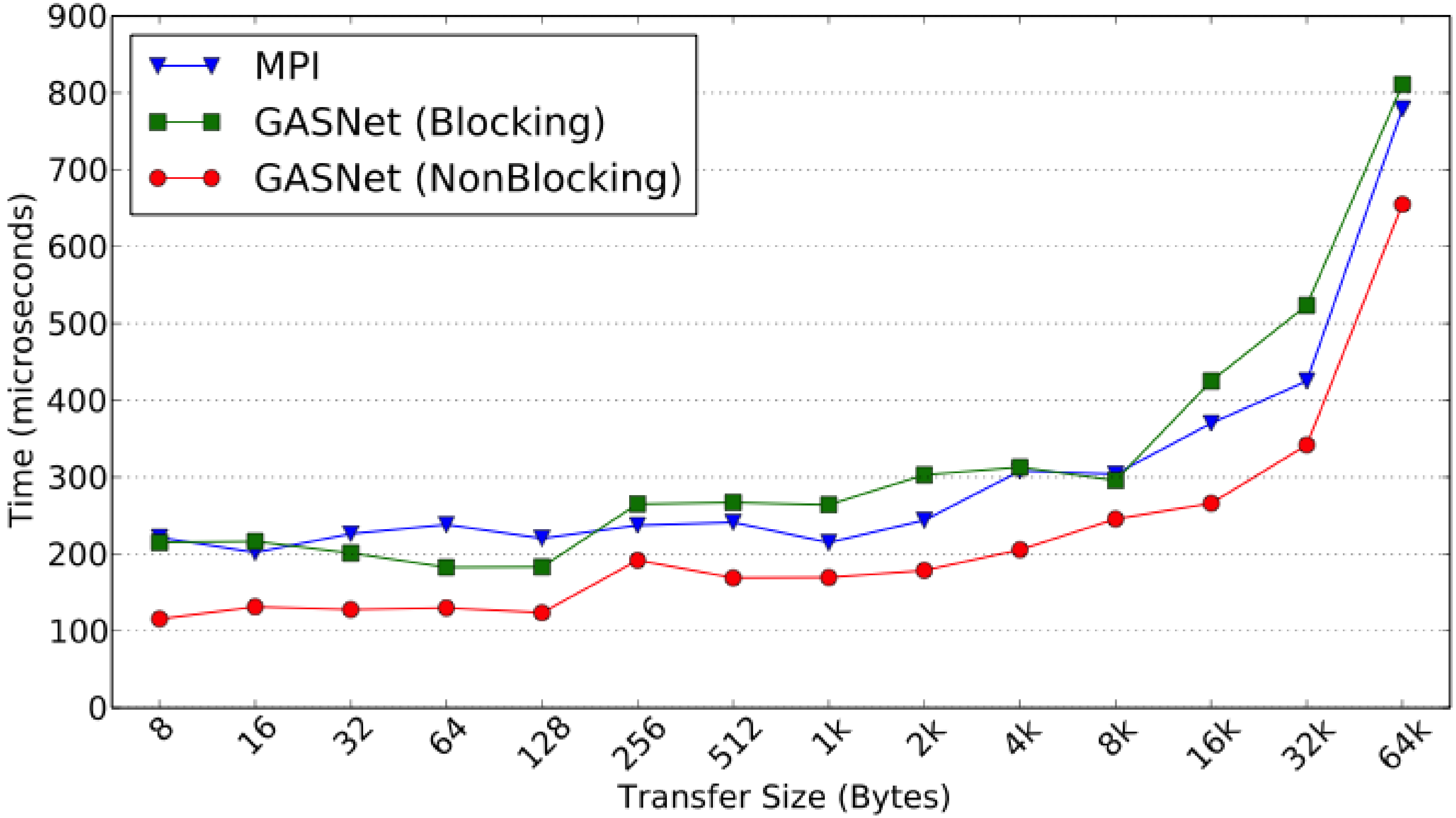
- Optimize for platform common characteristics
- Minimize runtime tuning overhead

Online tuning

- Optimize for application runtime characteristics
- Refine offline tuning results

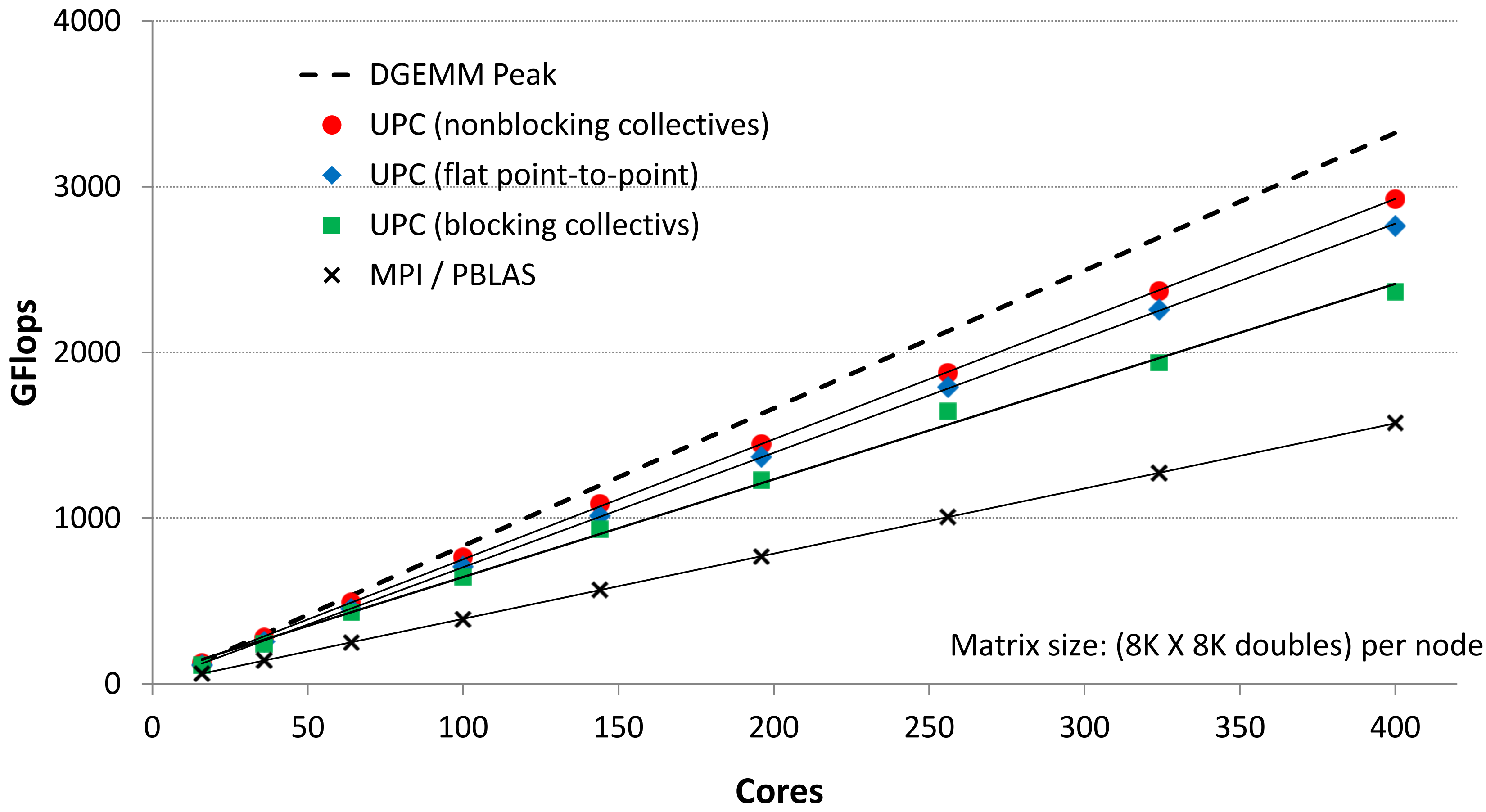
Performance Influencing Factors	**Performance Tuning Space**
Hardware ▪ CPU ▪ Memory system ▪ Interconnect Software ▪ Application ▪ System software Execution ▪ Process/thread layout ▪ Input data set ▪ System workload	Algorithm selection ▪ Eager vs. rendezvous ▪ Put vs. get ▪ Collection of well-known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

Broadcast Performance

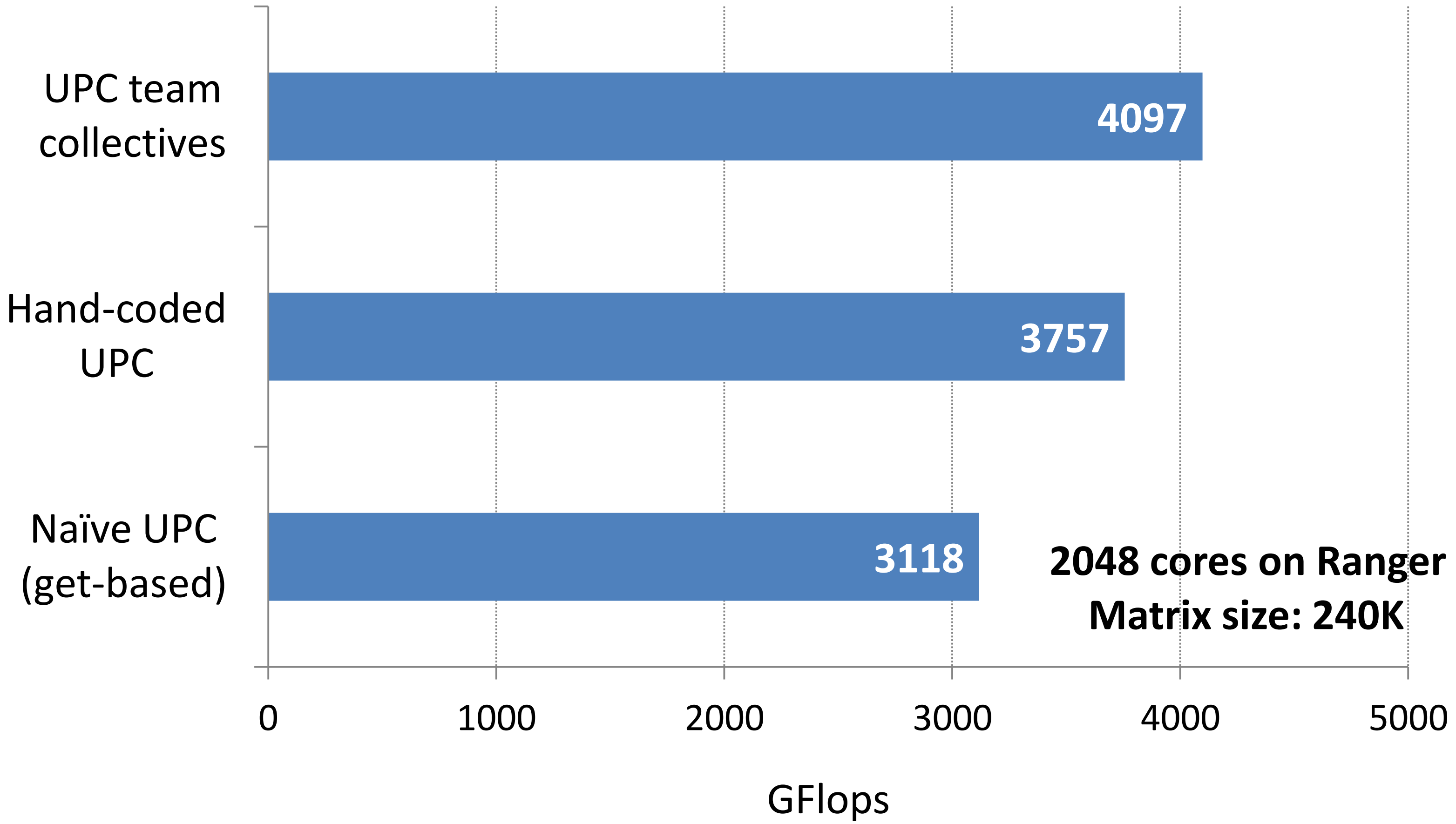


Cray XT4 Nonblocking Broadcast (1024 Cores)

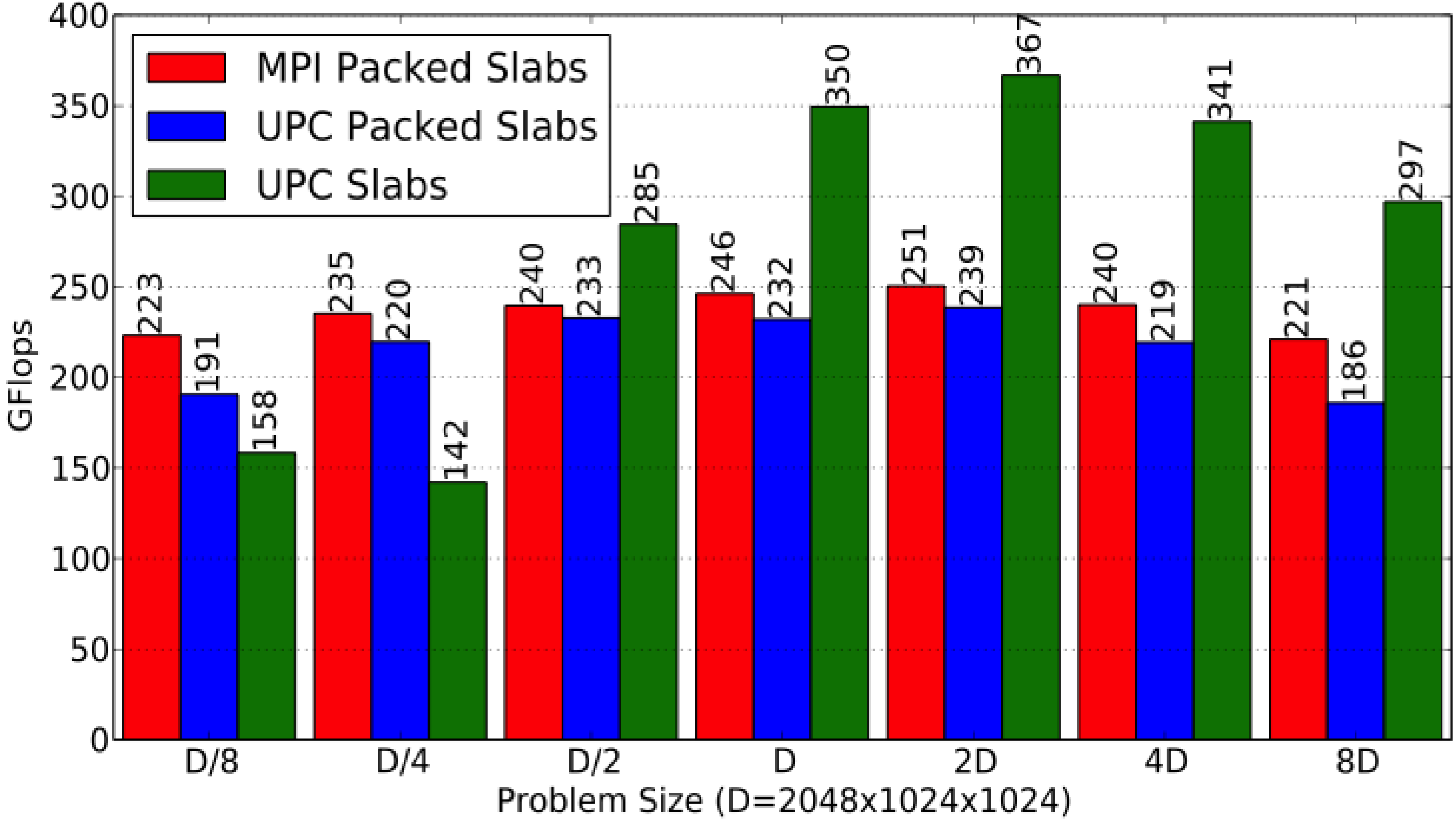
Matrix-Multiplication on Cray XT4



Choleskey Factorization on Sun Constellation (Infiniband)

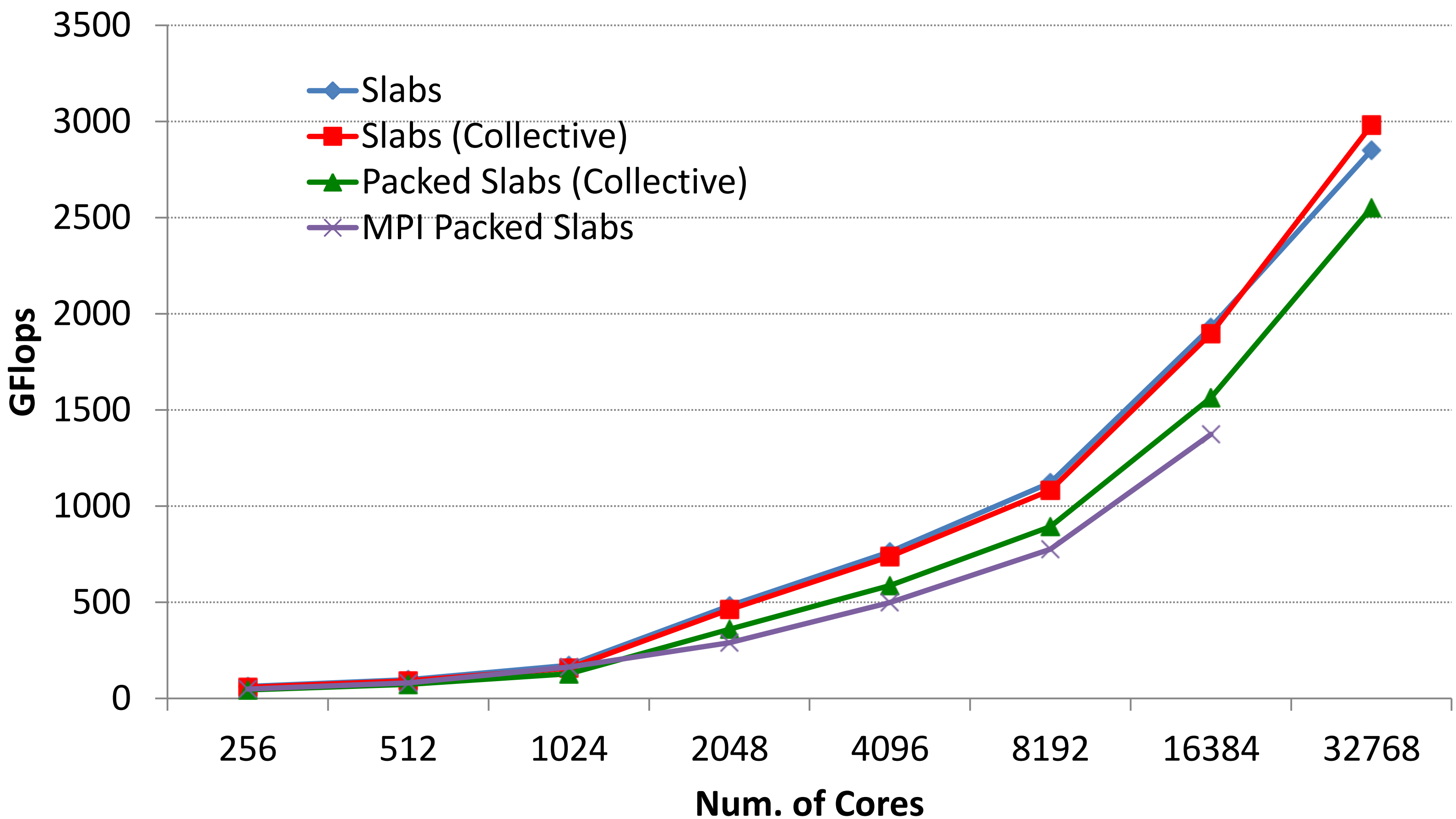


FFT Performance on Cray XT4



(1024 Cores)

FFT Performance on BlueGene/P



MPI FFT of HPC Challenge as of July 09 is ~4.5 Tflops on 128k Cores.

Summary

- PGAS provides programming convenience similar to shared-memory models
- UPC has demonstrated good performance comparable to MPI at large scale.
- Interoperable with other programming models and languages including MPI, FORTRAN and C++
- Growing UPC community with actively developed and maintained software implementations
  - Berkeley UPC and GASNet: http://upc.lbl.gov
  - Other UPC compilers: Cray UPC, GNU UPC, HP UPC and IBM UPC
  - Tools: TotalView and Parallel Performance Wizard (PPW)