THE OHIO STATE UNIVERSITY

Designing High Performance and Energy-Efficient MPI Collectives for Next Generation Clusters

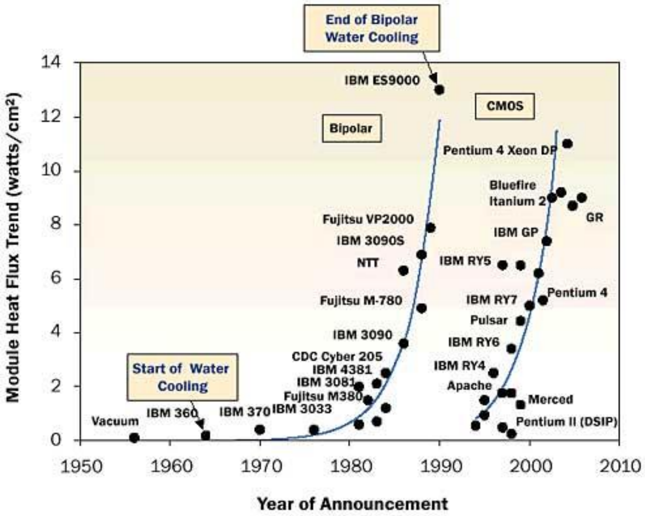
Akshay Venkatesh, 5th year Ph.D student

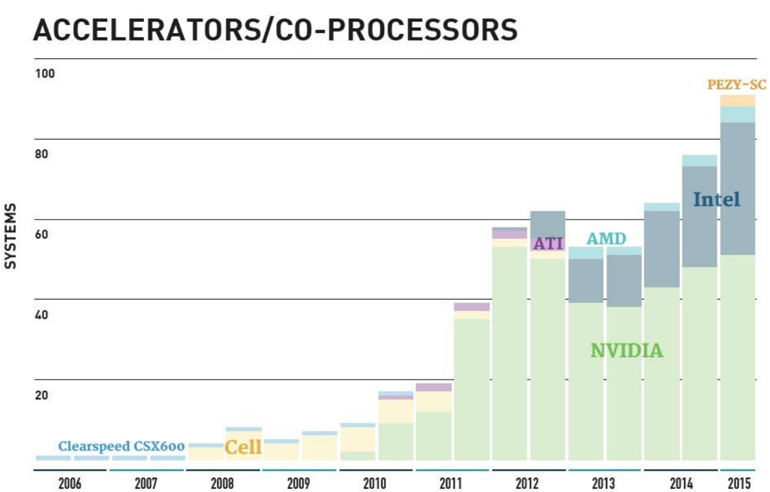
Advisor : DK Panda

Network-based Computing Lab, OSU

- Introduction
- Problem Statement
- Challenges
- Contributions and Results
- Future work
- Conclusions

- Introduction
- Problem Statement
- Challenges
- Contributions and Results
- Future work
- Conclusions

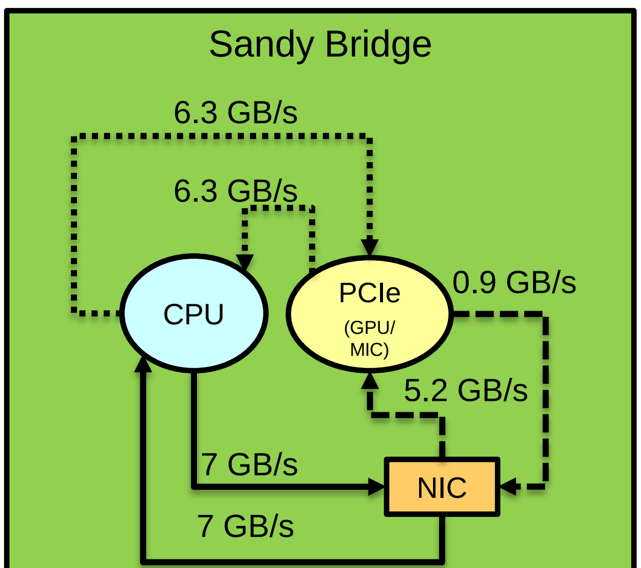




- Culmination of *Dennard's Scaling** has yielded manyfold increase in parallelism on processing chips and has placed emphasis on power/energy conservation of systems
- High performance computing domain has seen increased use accelerators/co-processors such as NVIDIA GPUs/ Intel MICs
- Scientific applications routinely use these specialized hardware to accelerate compute phases owing to their >= 1 Teraflops/device capability at comparatively lower power footprint
- MPI/PGAS serve as the de facto programming models to amalgamate capacities of several such distributed *heterogeneous* nodes

* Power density remains constant

- Introduction
- Problem Statement
- Challenges
- Contributions and Results
- Future work
- Conclusions



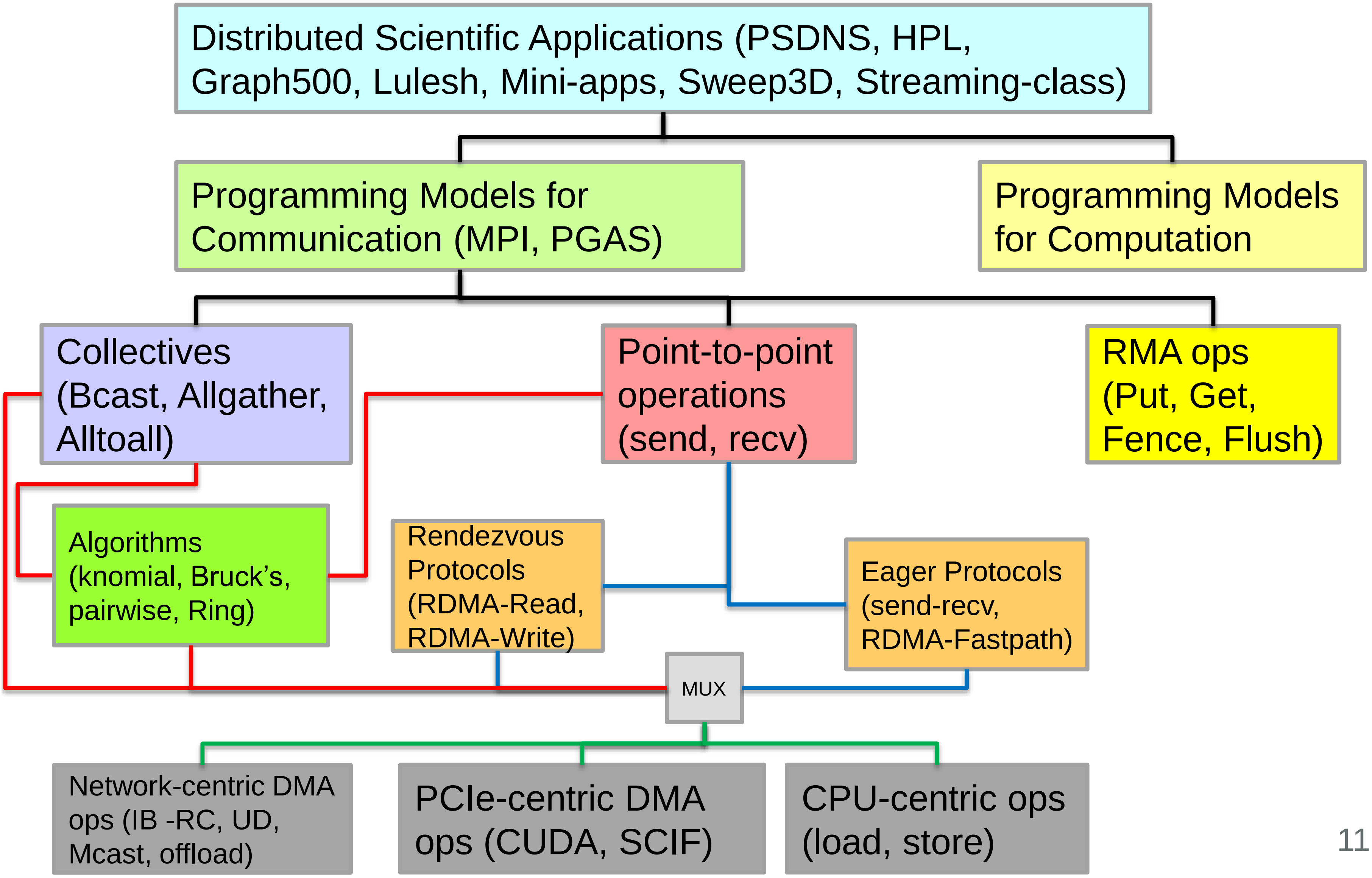
- With diversification of compute platforms, it is important to ensure that compute and communication phases of long running applications are efficient
- => Execution time and energy usage are two dimensions that demand attention
- NVIDIA GPUs and Intel MICs (available as PCIe devices) introduce differential compute and memory costs
- MPI collectives such as Broadcast, Alltoall and Allgather can contribute to a significant fraction of total application execution time and energy
- Minimizing latency, increasing overlap and minimizing energy of MPI collectives require rethinking of underlying algorithms

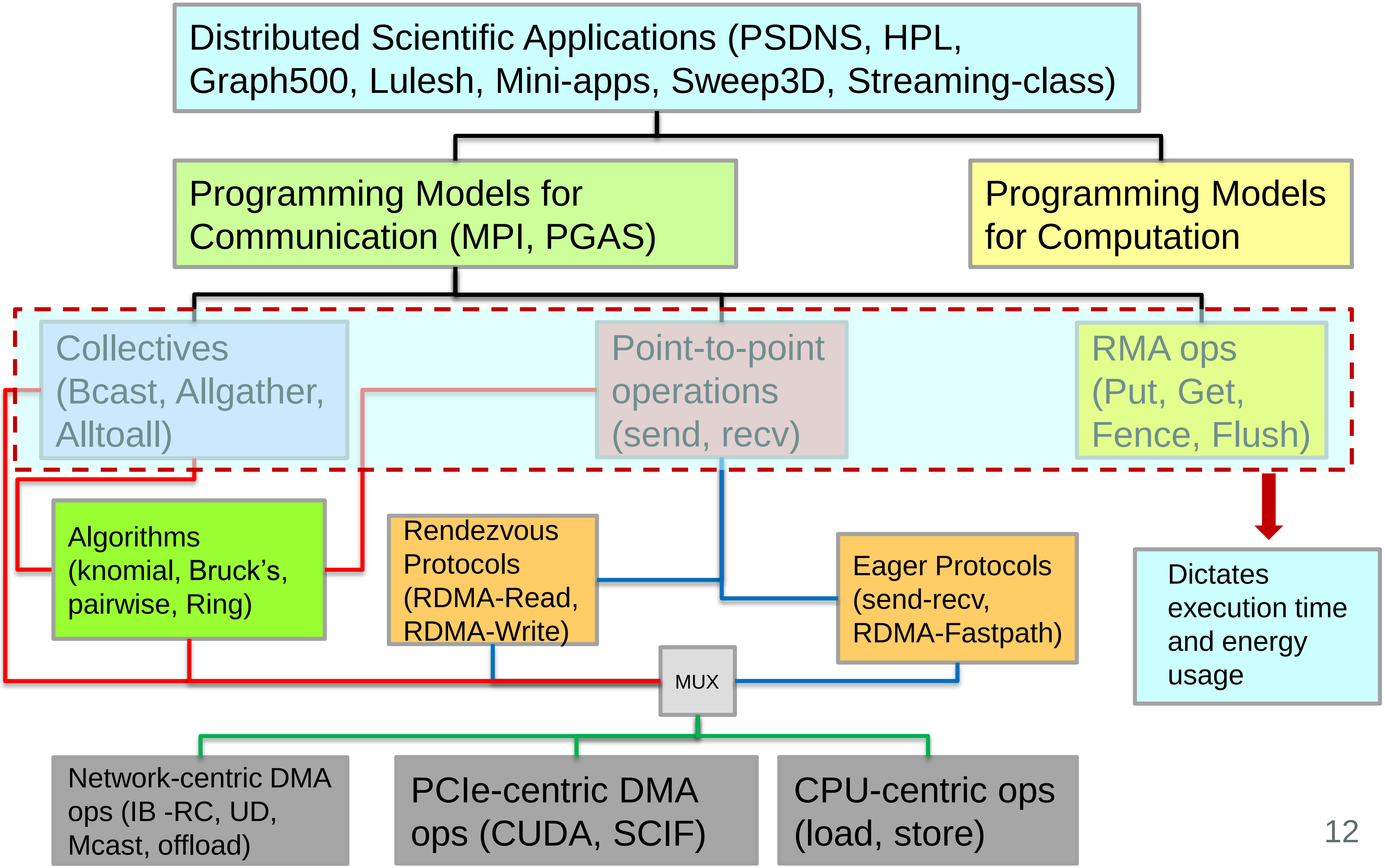
- Popular algorithms such as Bruck's allgather/alltoall algorithms, recursive doubling and ring algorithms assume uniformity of cost paths => *Repeated use of non-optimal paths and steps in heterogeneous systems*
- Existing runtimes that support communication operations from GPU buffers do not exploit novel mechanisms such as *GPUDirect RDMA* in throughput-critical scenarios
- Methods to hide latency (critical for GPUs) are unavailable in the form of non-blocking GPU collectives
- Rules to apply energy efficiency levers during MPI calls in an application-oblivious manner that works for irregular/regular communication patterns do not exist

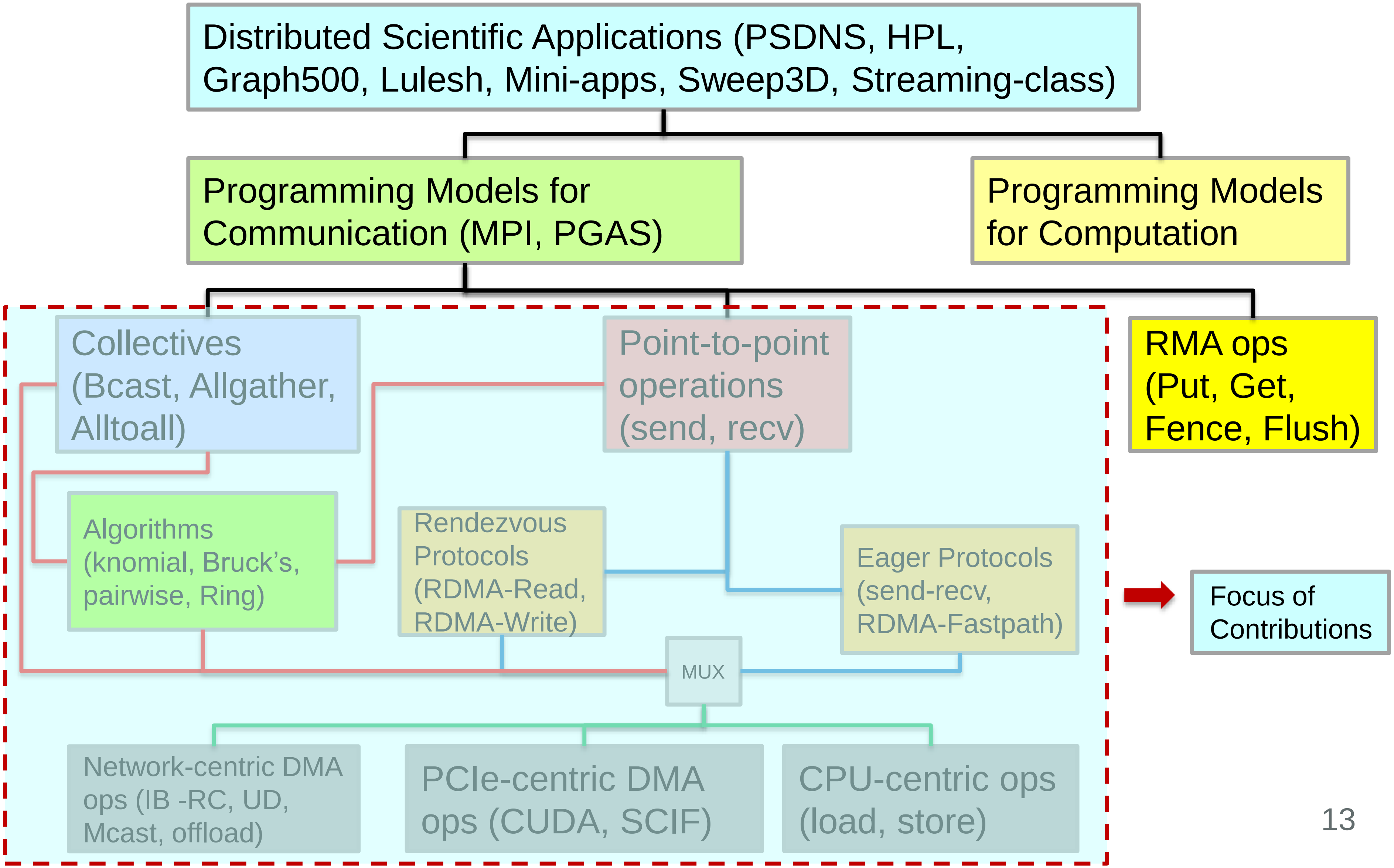
- Introduction
- Problem Statement
- Challenges
- Contributions and Results
- Future work
- Conclusions

- *Can variations of popular collective algorithms be proposed that is better suited towards platforms with heterogeneous communication cost paths and compute capacities?*
- *Can new heuristics lead to reduced collective communication cost in heterogeneous clusters?*
- *Can direct-GPU memory access mechanisms such NVIDIA GPUDirect-RDMA be coupled with existing paradigms such as the hardware multicast feature for throughput oriented applications?*
- *Can direct-GPU memory access mechanisms such as GPUDirect-RDMA and associated CUDA features be combined with network offload methods such as CORE-Direct to realize efficient non-blocking GPU collectives for good overlap and latency?*
- *Can a set of generic rules be proposed for point-to-point and collective routines such that energy savings are made only at relevant calls with negligible performance degradation?*
- *Can these rules ensure energy-savings in an application-oblivious manner and not just to well balanced applications?*

- Introduction
- Problem Statement
- Challenges
- Contributions and Results
- Future work
- Conclusions

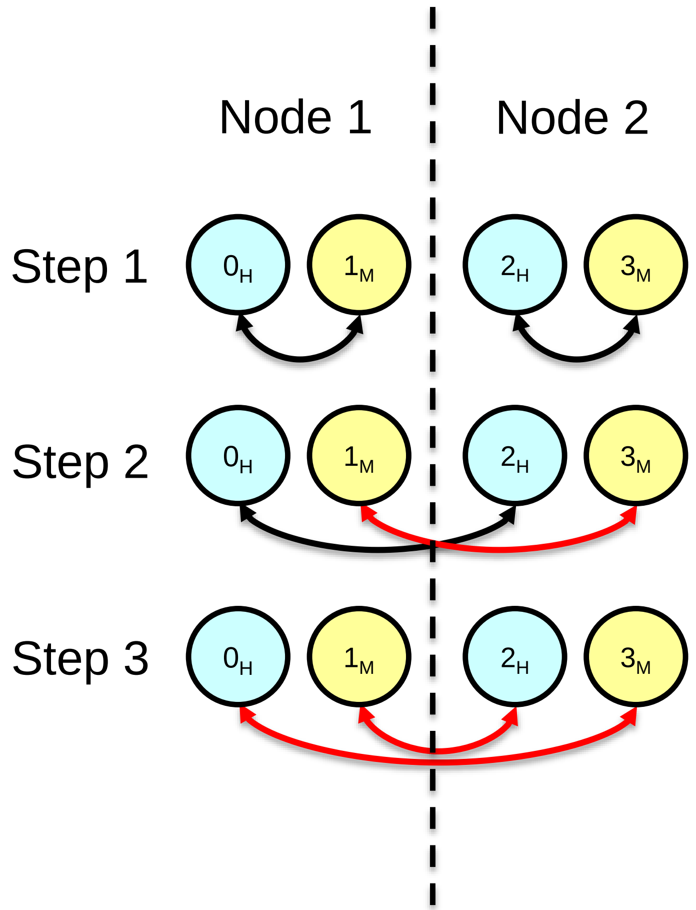

Distributed Scientific Applications (PSDNS, HPL, Graph500, Lulesh, Mini-apps, Sweep3D, Streaming-class)
Programming Models for Communication (MPI, PGAS)
Programming Models for Computation
Collectives (Bcast, Allgather, Alltoall)
Point-to-point operations (send, recv)
RMA ops (Put, Get, Fence, Flush)
Algorithms (knomial, Bruck's, pairwise, Ring)
Rendezvous Protocols (RDMA-Read, RDMA-Write)
Eager Protocols (send-recv, RDMA-Fastpath)
MUX
Network-centric DMA ops (IB -RC, UD, Mcast, offload)
PCIe-centric DMA ops (CUDA, SCIF)
CPU-centric ops (load, store)


Distributed Scientific Applications (PSDNS, HPL, Graph500, Lulesh, Mini-apps, Sweep3D, Streaming-class)
Programming Models for Communication (MPI, PGAS)
Programming Models for Computation
Collectives (Bcast, Allgather, Alltoall)
Point-to-point operations (send, recv)
RMA ops (Put, Get, Fence, Flush)
Algorithms (knomial, Bruck's, pairwise, Ring)
Rendezvous Protocols (RDMA-Read, RDMA-Write)
Eager Protocols (send-recv, RDMA-Fastpath)
Dictates execution time and energy usage
MUX
Network-centric DMA ops (IB -RC, UD, Mcast, offload)
PCIe-centric DMA ops (CUDA, SCIF)
CPU-centric ops (load, store)

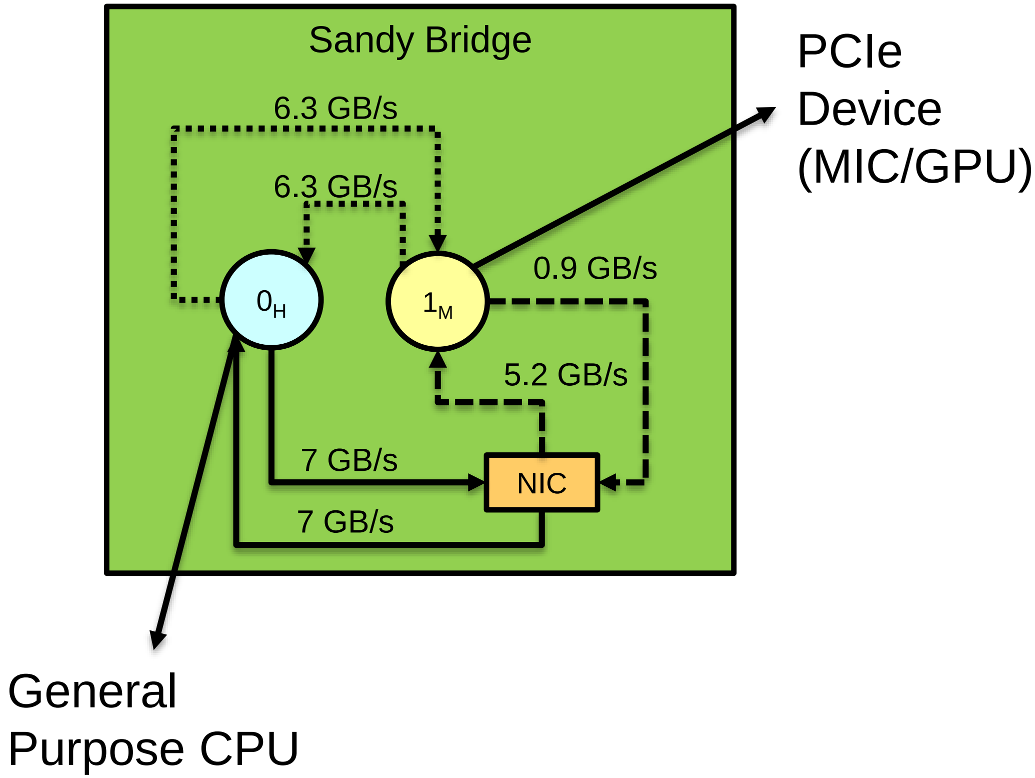

Distributed Scientific Applications (PSDNS, HPL, Graph500, Lulesh, Mini-apps, Sweep3D, Streaming-class)
Programming Models for Communication (MPI, PGAS)
Programming Models for Computation
Collectives (Bcast, Allgather, Alltoall)
Point-to-point operations (send, recv)
RMA ops (Put, Get, Fence, Flush)
Algorithms (knomial, Bruck's, pairwise, Ring)
Rendezvous Protocols (RDMA-Read, RDMA-Write)
Eager Protocols (send-recv, RDMA-Fastpath)
MUX
Network-centric DMA ops (IB -RC, UD, Mcast, offload)
PCIe-centric DMA ops (CUDA, SCIF)
CPU-centric ops (load, store)
Focus of Contributions

- Delegations mechanisms for dense collectives
- Path-cost aware collective adaptations
- Combining GPUDirect RDMA and hardware multicast for streaming apps
- Combining GPUDirect RDMA and CORE-Direct for non-blocking GPU collectives
- Application-oblivious Energy-Aware MPI (EAM) runtime

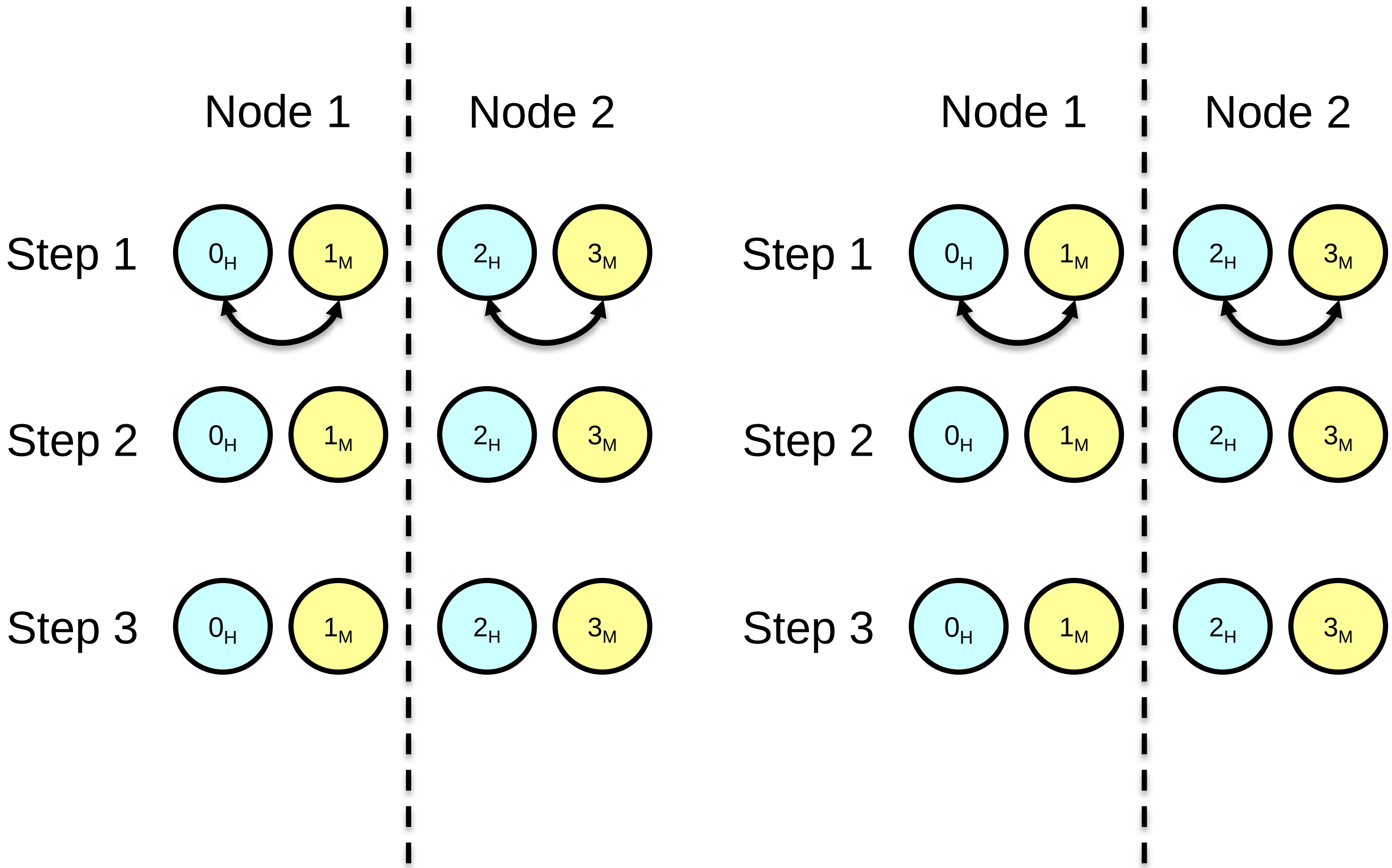
- Delegations mechanisms for dense collectives
- Path-cost aware collective adaptations
- Combining GPUDirect RDMA and hardware multicast for streaming apps
- Combining GPUDirect RDMA and CORE-Direct for non-blocking GPU collectives
- Application-oblivious Energy-Aware MPI (EAM) runtime



Default Pairwise Alltoall

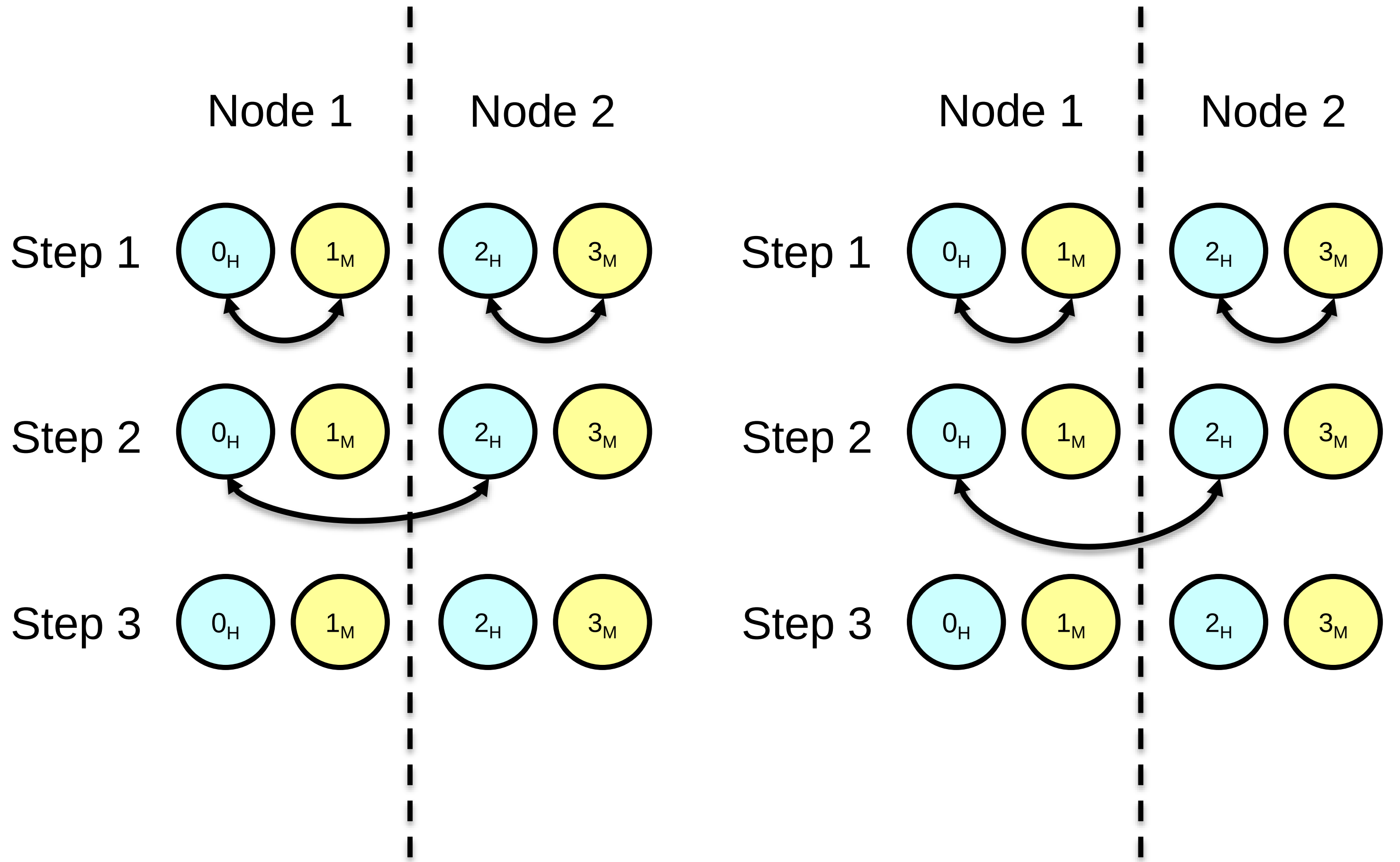


Pairwise Algorithm – used for large message alltoall operations



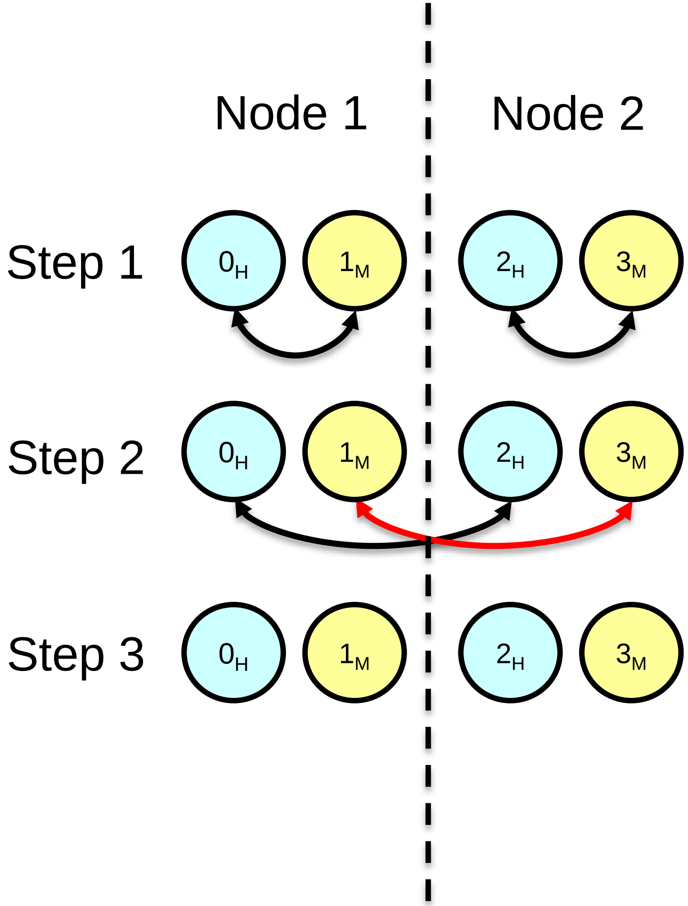
Default Pairwise Alltoall

Selective-rerouting Pairwise Alltoall (delegated)

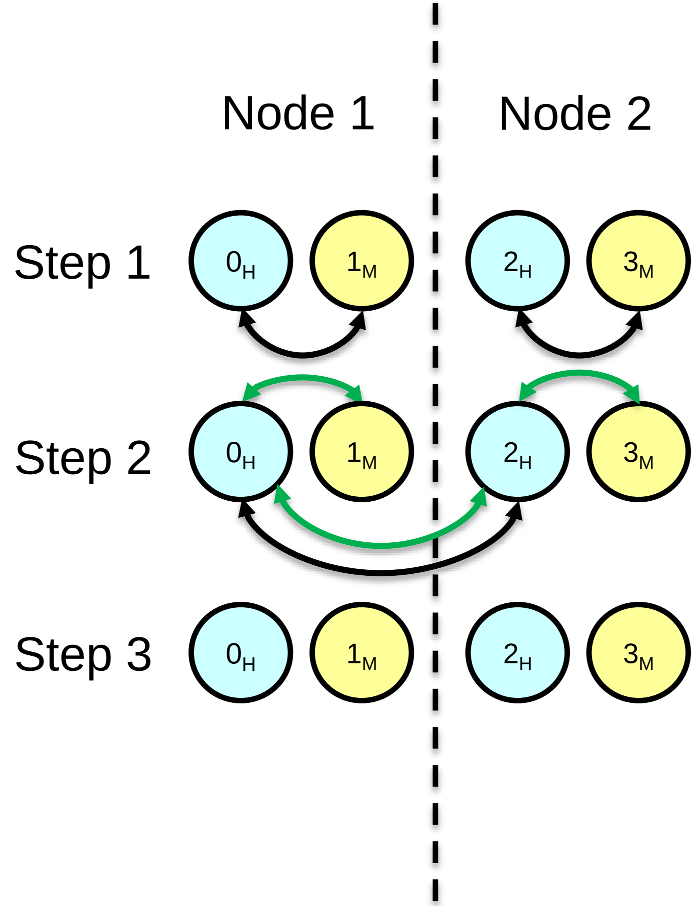


Default Pairwise Alltoall

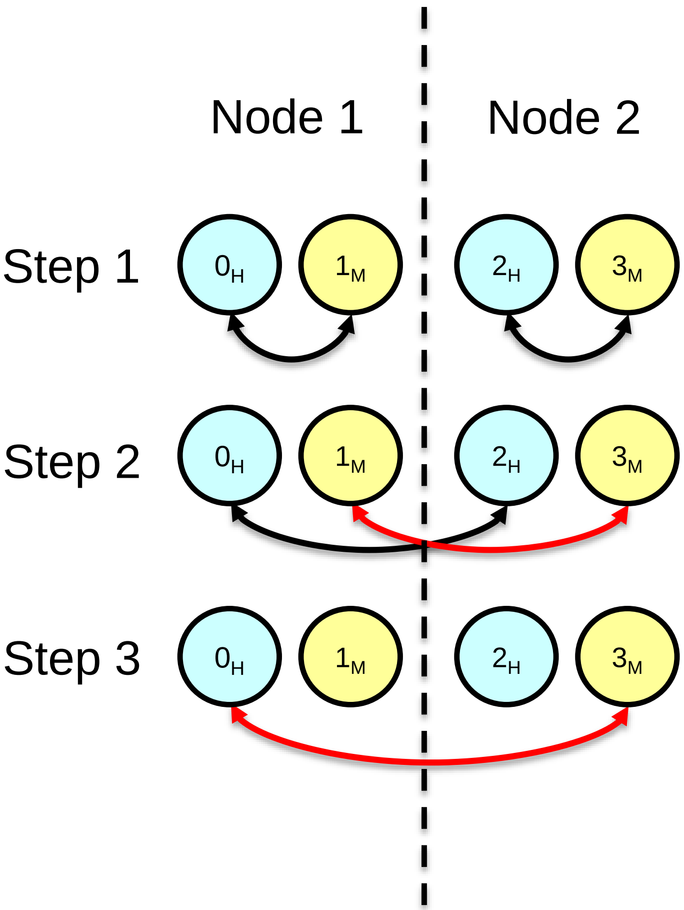
Selective-rerouting Pairwise Alltoall (delegated)



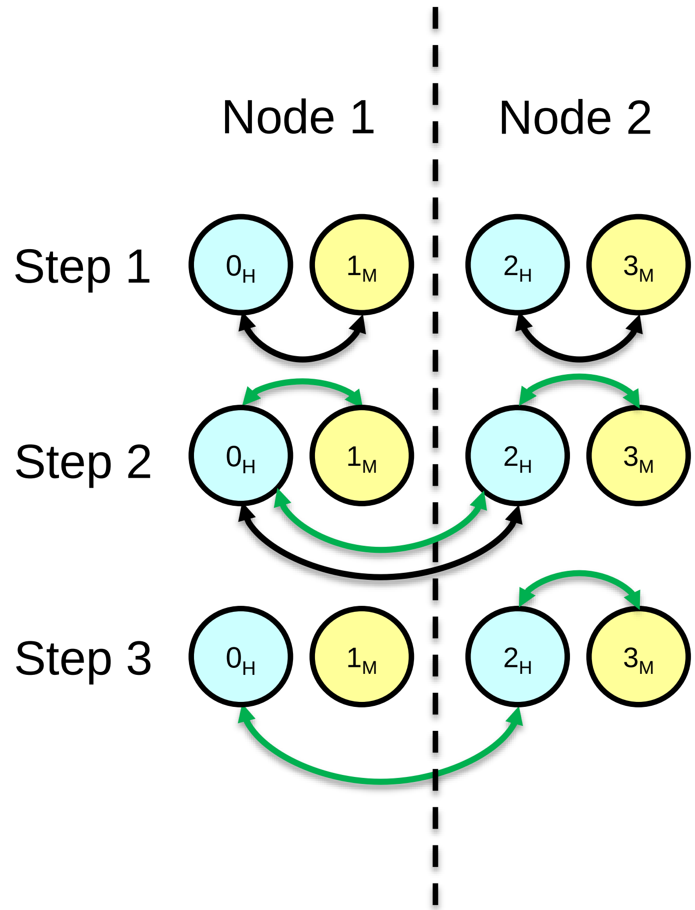
Default Pairwise Alltoall



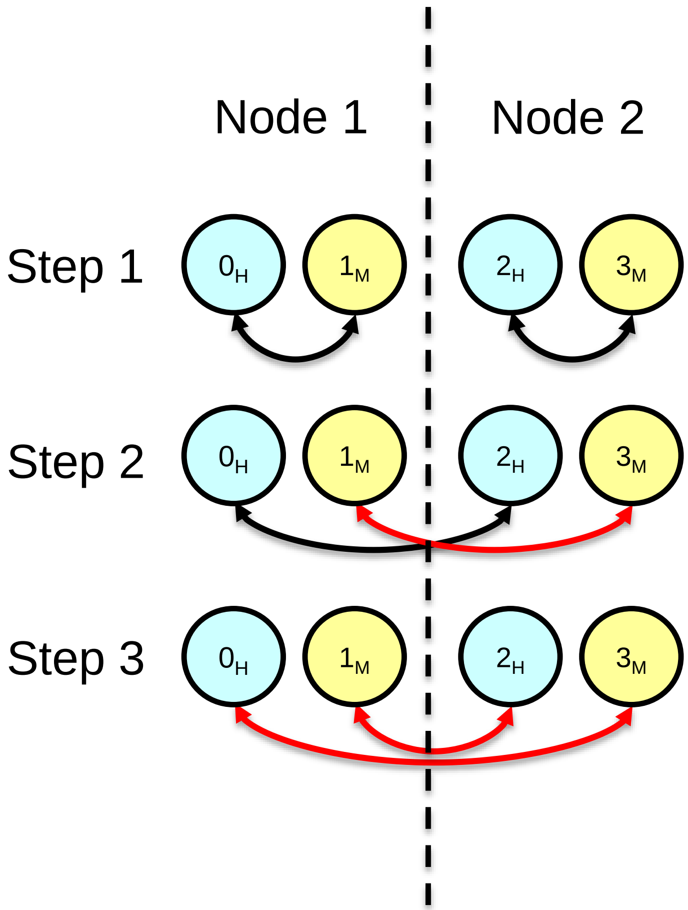
Selective-rerouting Pairwise Alltoall (delegated)



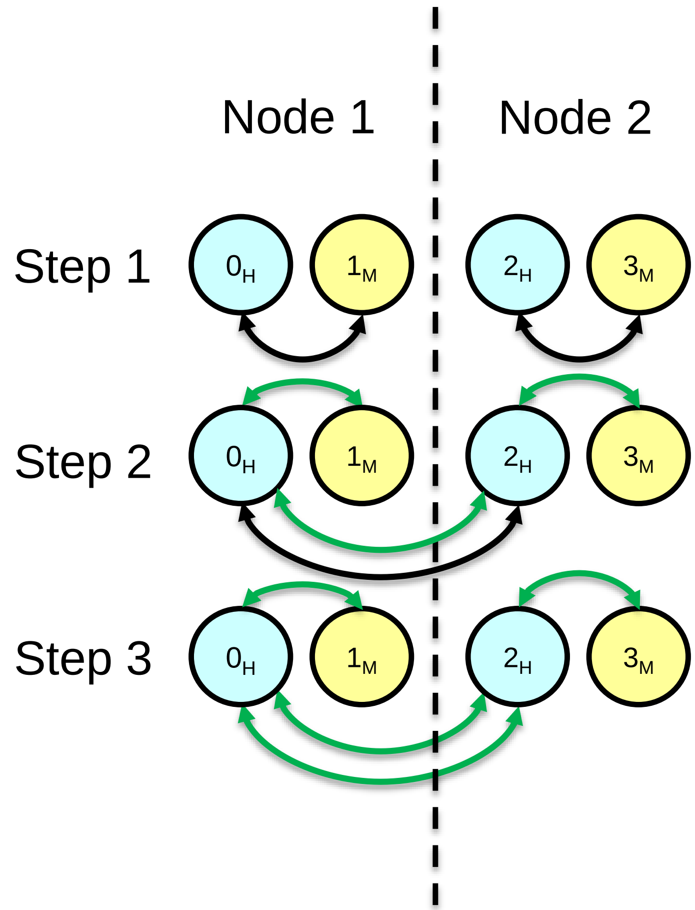
Default Pairwise Alltoall



Selective-rerouting Pairwise Alltoall (delegated)



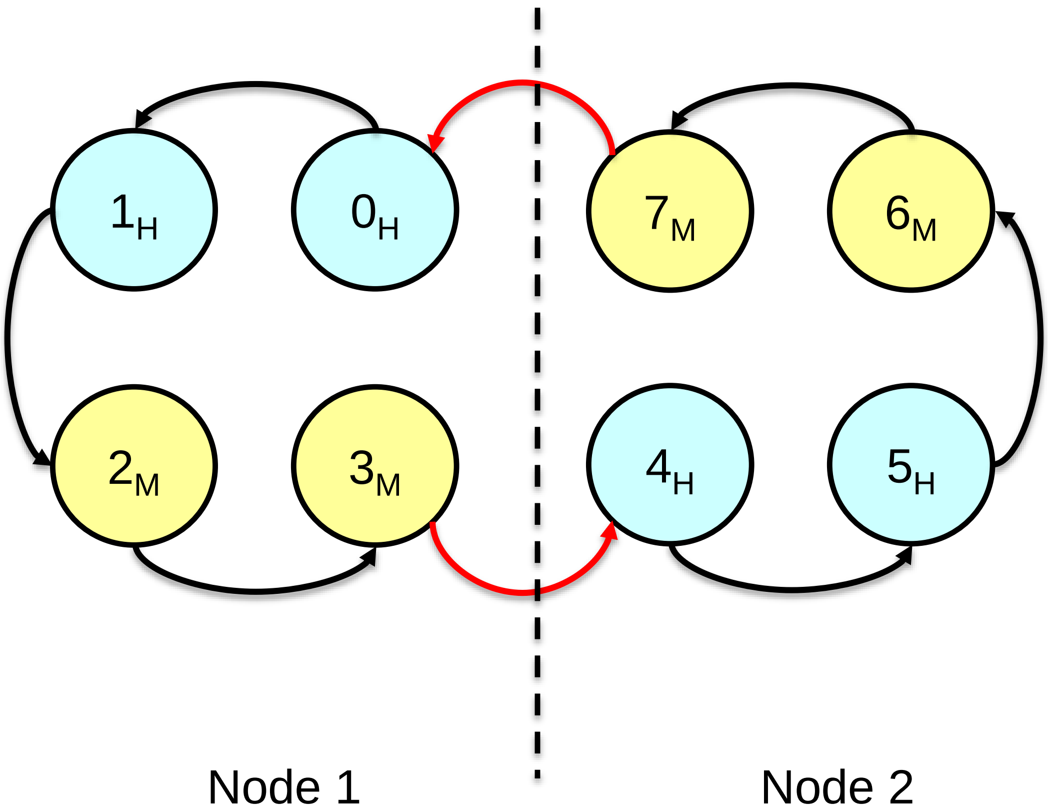
Default Pairwise Alltoall



Selective-rerouting Pairwise Alltoall (delegated)

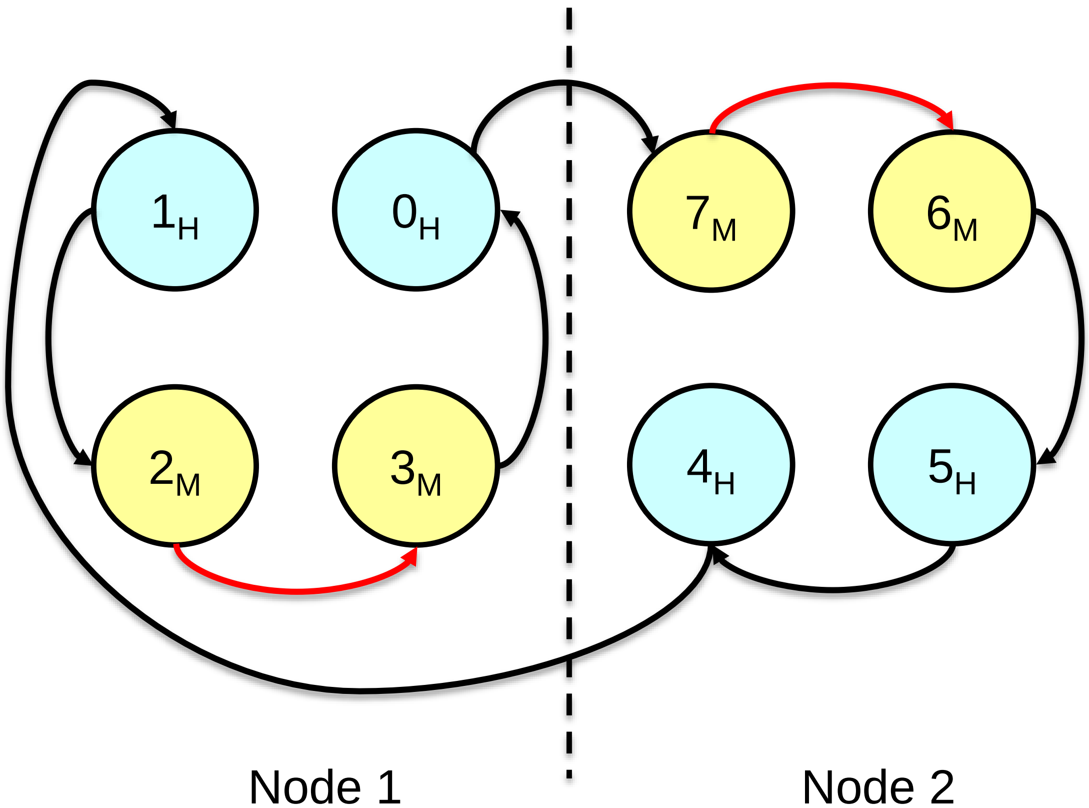
- Similar delegation approach applicable to other important collectives (Allgather, Allreduce, Bcast and Gather)
- Results

- Delegations mechanisms for dense collectives
- Path-cost aware collective adaptations
- Combining GPUDirect RDMA and hardware multicast for streaming apps
- Combining GPUDirect RDMA and CORE-Direct for non-blocking GPU collectives
- Application-oblivious Energy-Aware MPI (EAM) runtime



Default Ring algorithm

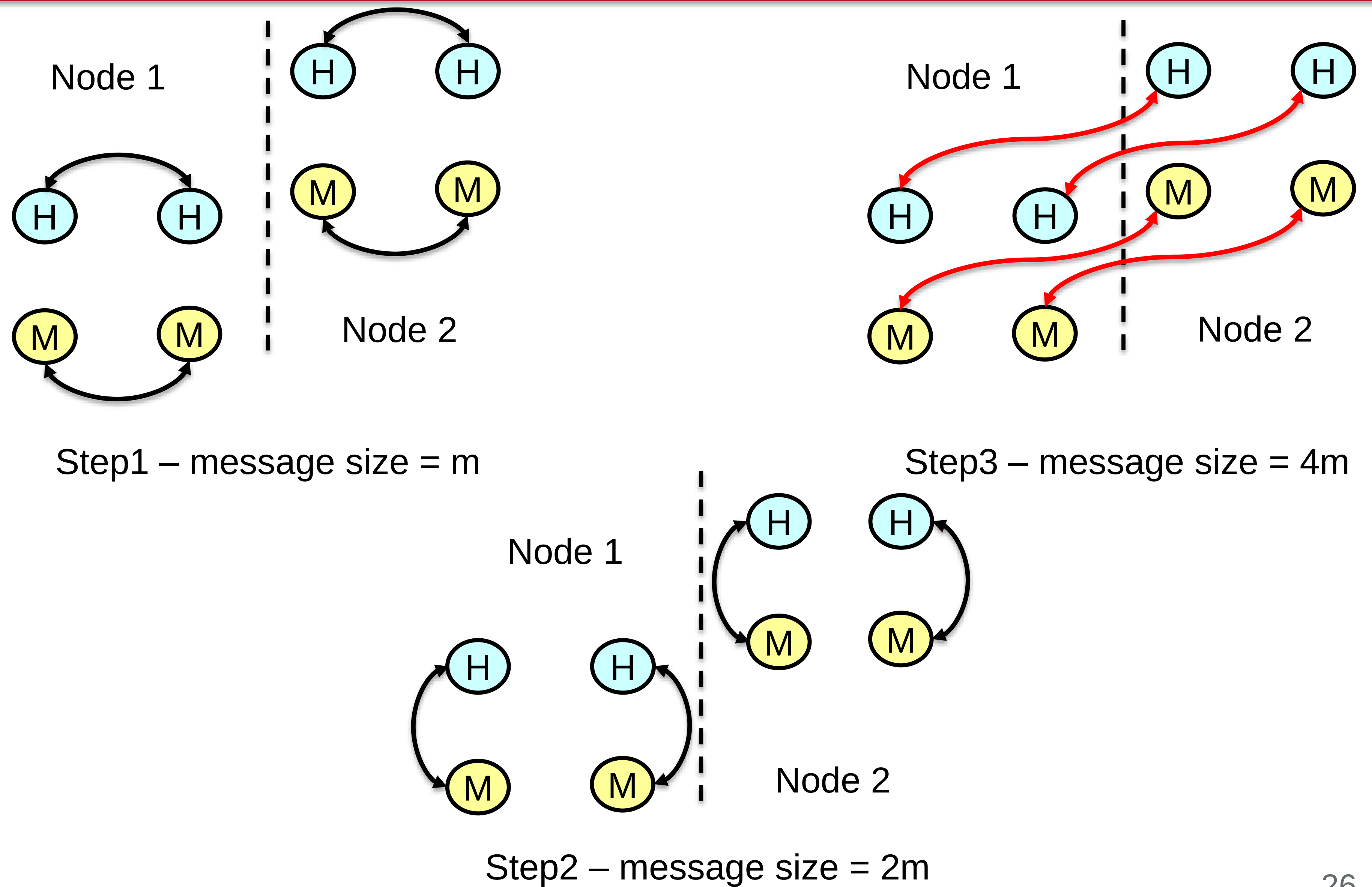
- Cost of the ring dictated by slowest sub-path in the ring
- All outgoing paths from the PCIe device are the slowest owing to read performance
- Total cost $= (n - 1) * T_{slowest}$



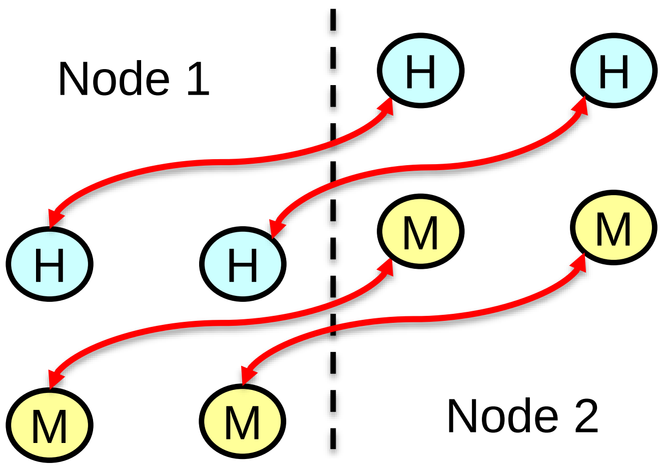
- The goal is to ensure that each node has a host processes lined up as border node
- If there is at least one host process/node then virtual ranks can be assigned such that no MIC processes are at the border
- Slow paths still exist but $Tnew_{slowest} < T_{slowest}$
- Total cost = $(n – 1) * Tnew_{slowest}$

Reordered Ring algorithm

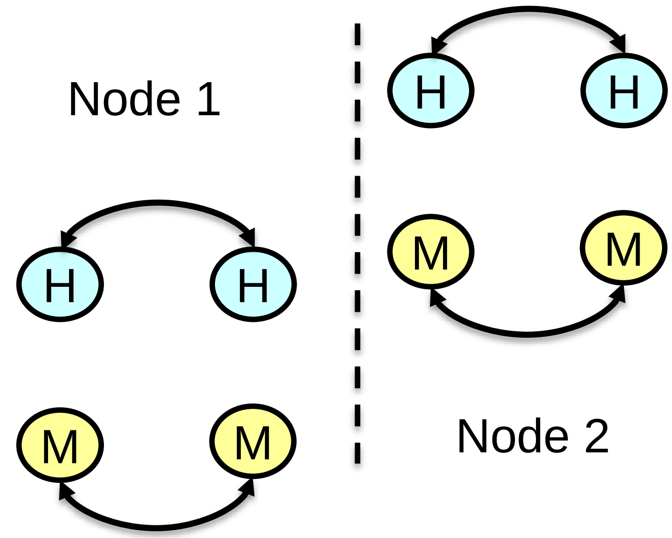
Default recursive doubling algorithm



Schedule-reordered recursive doubling algorithm

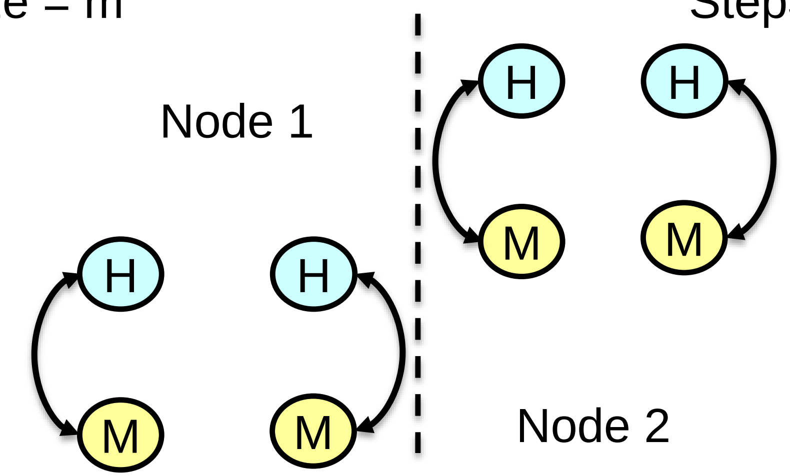


Step1 – message size = m



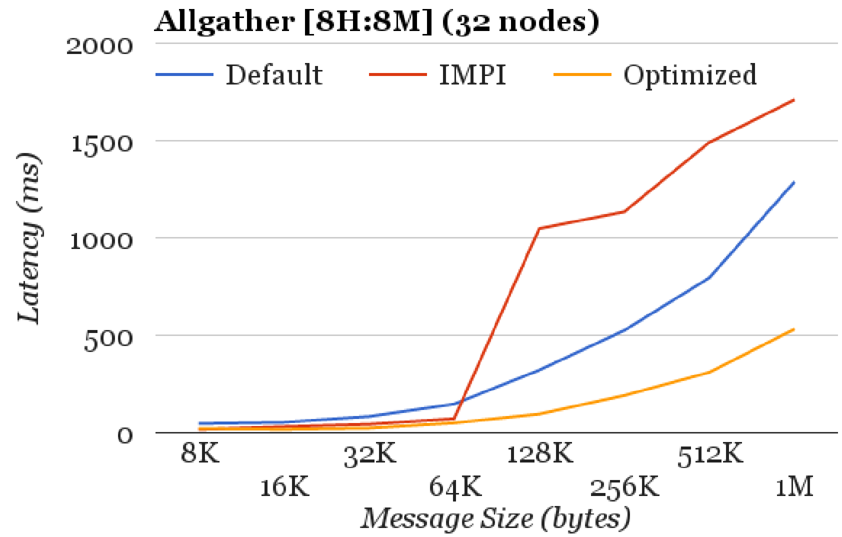
Step3 – message size = 4m

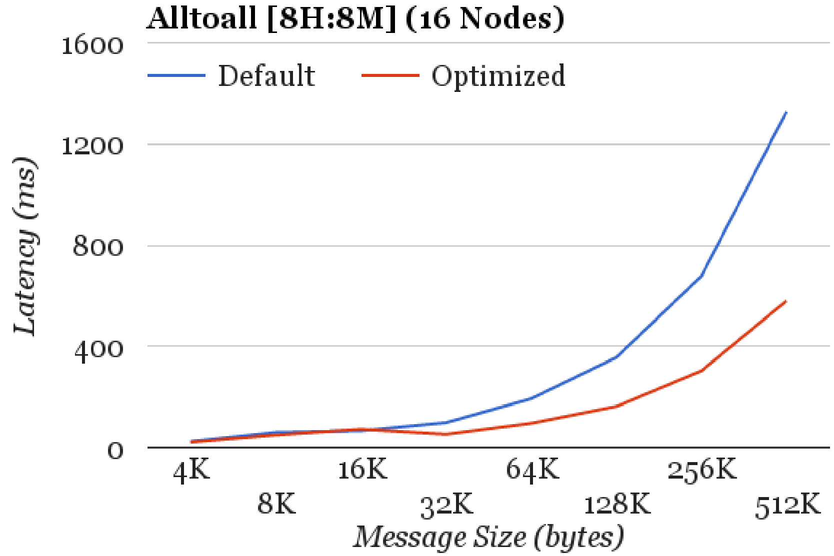
- Ensures that largest transfers don't occur on the slowest paths

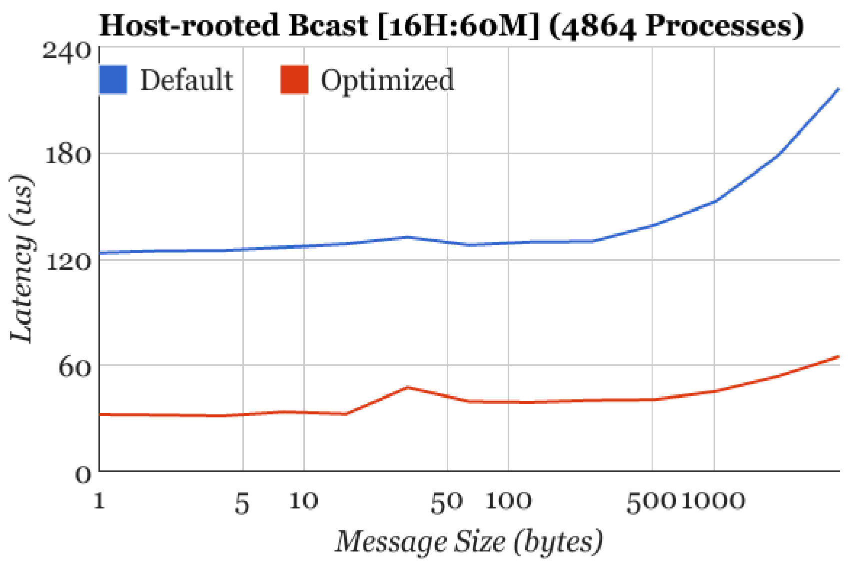


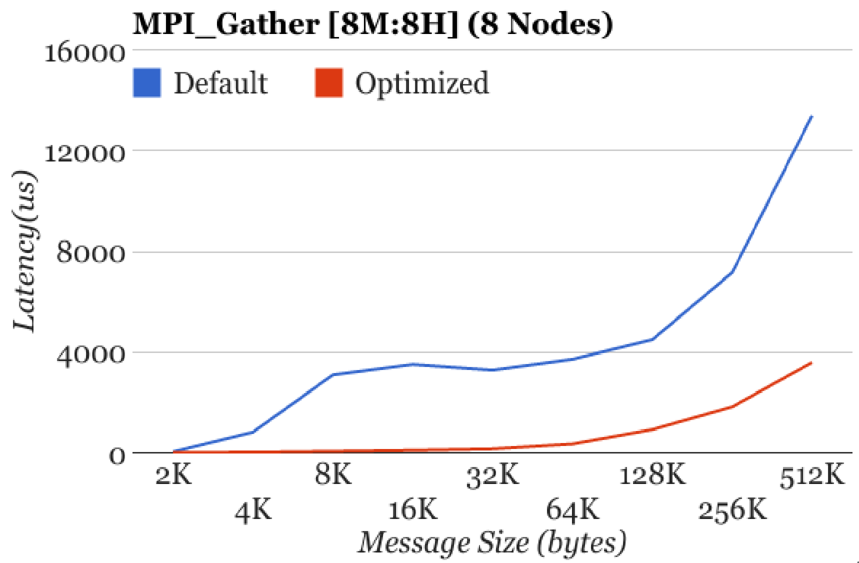
Step2 – message size = 2m

Results of delegation schemes and adaptations

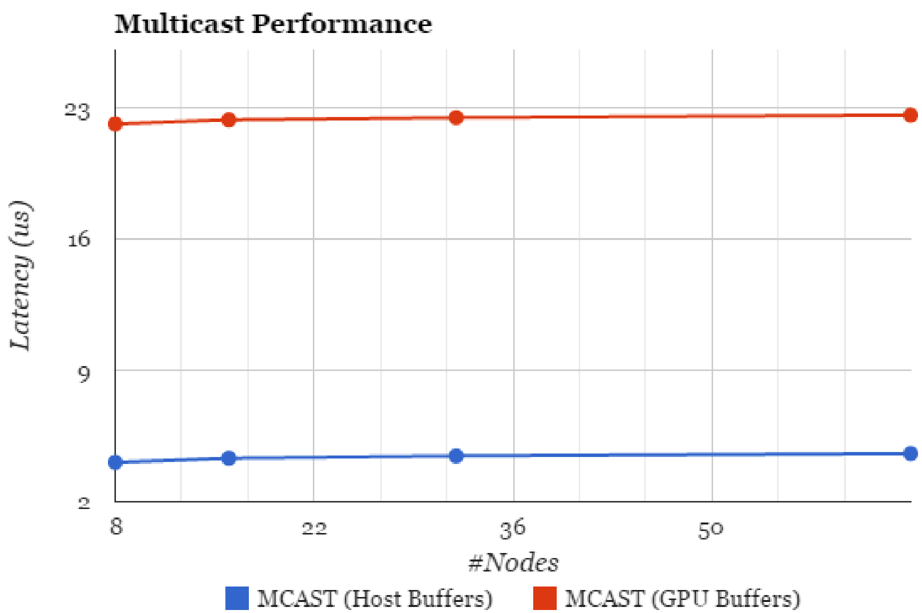


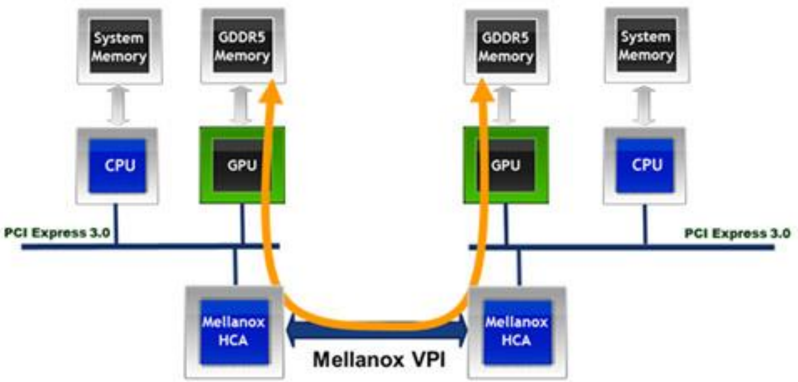






- Delegations mechanisms for dense collectives
- Path-cost aware collective adaptations
- Combining GPUDirect RDMA and hardware multicast for streaming apps
- Combining GPUDirect RDMA and CORE-Direct for non-blocking GPU collectives
- Application-oblivious Energy-Aware MPI (EAM) runtime





- Existing schemes that broadcast GPU data using hardware multicast did not exploit novel direct-GPU memory access mechanisms like GPUDirect RDMA (GDR)
- This leads to unexploited performance possibilities and detrimental to throughput-oriented streaming applications
- However, combining GDR with UD-based multicast is challenging

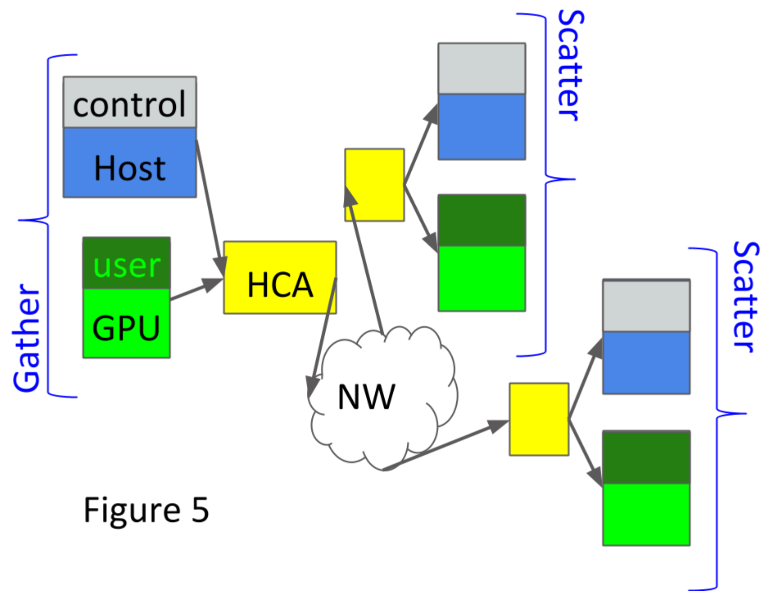
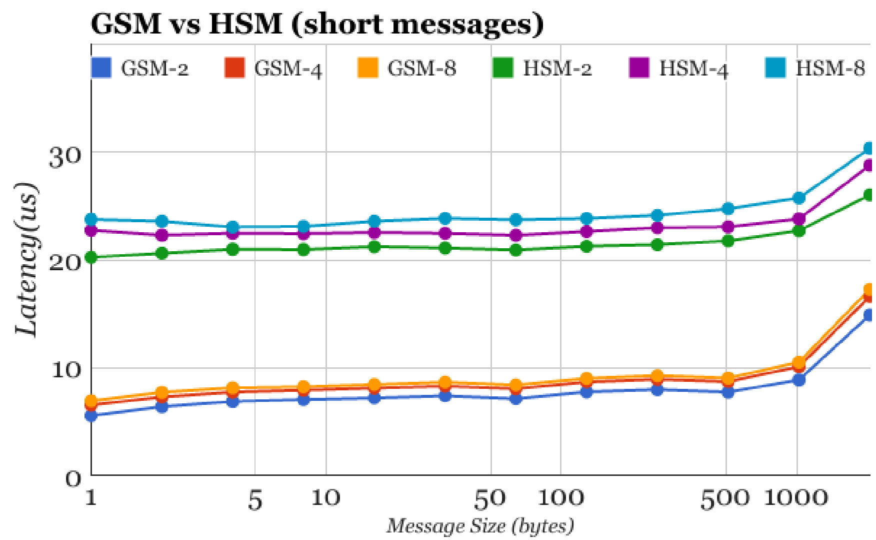


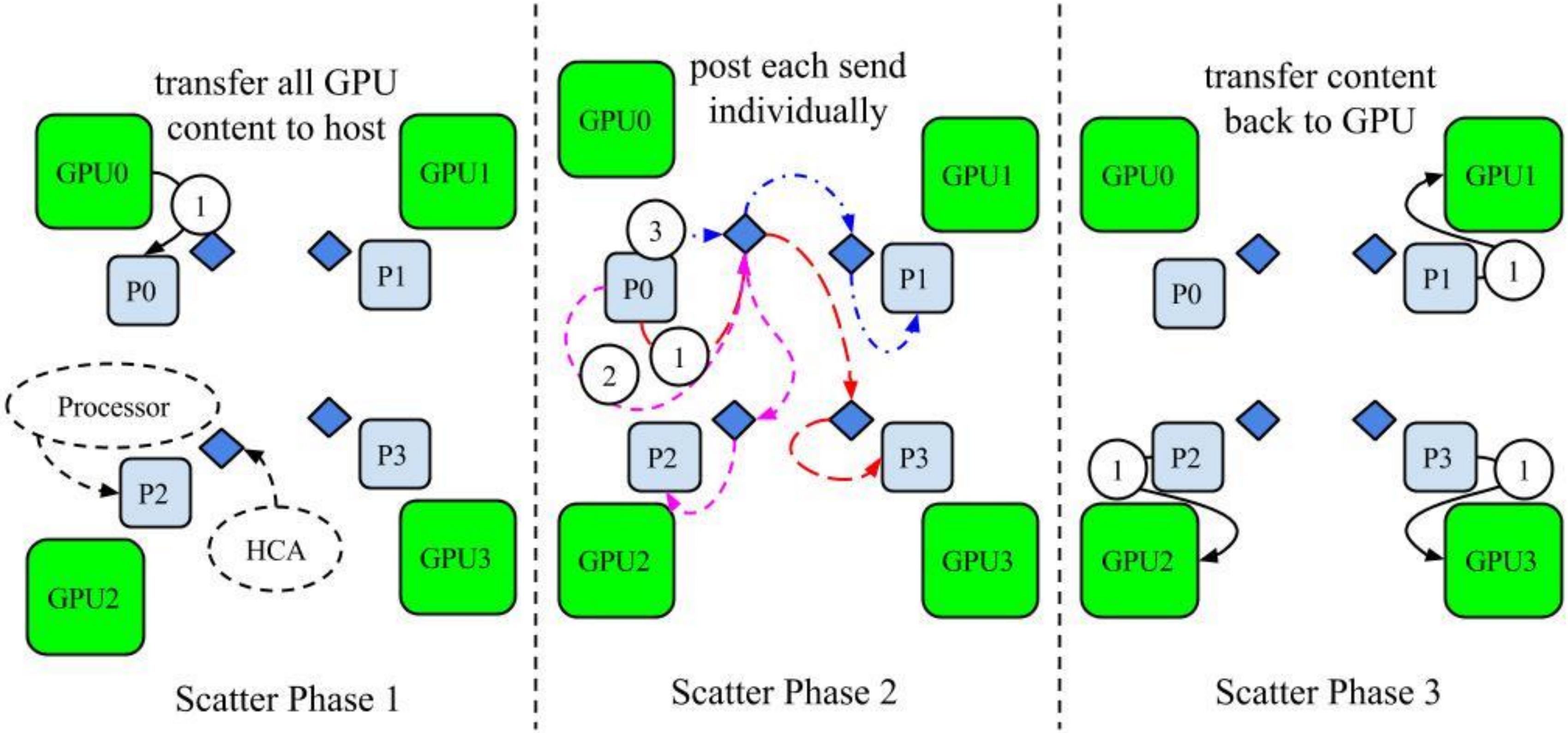
Figure 5

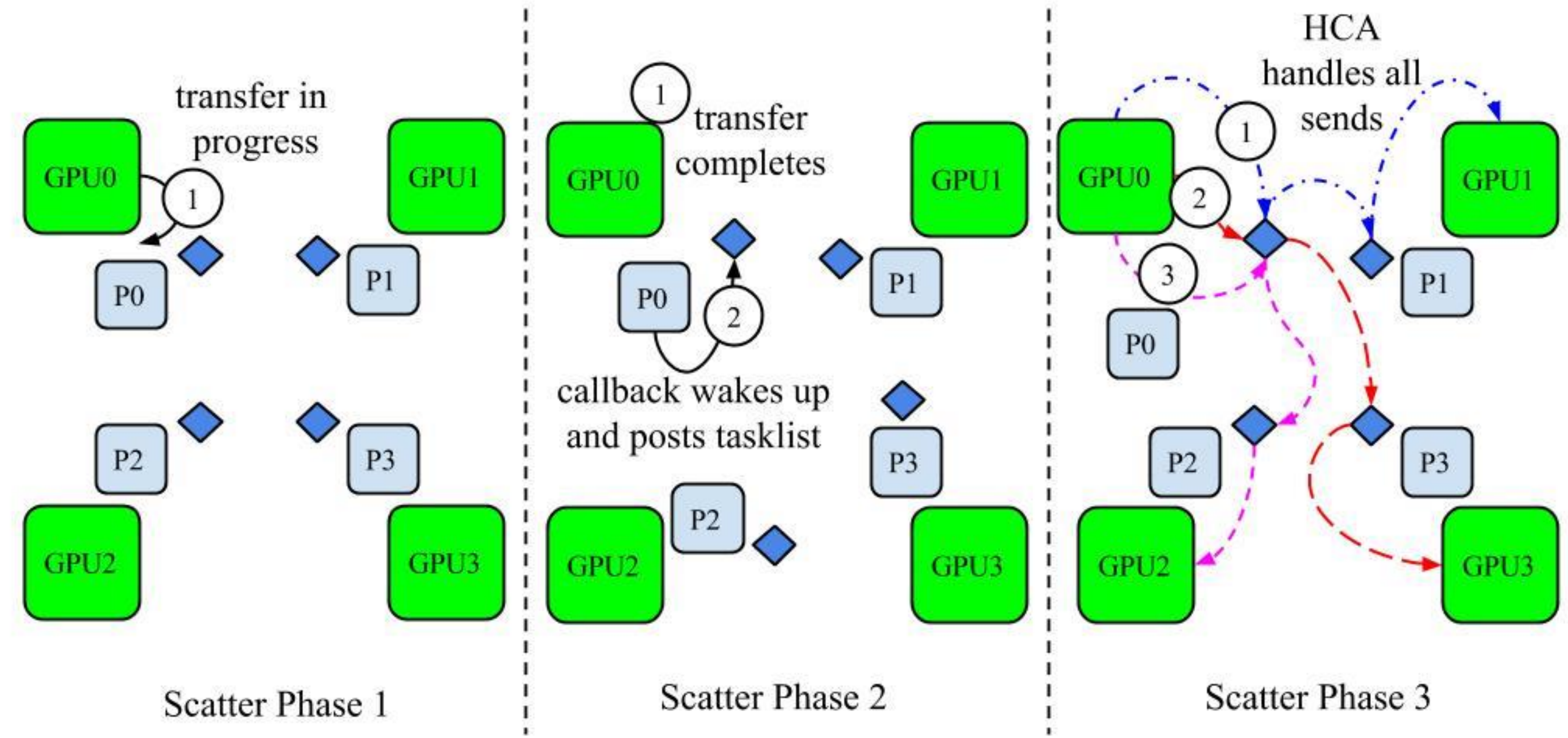


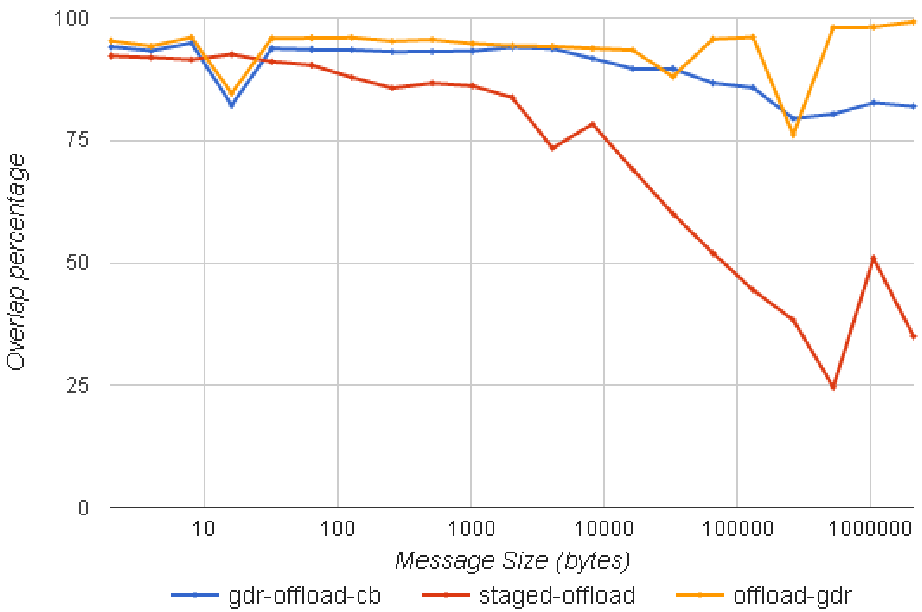
- We propose a scheme that leverages on the scatter-gather list abstraction to specify host and GPU memory regions and solve the problem of addressing UD-packet header data and GPU payloads
- An improvement of 50% reduction in latency is observed in comparison with host-staged approach with consistent scaling

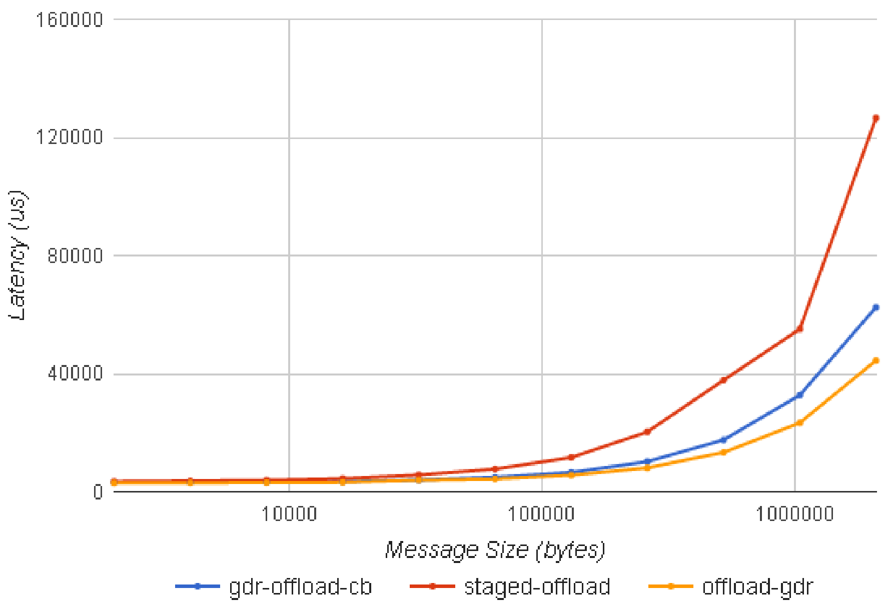
- Delegations mechanisms for dense collectives
- Path-cost aware collective adaptations
- Combining GPUDirect RDMA and hardware multicast for streaming apps
- Combining GPUDirect RDMA and CORE-Direct for non-blocking GPU collectives
- Application-oblivious Energy-Aware MPI (EAM) runtime

Default orchestration of non-blocking GPU Collectives



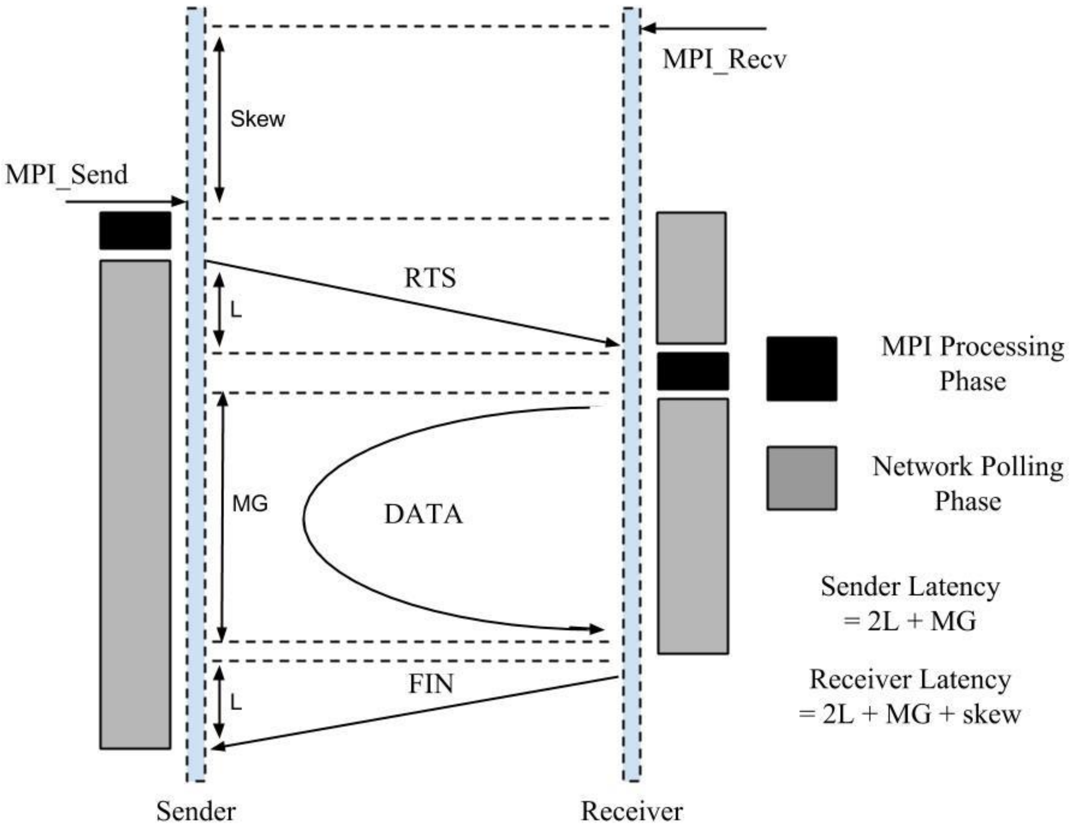

transfer in progress
GPU0
GPU1
1
P0
P1
P2
P3
GPU2
GPU3
Scatter Phase 1
1
transfer completes
GPU0
GPU1
P0
2
P1
callback wakes up and posts tasklist
P3
P2
GPU2
GPU3
Scatter Phase 2
HCA handles all sends
1
GPU0
2
GPU1
3
P1
P0
P2
P3
GPU2
GPU3
Scatter Phase 3



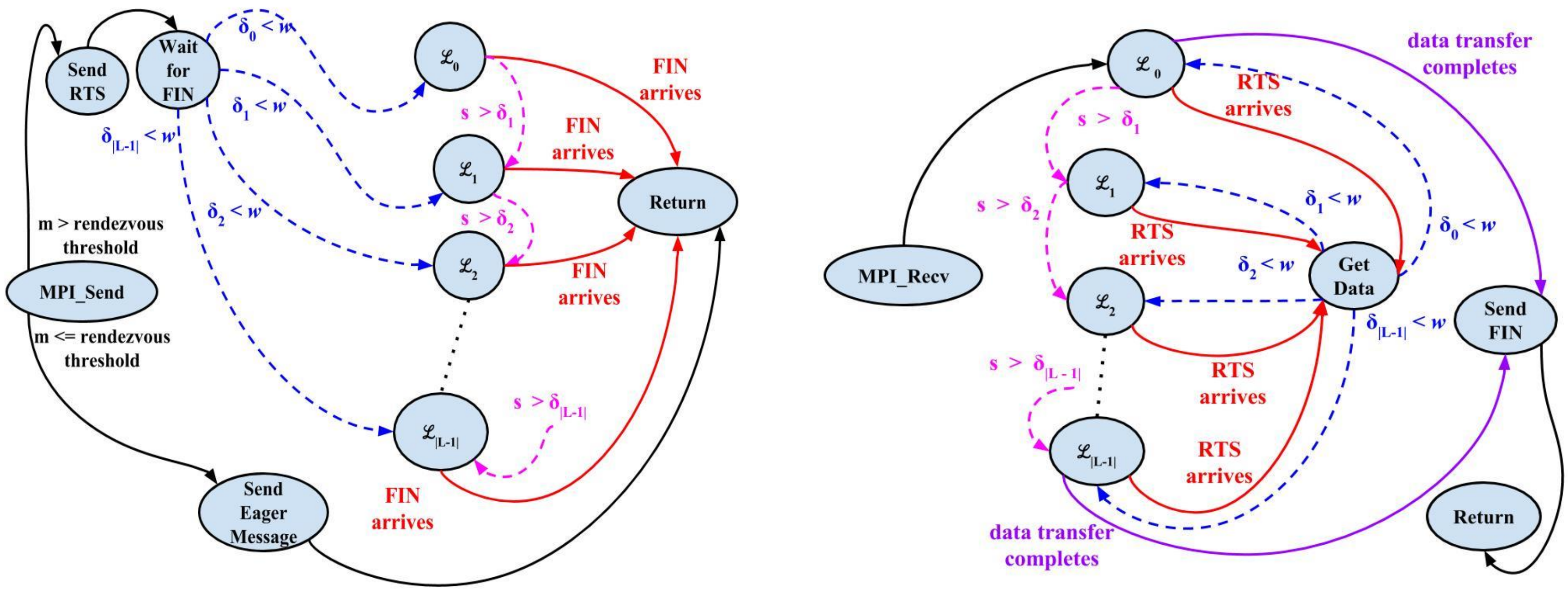


- We proposed schemes that leverages on CORE-Direct network offload technology and GPUDirect RDMA along with CUDA's callback mechanism to realize non-blocking GPU collectives
- For dense collectives such as Iallgather and Ialltoall, the proposed methods help achieve close to 100% overlap in the large message range and exhibit favorable latency in comparison with blocking counterparts

- Delegations mechanisms for dense collectives
- Path-cost aware collective adaptations
- Combining GPUDirect RDMA and hardware multicast for streaming apps
- Combining GPUDirect RDMA and CORE-Direct for non-blocking GPU collectives
- Application-oblivious Energy-Aware MPI (EAM) runtime



- State of the art approaches treat MPI as a blackbox and adopt aggressive power saving mechanisms which lead to degraded communication performance
- We propose rules that rely on intimate knowledge of the underlying MPI point-to-point and collective protocols in addition to communication time prediction models such as logGP



- Rules for applying appropriate energy levers for send and receive operations that use RGET protocol are shown.

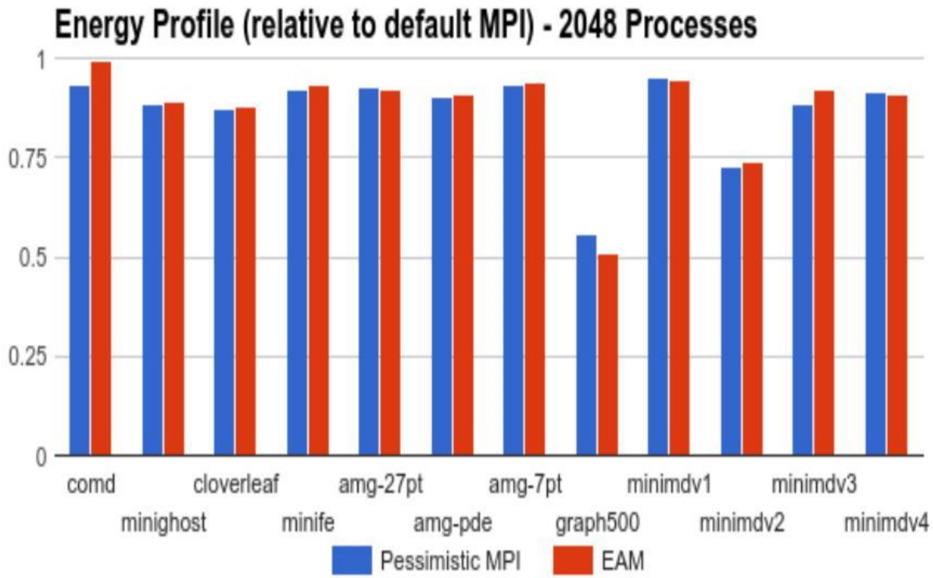


Figure 6

- Up to 40% improvement in energy usage of graph500
- Up to 10 application benchmarks showed no greater than user-allowed 5% degradation in overall performance
- Proposed approach works for both irregular and regular communication patterns

- Introduction
- Problem Statement
- Challenges
- Contributions and Results
- Future work and Conclusions

- The work proposes methods to reduce latency (heterogeneous clusters) and energy usage (homogeneous) of time consuming collective operations in heavily used MPI applications
- Results show methods are scalable and lead to application execution time improvement
- Future directions include formulating energy rules for RMA operations for both homogeneous and heterogeneous clusters as well designing novel asynchronous transfer mechanisms with NVIDIA's GPU offload technologies