

**APEIE 25**

Actual Problems  
of Electronic Instrument  
Engineering



# Landscapes and Challenges in Communication Middleware for HPC and AI Systems

*Mikhail Kurnosov*

*Siberian State University of Telecommunications  
and Information Sciences*

*Head of Parallel Computing Technologies Center,  
Doctor of Sciences, Professor*



# HPC and AI Systems

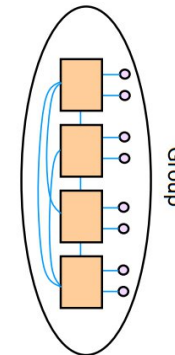
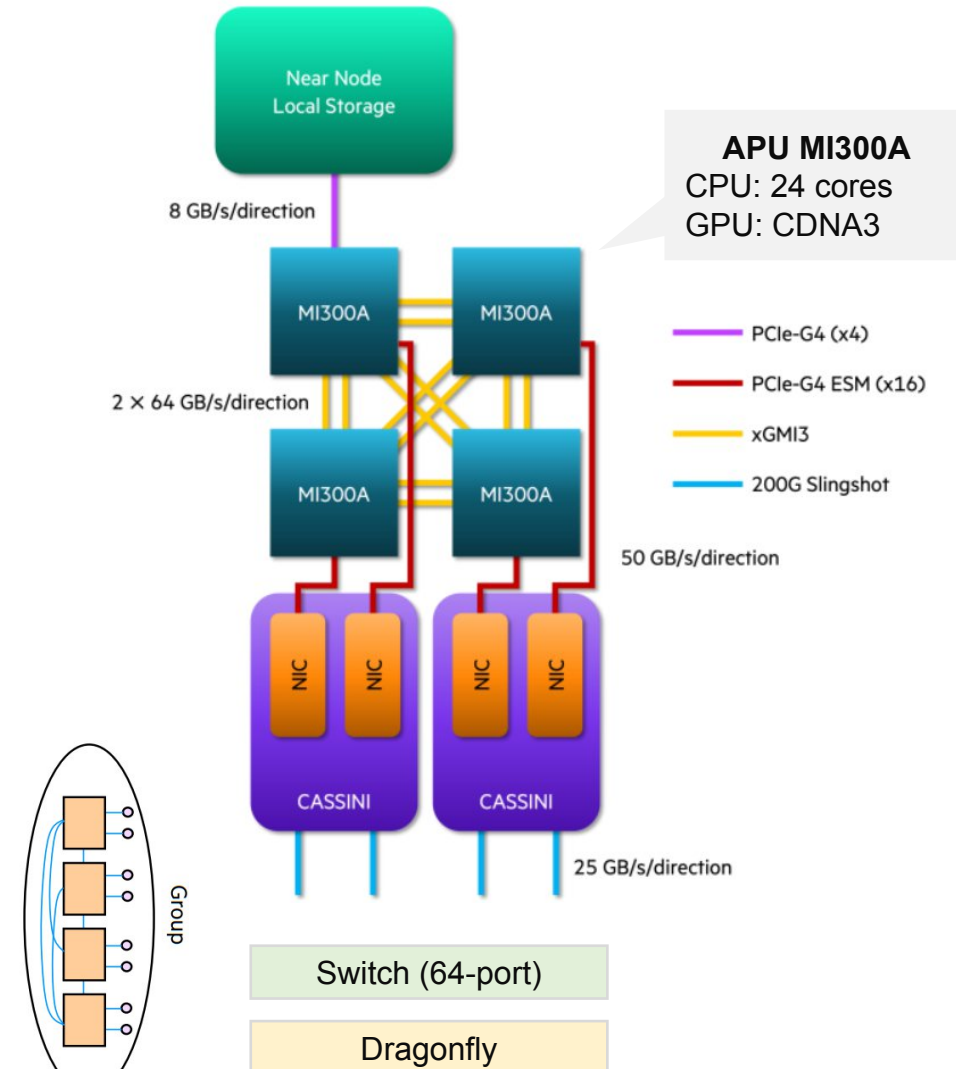
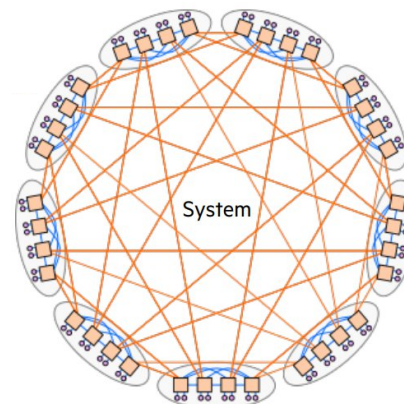


## Landscape

### El Capitan (Top500 #1, HPE Cray)

- **Compute nodes:** 11 136
  - APU (CPU + GPU): 4 AMD MI300A per node
  - Memory: 512 GiB HBM3 (4 NUMA nodes)
  - 4 NIC (one per APU)
- **Interconnect:** Slingshot-11
  - Level 1: 64-port switch
  - Level 2: group of fully connected L1 switches (1D Dragonfly)
  - Level 3: all groups are all-to-all connected
  - Maximum hop count: 3

- **Large-scale:** cores  $10^6$ , nodes  $10^5$
- **Heterogeneity**
  - CU: CPU + GPU/TPU/NPU
  - Memory: DDR, HBM (SDRAM), GDDR (SGRAM)
  - Data paths: multi-level Infiniband/Ethernet, PCIe, NVLink, CXL, UPI, Infinity Fabric



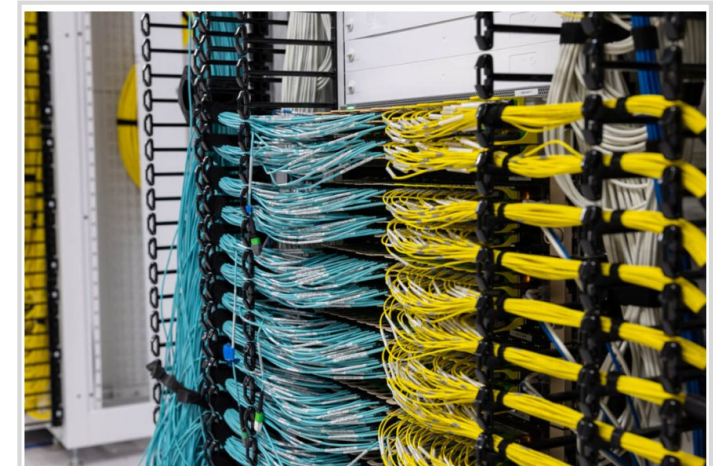
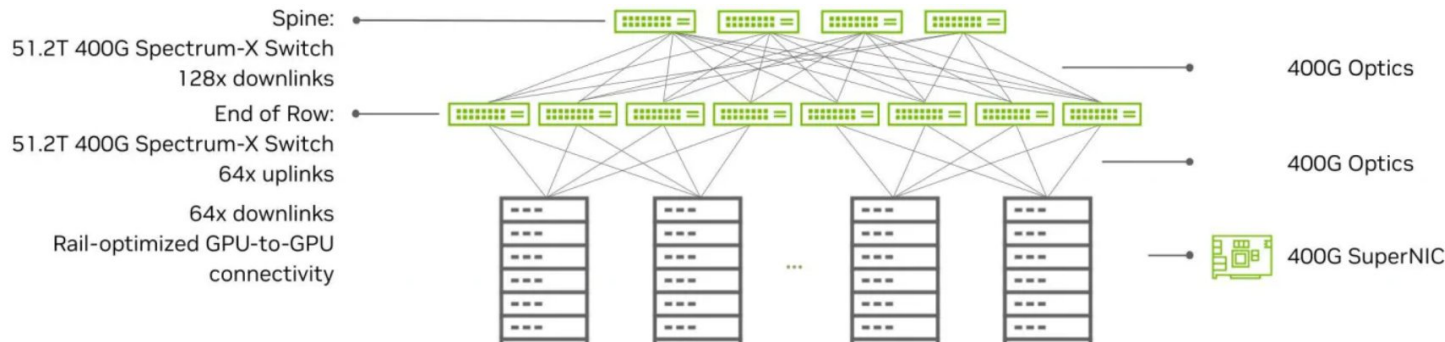
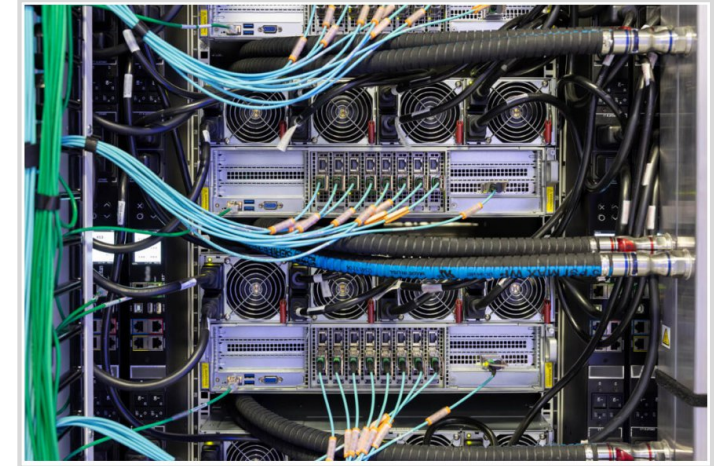
# HPC and AI Systems



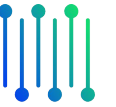
## Landscape

### Colossus (#1 AI supercomputer, xAI)

- **Training node (4U):** 8 NVIDIA H100/H200 + Intel Xeon CPUs
  - Rack: 8 nodes (64 GPU)
  - Array: 8 racks (512 GPU), > 200 arrays (> 100k GPUs)
- **Interconnect:** Ethernet RoCE
  - Spectrum-X Ethernet (RDMA), spine-leaf topology
  - Node NIC: 8 SuperNIC BlueField-3 (400GbE) per node

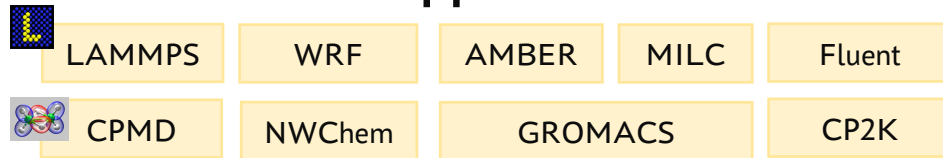


# Communication Middleware



## Landscape

### HPC Applications



### Point-to-point communication

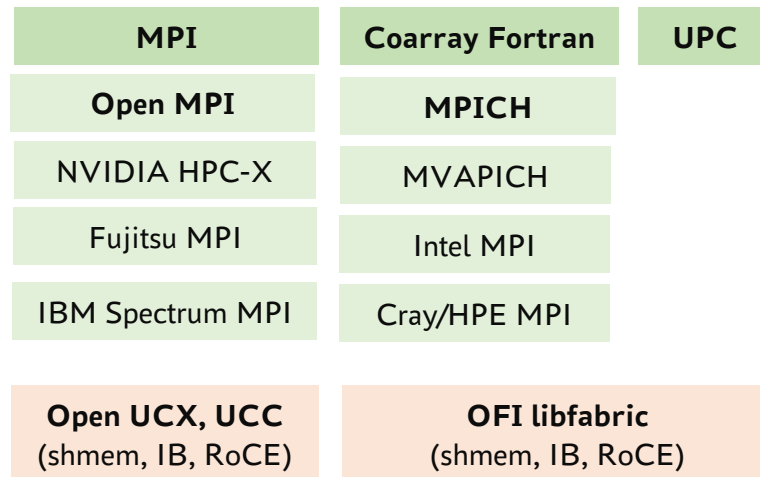
neighbor exchange (1D/2D/ 3D)

### Collective communication

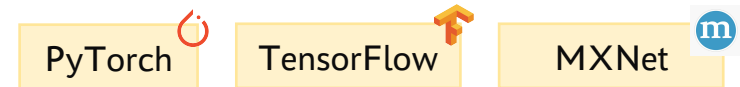
Allreduce/Reduce, All-to-all, All-to-one, One-to-all

### Communication Middleware

MPI 5.0  
June, 2025



### Distributed AI Training



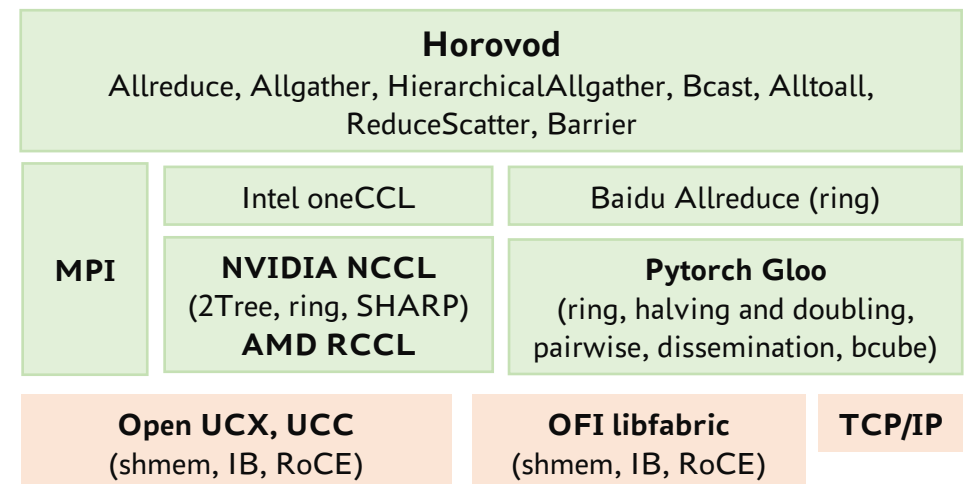
### Collective communication

Allreduce (gradient aggregation)

Broadcast (scatter parameters)

All-to-all, ReduceScatter

### Communication Middleware

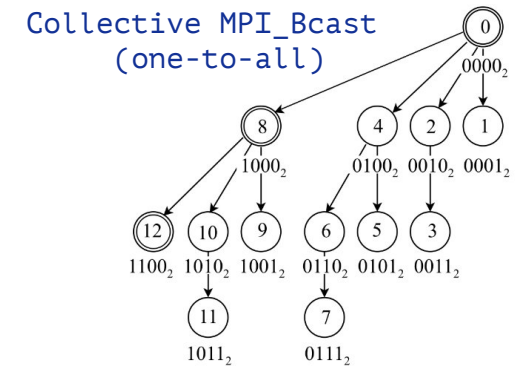
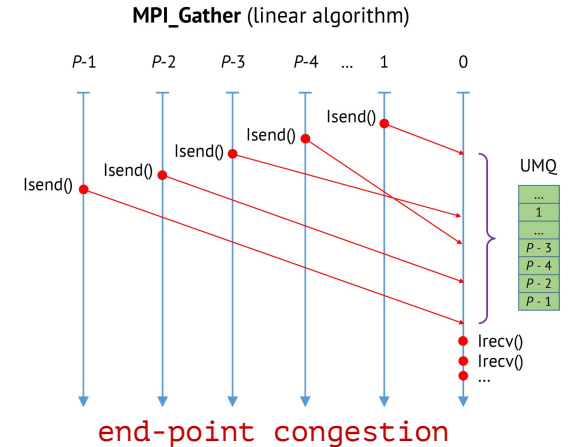
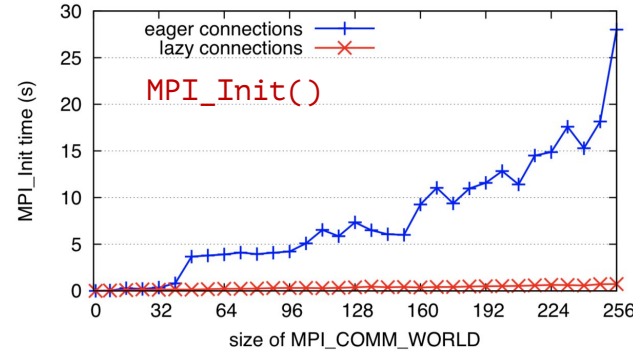


# Communication Middleware

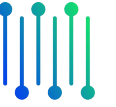


## Challenges

- **Scalable Parallel Job Startup**
    - Data structures (connection table):  $10^6$
  - **Point-to-Point and RMA operations**
    - Tag matching: lookup messages in receive queue
    - Congestion control
    - Comp. and comm. overlapping (offloading)
    - Heterogeneous memory (send/recv bufs): HBM, GDDR
  - **Resource-aware Process placement**
    - Min. distance to NUMA/HBM or GPU/NPU/TPU or NIC
    - Topology-aware process placement
  - **Collective Communication Algorithms** (Bcast/Scatter, Allreduce/Reduce, Allgather, Alltoall)
    - Nonblocking collectives (offload to NIC/SuperNIC/DPU)
    - Sparse/Neighborhood collectives (communication on process topologies)
- I. Vardas. **Improved Parallel Application Performance and Makespan by Colocation and Topology-aware Process Mapping**, 2024
- Träff J.L. **Optimal, Non-pipelined Reduce-scatter and Allreduce Algorithms with an Application to All-to-all Communication**, 2025
- J. S. García et al. **Offloaded MPI message matching: an optimistic approach**, SC-2024
- Y. Dai et al. **MIST: Towards MPI Instant Startup and Termination on Tianhe HPC Systems**, 2025
- S. Chakraborty, H. Subramoni, J. Perkins, A. Moody, M. Arnold, and D. K. Panda. **PMI Extensions for Scalable MPI Startup**, 2014



# MPI Collective Communication



## 1) Flat point-to-point based (send/recv)

Only number of processes and message size are known

- ❑ **One-to-all** (Broadcast, Scatter)
  - Binary tree, Binomial tree,  $k$ -nomial tree  $O(\log(p))$
  - Flat tree  $O(p)$ ,  $k$ -chain  $O(p)$ , pipeline
  - Scatter binomial tree + allgather recursive doubling
- ❑ **All-to-one** (Gather, Reduce, Scan)
  - Binomial tree,  $k$ -nomial tree, binary tree,
  - $k$ -chain, pipeline, linear
  - Rabesifner's algorithm
- ❑ **All-to-all** (Allgather, Alltoall, Allreduce, Reduce\_scatter, Reduce\_scatter\_block)
  - Bruck, recursive doubling, recursive halving algorithm, neighbor exchange
  - Linear, ring, gather + scatter
  - Rabenseifner's algorithm, Butterfly

## 2) Topology-aware algorithms

Topology/hierarchy of network is known (routes, distances, process placement)

- Algorithms for Torus/Dragonfly networks
- Algorithms for Fat-tree/spine-leaf topology
- Hierarchical collectives (NUMA-aware): network + shared memory (intra-node)

## 3) Hardware-accelerated algorithms

- NVIDIA/Mellanox SHARP (Allreduce, Reduce, Barrier, Bcast)
- HPE Cray Slingshot-11 (hw collectives)
- IBM BG tree network (Bcast)

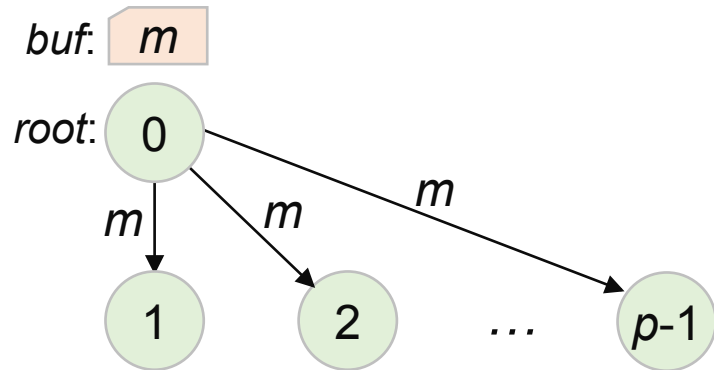
MPI 5.0 defines 17 collective operations  
> 50 algorithms (MPICH/MVAPICH, Open MPI)



# Flat Algorithms for Broadcast

Using Send/Recv operations only

Communication tree



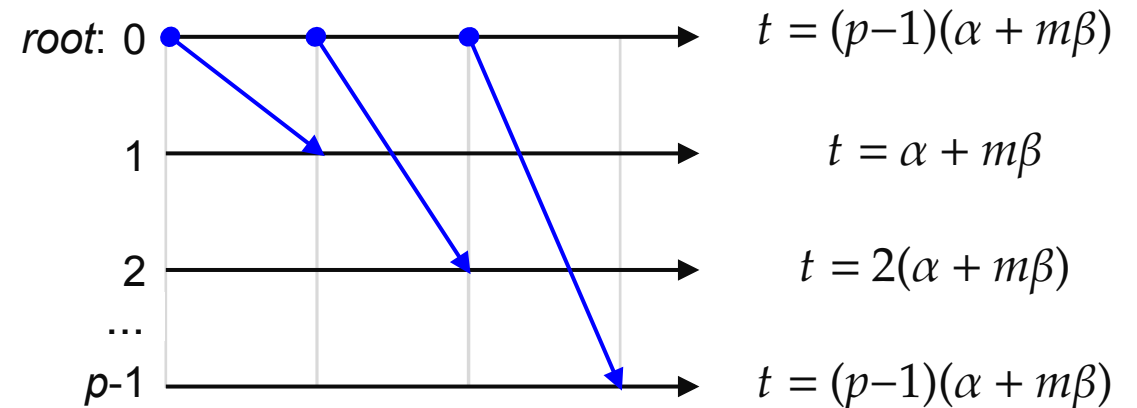
$$t_{\text{Linear}} = (p-1)(\alpha + m\beta) = O(p)$$

**Cost model (R. Hockney):**

- $p$  – number of processes,  $m$  – message size (bytes)
- $\alpha$  – latency per message,  $\beta$  – transfer time per byte

$$t_{\text{Send}} = \alpha + m\beta$$

Time diagram

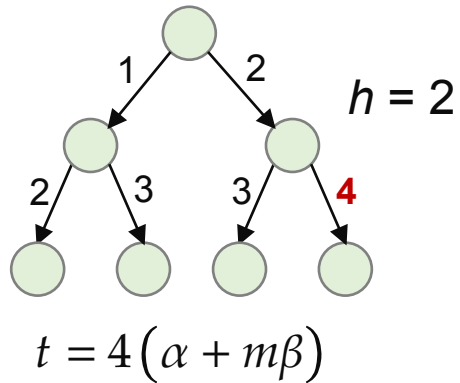


- ⊕ Easy to implement, widely used in MPI, ML frameworks
- ⊖ Ignore network topology, **poor scalability  $O(p)$**

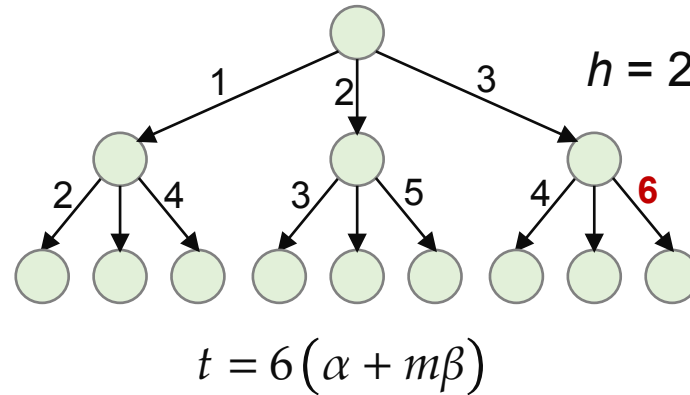
# Broadcast: $k$ -ary Tree



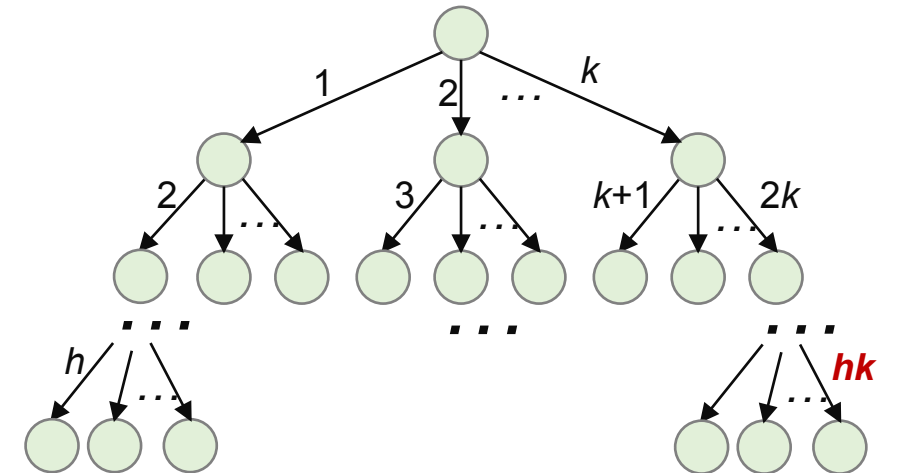
Binary tree ( $k=2, p=7$ )



Ternary tree ( $k=3, p=13$ )



$k$ -ary tree



$$t_{\text{Binary}} = 2 \lfloor \log_2 p \rfloor (\alpha + m\beta)$$

$$t_{\text{Ternary}} = 3 \lfloor \log_3 p \rfloor (\alpha + m\beta)$$

$$t_{\text{Kary}} = kh(\alpha + m\beta) = k \lfloor \log_k p \rfloor (\alpha + m\beta)$$

- Which tree is better?
- What is the optimal  $k$  in latency-bandwidth model?



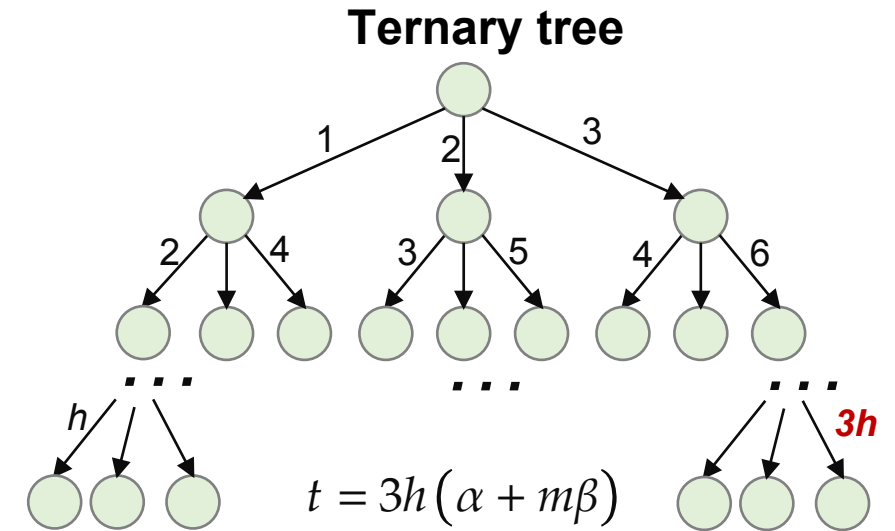
# Broadcast: $k$ -ary Tree

- Let us find the optimal  $k$  (node degree)

$$t_{\text{Kary}}(k) = k \log_k p (\alpha + m\beta)$$

$$\frac{dt_{\text{Kary}}(k)}{dk} = \frac{\ln p \ln k - \ln p}{\ln^2 k} = 0$$

$$k_{\text{opt}} = e \approx 2.7183$$



- An optimal  $k$ -ary tree in latency-bandwidth model is a ternary tree

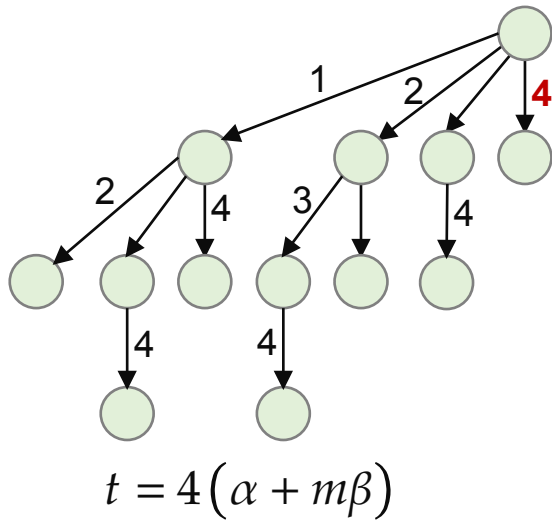
- Discussion

- Ternary tree is optimal in latency-bandwidth model only!
- Drawback of ternary-tree – each **root is idle** after sending three messages
- We need a new tree – all roots must keep sending

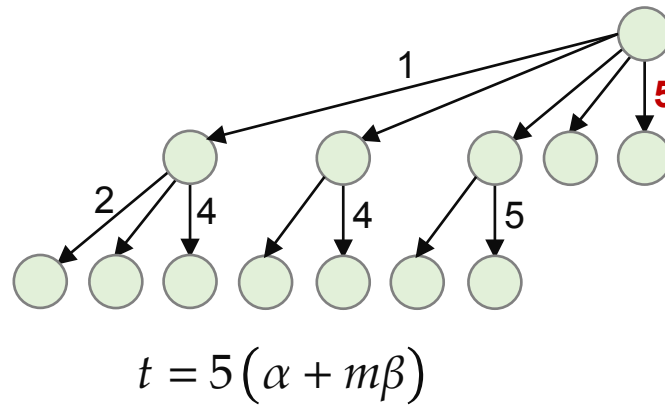
# Broadcast: $k$ -nomial Tree



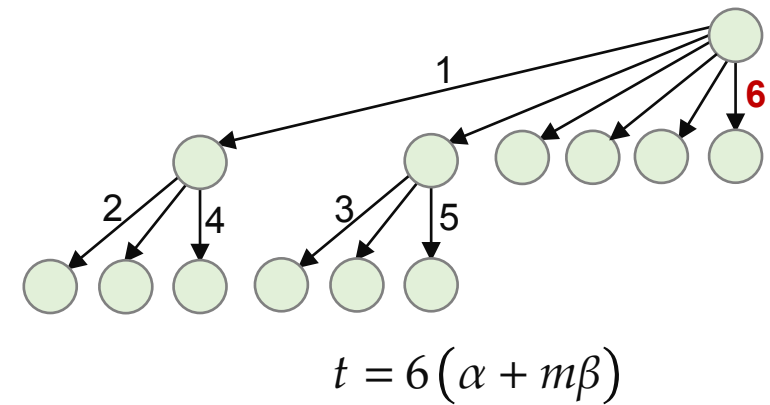
Binomial tree ( $k=2, p=13$ )



3-nomial tree ( $p=13$ )



4-nomial tree ( $p=13$ )



$$t_{\text{Binomial}} = \lceil \log_2 p \rceil (\alpha + m\beta)$$

$$t_{\text{3Nomial}} = \lceil \log_2 p \rceil (\alpha + m\beta)$$

$$t_{\text{KNomial}} = (k-1) \lceil \log_k p \rceil (\alpha + m\beta)$$

$$t_{\text{KNomial}} = (k-1) \lceil \log_k p \rceil (\alpha + m\beta) = O(\log p)$$

- What is the optimal  $k$  in latency-bandwidth model?

# Broadcast: $k$ -nomial Tree



- Let us find the optimal  $k > 1$

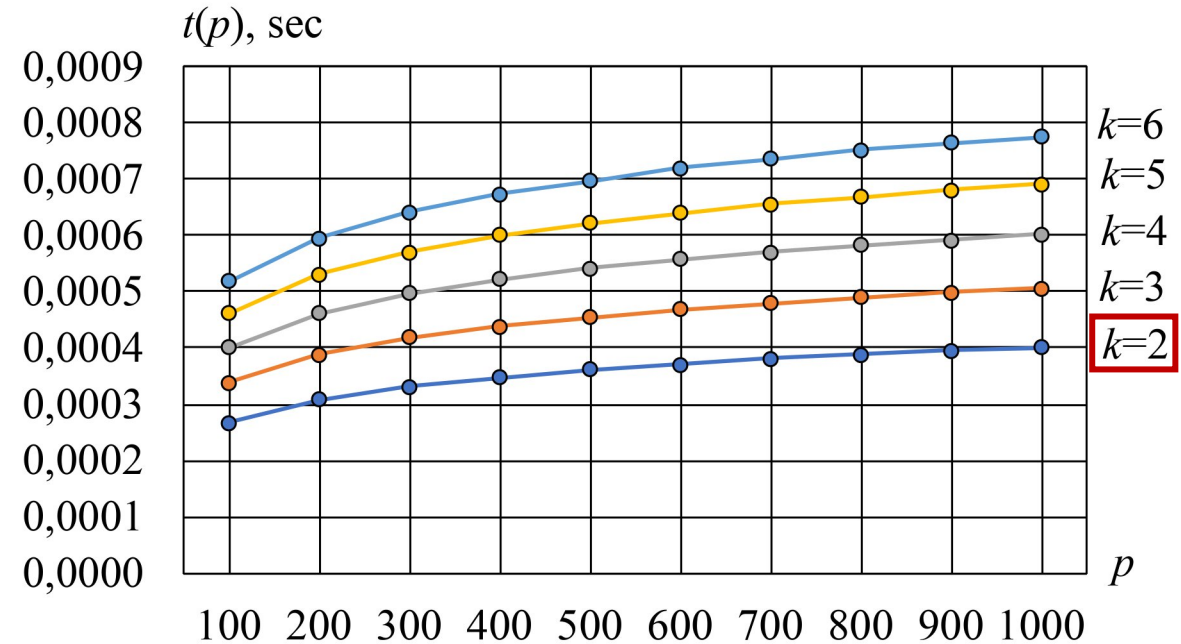
$$t_{\text{KNomial}} = (k-1) \lceil \log_k p \rceil (\alpha + m\beta)$$

- Time is monotonically increasing with  $k$

$$k_{\text{opt}} = 2$$

- Binomial tree is widely used for Bcast, Reduce, Gather, Scatter (Open MPI, MPICH, MVAPICH, Intel MPI)

- Can we broadcast faster than in  $k$ -nomial tree?



**$k$ -nomial tree broadcast time**

( $\alpha = 40 \mu\text{sec}$ ,  $\beta = 40 \text{ Gbps}$ ,  $m = 1024 \text{ byte}$ )



# Broadcast: lower bound

- Simple lower bound on the broadcast time

$$t_{\text{Bcast}} \geq \min \{ \lceil \log_2 p \rceil \alpha, m\beta \}$$

$$t_{\text{Binomial}} = \lceil \log_2 p \rceil \alpha + \lceil \log_2 p \rceil m\beta$$

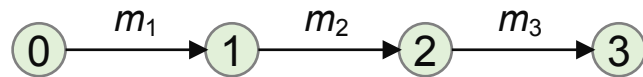
- $k$ -nomial tree is asymptotically optimal for small messages ( $m = 1$ ):  $O(\log p)$
- Binomial tree is a factor of  $\lceil \log_k p \rceil$  slower in bandwidth
- Can we broadcast faster for large  $m$ ?

- Sanders Peter, Speck Jochen, Träff Jesper Larsson. **Two-tree algorithms for full bandwidth broadcast, reduction and scan** // Parallel Computing. – 2009. – Vol. 35 (12). – pp. 581–594.
- Hoefler T., Moor D. **Energy, Memory, and Runtime Tradeoffs for Implementing Collective Communication Operations** // Journal of Supercomputing Frontiers and Innovations. – 2014. – Vol 1 (2). – pp. 58-75
- Thakur, R. Rabenseifner, W. Gropp. **Optimization of collective communication operations in MPICH** // Int. Journal of High Performance Computing Applications. – 2005. – Vol. 19 (1). – P. 49-66.



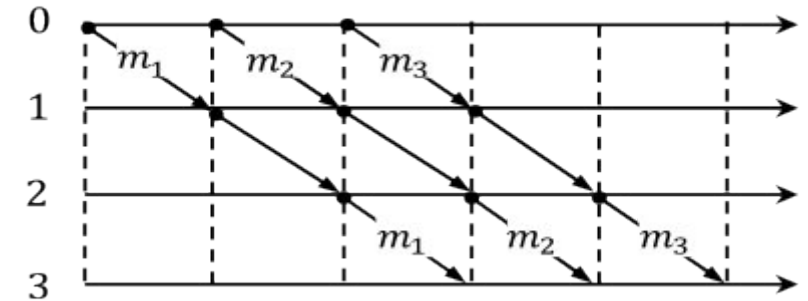
# Broadcast: Pipeline with Segmentation

- Message  $m$  is splitted into segments of size  $s$  bytes:  $m = (m_1, m_2, \dots, m_{m/s})$
- Process  $i$  sends current segment to process  $i + 1$



$$t_{\text{Pipeline}} = \left(p - 2 + \frac{m}{s}\right) (\alpha + s\beta) = O(p)$$

Pipeline is a factor of  $p / \log_2 p$  slower in latency (related to  $t_{\text{Bcast}}$ )



Example:  $p = 4$ , 3 segments,  
(full duplex channels)

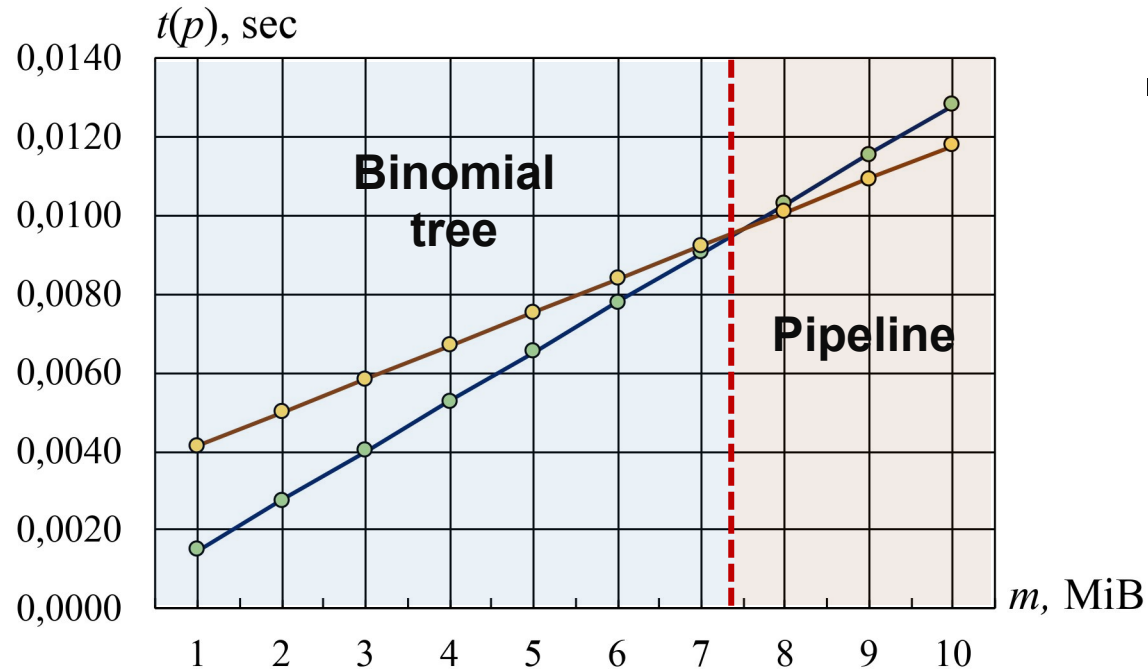
- Optimal segment size  $s$  in latency-bandwidth model

$$\frac{dt(s)}{ds} = -\frac{m}{s^2}\alpha + (p-2)\beta = 0$$

$$s_{\text{opt}} = \sqrt{\frac{m\alpha}{(p-2)\beta}}$$



# Broadcast: Binomial tree vs. Pipeline



**Segment size:**  $s = 4096$  bytes,  
 $\alpha = 40 \mu\text{sec}$ ,  $\beta = 40 \text{ Gbps}$ ,  $p = 64$

- **Open MPI, MPICH, MVAPICH:**

- Binomial tree – small sized messages
- Pipeline – large messages and small number of processes

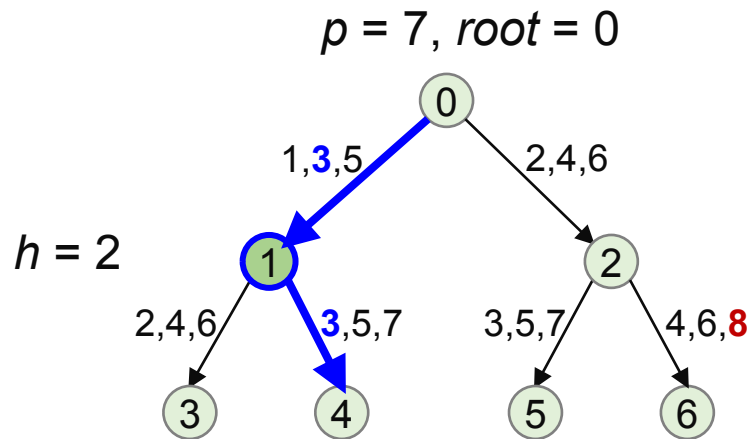
- **Faster algorithm for intermediate message sizes?**

Pipeline is a factor of  $p/\log_2 p$  slower in latency, related to  $t_{\text{Bcast}}$



# Broadcast: Pipelined Binary Tree

- Segmented message:  $m = (m_1, m_2, m_3)$ ,  $n = m/s$  – number of segments



	Time step							
	1	2	3	4	5	6	7	8
<b>Send-Recv operations</b>	0-1	0-2 1-3	0-1 1-4 2-5	0-2 2-6 1-3	0-1 1-4 2-5	0-2 2-6 1-3	1-4 2-5	2-6

- At step 3 process 1 recvs from 0 and sends to 4 simultaneously (full duplex model)

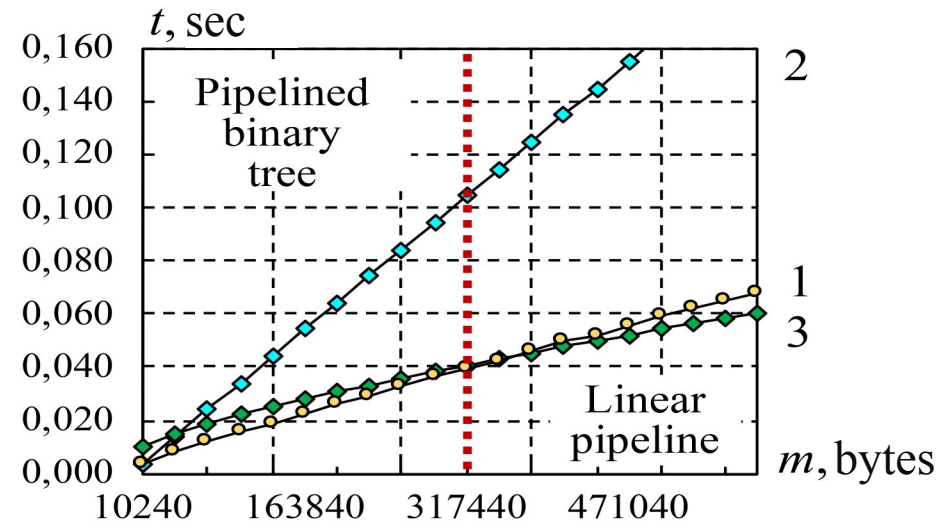
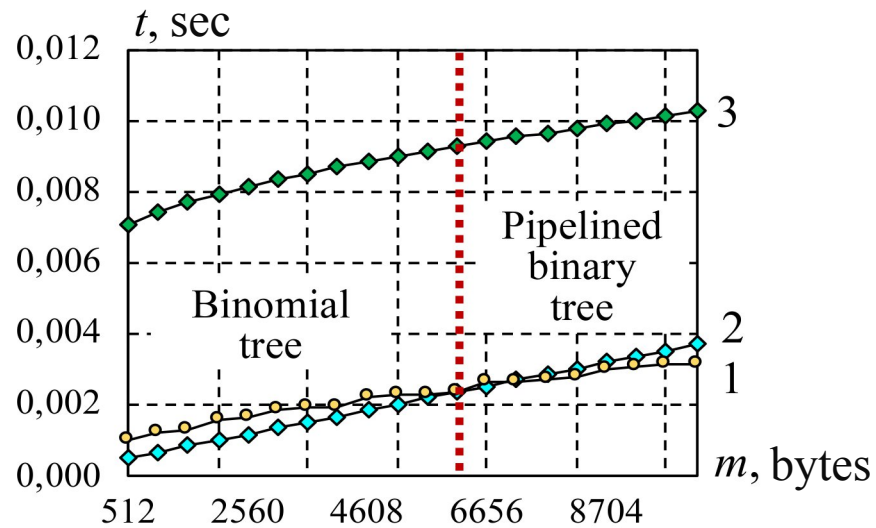
$$t_{\text{PipelinedTree}} = 2(h + n - 1)(\alpha + s\beta) = 2([\log_2 p] + \frac{m}{s} - 1)(\alpha + s\beta)$$

- Optimal segment size  $s$  in latency-bandwidth model**

$$\frac{dt(s)}{ds} = -\frac{2m}{s^2}\alpha + 2(\log_2 p - 1)\beta = 0 \quad s_{\text{opt}} = \sqrt{\frac{m\alpha}{(\log_2 p - 1)\beta}}$$



# Broadcast: Algorithm Selection



## Execution time of MPI\_Bcast (model):

1) pipelined binary tree (opt. segment); 2) binomial tree; 3) linear pipeline

( $\alpha = 0.00005$  s,  $\beta = 0.000000047$  s,  $p = 128$ ,  $root = 0$ ):

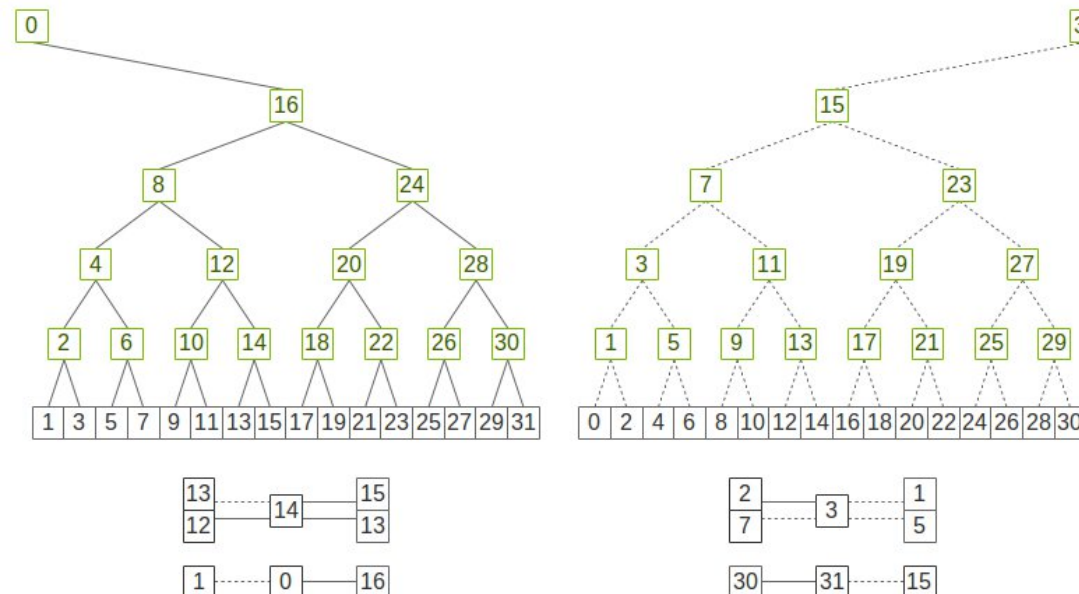
$$t_{\text{PipelinedTree}} = 2 \left( \lceil \log_2 p \rceil + \frac{m}{s} - 1 \right) (\alpha + s\beta)$$

- Small messages ( $m = 1$ ,  $s = 1$ ), large  $p$ :  $O(\log p)$
- Large messages,  $p$ ,  $\alpha$ ,  $\beta$  – constants:  $O(m\beta)$
- For large  $p$  and  $m$  bandwidth is off by a factor of 2



# Broadcast: Bandwidth-Optimal

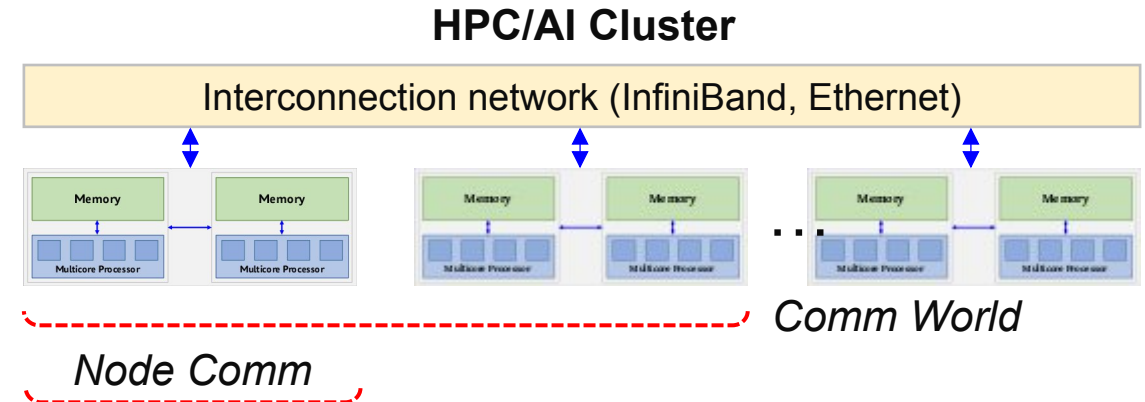
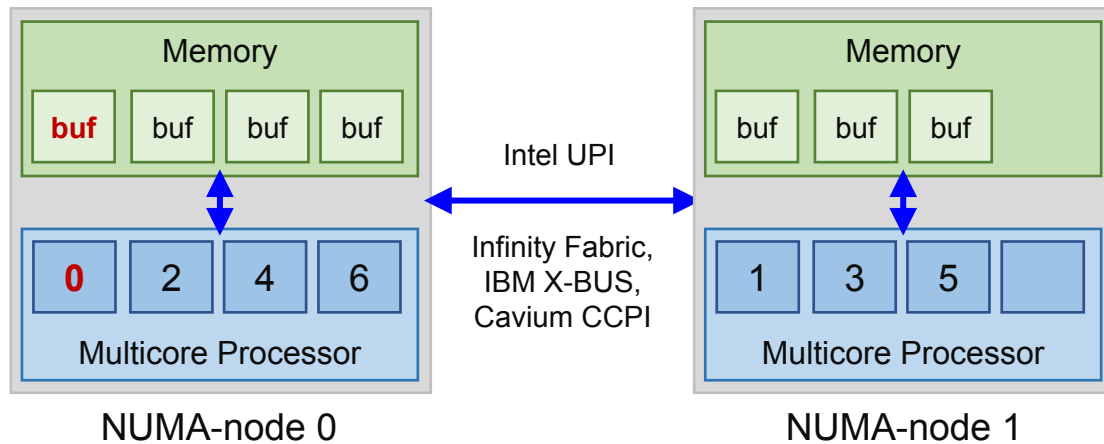
- Sanders P., Speck J. Träff J.L. **Two-tree Algorithms for Full bandwidth Broadcast, Reduction and Scan** // Parallel Computing. – 2009. – Vol. 35 (12). – pp. 581–594.
- **Key ideas:**
  - In binomial tree, all leaves only receive data and never send
  - Send along two simultaneous binary trees where the leaves of one tree are inner nodes of the other



- **NCCL:** <https://developer.nvidia.com/blog/massively-scale-deep-learning-training-nccl-2-4/>



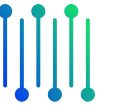
# Shared-Memory Collectives



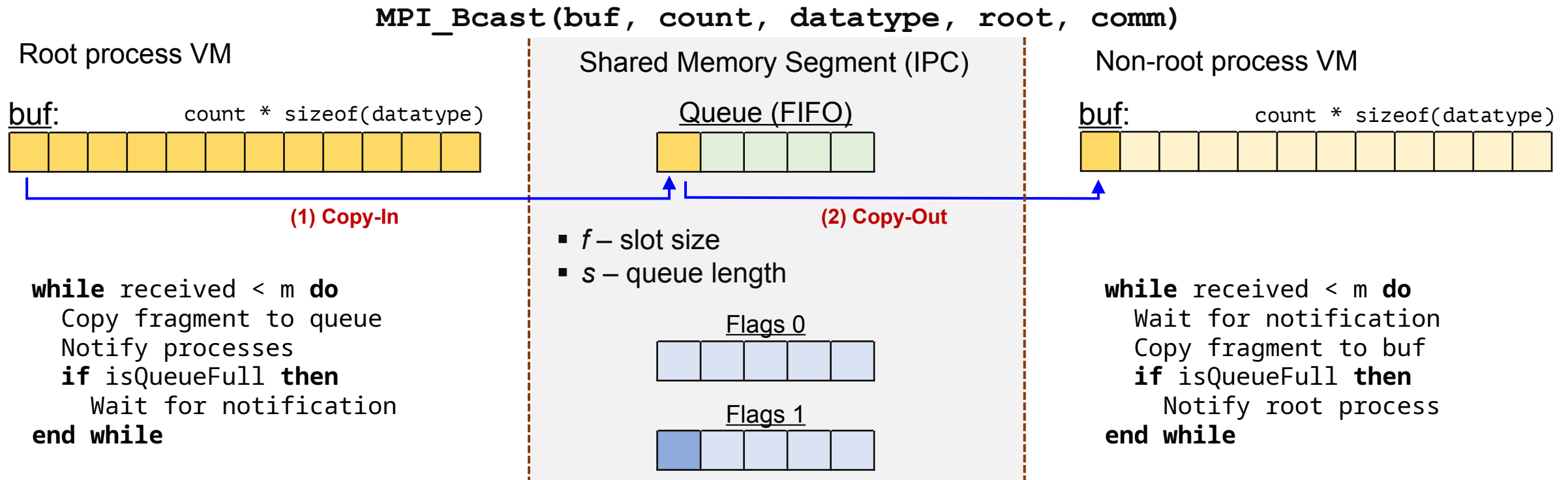
- **Key idea:** implement collective operation algorithm between processes on a same node using shared memory segments (mem. copy, load/store ops)
- **Challenges:** cache flushing, double copying

- Hooyoung Ahn. **MPI Allgather Utilizing CXL Shared Memory Pool in Multi-Node Computing Systems**, 2024
- Alexey Romanyuta, Mikhail Kurnosov. **Shared Memory-based Algorithms for MPI Irregular Collective Operations**, 2022.
- Jain S., Kaleem R., Balmana M. **Framework for Scalable Intra-Node Collective Operations using Shared Memory**, SC-2018
- Shigang Li, Torsten Hoefler. **NUMA-aware shared-memory collective communication for MPI**, 2018
- Graham R.L., Shipman G. **MPI Support for Multi-core Architectures: Optimized Shared Memory Collectives**, EuroMPI-2008

# Shared-Memory Collectives



- **Copy-In/Copy-Out algorithms** (Open MPI, MVAPICH, Intel MPI)
  - Create shared memory segment with a set of queues for each process
  - MPI\_Bcast: root copies message segments into queue slots and notifies other processes

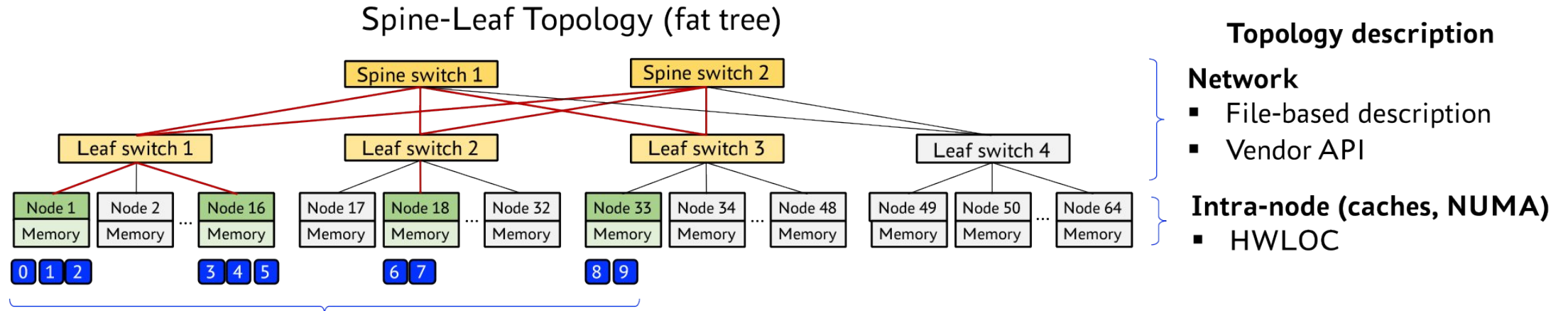


- **ZeroCopy Algorithms** (single-copy): kernel-assisted, KNEM (Linux), XPMEM (HP/Cray, Linux), CMA (Linux)

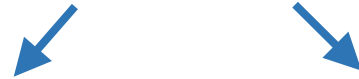
# Topology-aware Collective Algorithms



- **Input:** Network topology description + process placement => distance  $d_{ij}$  between processes  $i$  and  $j$



## Topology-aware algorithms (approaches)



### Hierarchical Collective Communication Algorithms

- Localizes communications on hierarchical levels (L3 cache, NUMA node, NUMA machine, leaf switch)
- A flat algorithm on each hierarchical level is used

### Dynamic optimization of communication graph for a give collective operation

- Builds collective communication graph for a given topology and process placement

# Hierarchical Broadcast Algorithm



## MPI\_Init()

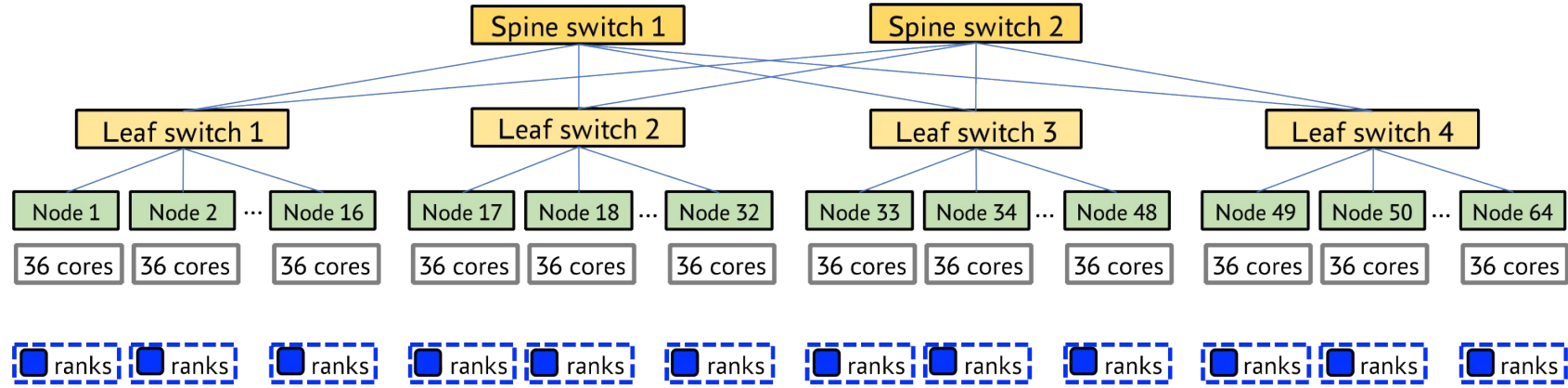
### Topology discovery and process grouping:

1. Discovery intra-node topology (HWLOC API) and network topo (config file)
2. Analyze process placement

### Topology description

```
# Topology config
# [nodename] [switch]
N6140-002 801SB
...
N6140-016 801SB
N6140-017 802SB
N6140-018 802SB
...
```

HWLOC + PMIx



# Hierarchical Broadcast Algorithm



## MPI\_Init()

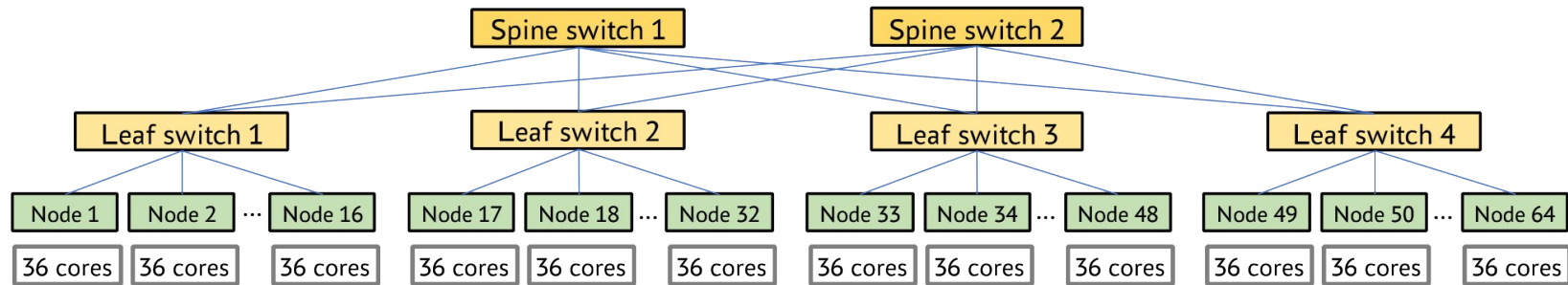
### Topology discovery and process grouping:

1. Discovery intra-node topology (HWLOC API) and network topo (config file)
2. Analyze process placement
3. **Group processes that share same communication device (L3 cache, NUMA node, HCA, switch)**

### Topology description

```
# Topology config
# [nodename] [switch]
N6140-002 801SB
...
N6140-016 801SB
N6140-017 802SB
N6140-018 802SB
...
```

HWLOC + PMIx



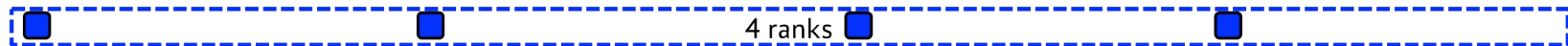
#### Intra-node level



#### Inter-node level



#### Inter-switch level



Share NUMA node

Share HCA (node)

Share switch

# Hierarchical Broadcast Algorithm



## MPI\_Init()

### Topology discovery and process grouping:

1. Discovery intra-node topology (HWLOC API) and network topo (config file)
2. Analyze process placement
3. Group processes that share same communication device (L3 cache, NUMA node, HCA, switch)

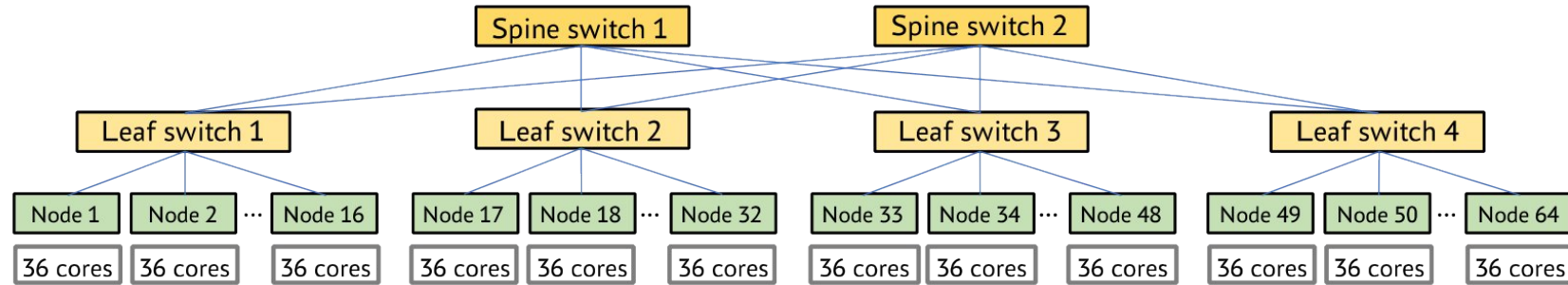
## MPI\_Bcast():

1. Inter-switch Bcast across leaders of nodes
2. Inter-node Bcast across leaders of NUMA nodes
3. Intra-node Bcast across NUMA node ranks

### Topology description

```
# Topology config
# [nodename] [switch]
N6140-002 801SB
...
N6140-016 801SB
N6140-017 802SB
N6140-018 802SB
...
```

HWLOC + PMIx



Share NUMA node



(3) Intra-node Bcast

Share HCA (node)



(2) Inter-node Bcast

Share switch

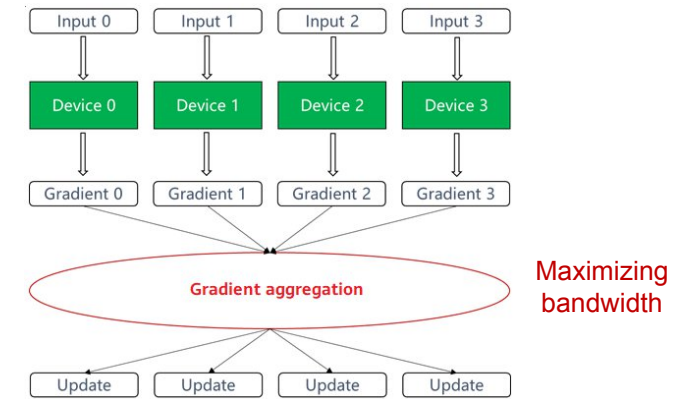
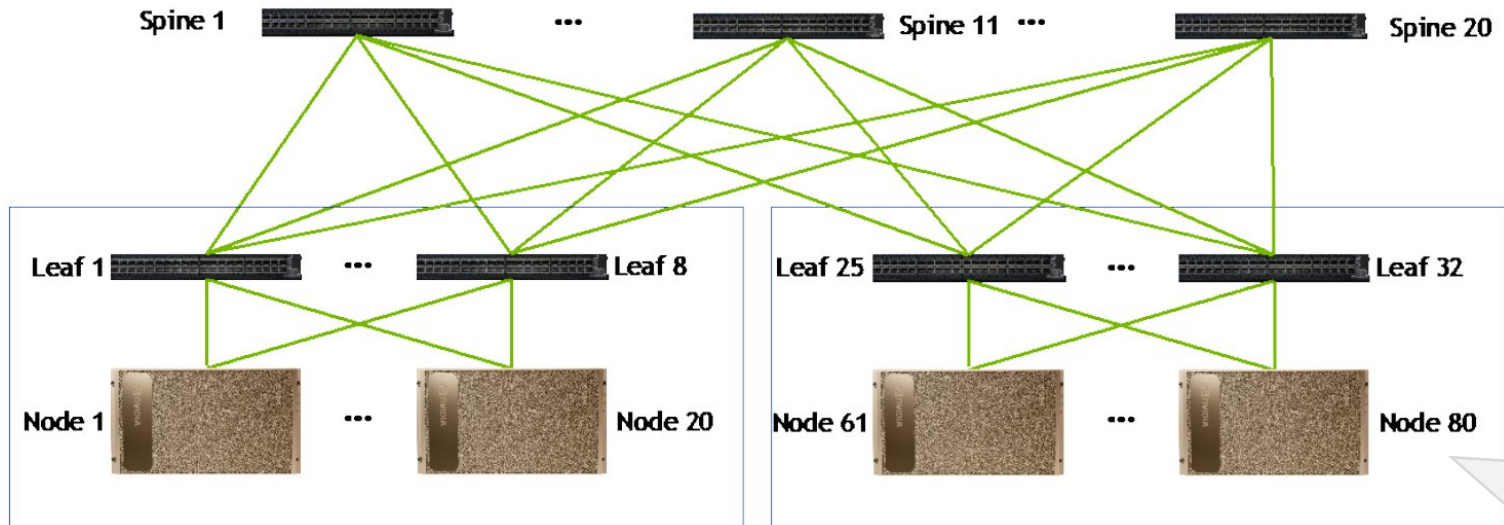


(1) Inter-switch Bcast

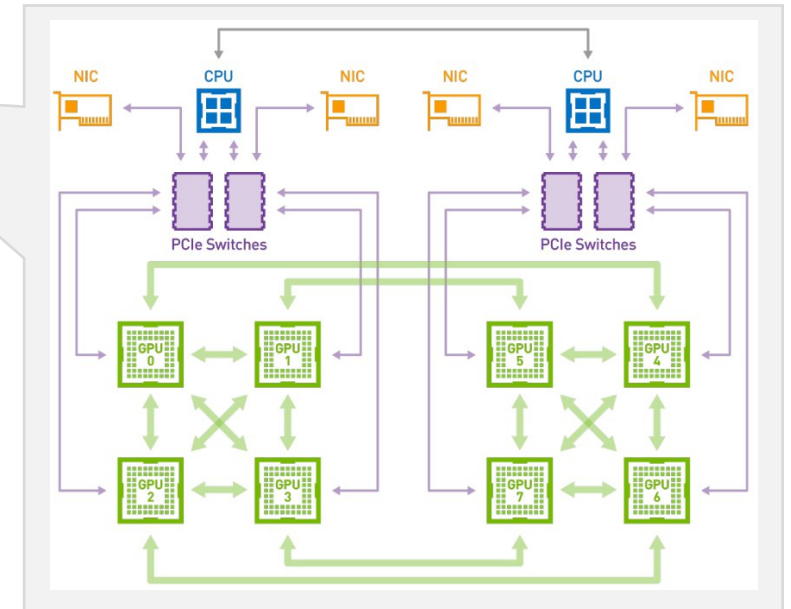
# Collective Operations for AI



## NVIDIA 80-node DGX SuperPOD



### DGX-1 Node



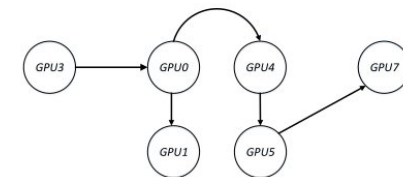
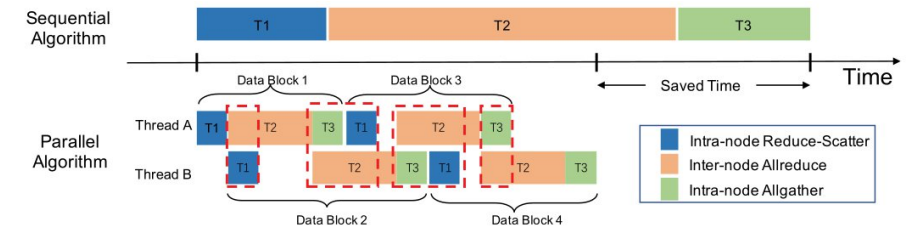
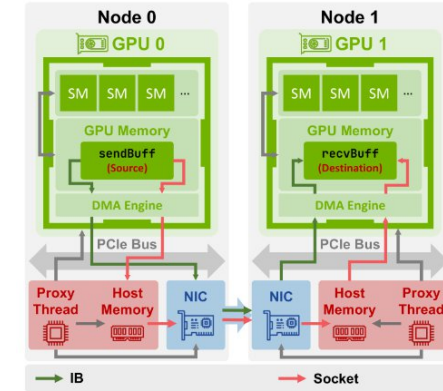
- **Nodes:** 80 nodes DGX-1 (8 GPU, 2 CPU, 4 NIC)
- **Intra-node** interconnect: NVLink, PCI Express
- **Inter-node:** Infiniband fabric 32 leaf switches + 20 spine
- Each node connected to 4 isolated networks (planes)
- Node 1/GPU0 – Node 02/GPU0: leaf switch 1
- Node 1/GPU0 – Node 61/GPU0: leaf 1, spine 1, leaf switch 25
- Node 1/GPU0 – Node 02/GPU1: NVLink to GPU1 (plane 1), leaf switch 2

# Collective Operations for AI



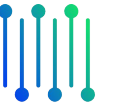
## Topology-aware Collective Communication Algorithms

1. Venkata M.G. et al. **Unified Collective Communication: A Unified Library for CPU, GPU, and DPU Collectives** // IEEE Micro. Vol. 45, No. 2. 2025. [OpenUCC](#)
2. Zhiyi Hu et al. **Demystifying NCCL: An In-depth Analysis of GPU Communication Protocols and Algorithms** // HOT Interconnects, 2025. [NCCL](#)
3. Dong J. **ACCL: Architecting Highly Scalable Distributed Training Systems With Highly Efficient Collective Communication Library** // IEEE Micro. 2021. Vol. 41. No. 5. [Alibaba](#)
4. Wang G., Venkataraman S, Phanishayee A., Thelin J. Devanur N., Stoica I. **Blink: Fast and Generic Collectives for Distributed ML** // Proc. of the Conference on Machine Learning and Systems (MLSys). 2020. [Microsoft](#)
5. Dong J. **EFLOPS: Algorithm and System Co Design for a High Performance Distributed Training Platform** // Proc. of IEEE International Symposium on High Performance Computer Architecture (HPCA), 2020.
6. Shah A. et al. **TACCL: Guiding Collective Algorithm Synthesis using Communication Sketches** // Proc. of the USENIX Symposium on Networked Systems Design and Implementation. 2023.
7. Jouppi N.P. et al. **TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings** // Proc. of International Symposium on Computer Architecture (ISCA). 2023. [Google](#)



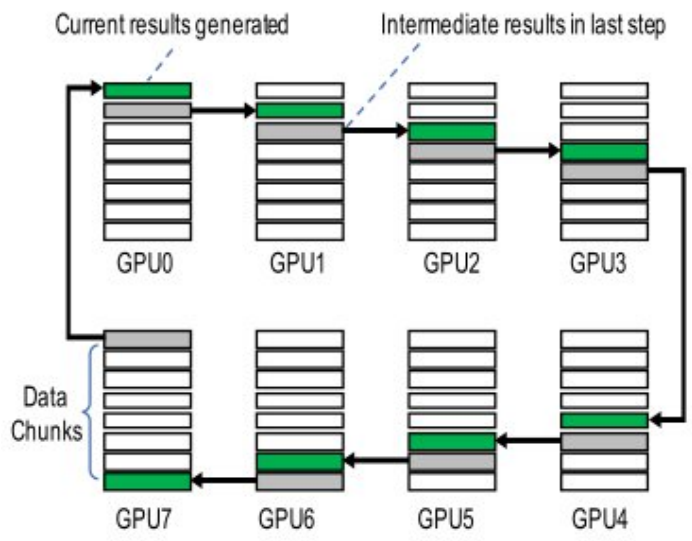
(c) Blink 6-GPU spanning trees.

# Collective Operations for AI



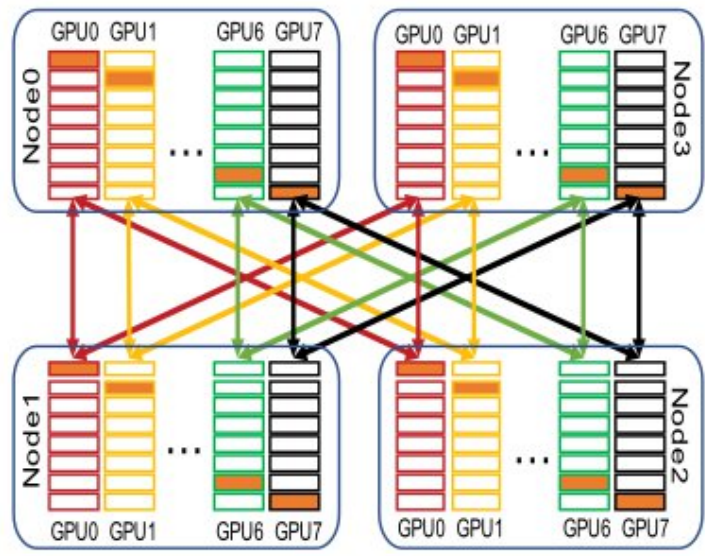
## Topology-aware Allreduce *sendBuf, recvBuf, m, op*

P	sbuf	+	P	rbuf
P0	[1, 2, 3]	→	P0	[8, 6, 15]
P1	[4, 0, 1]		P1	[8, 6, 15]
P2	[2, 1, 2]		P2	[8, 6, 15]
P3	[1, 3, 9]		P3	[8, 6, 15]



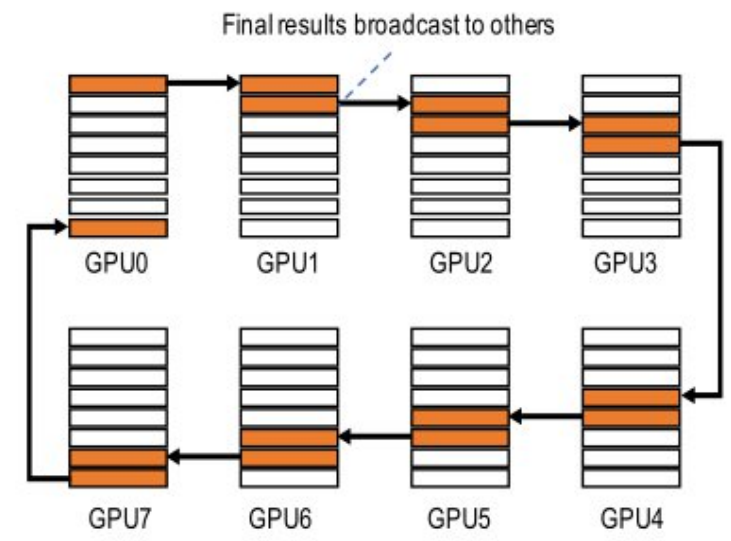
### 1. Intra-node ReduceScatter between $k$ GPUs

- Ring steps:  $k-1$
- Message size:  $m / k$



### 2. Inter-node Allreduce (dedicated NIC for each GPU)

- $k$  Allreduce in parallel
- Message size:  $m / k$



### 3. Intra-node Allgather between $k$ GPUs

# Hardware Collectives



- **NVIDIA/Mellanox SHARP**

- Operations: Barrier, Allreduce, Reduce, Bcast, Reduce-scatter, Allgather
- Data Types: 16/32/64-bit int/FP, 16-bit Bfloat and 8-bit Int)

- **Высокоскоростная коммуникационная сеть АЛЬФА**

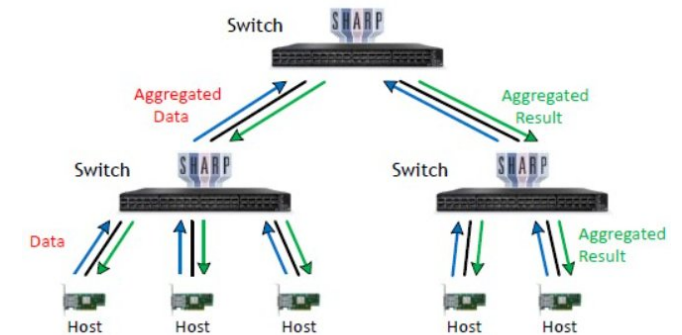
- Топология: кольцо, 2D-тор, 3D-тор
- Реализация MPI на базе MPICH 4.0
- ФГУП РФЯЦ ВНИИТФ им. Е.И. Забабахина (Снежинск)

- **Высокоскоростная сеть Ангара**

- Топология: 1D-4D-тор
- Реализация MPI на базе MPICH
- НИЦЭВТ

- **Коммуникационная система СМПО**

- Топология: мульти-тор, KNS
- Реализации MPI: Open MPI, MVAPICH, S-MPI (PSM2)
- РФЯЦ ВНИИЭФ (Саров)



- Graham R. et al. **Scalable Hierarchical Aggregation and Reduction Protocol (SHARP): Streaming-Aggregation Hardware Design and Evaluation**, 2020
- Ramesh et al. **Scalable MPI Collectives using SHARP: Large Scale Performance Evaluation on the TACC Frontera System**, 2020
- Симонов А.С. и др. **Высокоскоростная сеть Ангара: архитектура и результаты применения**, 2019.
- Степаненко С.А. **Мультипроцессорные среды суперЭВМ. Масштабирование эффективности**, 2016

# Thank you

Questions?

**Mikhail Kurnosov**

Siberian State University of Telecommunications  
and Information Sciences

Head of Parallel Computing Technologies Center,  
Doctor of Sciences, Professor

**E-mail:** [mkurnosov@sibguti.ru](mailto:mkurnosov@sibguti.ru)

**WWW:** [www.mkurnosov.net](http://www.mkurnosov.net)



**Монография**

Курносов М.Г.

**Алгоритмы коллективных  
операций стандарта MPI.**

2025, ISBN 978-5-9912-1149-9