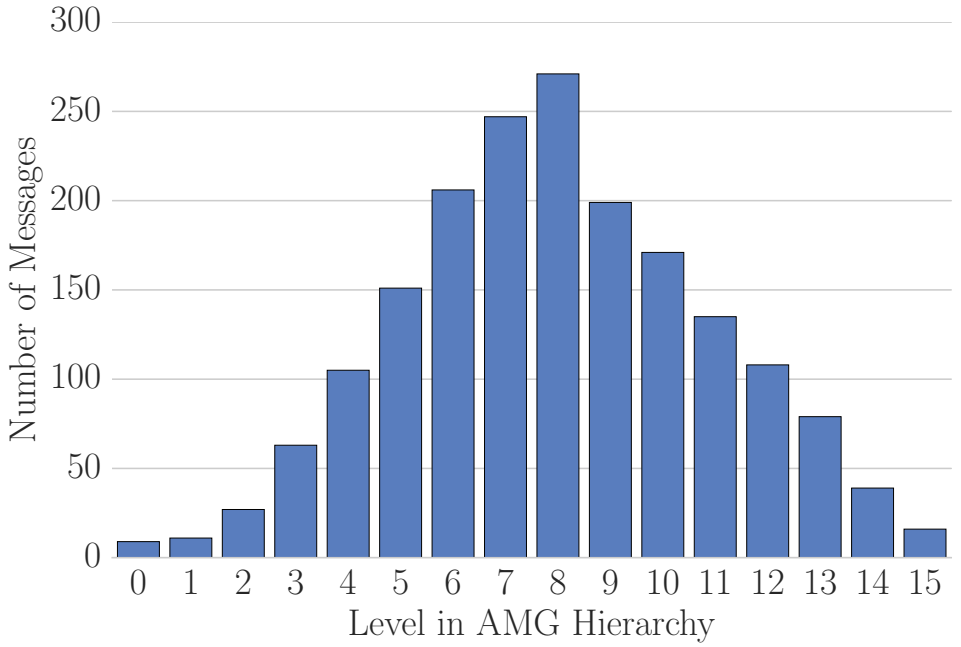
Locality-Aware Communication

Amanda Bienz

Assistant Professor
Department of Computer Science
University of New Mexico

Motivation

- Communication is typically the bottleneck in irregular parallel applications
- Often, each application or solver will implement their own communication optimizations
  - Some really clever approaches! But no central knowledge, so people keep reinventing the wheel
- Many parallel codebases have existed for decades
  - Want to optimize performance with minimal changes to existing codebases

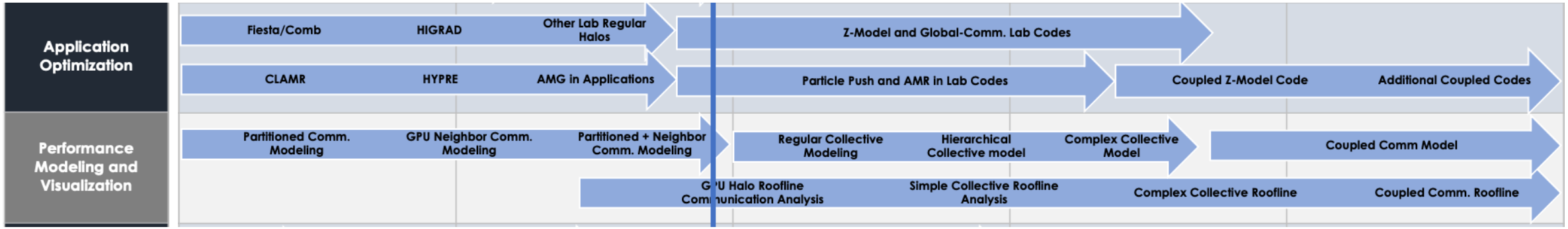


Approach

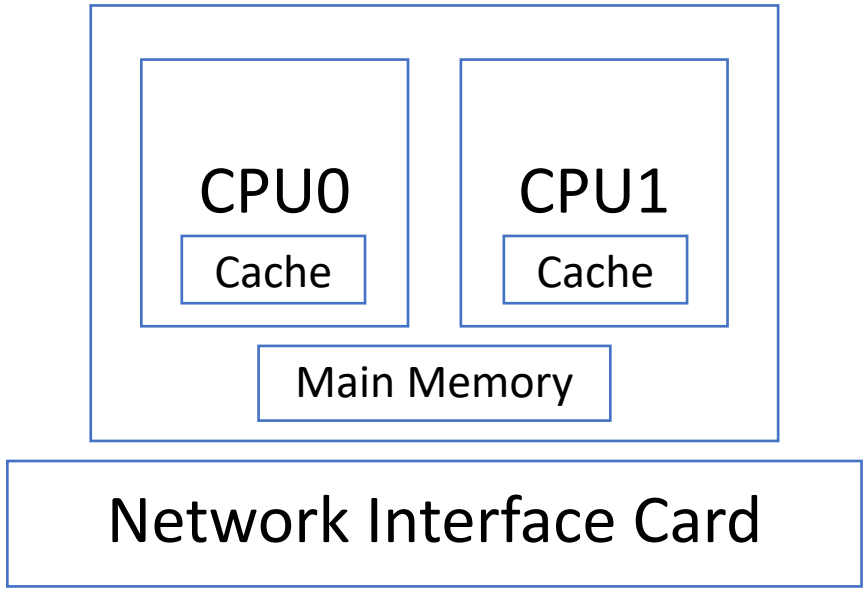
1. Profile systems and form representative performance models

2. Use performance models to create communication optimizations

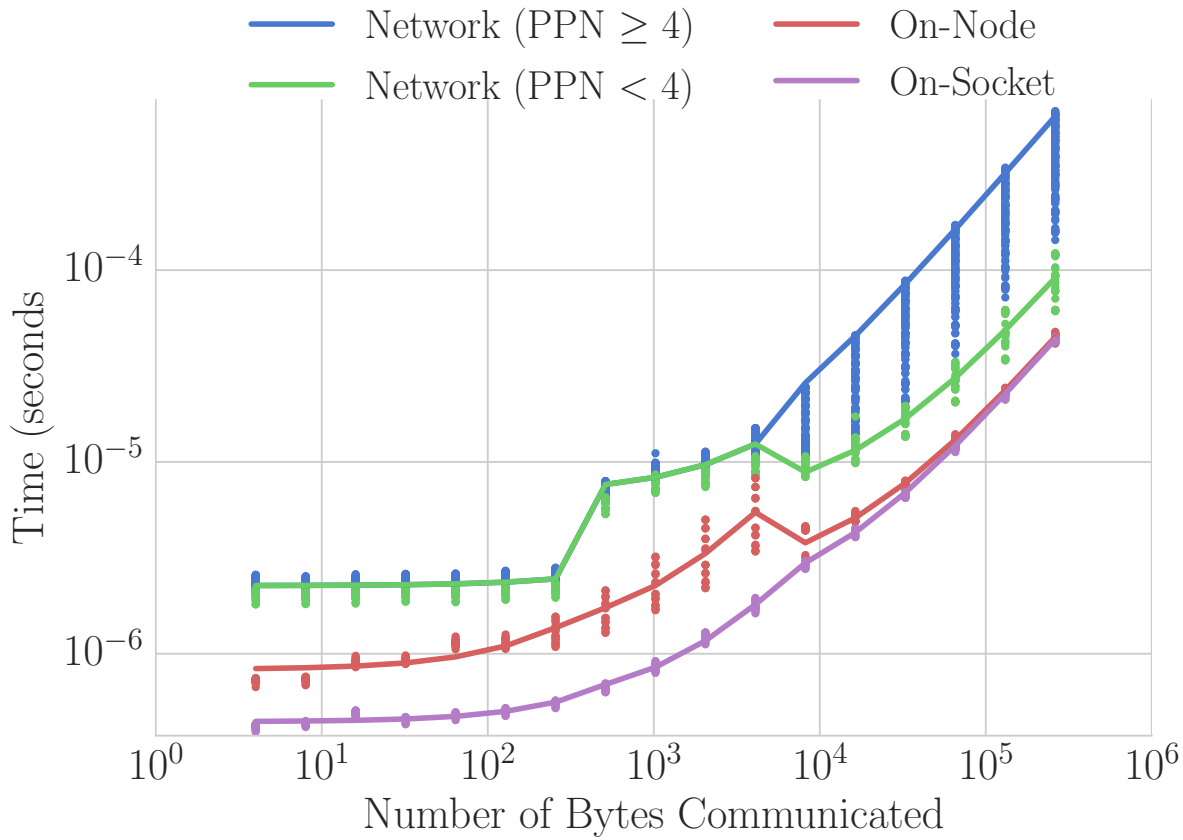
3. Add optimizations to MPI Advance to improve performance of existing applications : **Derek Schafer discussed research infrastructure**



Communication on SMP Architectures

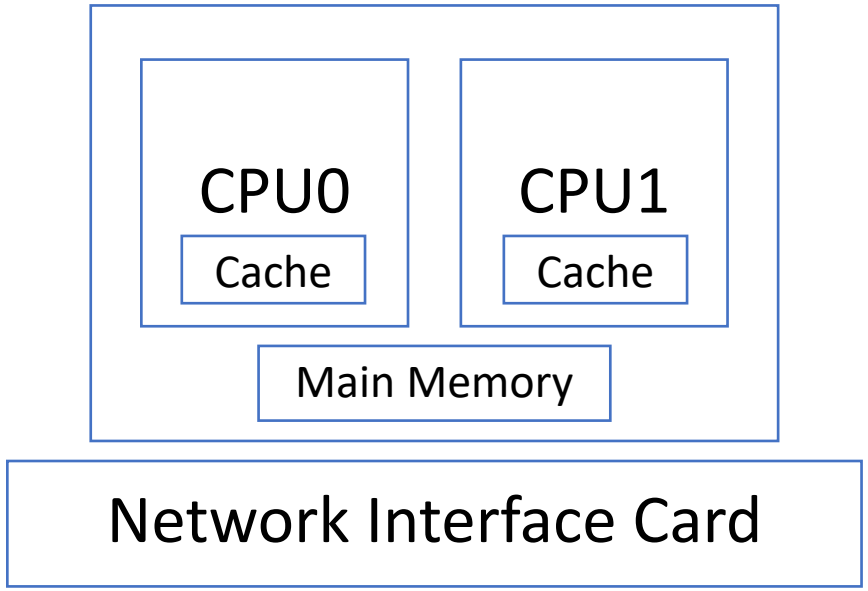


Symmetric Multiprocessing (SMP) Node

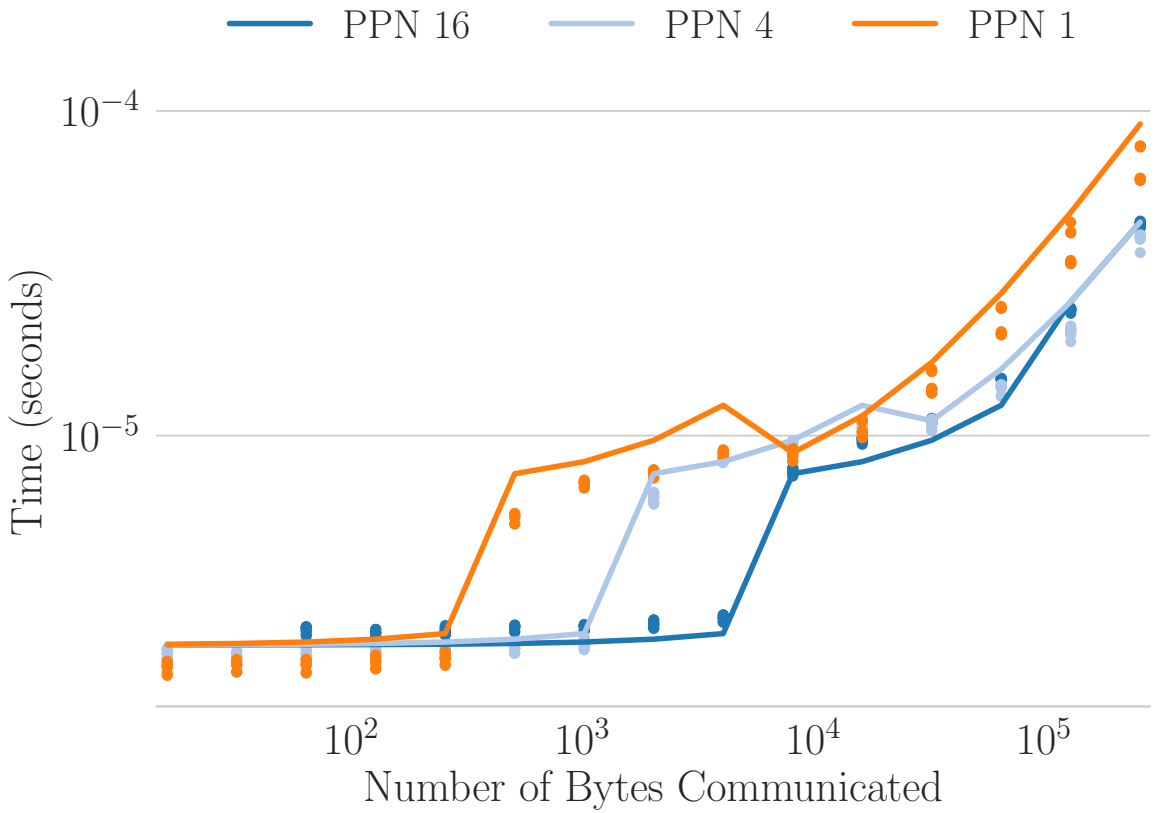


Key Takeaway : Intra-socket << Intra-node/Inter-socket << Inter-node

Communication on SMP Architectures



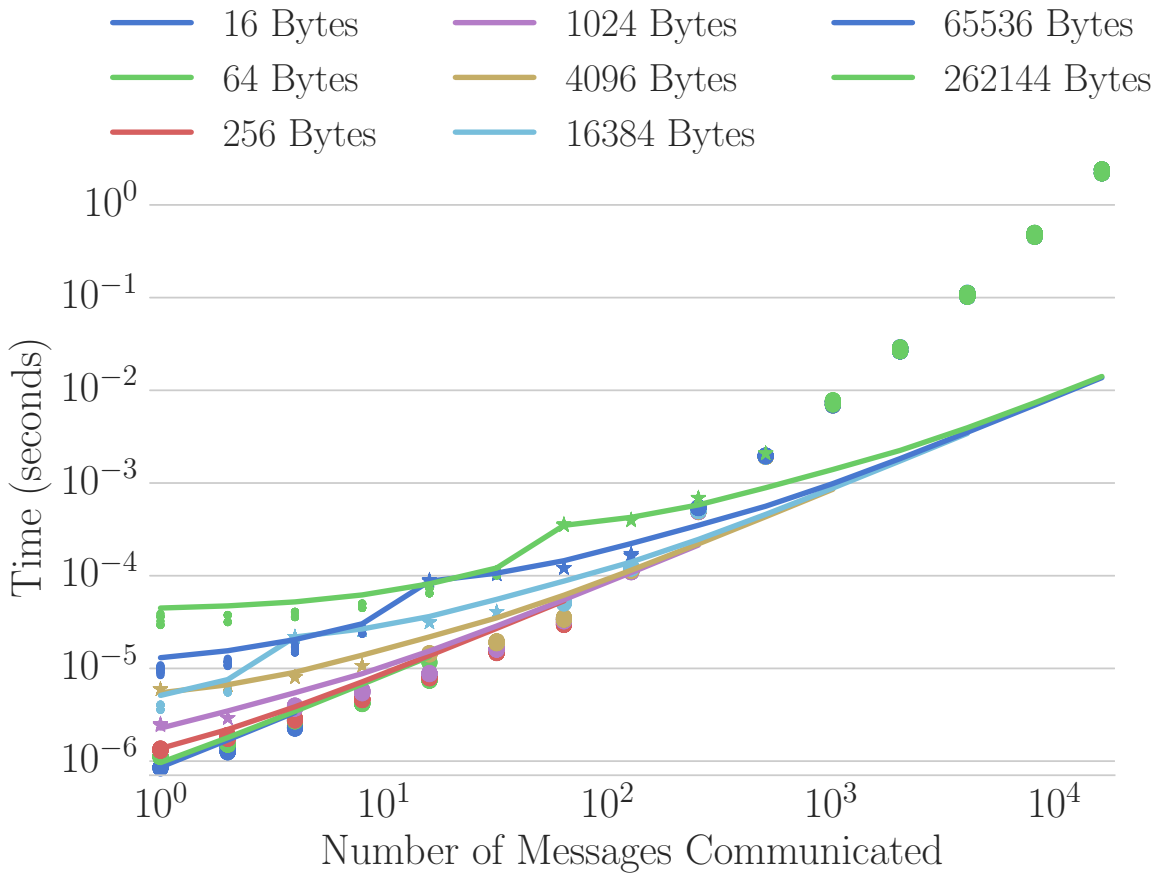
Symmetric Multiprocessing (SMP) Node



Key Takeaway : Inter-node data should be evenly distributed across all CPU cores per node

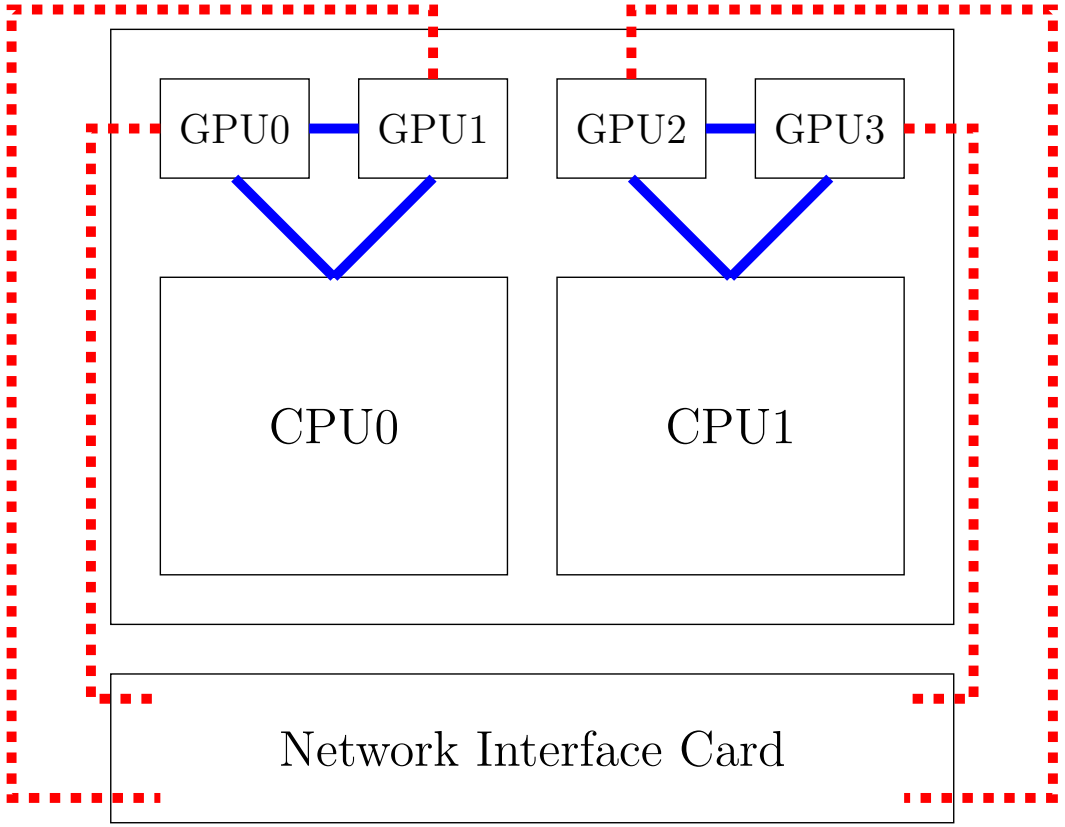
Communication on SMP Architectures

- Sending multiple messages, max-rate model vs timings (dots)
- Receiving out of order (first message matches last MPI_Irecv)
- Large message counts : orders of magnitude overhead from queue search (matching) costs



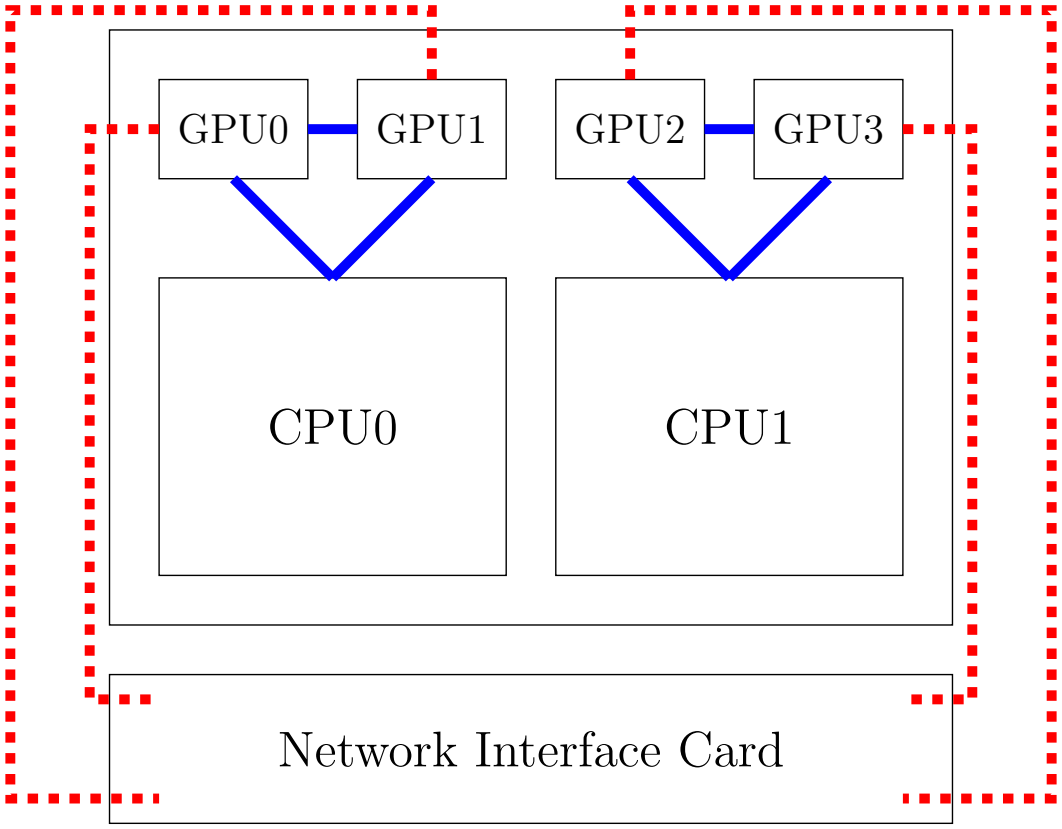
Key Takeaway : Large overhead from receiving large number of messages at once

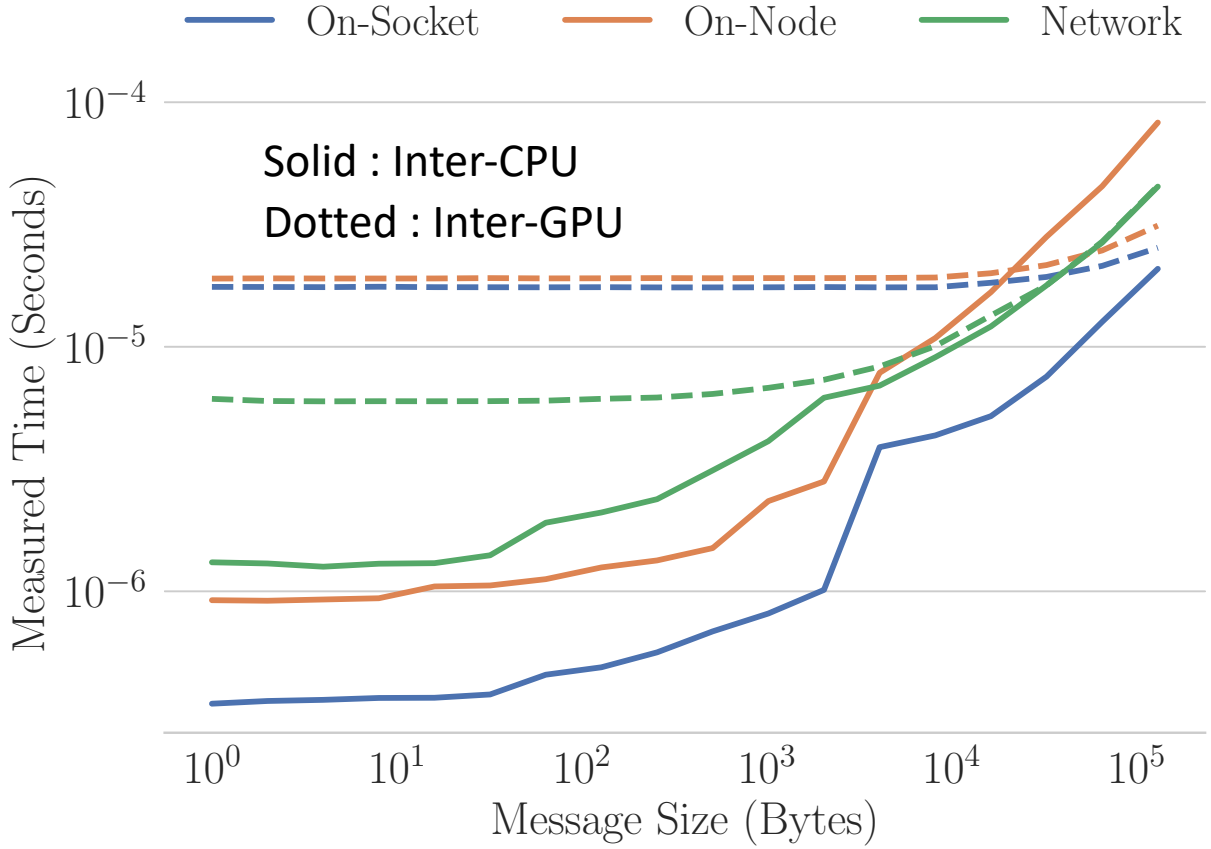
Communication on Heterogeneous Architectures



- Many of fastest computers have GPUs on each node
- Computation is performed on GPUs
- Need to communicate between GPUs
- Much more complicated
  - Many paths of communication

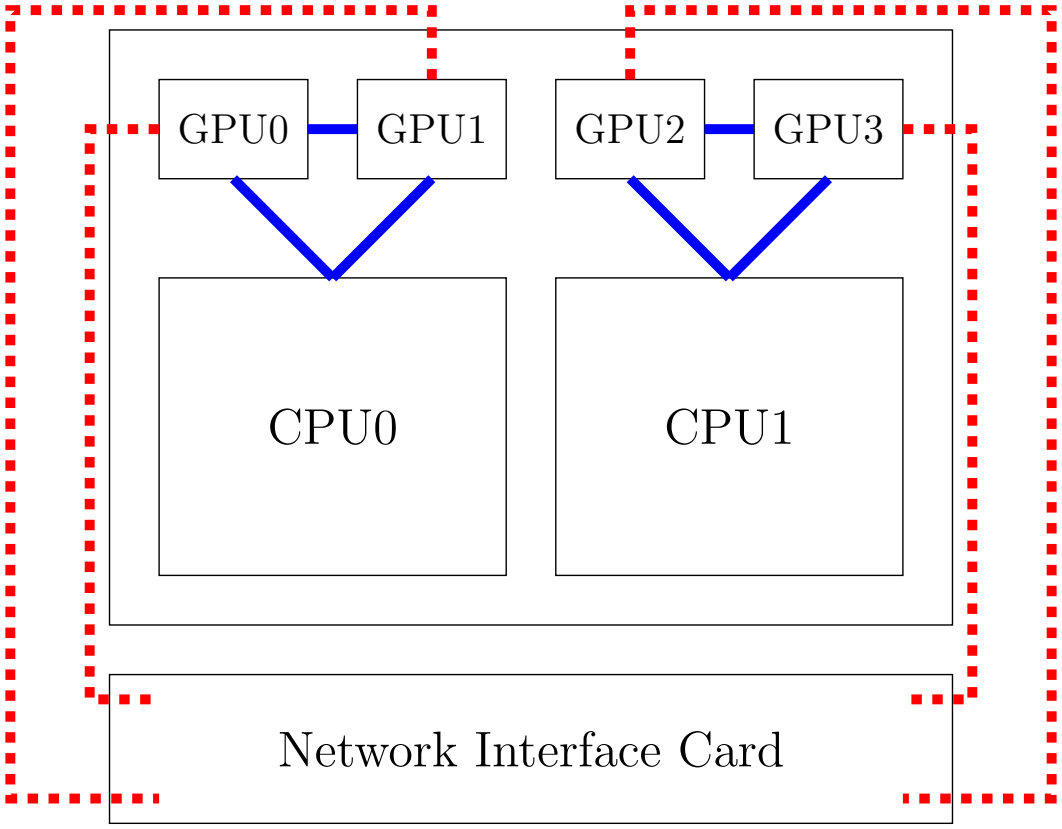
Communication on Heterogeneous Architectures

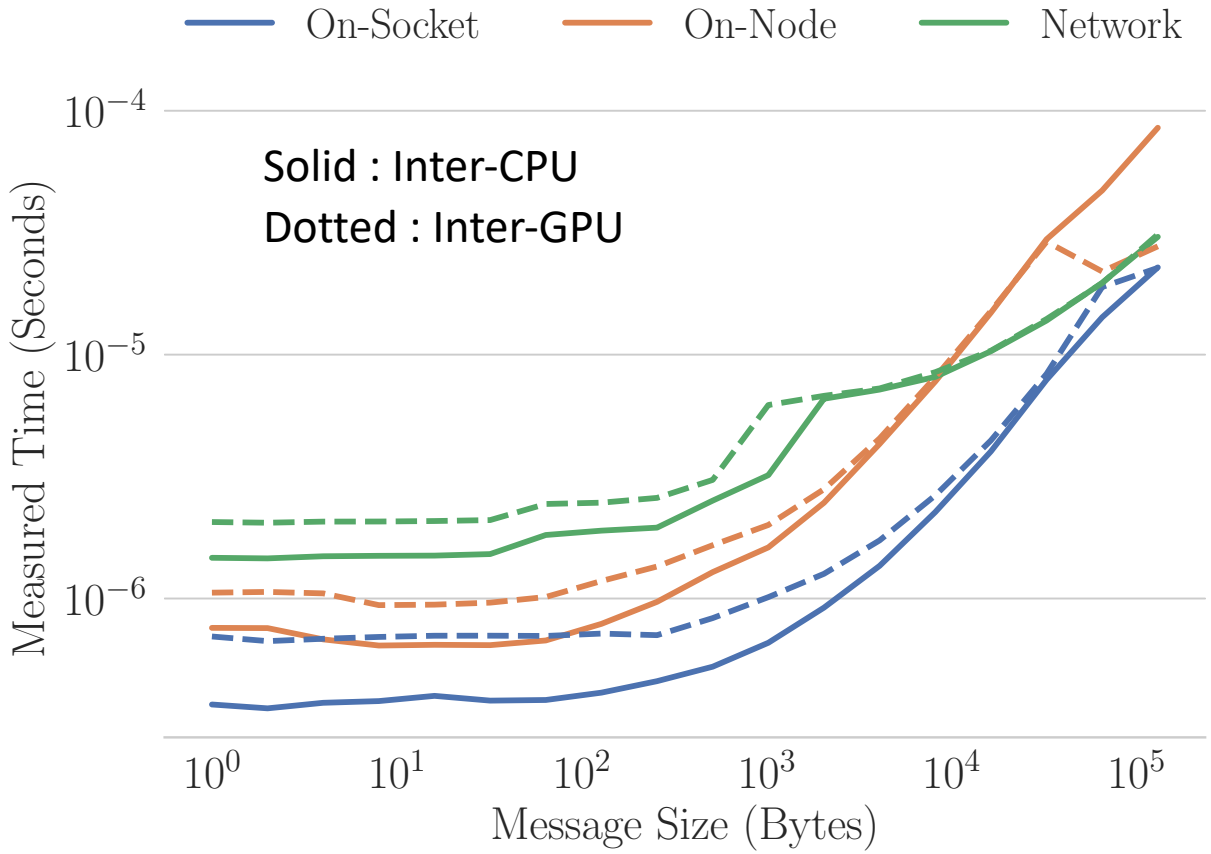




Key Takeaway : Intra-socket << Inter-socket, inter-CPU << inter-GPU

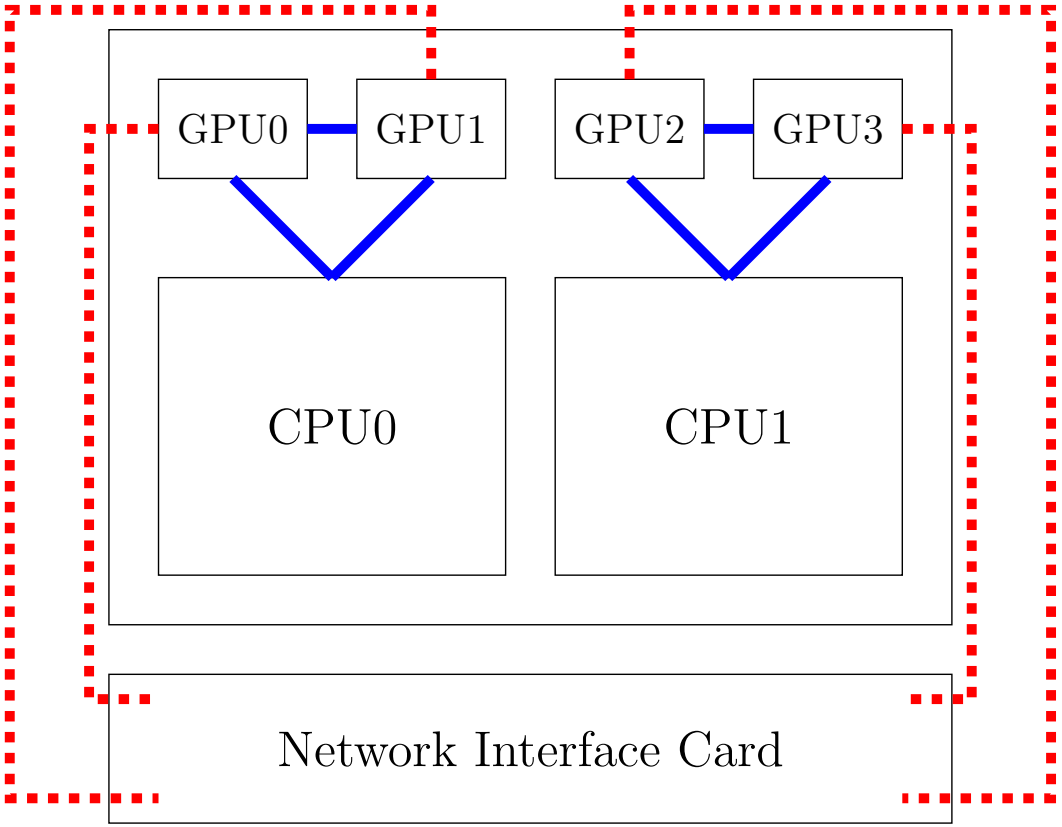
Communication on Heterogeneous Architectures

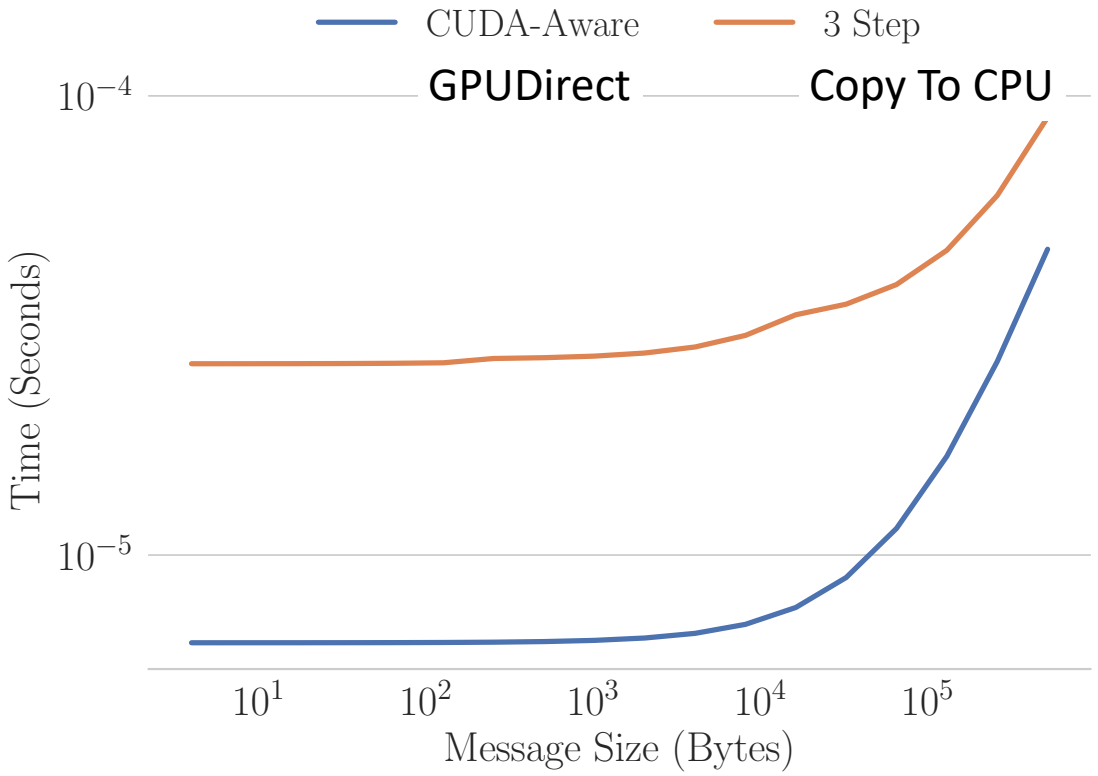




Benchmarks look more reasonable with MVAPICH2-GDR

Communication on Heterogeneous Architectures



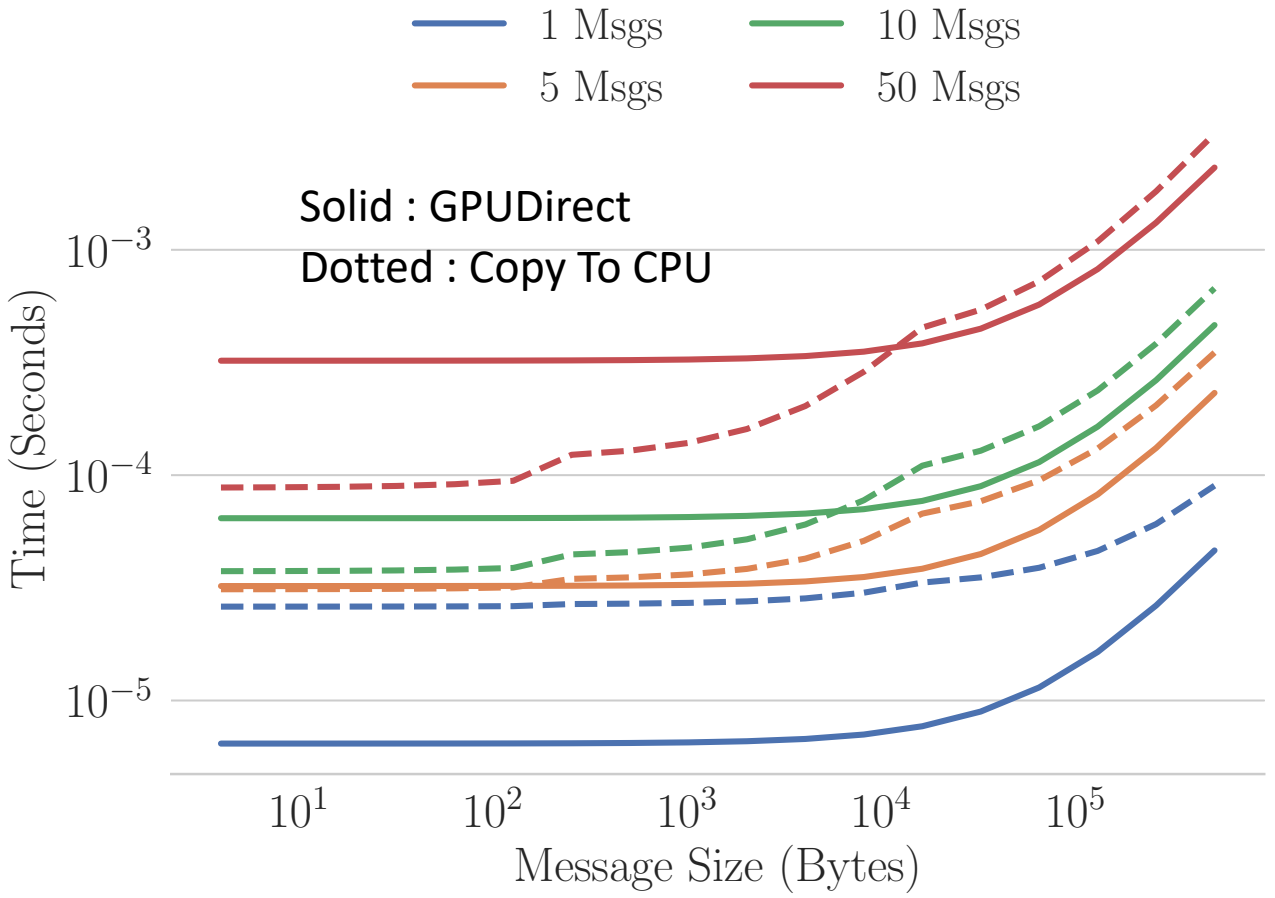


Key Takeaway : GPUDirect is cheaper for a single (reasonably sized) message

Communication on Heterogeneous Architectures

Multiple Messages

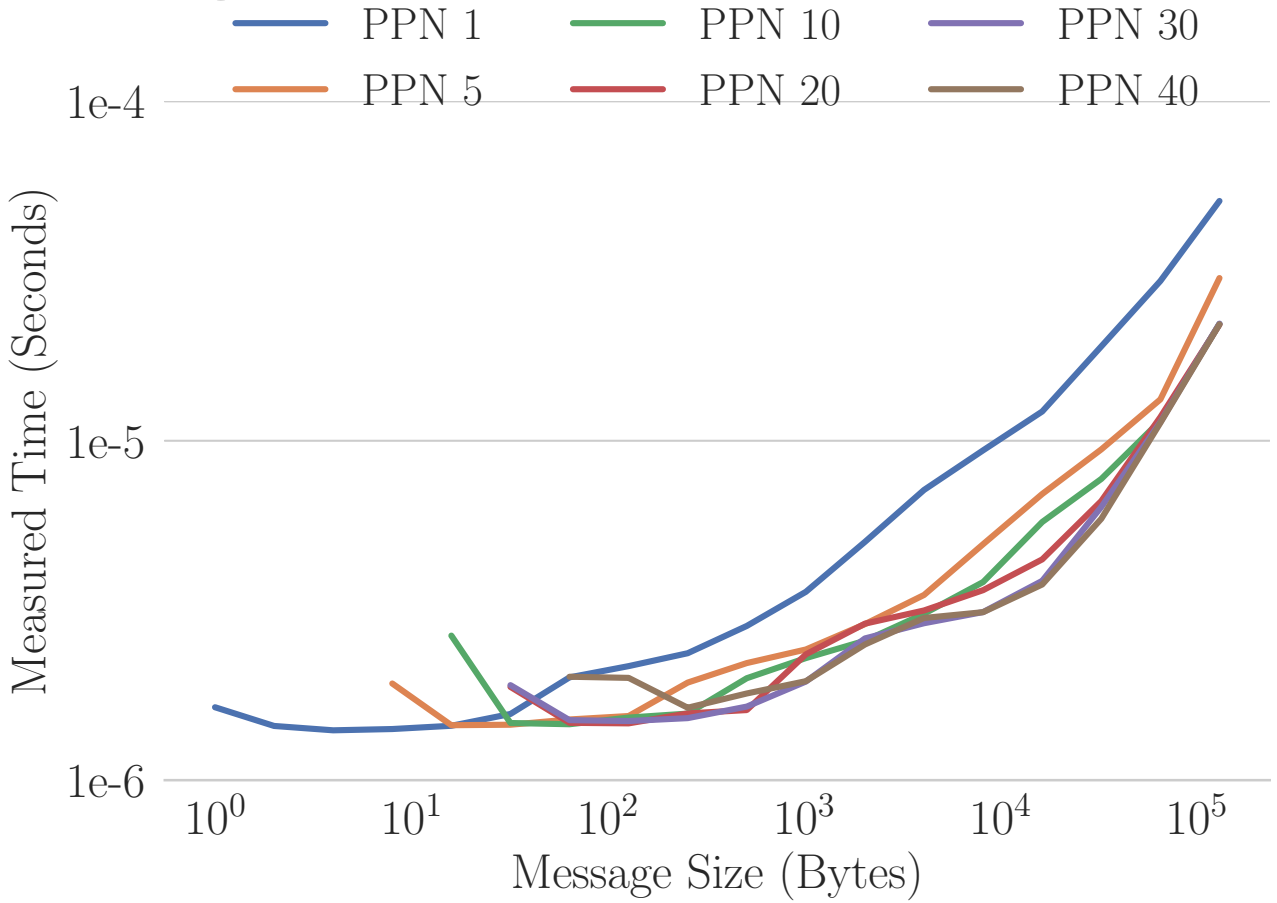
- Copy to CPU method :
  - Copy all data from GPU to CPU
  - Send many messages between CPUs
  - Copy all data to destination GPU



Key Takeaway : When sending large number of messages, cheaper to copy to CPU

Communication on Heterogeneous Architectures

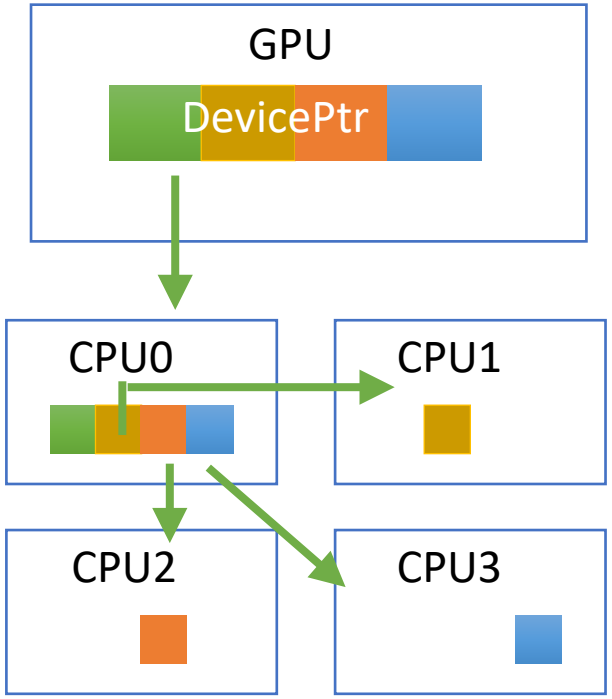
- Many CPU cores per GPU available
- Can distribute data from GPU across multiple CPU cores
- Reduces cost of inter-node communication
- Additional overhead of data distribution



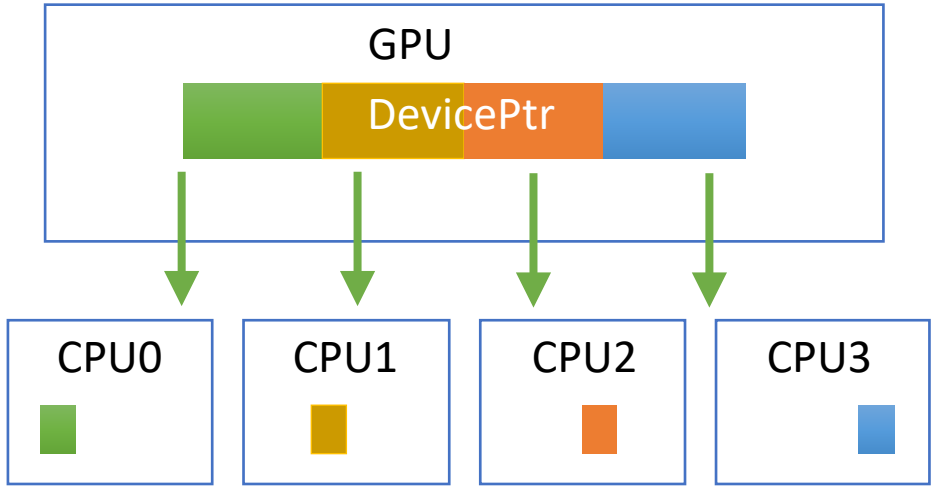
Key Takeaway : Cheapest to use multiple CPU cores per GPU during Copy-to-CPU communication

A. Bienz, L. N. Olson, W. D. Gropp and S. Lockhart, "Modeling Data Movement Performance on Heterogeneous Architectures," *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, 2021, pp. 1-7

Using Multiple CPU Cores Per GPU



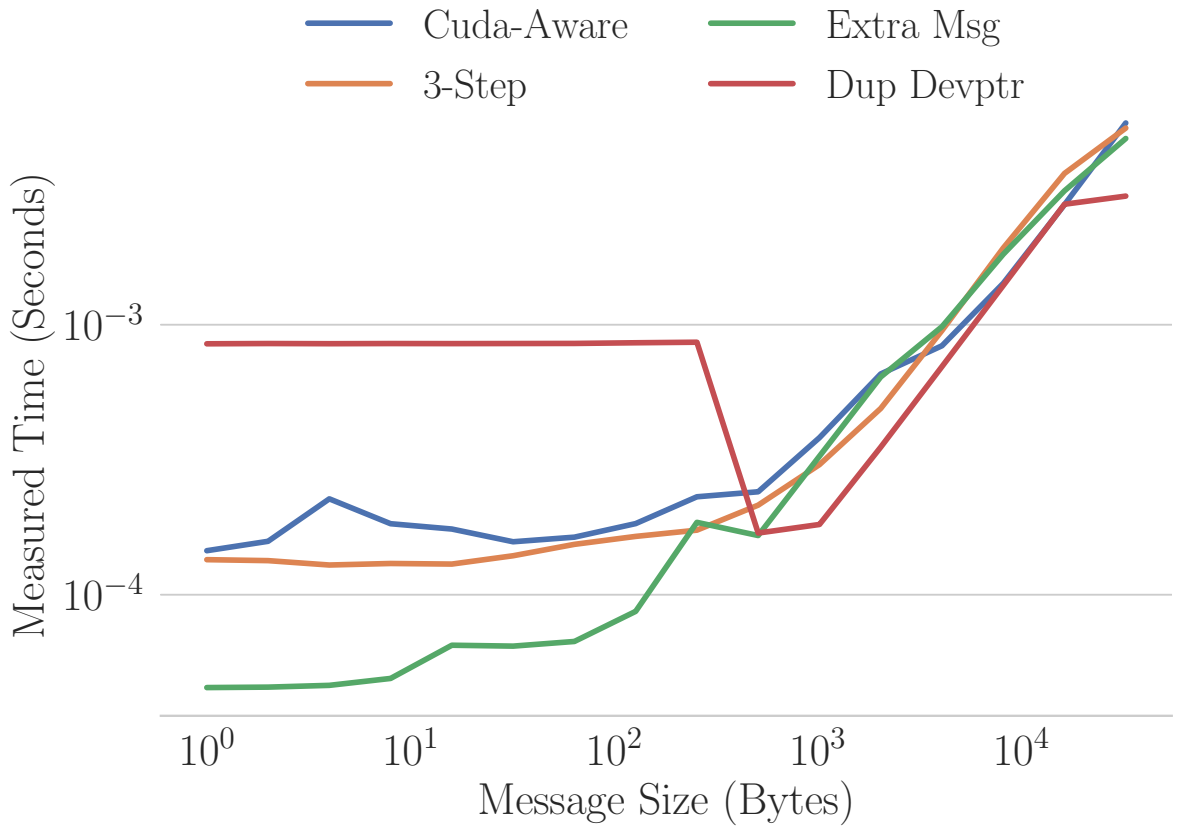
Extra Message Approach



Duplicate Device Pointer Approach

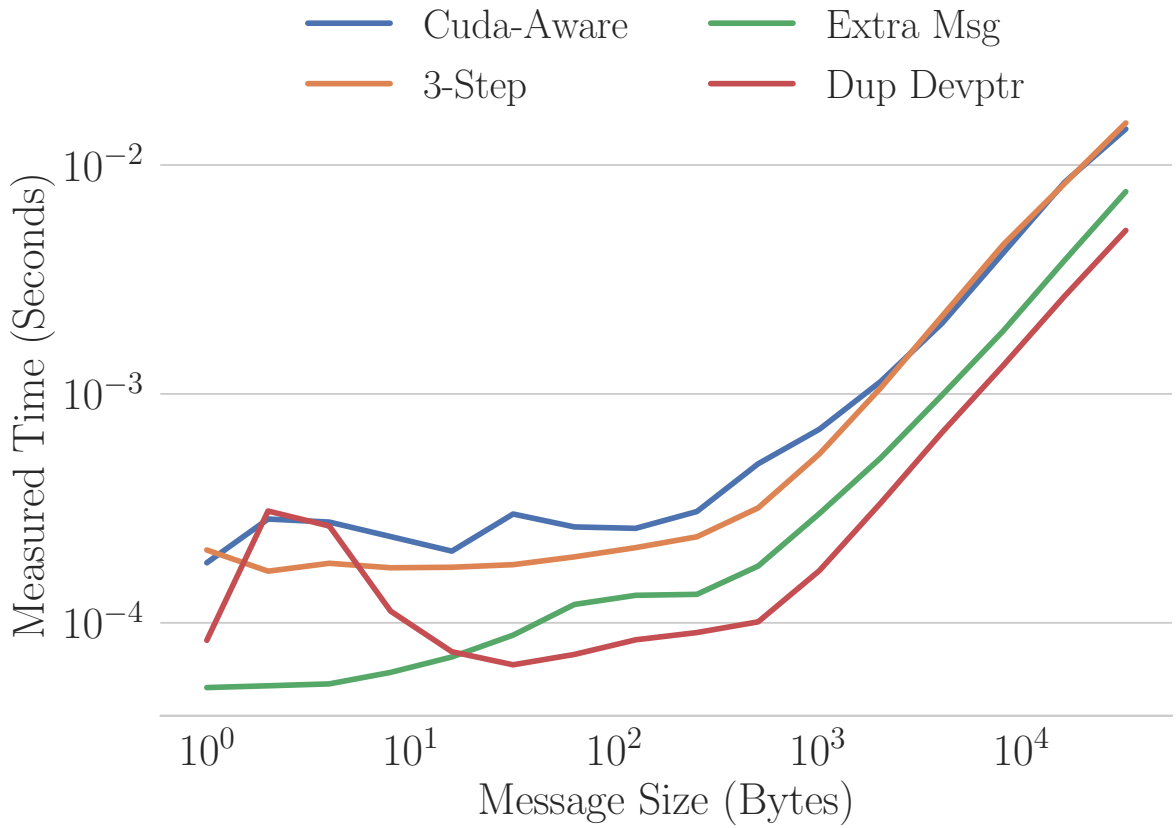
Using Multiple CPU Cores Per GPU

- Simple test : MPI_Alltoallv on 32 nodes
- Large overhead with duplicate device pointers
  - Only on Lassen



Using Multiple CPU Cores Per GPU

- Simple test : MPI_Alltoallv on 32 nodes
- Overhead is more reasonable on Summit

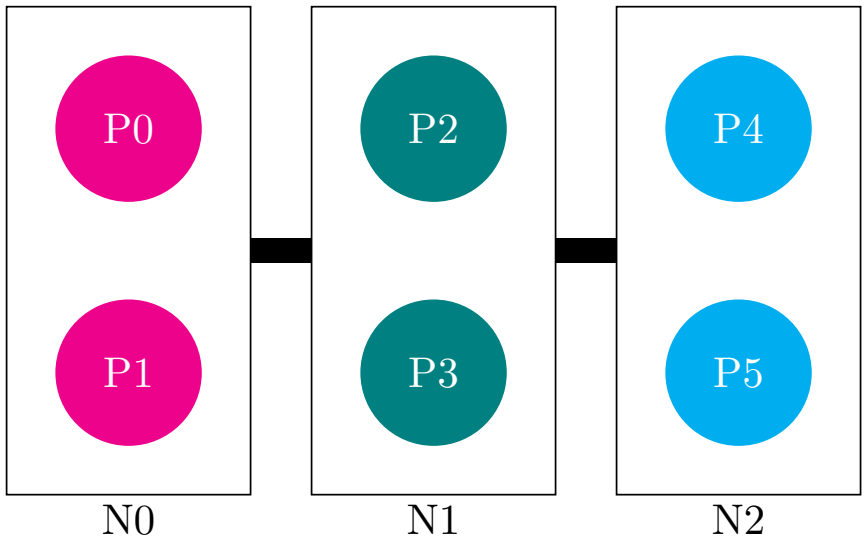


A. Bienz, L. N. Olson, W. D. Gropp and S. Lockhart, "Modeling Data Movement Performance on Heterogeneous Architectures,"

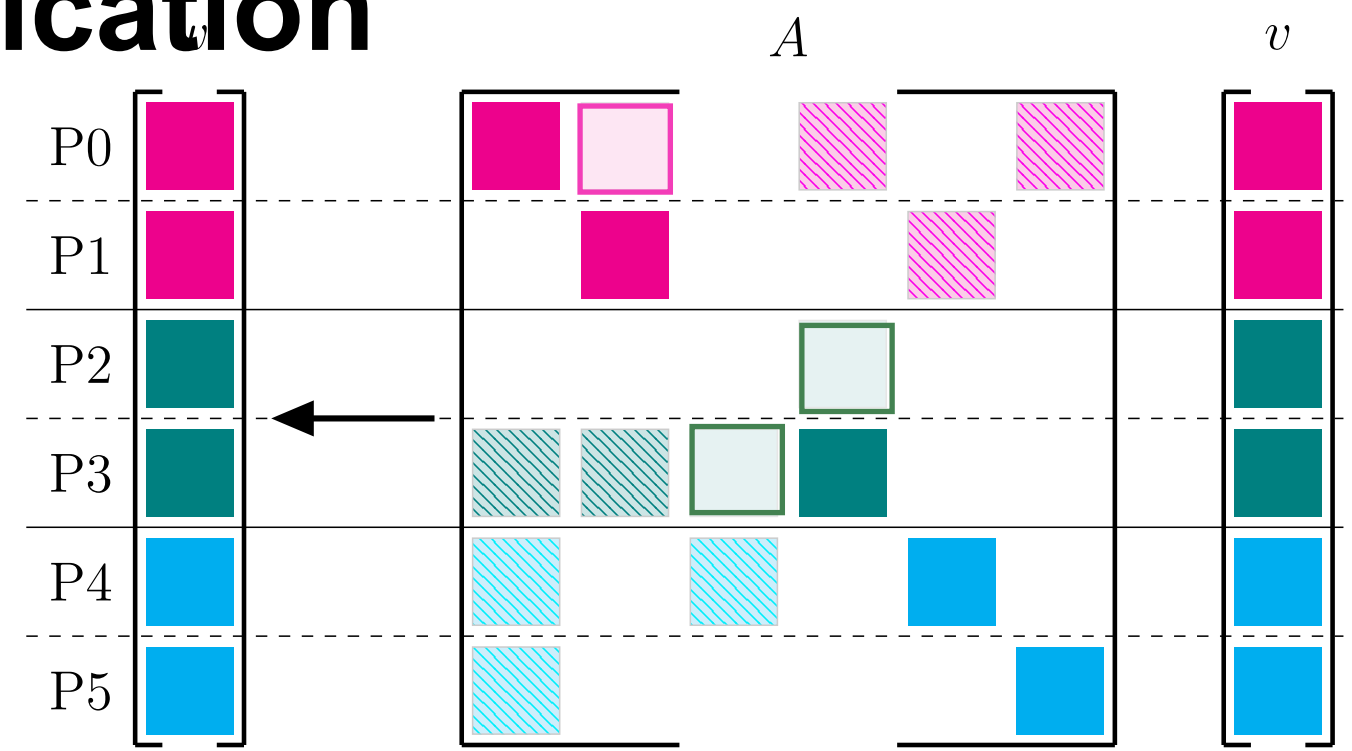
Performance Optimizations

- Reduce costly communication
- Minimize the total number of messages received at a given time (queue search costs)
- Distribute data evenly across all available CPU cores
- Focusing on inter-CPU communication
  - Locality-aware irregular communication (neighborhood collectives)
  - Locality-aware collectives

Irregular Communication

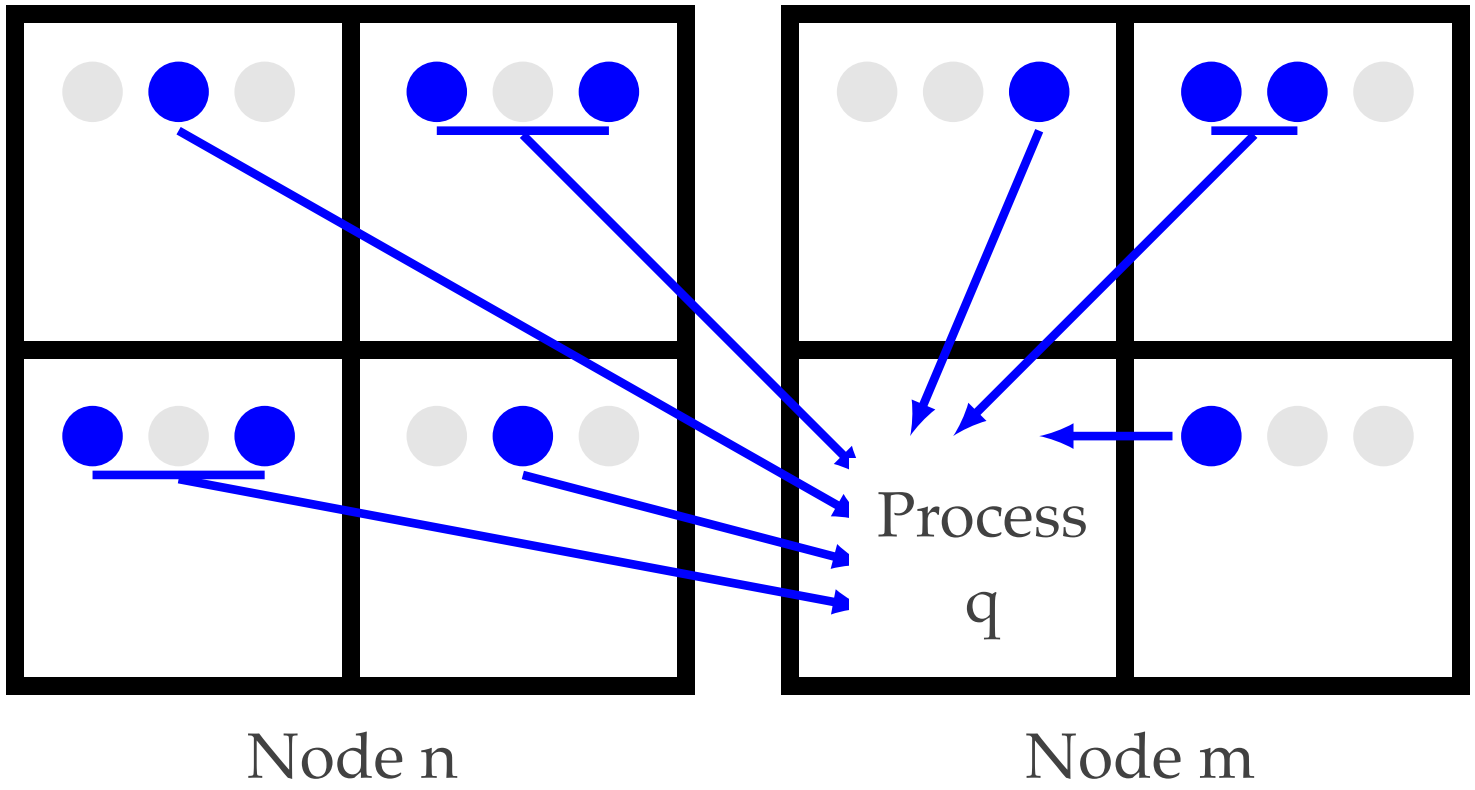


Six processes distributed across three nodes



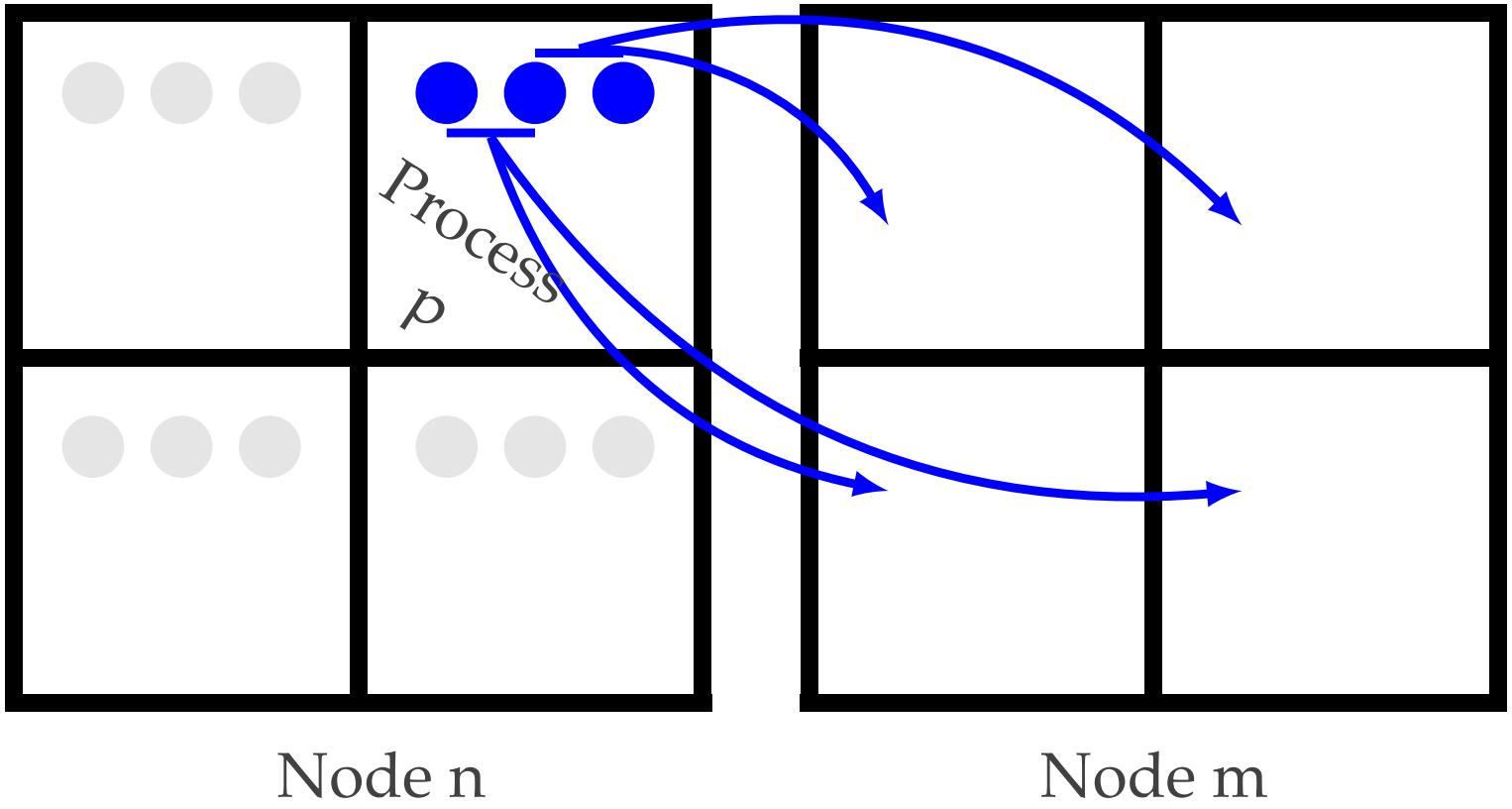
Linear system distributed across the processes

Standard Communication



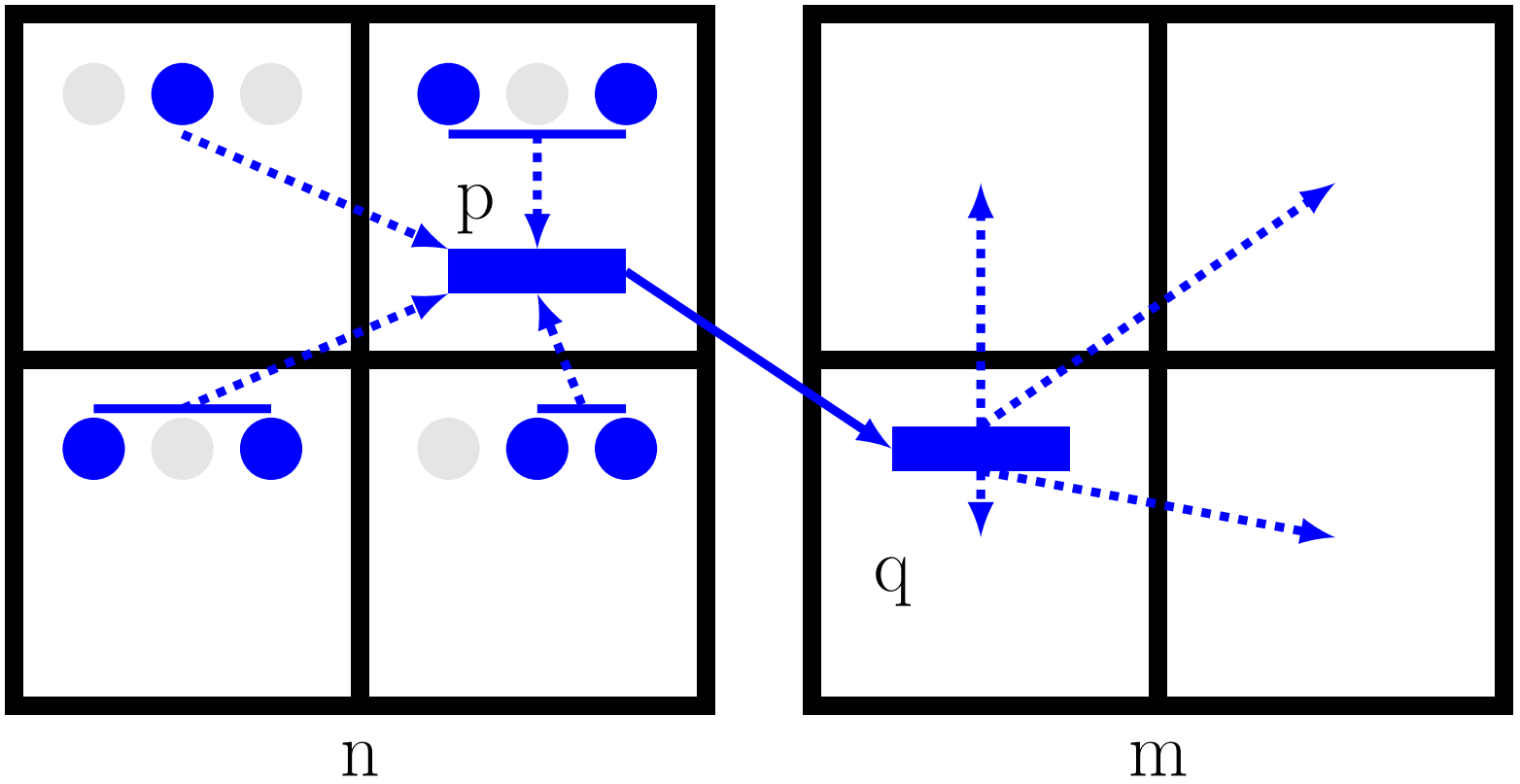
Multiple messages between set of nodes

Standard Communication

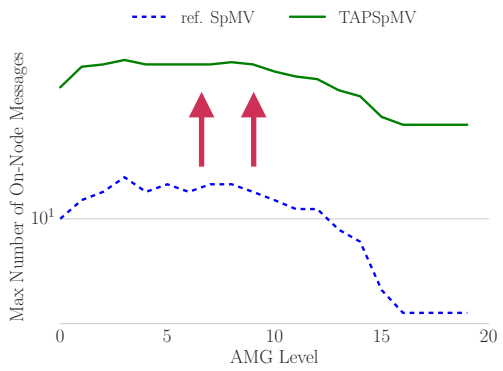
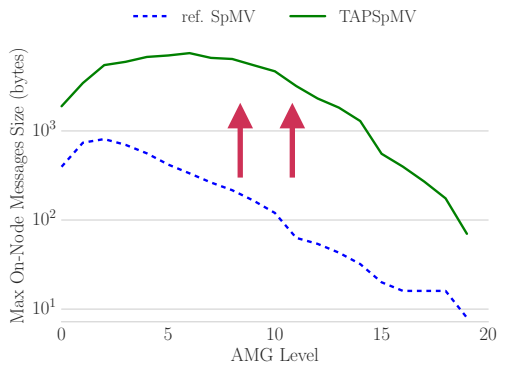
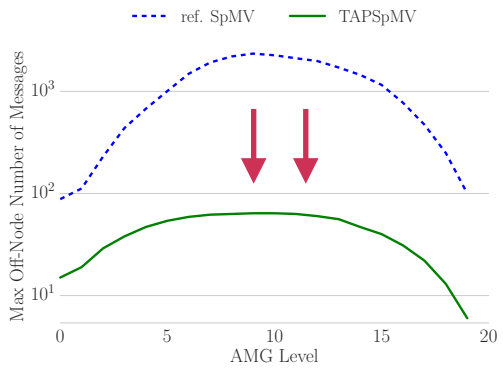
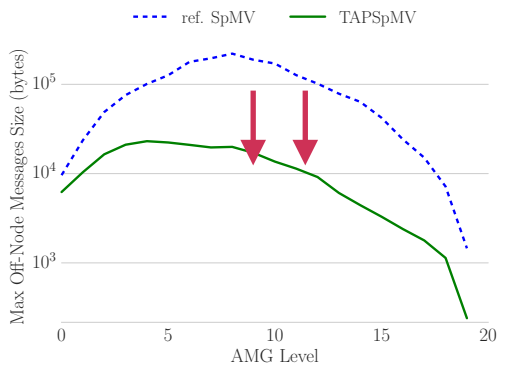


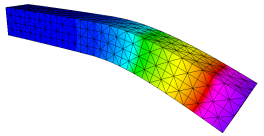
Multiple messages and duplicate data between set of nodes

Locality-Aware Communication : Small Messages



Locality-Aware Communication

	Max Number of Messages	Max Size of Messages
Intra-node		
Inter-node		

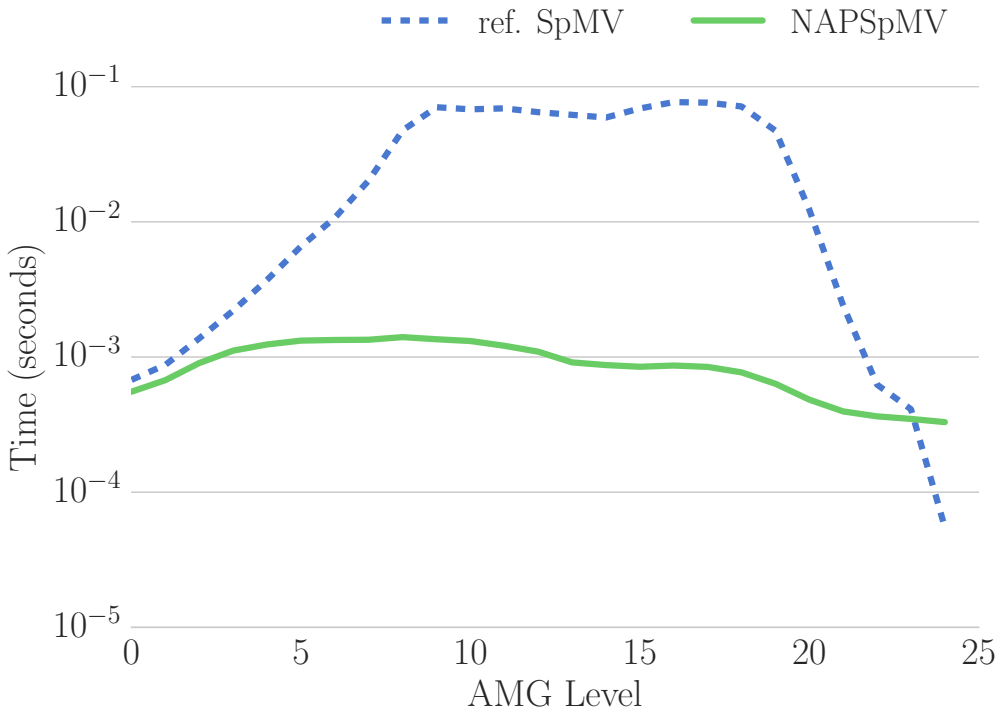


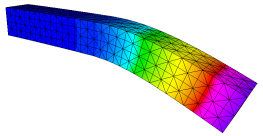
linear elasticity hierarchy
16,284 processes

Blue Dotted Lines :
Standard Communication

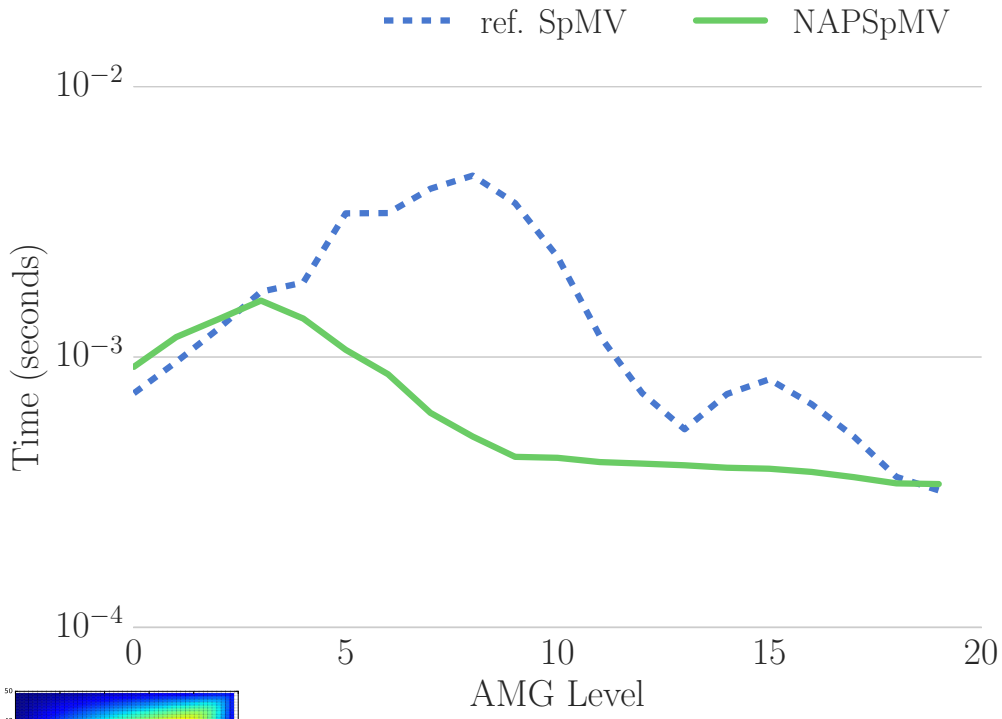
Green Lines:
Locality-Aware

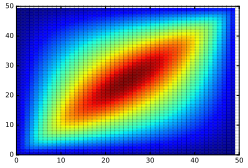
Locality-Aware SpMVs





Linear Elasticity (MFEM)





2D Rotated Anisotropic Diffusion

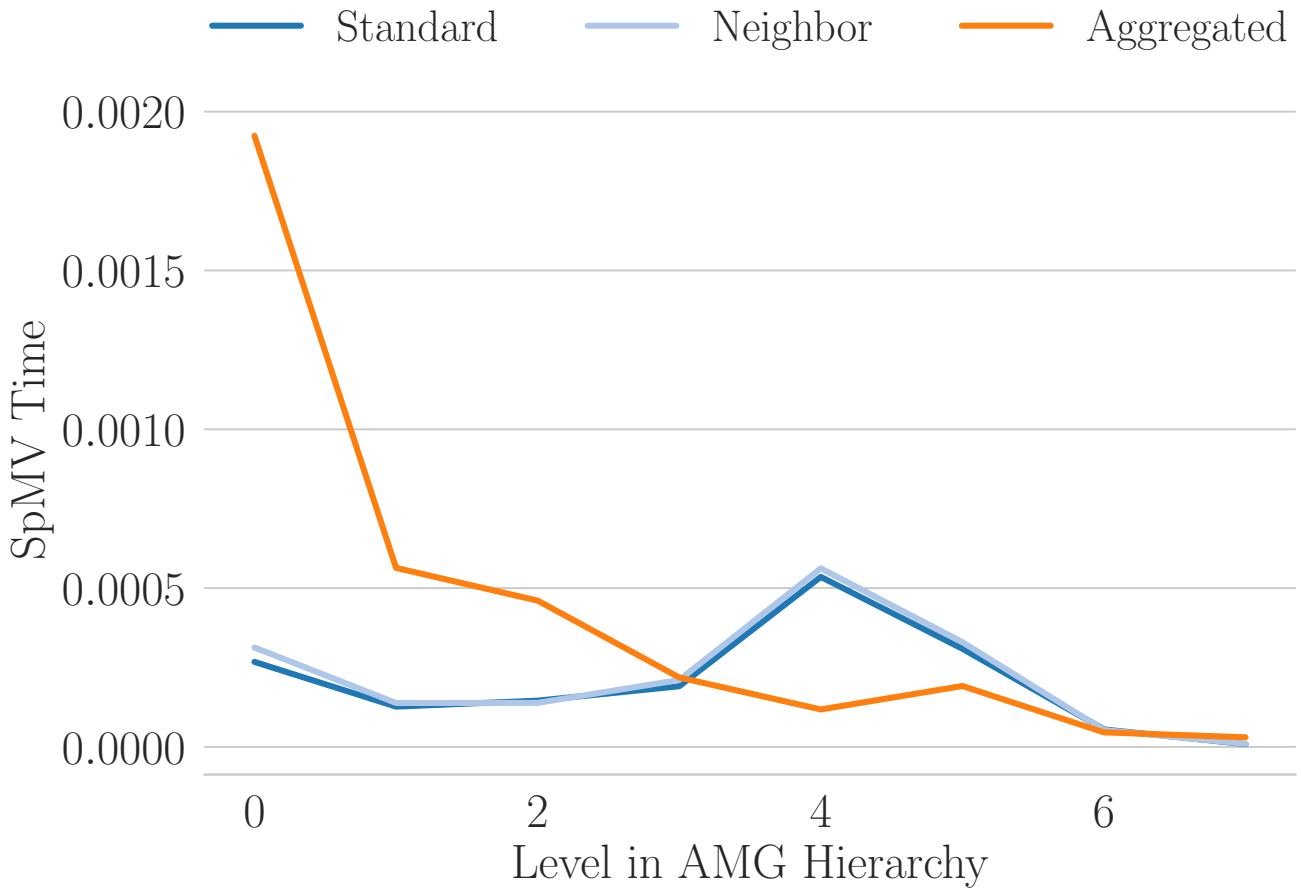
Node-Aware Sparse Matrix Vector Multiplication (https://doi.org/10.1016/j.jpdc.2019.03.016)

Irregular Communication in MPI Advance

- No easy way to aggregate MPI_Isend and MPI_Irecv calls without knowing all calls that are being made
- The API to optimize irregular communication does exist in MPI:
  - Neighborhood Collectives
- Optimizations added to MPI Advance for persistent version of MPI_Neighbor_alltoallv

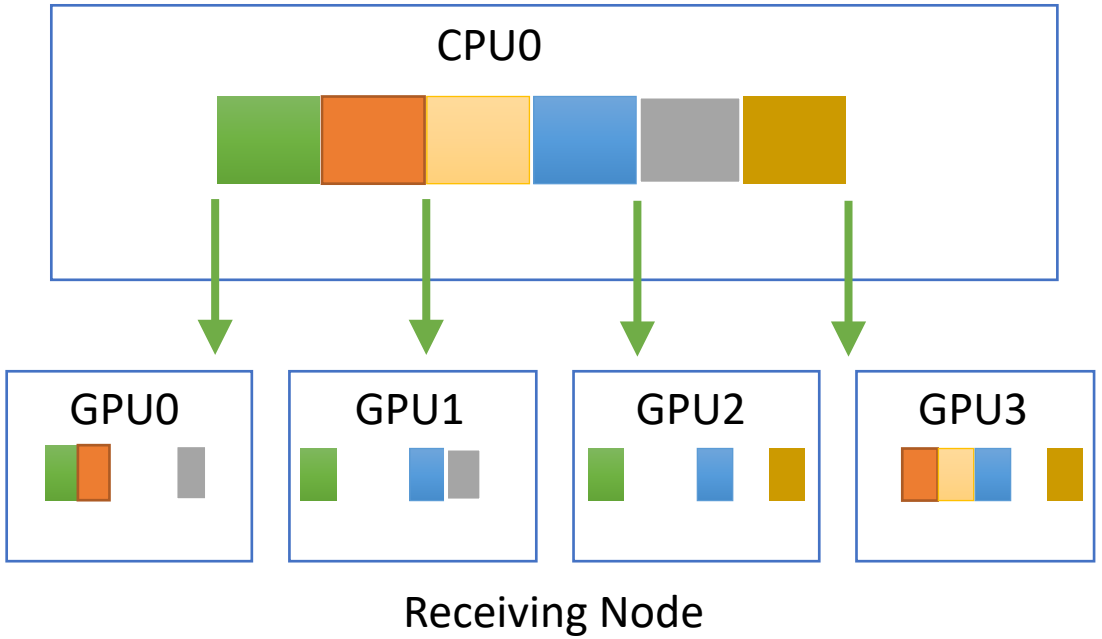
Aggregated Neighborhood Collectives in Hypre

- Results currently outside of Hypre
  - Form hierarchy and then add neighbor collectives
- **Gerald Collom will talk about his work adding neighbor collectives within Hypre**



Open Questions : Neighborhood Collective Optimizations

- Open question : how to efficiently remove duplicate values on heterogeneous architectures
- Issue : must re-pack received data
  - Potentially re-pack on one GPU and then communicate locally among GPUs

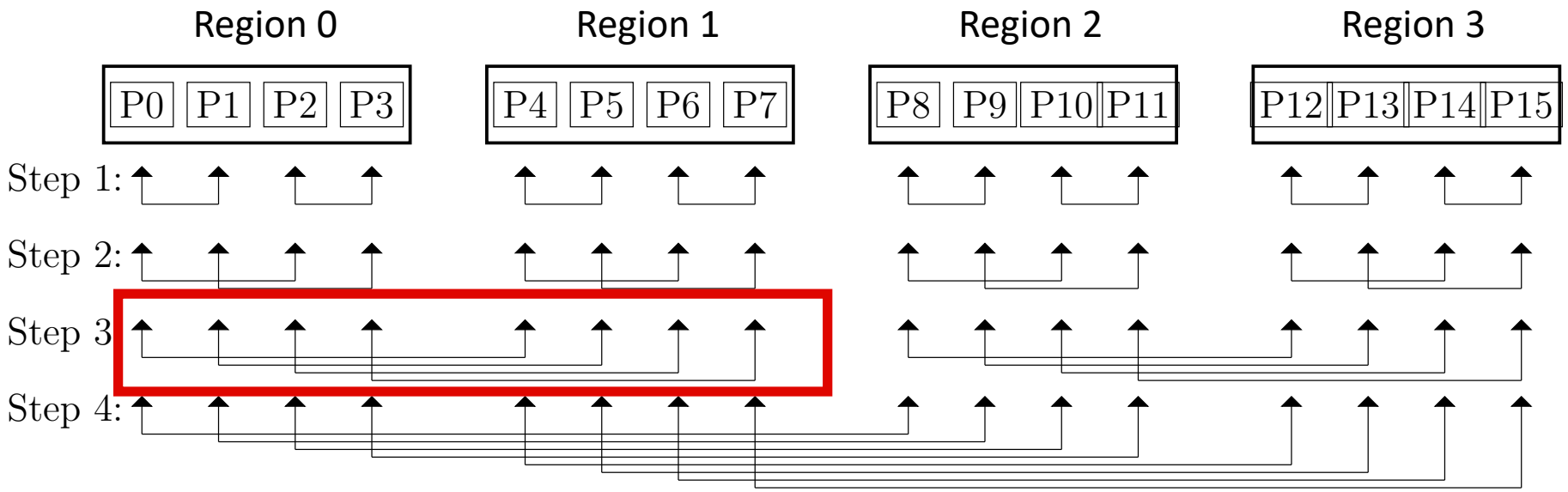


Optimizing Collective Operations

- Collective operations typically are optimized algorithmically
  - Minimize message count for small collectives
  - Minimize amount of data communicated for large collectives
- No attention to relative locations of sending and receiving processes

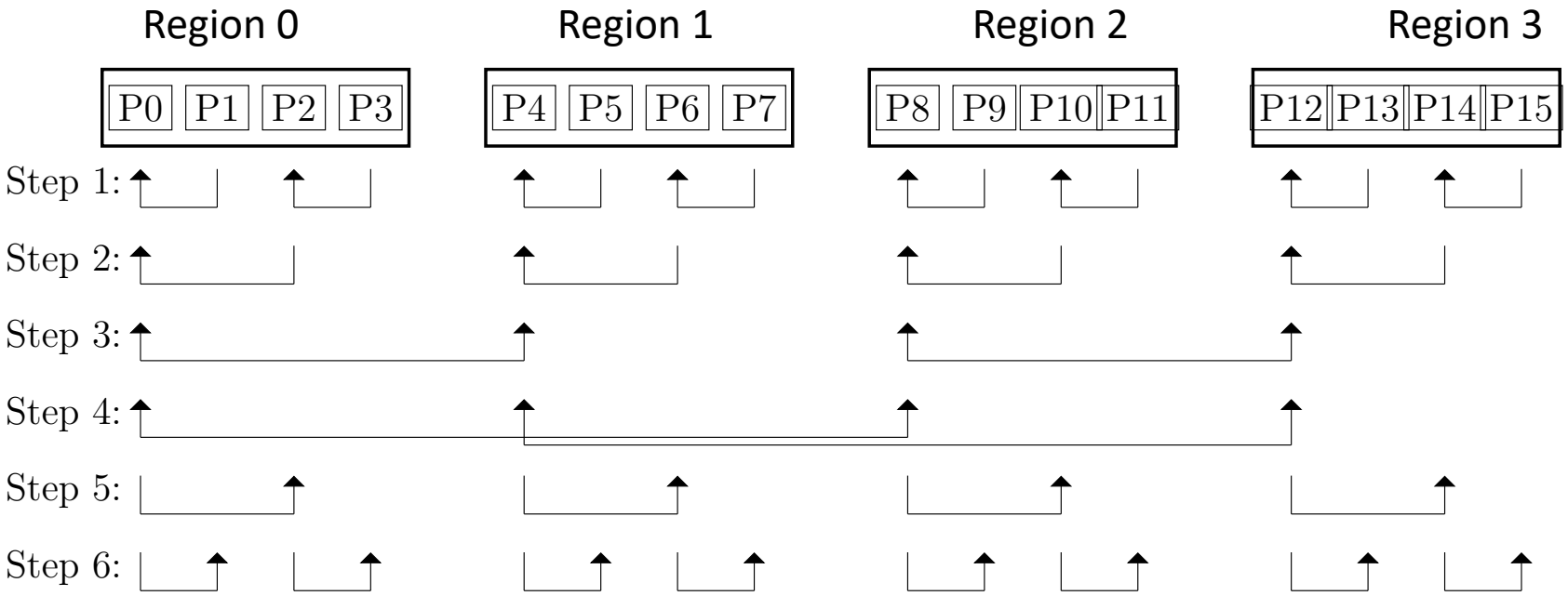
- **Add locality-awareness : minimize non-local (expensive) communication, exchanging for additional local communication**

Recursive Doubling : Standard



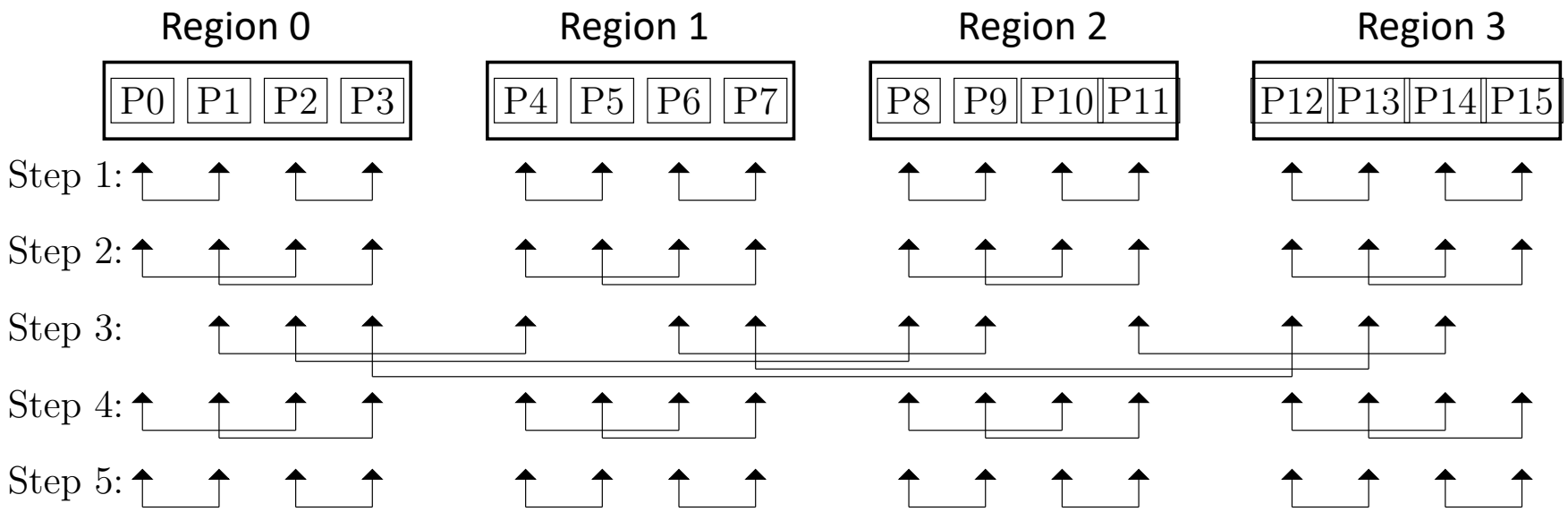
Multiple messages between nodes and sending the same values through the network multiple times

Recursive Doubling : Hierarchical



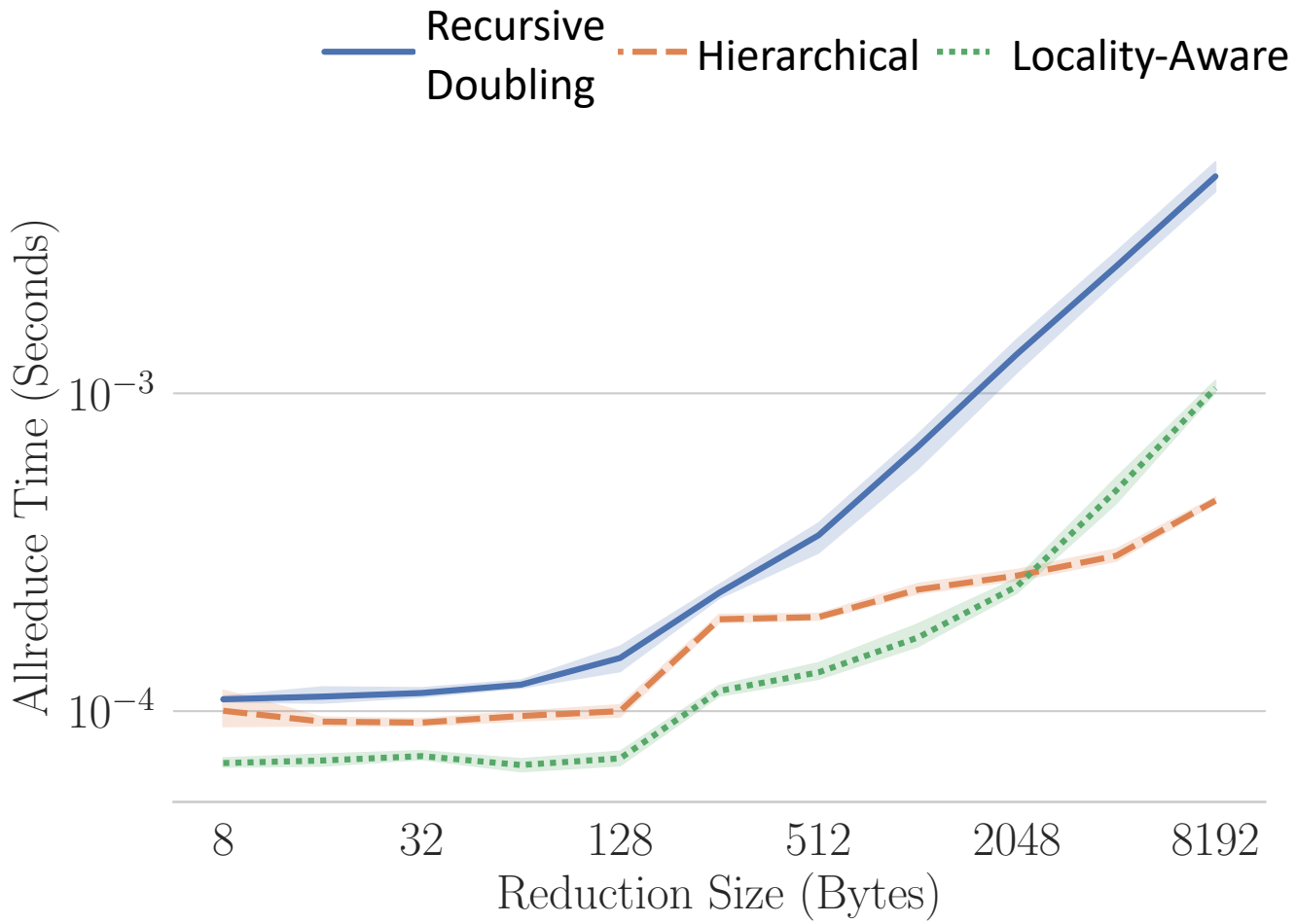
No duplicate messages, but many idle processes per node

Recursive Doubling : Locality-Aware



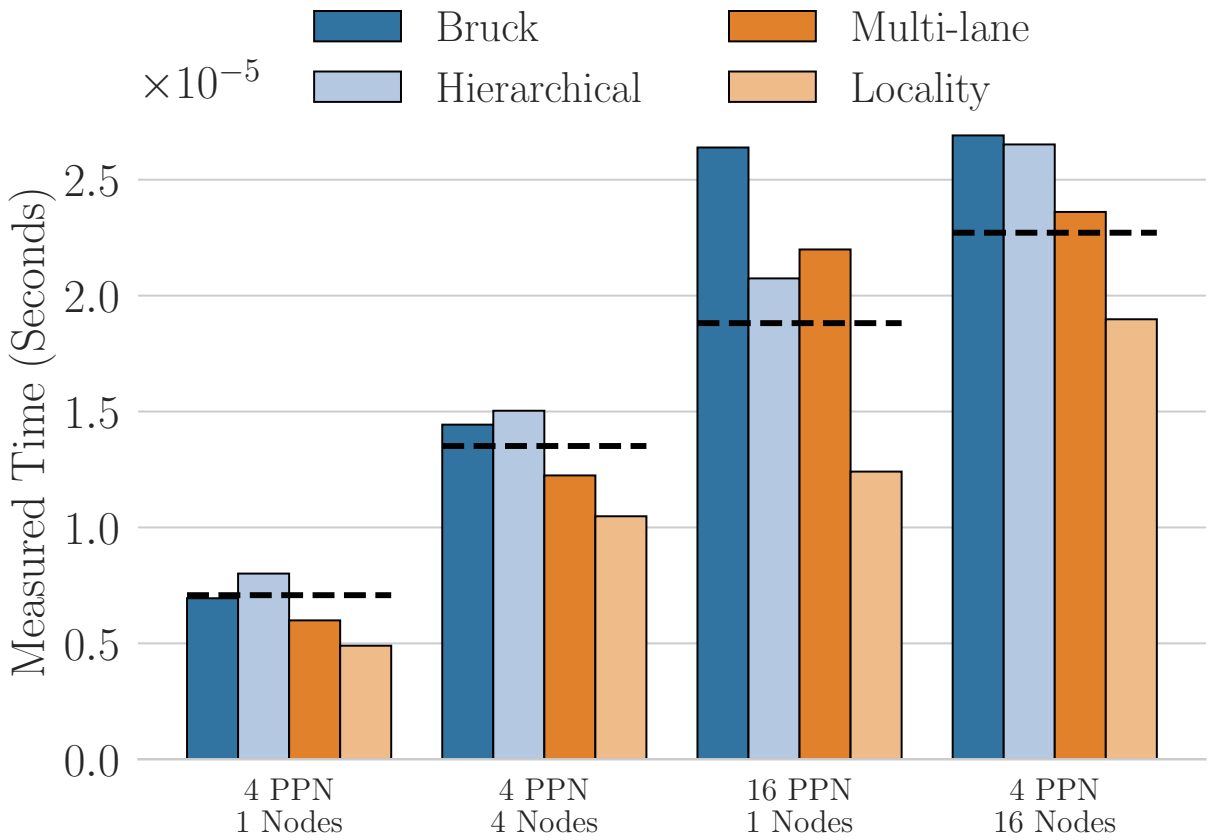
Use extra processes to exchange data with multiple non-local regions in a single step

Locality-Aware Allreduce



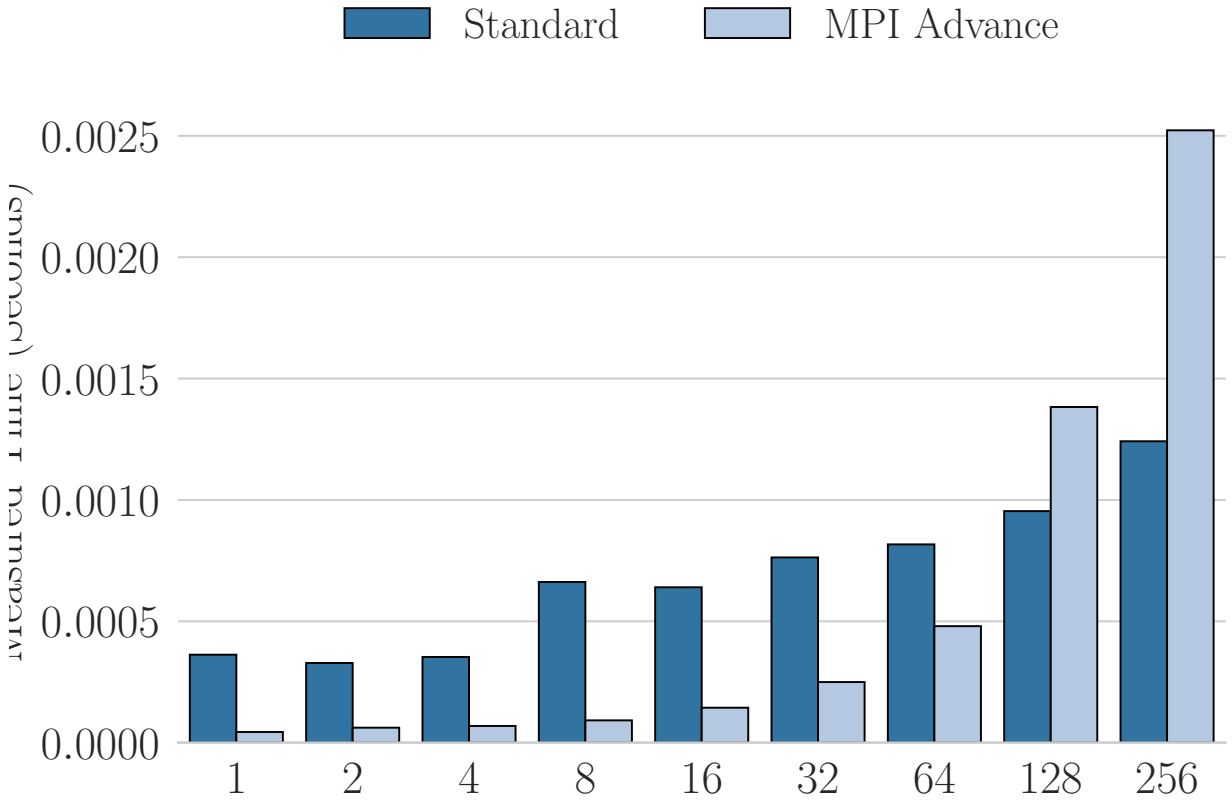
A. Bienz, L. Olson and W. Gropp, "Node-Aware Improvements to Allreduce,"

Locality-Aware Bruck Allgather



A. Bienz, S. Gautam, and A. Kharel, "A Locality-Aware Bruck Allgather"

Locality-Aware Alltoallv



- Fast Fourier transforms (such as Heffte) depend on performance of MPI_Alltoall and MPI_Alltoallv
- Evelyn Namugwanya is beginning to look at effects of locality-aware collectives on FFT solvers

Open Questions : Collective Operations

- How do we determine which implementation to use in any given instance?
- Persistent collectives : can load balance, eliminate setup overhead
- Heterogeneous Architectures : how to use all available CPU cores per GPU in a portable way?
- Newer architectures : can GPUDirect pay off?

Work in Progress

- Want to assess algebraic multigrid within applications
  - Adding locality-aware support into
    - Hypre
    - Trilinos
  - Implicit solve in Higrad will call Hypre
  - EMPIRE depends on Meulu from Trilinos
- Working with labs to identify other relevant applications

Questions?