IXPUG 2024

Modeling and Simulation of Collective Algorithms on HPC Network Topologies Using Structural Simulation Toolkit

Sai P. Chenna, Michael Steyer, Nalini Kumar,
Maria Garzaran, Philippe Thierry

intel

Introduction

Communication is **the** significant scaling bottleneck for large-scale AI training:

- Larger model sizes, larger datasets => more network traffic
- Growth of network BW lagging behind

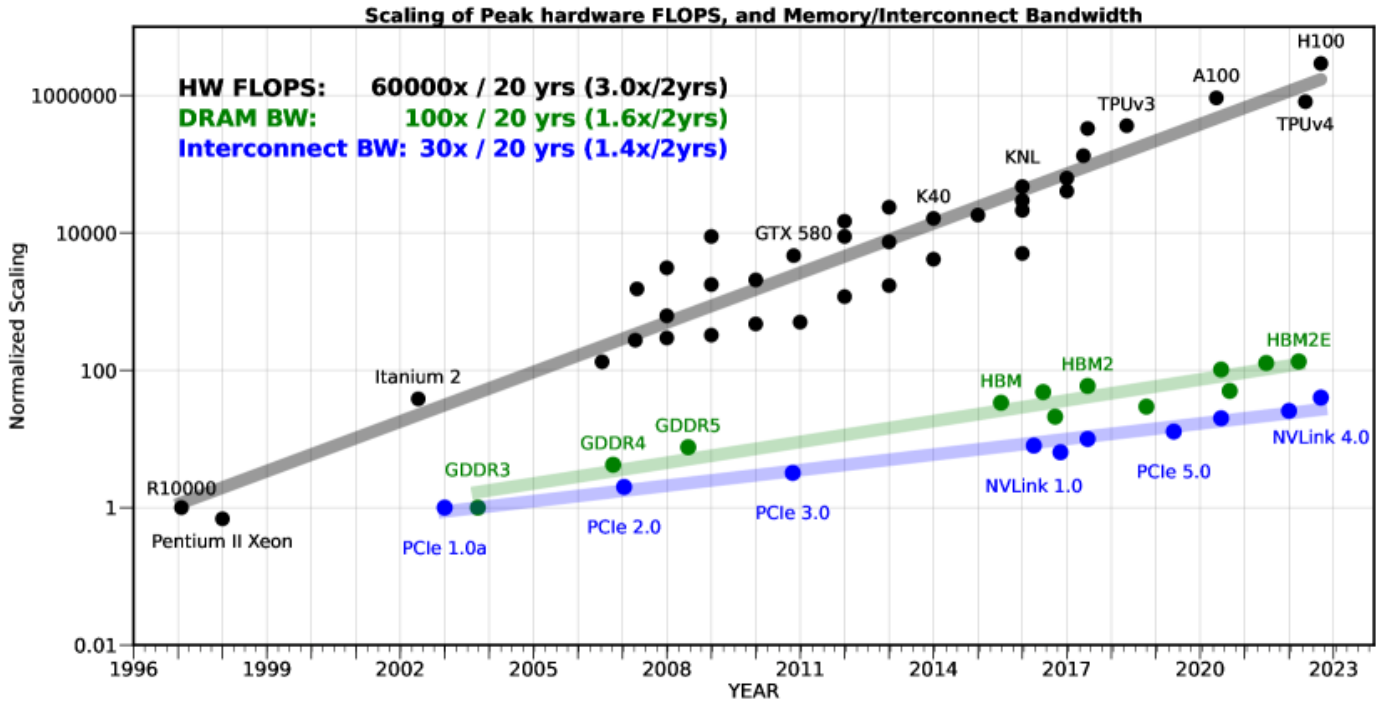
Collectives – dominant comm. pattern for Dist. Training

Various algorithms proposed to optimize communication performance

- Ring, Rabenseifner, Double Binary Tree, Topo-aware etc..
- Modeling these algorithms crucial for co-design

Our proposed solution : leverage scalable system-level simulator (SST*) for collective modeling

- We leverage SST's network modeling stack and extend it to build models for various collective algorithms
- Evaluate scaling efficiency of these collectives on different network topologies

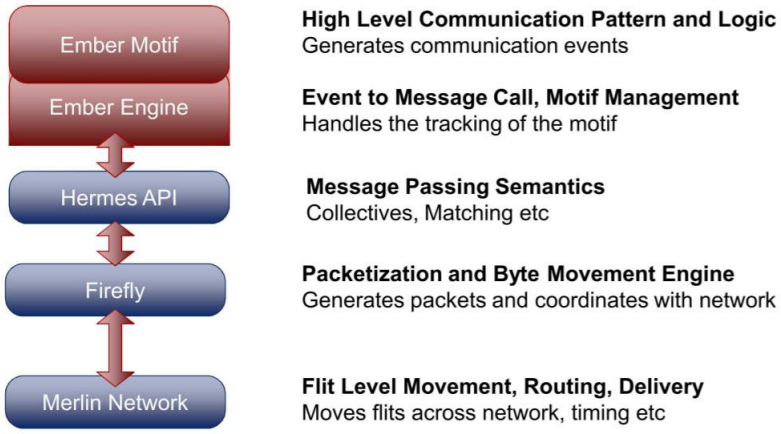


Evolution of peak Compute, Memory and Interconnect performance over the years
Image Courtesy: AI and Memory Wall

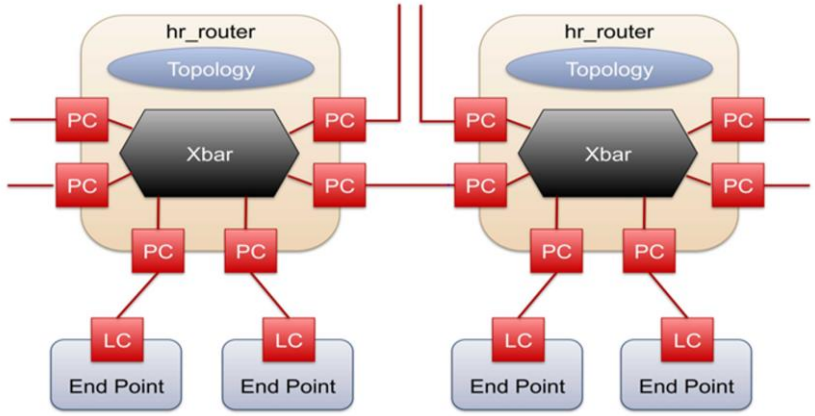
*Structural Simulation Toolkit (http://sst-simulator.org/)

Structural Simulation Toolkit: Network Modeling Stack

SST	▪ Scalable, parallel DES* aimed for HW/SW co-design ▪ Developed by Sandia National Laboratories (http://sst-simulator.org/)
Ember	▪ Network traffic generator based on application communication patterns ▪ Packages HPC comm patterns as motifs ▪ Provides a collection of diverse HPC comm patterns
Firefly	▪ Interface b/w network driver (Ember) and the router models (Merlin) ▪ Provides packetization and byte movement engine ▪ Implementation of communication protocols
Merlin	▪ Provides low-level, flexible networking components to simulate high-speed networks ▪ Models flit-level movement, physical routing and delivery of packets ▪ Supported topologies – Fat-tree, Dragonfly, N-dimension Torus



SST Network modeling stack
Image courtesy: Sandia National Laboratories



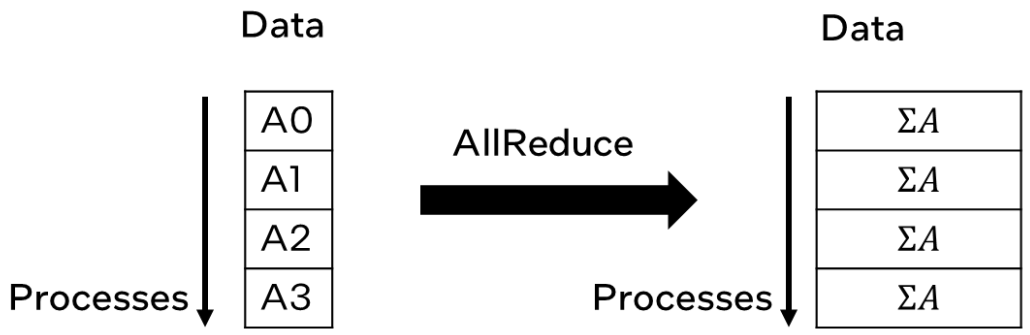
Merlin high-level overview
Image courtesy: Sandia National Laboratories

*Discrete Event Simulator

Collectives – AllReduce & Barrier

AllReduce

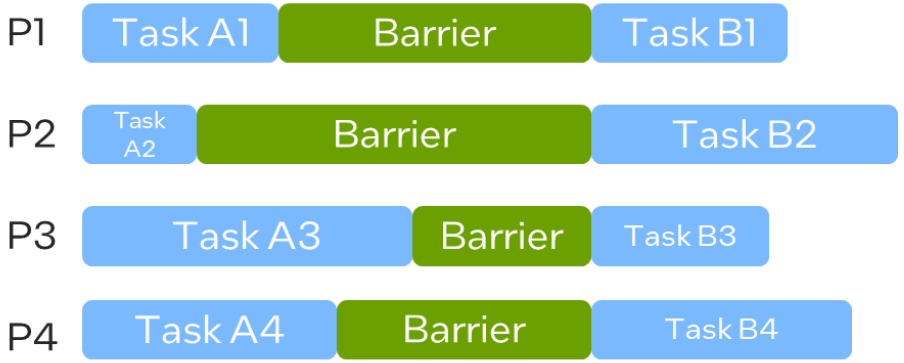
- Perform reduction on data distributed across processes
- E.g: sum model gradients across various processes in AI training
- Algorithms:
  - Binary Tree (default)
  - Rabenseifner
  - Ring



AllReduce

Barrier

- Synchronize across processes
- E.g: Init/Finalize, coordinating iterations, collectives
- Algorithms:
  - Binary Tree (default)
  - Dissemination

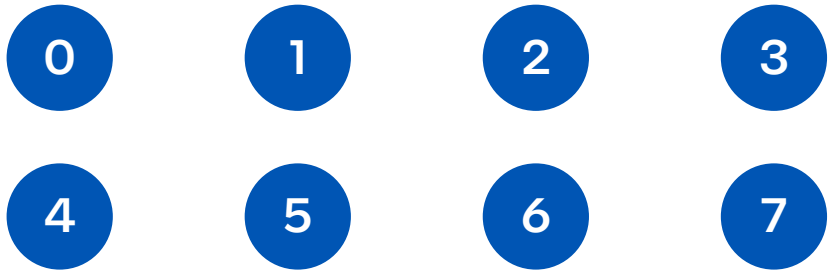


Barrier

AllReduce: Binary Tree

Binary Tree:

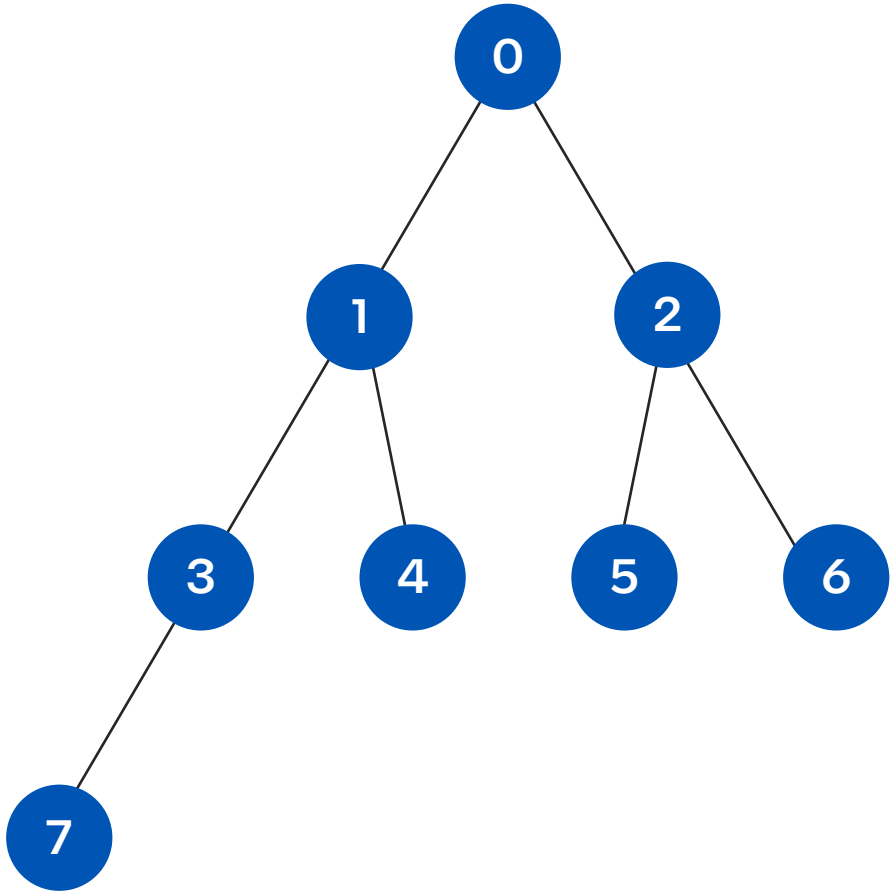
- Default implementation in SST (Firefly)
- Algorithm:
  - Two-phases: Reduce & Broadcast



AllReduce: Binary Tree

Binary Tree:

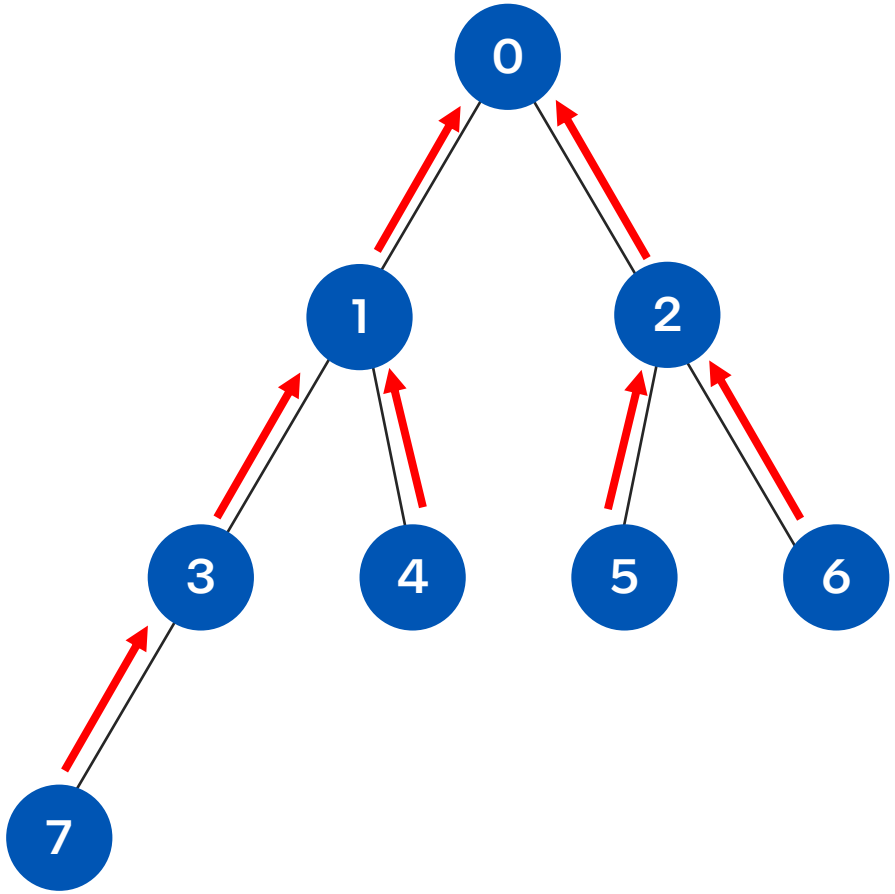
- Default implementation in SST (Firefly)
- Algorithm:
  - Two-phases: Reduce & Broadcast
  - Step 1: Initialize the tree



AllReduce: Binary Tree

Binary Tree AllReduce:

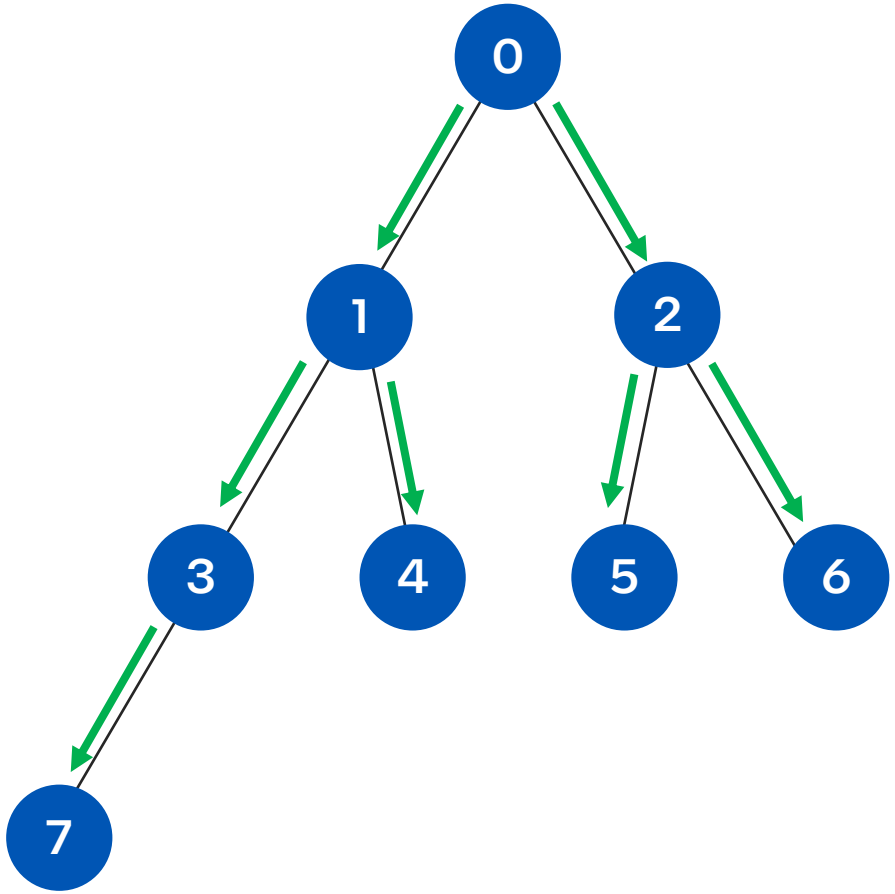
- Default implementation in SST (Firefly)
- Algorithm:
  - Two-phases: Reduce & Broadcast
  - Step 1: Initialize the tree
  - Step 2: Reduce
    - Receive data from children
    - Perform reduction
    - Send data to parent



AllReduce: Binary Tree

Binary Tree AllReduce:

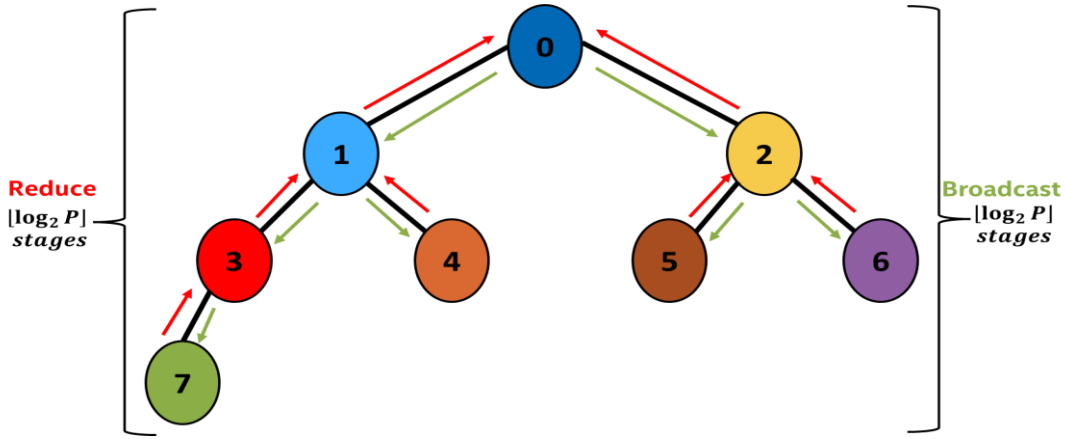
- Default implementation in SST (Firefly)
- Algorithm:
  - Two-phases: Reduce & Broadcast
  - Step 1: Initialize the tree
  - Step 2: Reduce
  - Step 3: Broadcast
    - Receive data from parent
    - Send data to children



AllReduce: Binary Tree

Binary Tree AllReduce:

- Default implementation in SST (Firefly)
- Algorithm:
  - Two-phases: Reduce & Broadcast
  - Step 1: Initialize the tree
  - Step 2: Reduction
  - Step 3: Broadcast
- Cost model
  - Assuming P processes, N message size
  - Latency per message – α
  - Transfer time per byte – β
  - Reduction cost per byte – γ



$$T_{cost\_tree\_reduce} = [\alpha \log_2 P] + [N \log_2 P \, \beta] + [N \log_2 P \, \gamma]$$

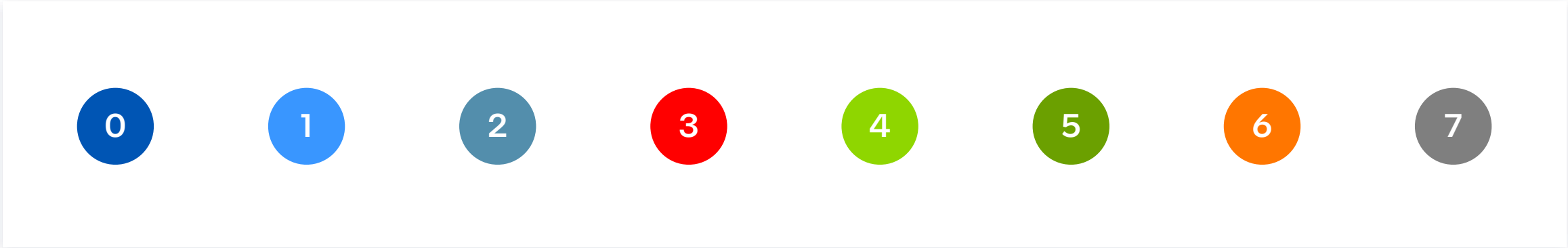
$$T_{cost\_tree\_broadcast} = [\alpha \log_2 P] + [N \log_2 P \, \beta]$$

$$T_{cost\_tree\_allreduce} = [2\alpha \log_2 P] + [2\log_2 PN\beta] + [N\log_2 P\gamma]$$

AllReduce: Rabenseifner

Rabenseifner:

- Provides better rank proximity over binary tree
- Algorithm:
  - Two phases: Reduce-Scatter & AllGather



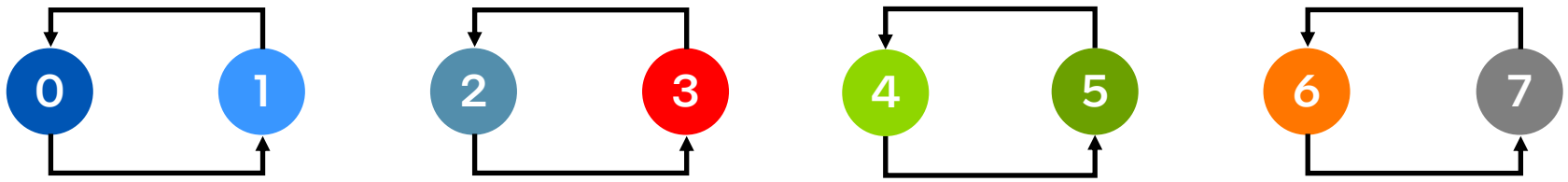
AllReduce: Rabenseifner

Rabenseifner:

- Provides better rank proximity over binary tree
- Algorithm:
  - Two phases: Reduce-Scatter & AllGather
  - Phase 1: Reduce-Scatter
    - Log P steps
    - Message size decreases by 2
    - Rank distance increases by 2

Stage 1

Message size: N/2



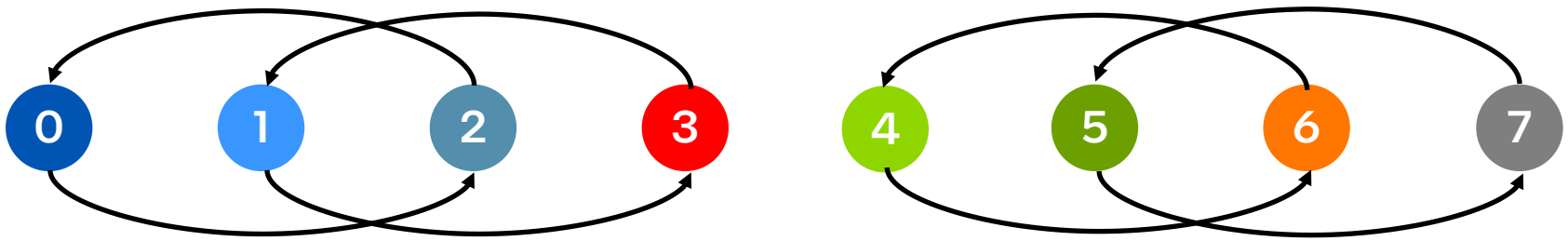
AllReduce: Rabenseifner

Rabenseifner:

- Provides better rank proximity over binary tree
- Algorithm:
  - Two phases: Reduce-Scatter & AllGather
  - Phase 1: Reduce-Scatter
    - Log P steps
    - Message size decreases by 2
    - Rank distance increases by 2

Stage 2

Message size: N/4



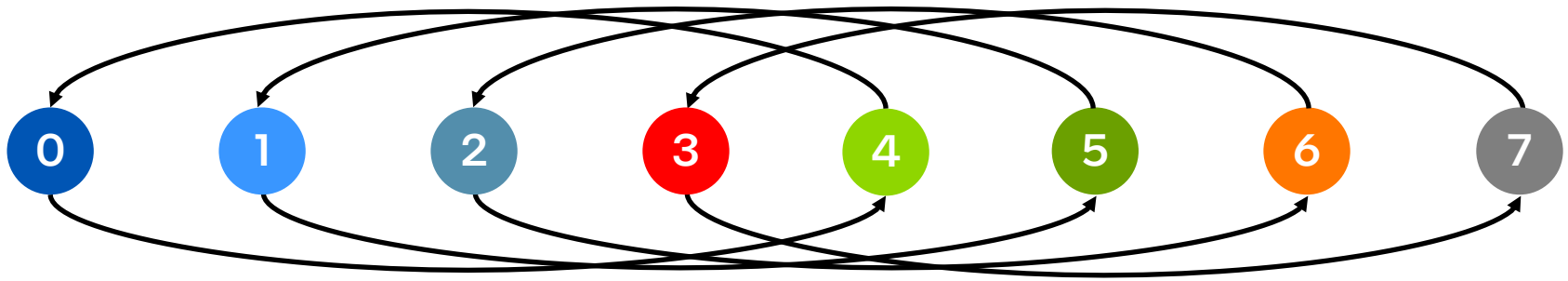
AllReduce: Rabenseifner

Rabenseifner:

- Provides better rank proximity over binary tree
- Algorithm:
  - Two phases: Reduce-Scatter & AllGather
  - Phase 1: Reduce-Scatter
    - Log P steps
    - Message size decreases by 2
    - Rank distance increases by 2

Stage 3

Message size: N/8



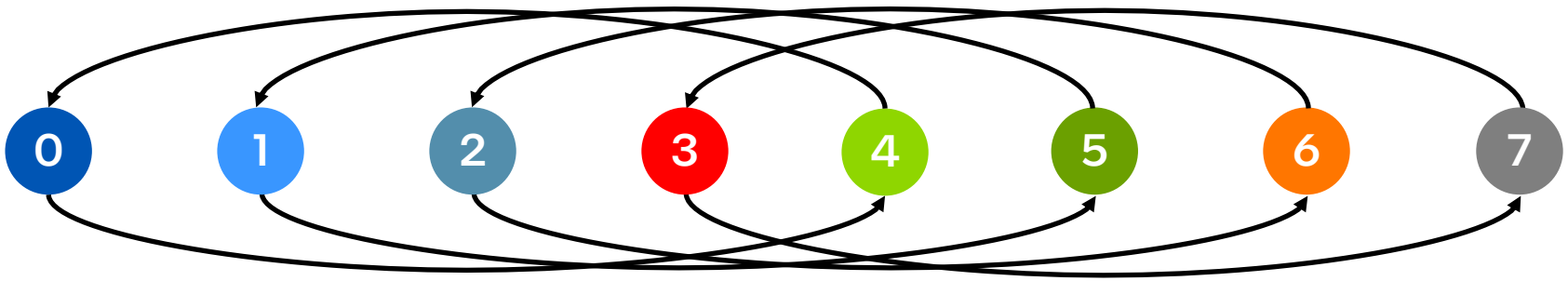
AllReduce: Rabenseifner

Rabenseifner:

- Provides better rank proximity over binary tree
- Algorithm:
  - Two phases: Reduce-Scatter & AllGather
  - Phase 1: Reduce-Scatter
  - Phase 2: AllGather
    - Log P steps
    - Message size increases by 2
    - Rank distance decreases by 2

Stage 1

Message size: N/8



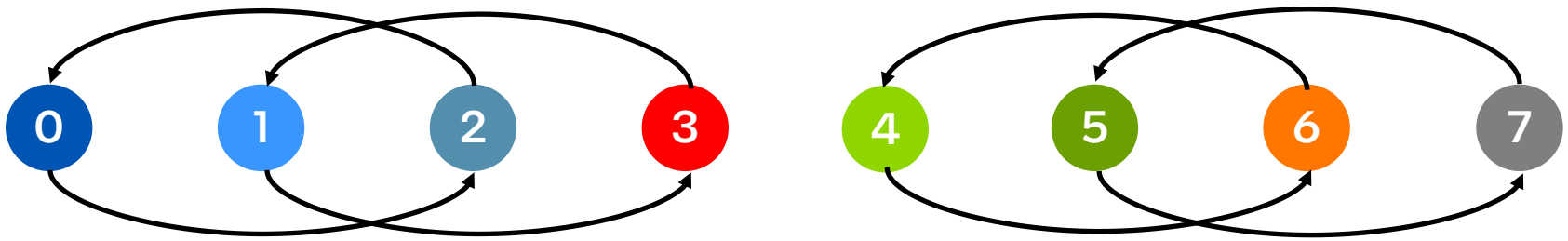
AllReduce: Rabenseifner

Rabenseifner:

- Provides better rank proximity over binary tree
- Algorithm:
  - Two phases: Reduce-Scatter & AllGather
  - Phase 1: Reduce-Scatter
  - Phase 2: AllGather
    - Log P steps
    - Message size increases by 2
    - Rank distance decreases by 2

Stage 2

Message size: N/4



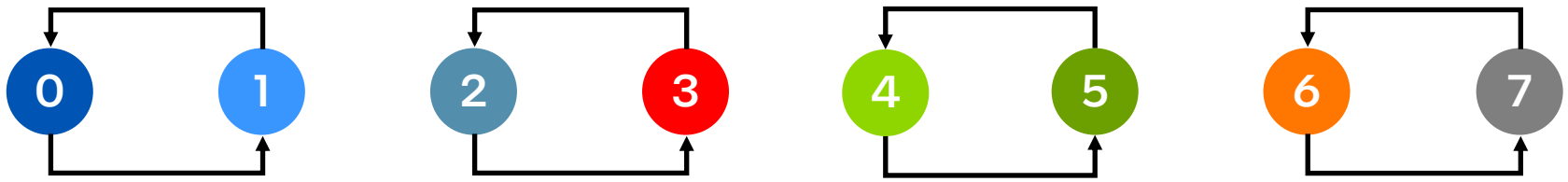
AllReduce: Rabenseifner

Rabenseifner AllReduce:

- Provides better rank proximity over binary tree
- Algorithm:
  - Two phases: Reduce-Scatter & AllGather
  - Phase 1: Reduce-Scatter
  - Phase 2: AllGather
    - Log P steps
    - Message size increases by 2
    - Rank distance decreases by 2

Stage 3

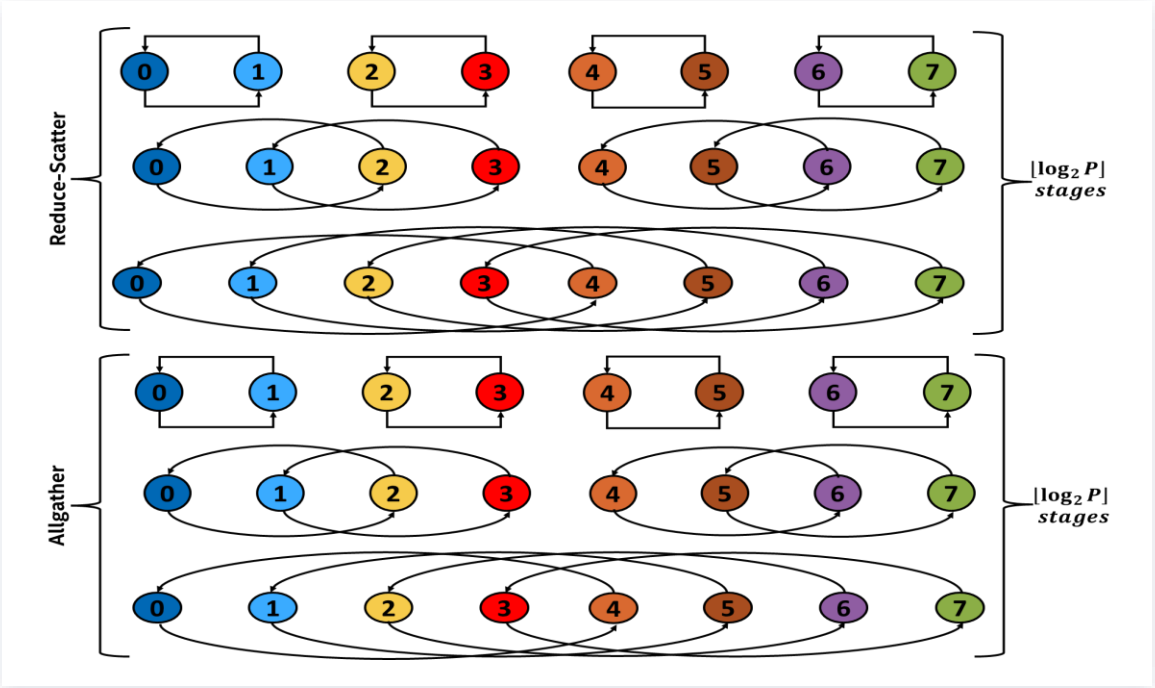
Message size: N/2



AllReduce: Rabenseifner

Rabenseifner :

- Provides better rank proximity over binary tree
- Algorithm:
  - Two phases: Reduce-Scatter & AllGather
  - Phase 1: Reduce-Scatter
  - Phase 2: AllGather



$$T_{cost\_rabenseifner\_reduce\_scatter} = [\alpha \log_2 P] + \left[\left(\frac{P-1}{P}\right) N\beta\right] + \left[\left(\frac{P-1}{P}\right) N\gamma\right]$$

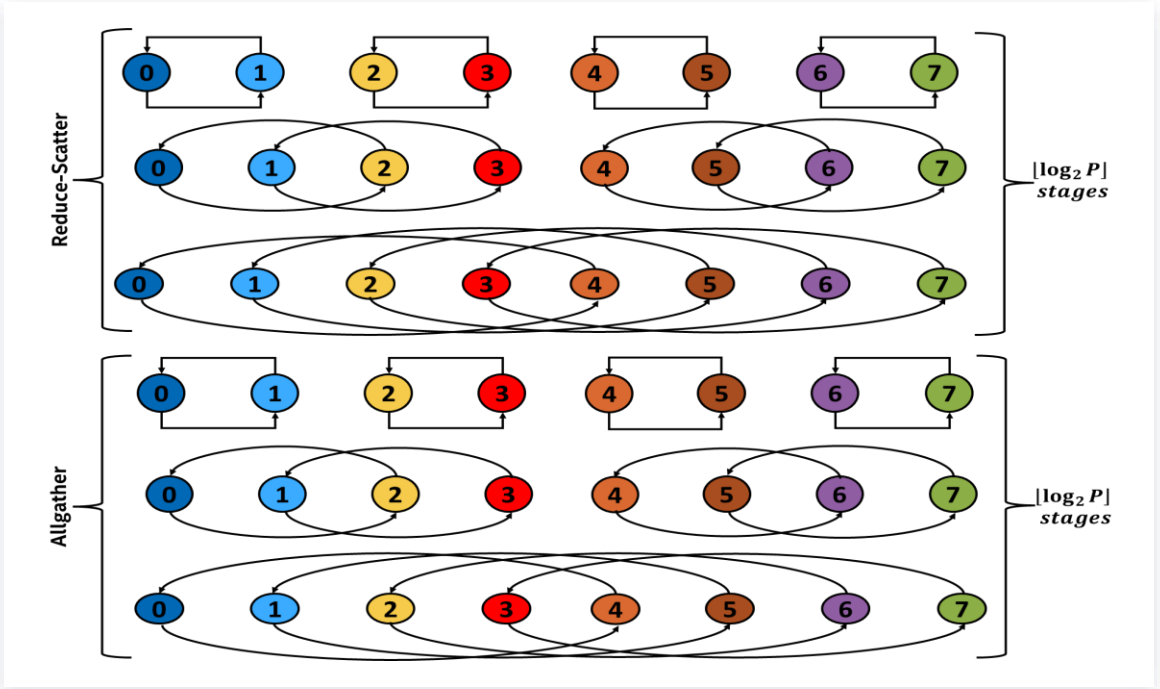
$$T_{cost\_rabenseifner\_allgather} = [\alpha \log_2 P] + \left[\left(\frac{P-1}{P}\right) N\beta\right]$$

$$T_{cost\_rabenseifner\_allreduce} = [2\alpha \log_2 P] + \left[2\left(\frac{P-1}{P}\right) N\beta\right] + \left[\left(\frac{P-1}{P}\right) N\gamma\right]$$

AllReduce: Rabenseifner

Rabenseifner :

- Provides better rank proximity over binary tree
- Algorithm:
  - Two phases: Reduce-Scatter & AllGather
  - Phase 1: Reduce-Scatter
  - Phase 2: AllGather
  - Two additional steps of data exchange for non-powers of 2



$$T_{cost\_rabenseifner\_reduce\_scatter} = [\alpha\,(\log_2 P + 1)] + \left[\left[\left(\frac{P-1}{P}\right) + 1\right]N\beta\right] + \left[\left[\left(\frac{P-1}{P}\right) + 1\right]N\gamma\right]$$

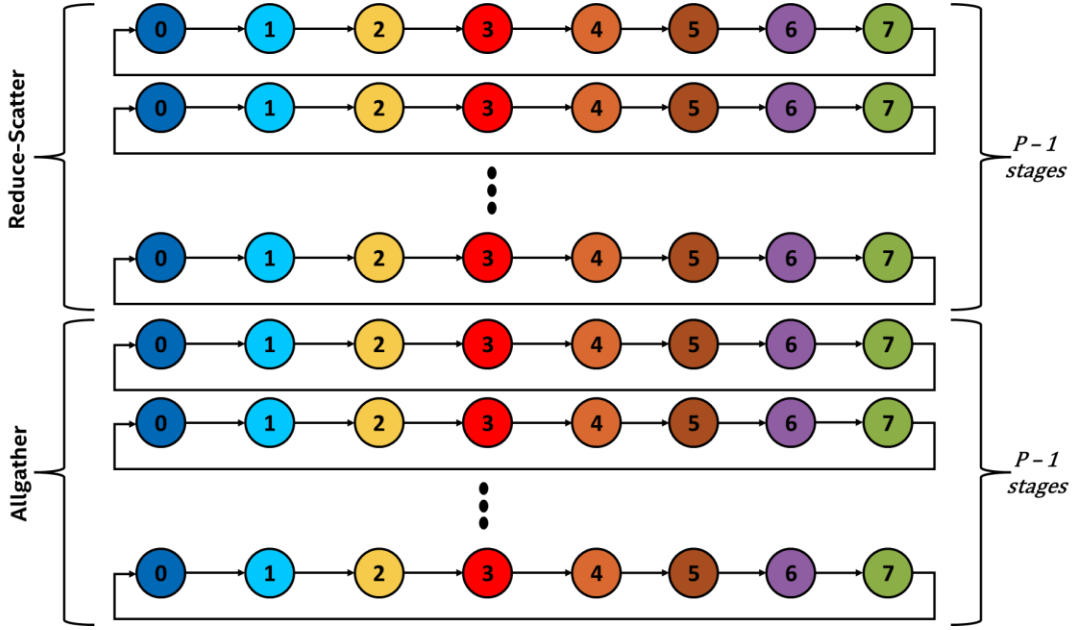
$$T_{cost\_rabenseifner\_allgather} = [\alpha\,(\log_2 P + 1)] + \left[\left[\left(\frac{P-1}{P}\right) + 1\right]N\beta\right]$$

$$T_{cost\_rabenseifner\_allreduce} = [2\alpha\,(\log_2 P + 1)] + \left[2\left[\left(\frac{P-1}{P}\right) + 1\right]N\beta\right] + \left[\left[\left(\frac{P-1}{P}\right) + 1\right]N\gamma\right]$$

AllReduce: Ring

Ring AllReduce:

- Optimal rank proximity
- Algorithm:
  - Two phases: Reduce-Scatter & AllGather
  - Fixed communicating pairs
    - Rank distance 1
    - Constant message size: N/P



$$T_{cost\_ring\_reduce\_scatter} = [\alpha(P-1)] + \left[\left(\frac{P-1}{P}\right)N\beta\right] + \left[\left(\frac{P-1}{P}\right)N\gamma\right]$$

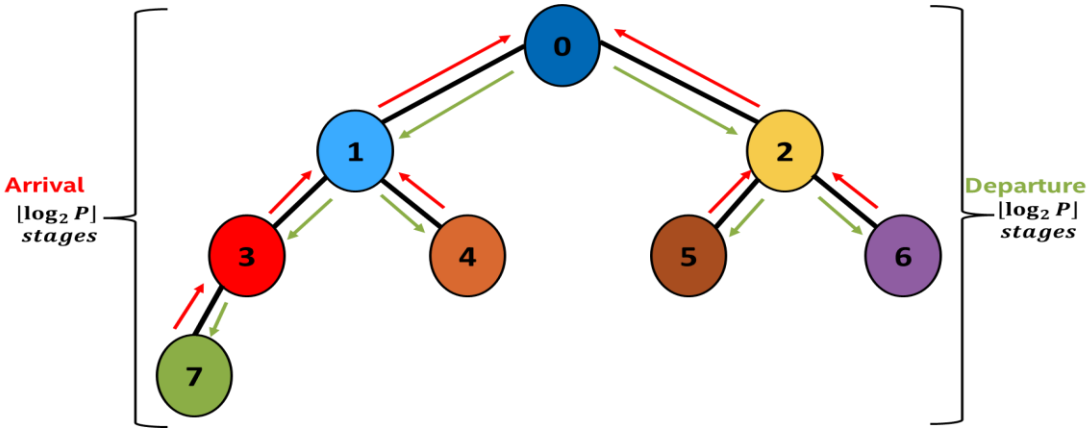
$$T_{cost\_ring\_allgather} = [\alpha(P-1)] + \left[\left(\frac{P-1}{P}\right)N\beta\right]$$

$$T_{cost\_ring\_allreduce} = [2\alpha(P-1)] + \left[2[\left(\frac{P-1}{P}\right)]N\beta\right] + \left[[\left(\frac{P-1}{P}\right)]N\gamma\right]$$

Barrier: Binary Tree

Binary Tree Barrier:

- Default implementation in SST (Firefly)
- Algorithm:
  - Like Tree AllReduce – but without
    - Data transfer and reduction costs
  - Two Phases: Arrival & Departure
  - Arrival
    - All processes inform the parent they reached barrier (until root node)
  - Departure
    - Each process signals children to exit the barrier state
- Cost model – only involves latency cost
  - Assuming P processes, N message size
  - Latency per message – α

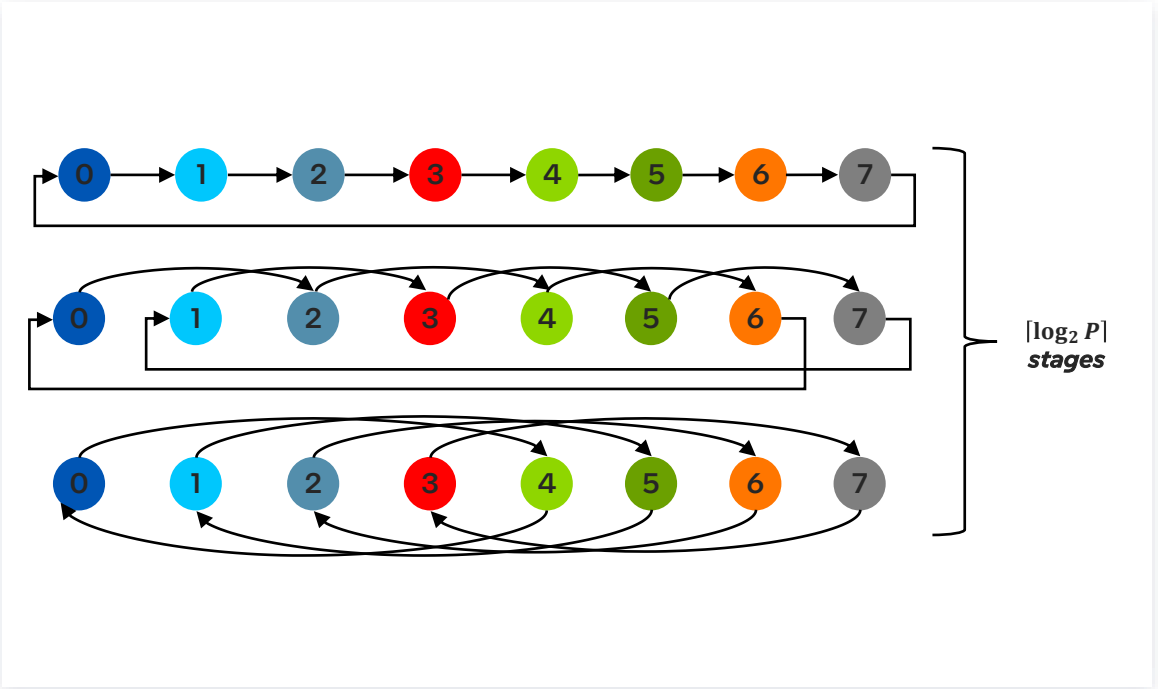


$$T_{cost\_tree\_barrier} = 2\alpha\lceil \log_2 P \rceil$$

Barrier: Dissemination

Dissemination Barrier:

- Algorithm:
  - Decentralized approach for process synchronization
  - Takes $\log_2 P$ steps
  - At the end of step i
    - Guaranteed that atleast $(1+2^i)$ processes reach barrier
- Cost model – only involves latency cost
  - Assuming P processes, N message size
  - Latency per message – α



$$T_{cost\_dissemination\_barrier} = \alpha \lceil \log_2 P \rceil$$

Experimental Setup

Network configuration

Topology	Fat-tree, 2D HyperX, 3D HyperX, 2D Torus, 3D Torus, Dragonfly
Endpoints	1024, 4096, 16K, 64K, 256K
Link Bandwidth	50 GB/s
Link Latency	20 ns

Router configuration

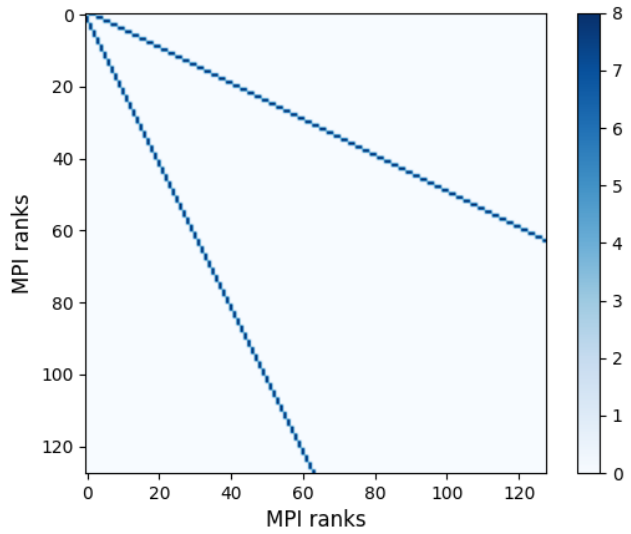
Crossbar BW	50 GB/s
Input latency	30ns
Output latency	30ns
Input buffer size	64 kB
Output buffer size	32 kB

Communication Pattern

AllReduce	Binary Tree, Rabenseifner, Ring
Message Size	8kB, 32kB, 128kB, 512kB, 2MB, 8MB, 16MB, 64MB, 256MB
Barrier	Binary Tree, Dissemination

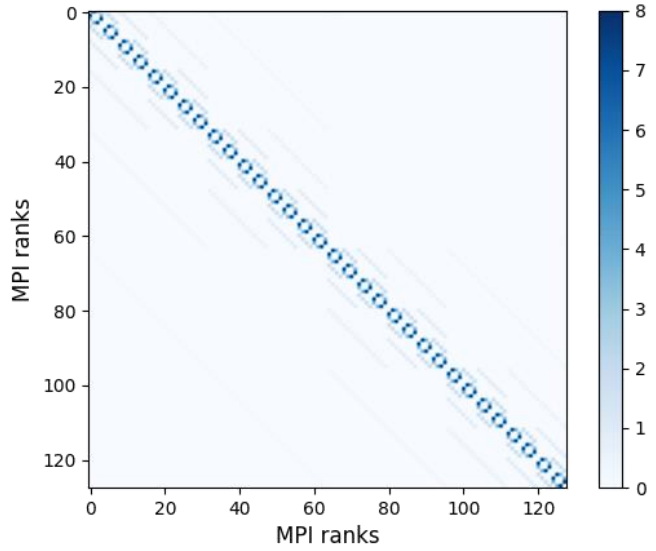
Simulation Results: Communication Matrix (AllReduce: P – 128 ; N – 8MB)

Binary Tree



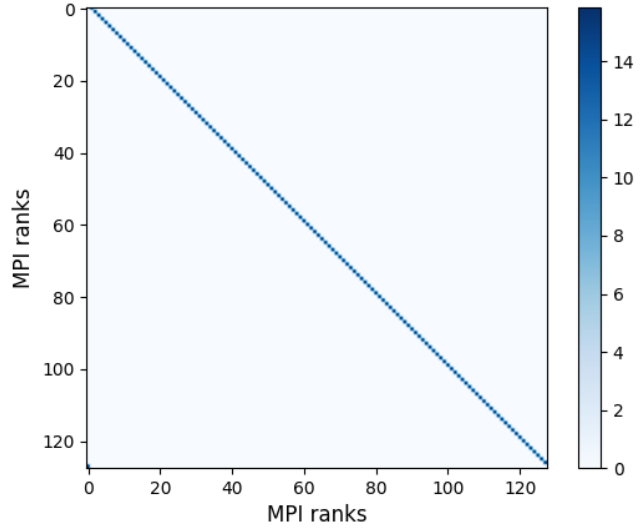
- Naïve approach – no rank proximity
- Sequential pattern – longer wait times
  - Not much overlap b/w comm. pairs
- Pairwise comm. involves full message size

Rabenseifner



- Better decomposition
- Pairwise comm involves reduced msg sizes at each comm stage
- Better overlap of comm. pairs

Ring

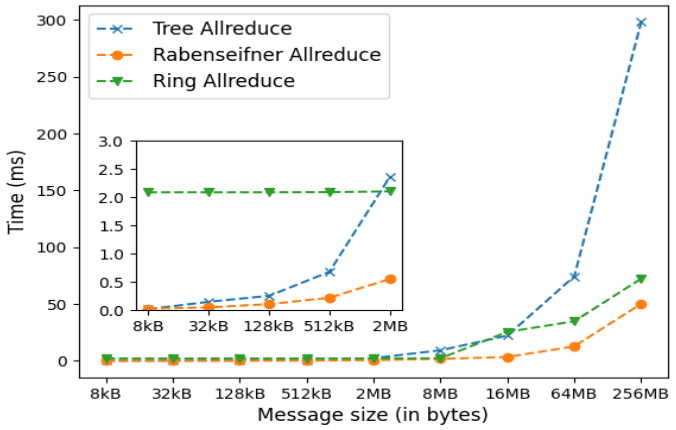


- Optimal rank proximity
- Fixed send/recv pairs

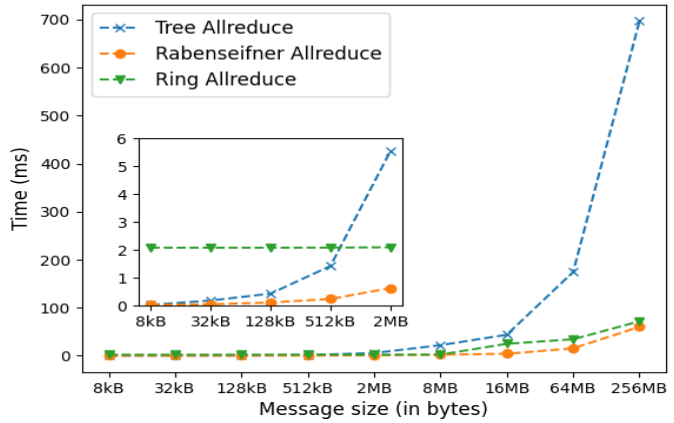
Simulation Results: Communication time (AllReduce: P – 1024)

- Ring AllReduce has higher latency for smaller messages
  - More steps – 2(P-1) vs $2\log_2 P$
- Rabenseifner and Ring optimal for large message sizes
  - Better rank proximity over binary tree
- Ring AllReduce is optimal for non-power of 2 processes
  - Comm volume per rank
    - $2\left(\frac{P-1}{P}\right)N$ vs $2\left(\left(\frac{P-1}{P}\right)+1\right)N$

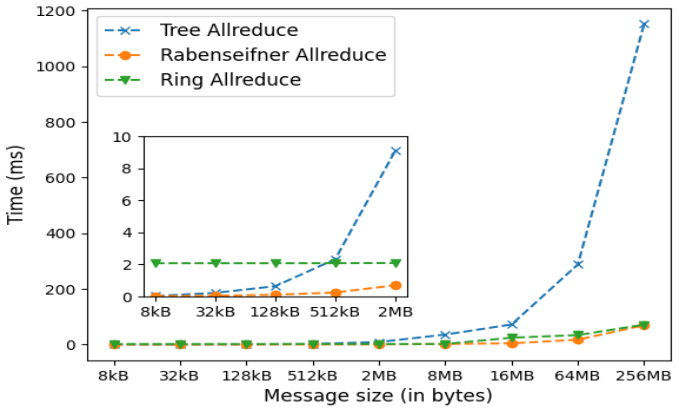
Three-level Fat-tree



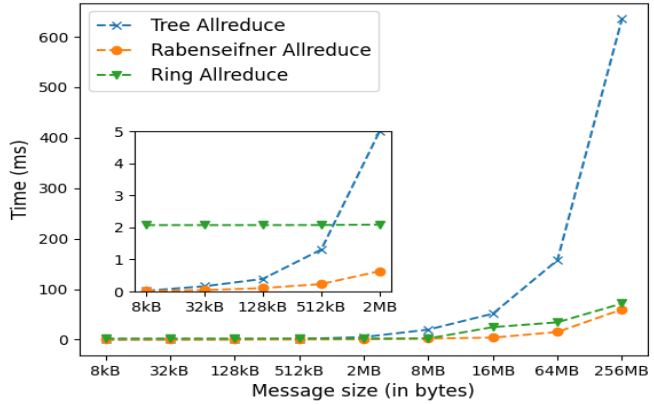
2D HyperX



2D Torus

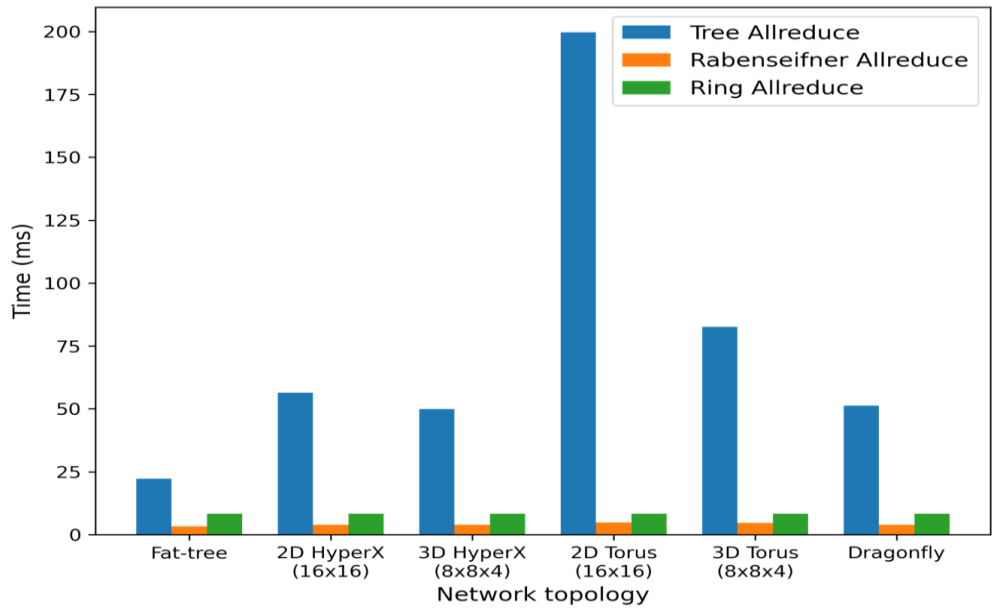


Dragonfly

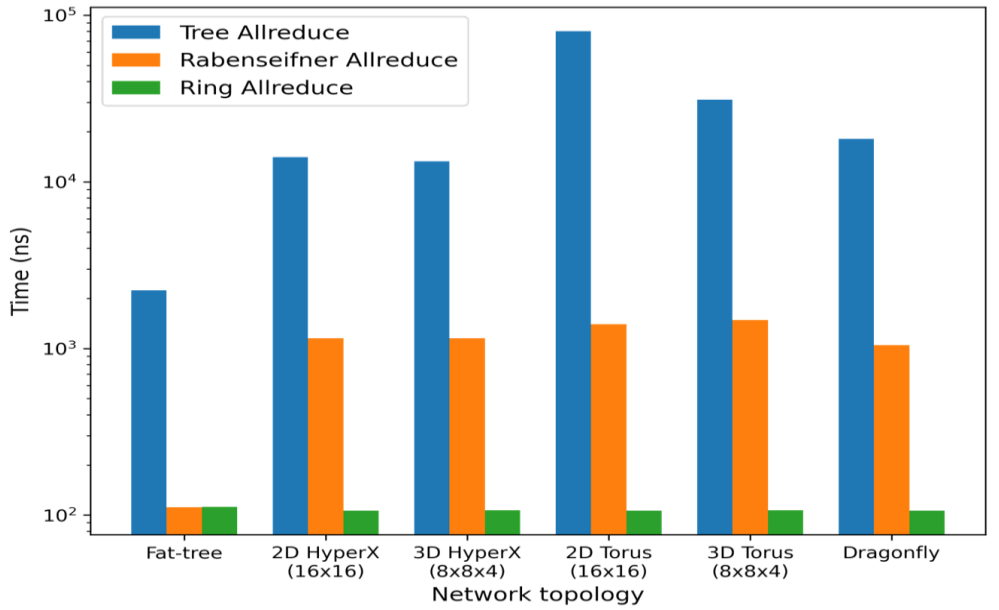


Simulation Results: Impact of N/W (AllReduce: P – 4096 ; N – 16MB)

Communication time



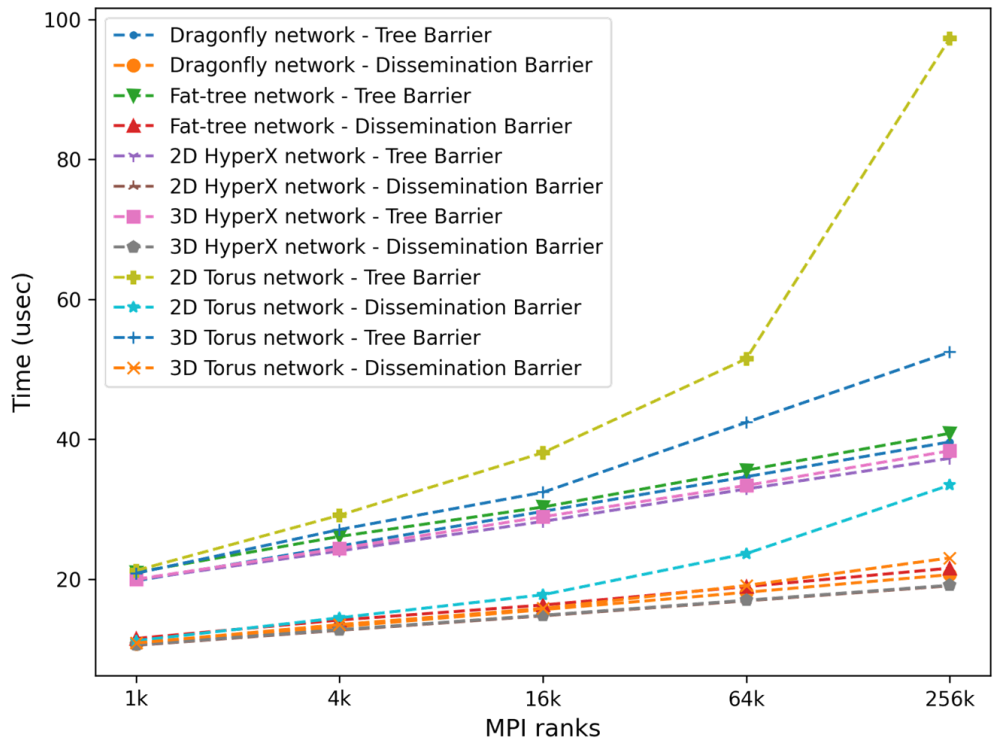
Average packet latency



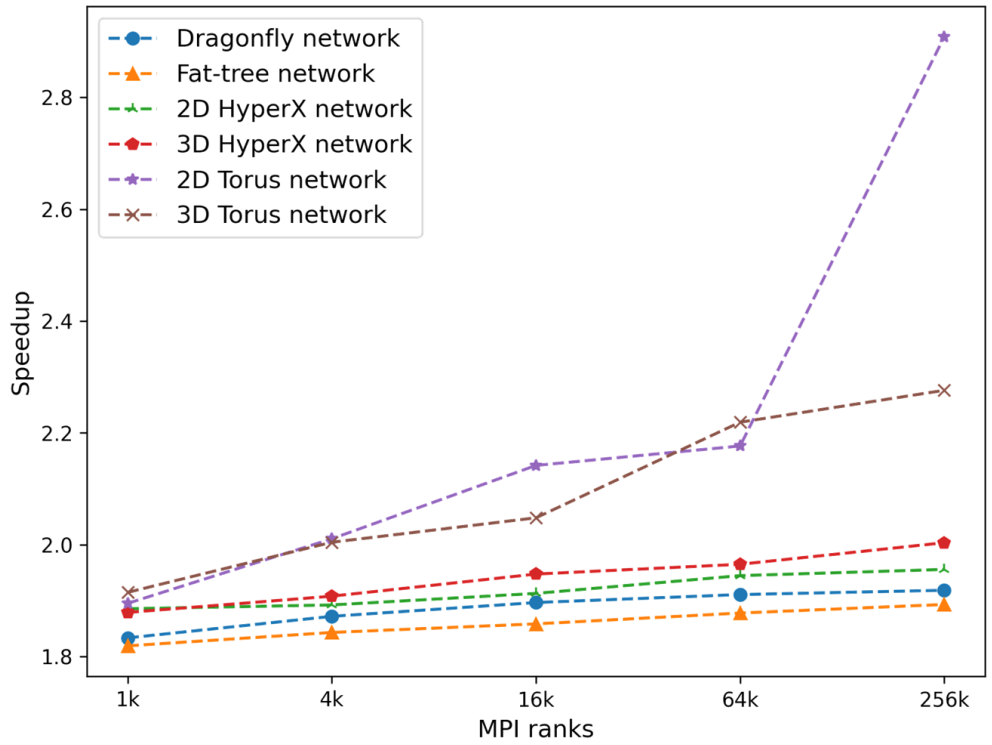
- Binary Tree AllReduce is sensitive to network topology
  - High diameter N/Ws (2D/3D Torus) resulted in longer comm. times
- Rabenseifner and Ring had minimal impact w.r.t network topology

Simulation Results: Barrier Scalability

Communication time



Dissemination Barrier – Speedup



- Dissemination barrier outperforms Tree barrier
  - 2x fewer steps – $\lceil \log_2 P \rceil$ vs $2\lceil \log_2 P \rceil$

Summary & Ongoing Work

- Collectives – dominant communication pattern in large-scale HPC & AI workloads
- Various algorithms have been proposed to optimize collective performance
  - Modeling and simulation is crucial for identifying optimal candidates
- Our approach: Leverage and extend scalable system-level simulator (SST) for collective modeling
- We built models in SST for AllReduce (Rabenseifner, Ring) and Barrier (Dissemination) collectives
  - Evaluated collective performance on different network topologies
- Ongoing work include
  - Upstreaming these models into SST repo – pending legal approvals
  - Build additional collective models – Double Binary Tree, Topology-aware collectives (e.g. Hierarchical AllReduce)
  - Validation w/ Benchmarks and analytical models

Notices & Disclaimers

Performance varies by use, configuration and other factors. Learn more at intel.com/performanceindex.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

Your costs and results may vary.

Intel technologies may require enabled hardware, software or service activation.

Configurations:

Structural Simulation Toolkit version: 13.0.0; modified to include support for additional collectives. The modified simulator code is not yet publicly available.


intel