ForestColl: Efficient Collective Communications on Heterogeneous Network Fabrics

Collective Communication

- Originally a topic in HPC, it is now extensively used for gradient, parameter, and activation synchronization in distributed ML training and inferencing.
- **Allgather** is an operation where every node/GPU broadcasts a shard of data.
  - reduce-scatter = *reversed* allgather
  - allreduce = reduce-scatter + allgather

Before			After		
Reduce-Scatter					
Node 0	Node 1	Node 2	Node 0	Node 1	Node 2
$S_0^{(0)}$	$S_0^{(1)}$	$S_0^{(2)}$	$\bigoplus_i S_0^{(i)}$		
$S_1^{(0)}$	$S_1^{(1)}$	$S_1^{(2)}$		$\bigoplus_i S_1^{(i)}$	
$S_2^{(0)}$	$S_2^{(1)}$	$S_2^{(2)}$			$\bigoplus_i S_2^{(i)}$
Allgather					
Node 0	Node 1	Node 2	Node 0	Node 1	Node 2
$S_0^{(0)}$			$S_0^{(0)}$	$S_0^{(0)}$	$S_0^{(0)}$
	$S_1^{(1)}$		$S_1^{(1)}$	$S_1^{(1)}$	$S_1^{(1)}$
		$S_2^{(2)}$	$S_2^{(2)}$	$S_2^{(2)}$	$S_2^{(2)}$
Allreduce					
Node 0	Node 1	Node 2	Node 0	Node 1	Node 2
$S_0^{(0)}$	$S_0^{(1)}$	$S_0^{(2)}$	$\bigoplus_i S_0^{(i)}$	$\bigoplus_i S_0^{(i)}$	$\bigoplus_i S_0^{(i)}$
$S_1^{(0)}$	$S_1^{(1)}$	$S_1^{(2)}$	$\bigoplus_i S_1^{(i)}$	$\bigoplus_i S_1^{(i)}$	$\bigoplus_i S_1^{(i)}$
$S_2^{(0)}$	$S_2^{(1)}$	$S_2^{(2)}$	$\bigoplus_i S_2^{(i)}$	$\bigoplus_i S_2^{(i)}$	$\bigoplus_i S_2^{(i)}$

We aim to derive efficient communication schedules for any given network topology.

- **Heterogeneity:** today's ML network architectures are highly diverse and heterogeneous.
- **Scalability:** optimizing aggregation and multicast traffic requires strict data dependency, often results in NP-hard discrete optimization.

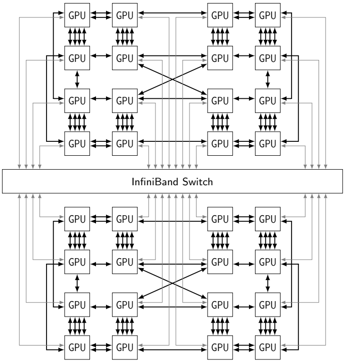


Figure: AMD MI250 Topology

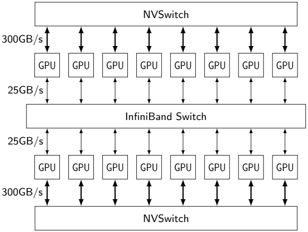


Figure: NVIDIA DGX A100 Topology

# of nodes	4	9	16	25	36
SCCL [PPoPP '21]	0.61s	1.00s	60s	3286s	$>10^4$s
TACCL [NSDI '23]	0.45s	67.8s	1801s	1802s	n/a

Table: Generation Time on 2D Torus ($n \times n$)

ForestColl: construct a set of spanning trees with k trees rooted at each node/GPU.

- Each tree broadcasts $1/k$ of the data from each root.
- **Performance:** the trees achieve mathematically **minimum overlap/congestion**.
- **Scalability:** computation is in **strongly polynomial time**.

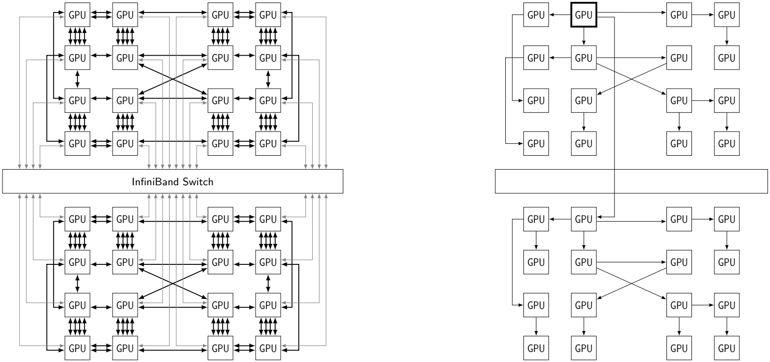


Figure: 2-Node AMD MI250

Previous Work

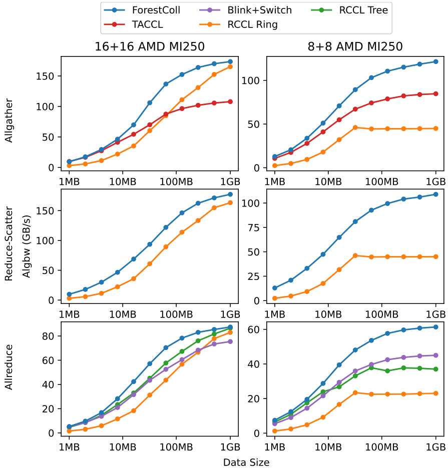
	SCCL	TACCL	BFB	Blink	TE-CCL	**ForestColl**
Switch-based Network	×	✓	×	×	✓	✓
Optimal Schedule	✓	×	×	×	×	✓
Scalable Runtime	×	×	✓	✓	×	✓

Table: Comparison of schedule generation methods.

Compared to previous work:

- ForestColl does not require input parameters such as the number of chunks or chunk size, eliminating the need for parameter sweeps.
- While converting switch topologies into equivalent switchless logical topologies, ForestColl ensures no compromise to performance.
- Blink constructs trees rooted at a single node. Thus, Blink does not support allgather/reduce-scatter and performs allreduce as reduce+broadcast, which can suffer from a bottleneck at a single GPU.

Evaluation



16+16 Setting:

- ForestColl outperforms RCCL by 37%, 32%, and 13% on average in allgather, reduce-scatter, and allreduce.

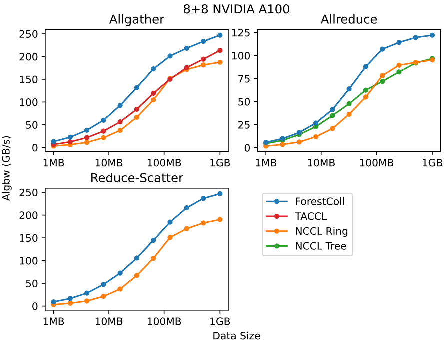
8+8 Setting (half of the GPUs per node):

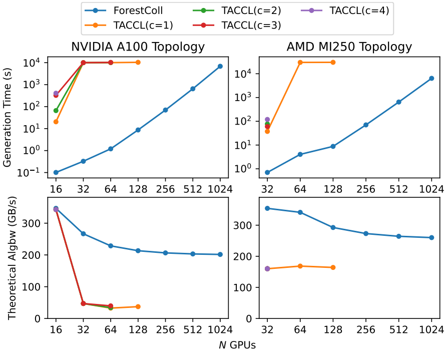
- ForestColl outperforms RCCL by 2.52x, 2.29x, and 1.47x on average in allgather, reduce-scatter, and allreduce.

Evaluation

ForestColl is

- on average, 52%, 38%, and 30% faster than NCCL on 8+8 A100 in allgather, reduce-scatter, and allreduce, respectively.
- orders of magnitude faster schedule generation than TACCL.





Q: Why not just use rings?

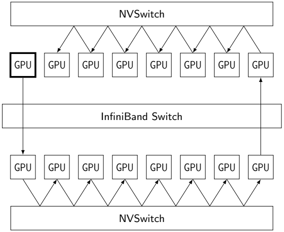


Figure: NCCL Ring

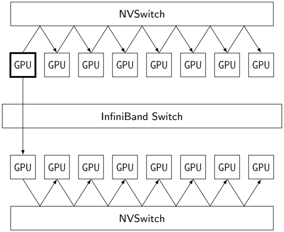


Figure: ForestColl

Ring vs ForestColl

Q: Why not just use rings?

- Intra-node bandwidth is much more abundant than inter-node bandwidth.

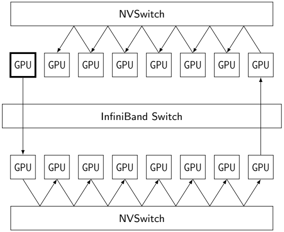


Figure: NCCL Ring

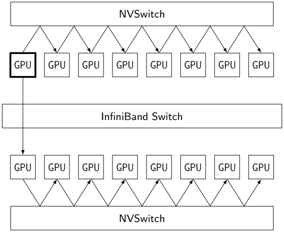


Figure: ForestColl

Ring vs ForestColl

Q: Why not just use rings?

- Intra-node bandwidth is much more abundant than inter-node bandwidth.
- Rings often overuse inter-node bandwidth, even though the traffic can be delivered within the node.
    - In ring allgather, every GPU performs a ring broadcast. In total, ring allgather has 2x amount of inter-node traffic as ForestColl.

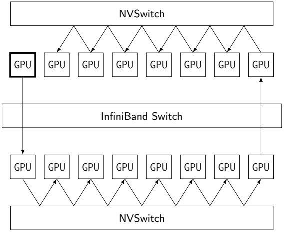


Figure: NCCL Ring

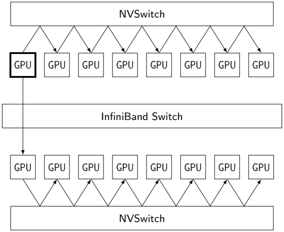


Figure: ForestColl

Q: Why not just use rings?

- Intra-node bandwidth is much more abundant than inter-node bandwidth.
- Rings often overuse inter-node bandwidth, even though the traffic can be delivered within the node.
  - In ring allgather, every GPU performs a ring broadcast. In total, ring allgather has 2x amount of inter-node traffic as ForestColl.

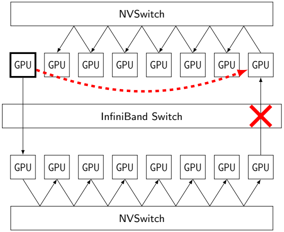


Figure: NCCL Ring

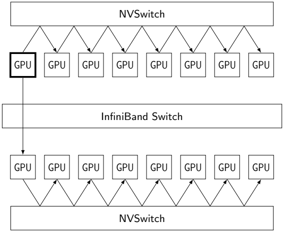


Figure: ForestColl

Ring vs ForestColl

Q: Why not just use rings?

- Intra-node bandwidth is much more abundant than inter-node bandwidth.
- Rings often overuse inter-node bandwidth, even though the traffic can be delivered within the node.
  - In ring allgather, every GPU performs a ring broadcast. In total, ring allgather has 2x amount of inter-node traffic as ForestColl.
- **Full Bandwidth Utilization $\neq$ Optimal Performance**

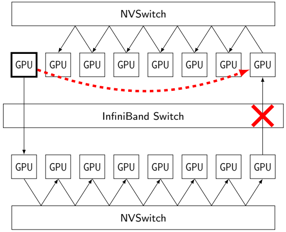


Figure: NCCL Ring

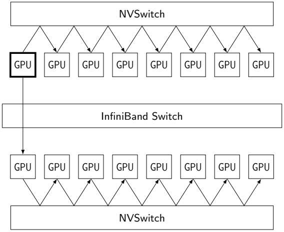


Figure: ForestColl

Q: What is the optimal allgather performance given a topology?

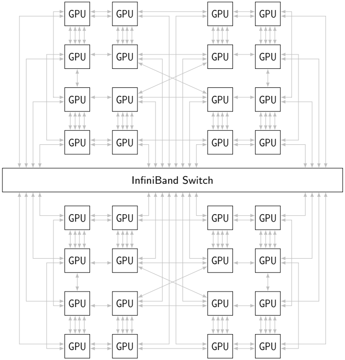


Figure: 2-Node AMD MI250

Q: What is the optimal allgather performance given a topology?

- Previous works often use $\frac{M}{B} \cdot \frac{N-1}{N}$.
  - The minimum amount of data needs to be received by a GPU is $\frac{M}{N}(N-1)$.
  - The total ingress bandwidth of the GPU is B.

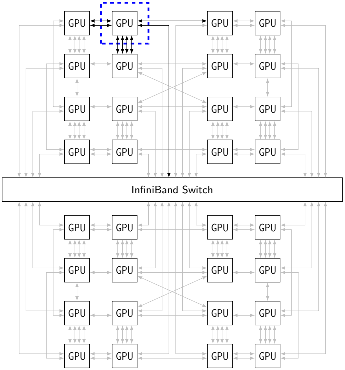


Figure: 2-Node AMD MI250

Q: What is the optimal allgather performance given a topology?

- Previous works often use $\frac{M}{B} \cdot \frac{N-1}{N}$.
  - The minimum amount of data needs to be received by a GPU is $\frac{M}{N}(N-1)$.
  - The total ingress bandwidth of the GPU is B.
- What if the performance is bounded by some network cut elsewhere?

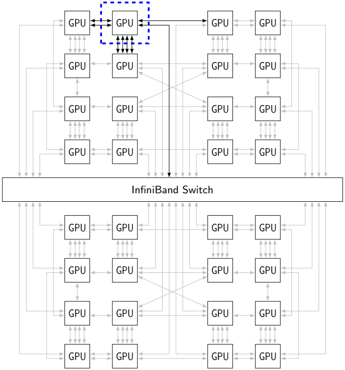


Figure: 2-Node AMD MI250

Optimality

Q: What is the optimal allgather performance given a topology?

- Consider a cut S from the network:

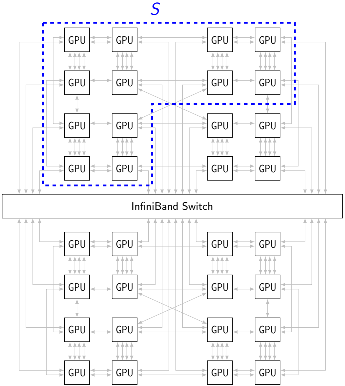


Figure: 2-Node AMD MI250

Optimality

Q: What is the optimal allgather performance given a topology?

- Consider a cut S from the network:
  - The amount of data needs to travel out of S is at least $\frac{M}{N} \cdot |S \cap V_c|$. ($V_c$ = set of GPUs)

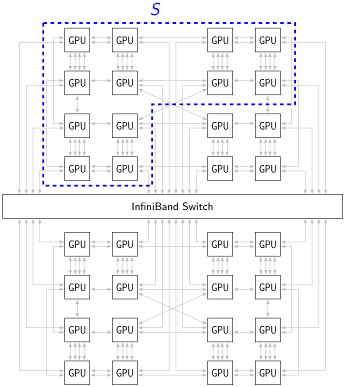


Figure: 2-Node AMD MI250

Q: What is the optimal allgather performance given a topology?

- Consider a cut S from the network:
  - The amount of data needs to travel out of S is at least $\frac{M}{N} \cdot |S \cap V_c|$. ($V_c =$ set of GPUs)
  - The exiting bandwidth of S is $B^+(S)$.

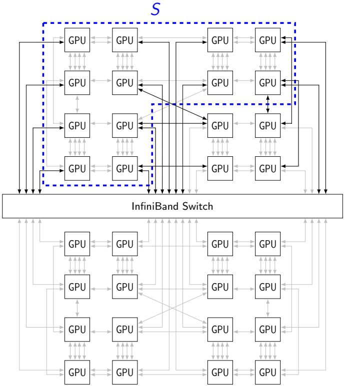


Figure: 2-Node AMD MI250

Optimality

Q: What is the optimal allgather performance given a topology?

- Consider a cut S from the network:
  - The amount of data needs to travel out of S is at least $\frac{M}{N} \cdot |S \cap V_c|$. ($V_c =$ set of GPUs)
  - The exiting bandwidth of S is $B^+(S)$.

  The allgather runtime is at least $\frac{M}{N} \cdot \frac{|S \cap V_c|}{B^+(S)}$.

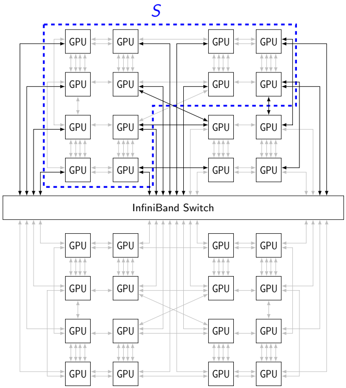


Figure: 2-Node AMD MI250

Optimality

Q: What is the optimal allgather performance given a topology?

- Consider a cut S from the network:
  - The amount of data needs to travel out of S is at least $\frac{M}{N} \cdot |S \cap V_c|$. $\quad (V_c = \text{set of GPUs})$
  - The exiting bandwidth of S is $B^+(S)$.

  The allgather runtime is at least $\frac{M}{N} \cdot \frac{|S \cap V_c|}{B^+(S)}$.
- The optimal performance is determined by a bottleneck cut S^* such that

$$\frac{|S^* \cap V_c|}{B^+(S^*)} = \max_{S \subset V, S \not\supseteq V_c} \frac{|S \cap V_c|}{B^+(S)}.$$

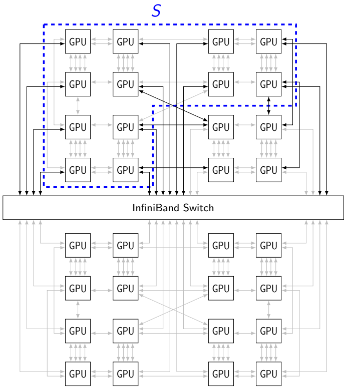


Figure: 2-Node AMD MI250

Q: What is the optimal allgather performance given a topology?

- Consider a cut S from the network:
  - The amount of data needs to travel out of S is at least $\frac{M}{N} \cdot |S \cap V_c|$. ($V_c =$ set of GPUs)
  - The exiting bandwidth of S is $B^+(S)$.

  The allgather runtime is at least $\frac{M}{N} \cdot \frac{|S \cap V_c|}{B^+(S)}$.
- The optimal performance is determined by a bottleneck cut S^* such that

$$\frac{|S^* \cap V_c|}{B^+(S^*)} = \max_{S \subset V, S \not\supseteq V_c} \frac{|S \cap V_c|}{B^+(S)}.$$

ForestColl can generate spanning trees to achieve the above optimality in strongly polynomial time.

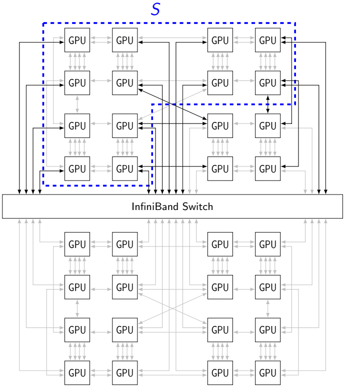
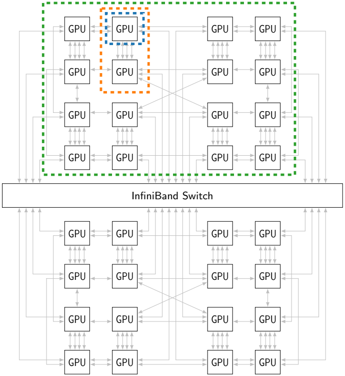


Figure: 2-Node AMD MI250

Optimality

AMD MI250:

- When number of nodes < 4, the ingress bandwidth of an OAM is the bottleneck.
- When number of nodes ≥ 4, the ingress bandwidth of a node is the bottleneck.



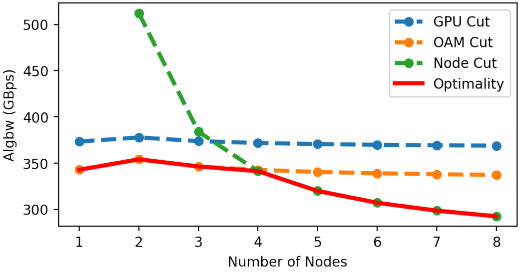
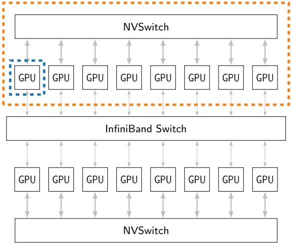


Figure: Optimality and performance bounds from different cuts of AMD MI250 topologies

Optimality

NVIDIA DGX A100:

- When number of nodes < 3, the ingress bandwidth of a GPU is the bottleneck.
- When number of nodes ≥ 3, the ingress bandwidth of a node is the bottleneck.



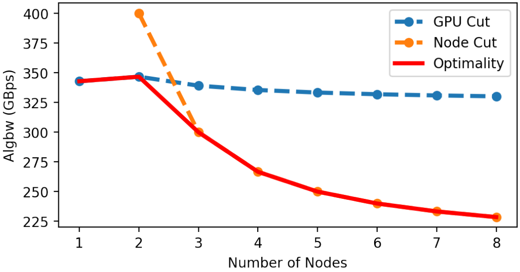


Figure: Optimality and performance bounds from different cuts of NVIDIA DGX A100 topologies

Summary

ForestColl is a schedule generation algorithm for collective communications that

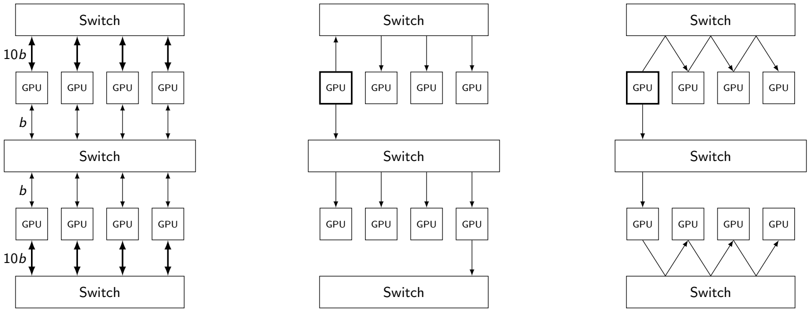
- provides **provably optimal** schedule;
- works on **any network topology** (direct-connect or switch topology);
- runs in **strongly polynomial time** (scalable to large number of nodes);
- outperforms state-of-the-art solutions.

Paper: https://arxiv.org/abs/2402.06787
GitHub: https://github.com/liangyuRain/ForestColl

Switch Topology

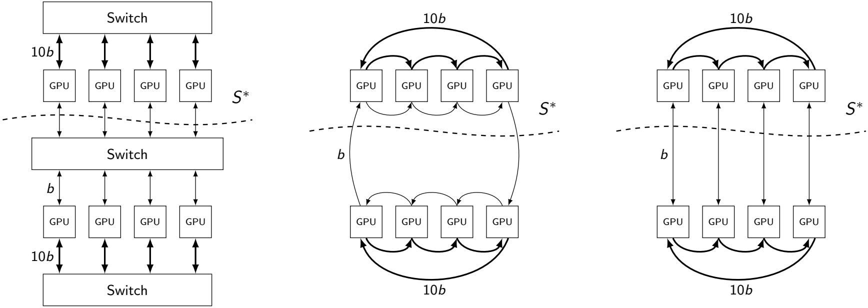
In switch topology, the vertex set consists of **compute nodes** and **switch nodes**.

- **Problem:** allgather is no longer defined by spanning out-trees.
  - Non-Spanning: unnecessary to broadcast data to every switch node.
  - Non-Tree: switch may not be able to multicast.
- **Solution:** convert switch topology into a logical topology without switches.



Edge Splitting

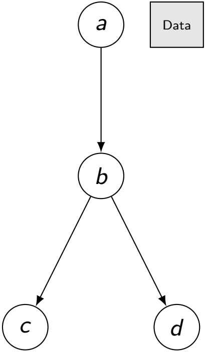
- **Previous work** proposed ways such as unwinding a switch into a ring.
- **Edge Splitting:** for each switch node w, iteratively choose edges $(u, w), (w, t)$ and replace them by (u, t) without sacrificing connectivity.
  - Originally used to prove connectivity properties of Eulerian graph. (Jackson, 1988; Frank, 1988; Bang-Jensen et al., 1995)
  - Now to remove switch nodes without compromising allgather performance.



Cut Bandwidth: $4b$ $\quad$ b $\quad$ $4b$

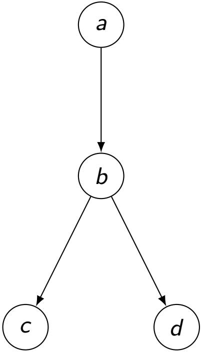
Pipeline Schedule

Non-Pipeline Schedule



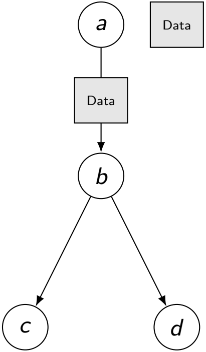
Time Cost: 0

Pipeline Schedule



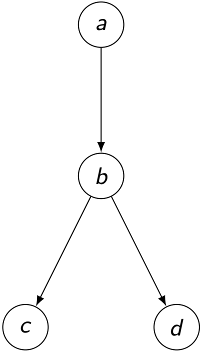
Pipeline Schedule

Non-Pipeline Schedule



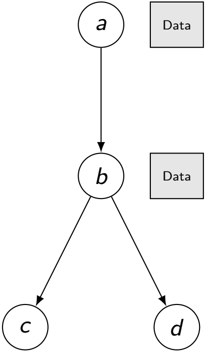
Time Cost: 1

Pipeline Schedule



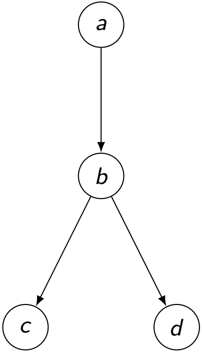
Pipeline Schedule

Non-Pipeline Schedule

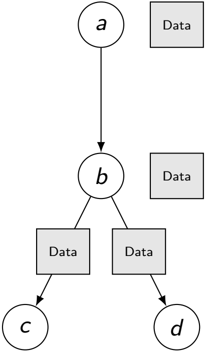


Time Cost: 1

Pipeline Schedule

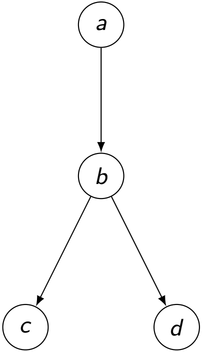


Non-Pipeline Schedule



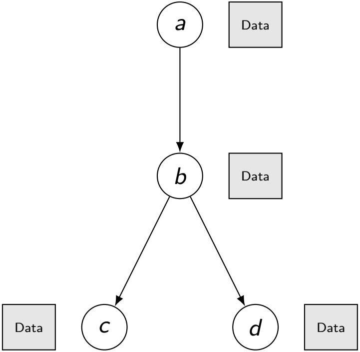
Time Cost: 2

Pipeline Schedule



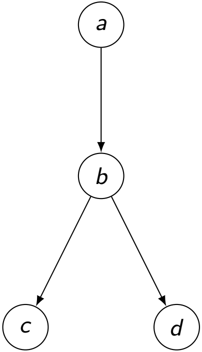
Pipeline Schedule

Non-Pipeline Schedule



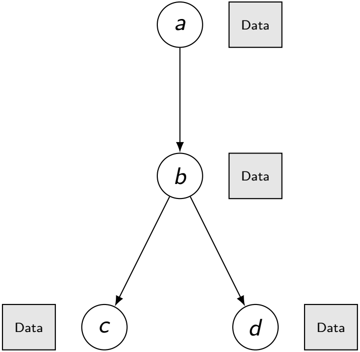
Time Cost: 2

Pipeline Schedule



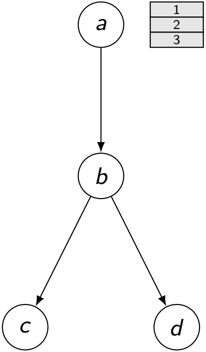
Pipeline Schedule

Non-Pipeline Schedule



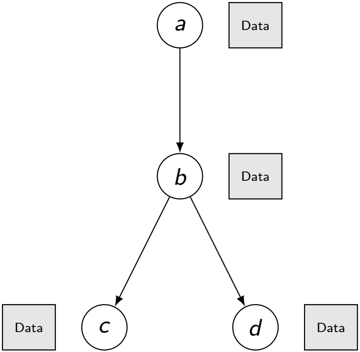
Time Cost: 2

Pipeline Schedule



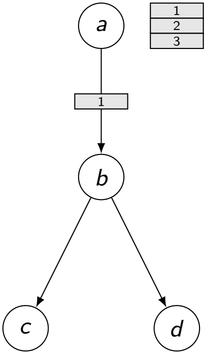
Time Cost: 0

Non-Pipeline Schedule



Time Cost: 2

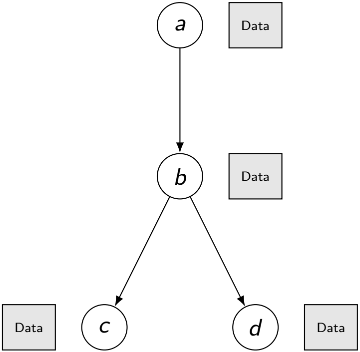
Pipeline Schedule



Time Cost: 1/3

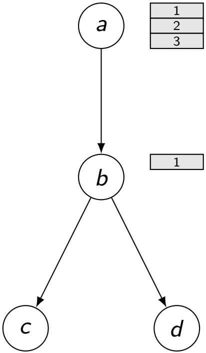
Pipeline Schedule

Non-Pipeline Schedule



Time Cost: 2

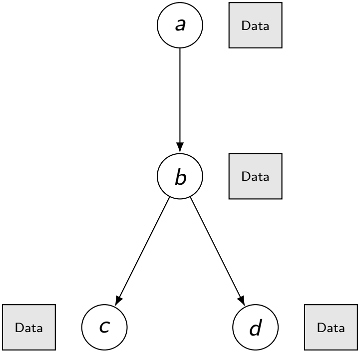
Pipeline Schedule



Time Cost: 1/3

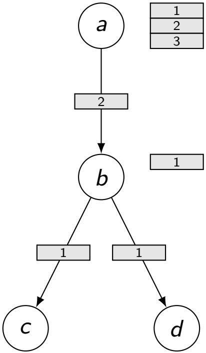
Pipeline Schedule

Non-Pipeline Schedule



Time Cost: 2

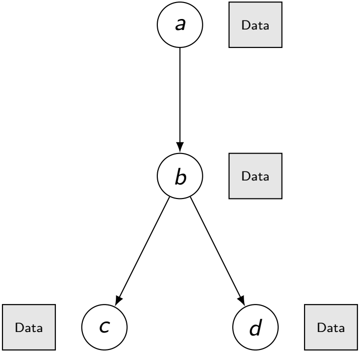
Pipeline Schedule



Time Cost: 2/3

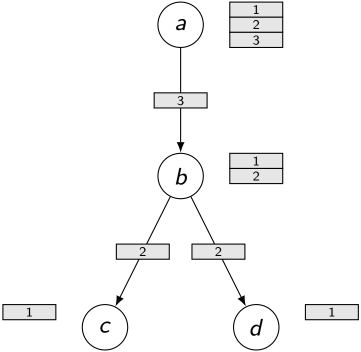
Pipeline Schedule

Non-Pipeline Schedule



Time Cost: 2

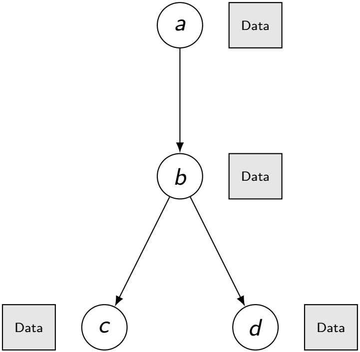
Pipeline Schedule



Time Cost: 3/3

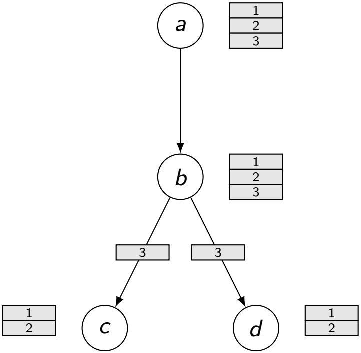
Pipeline Schedule

Non-Pipeline Schedule



Time Cost: 2

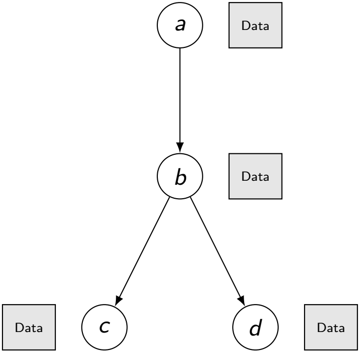
Pipeline Schedule



Time Cost: 4/3

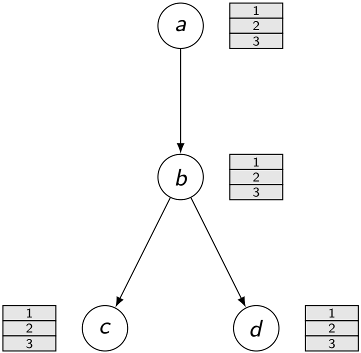
Pipeline Schedule

Non-Pipeline Schedule



Time Cost: 2

Pipeline Schedule

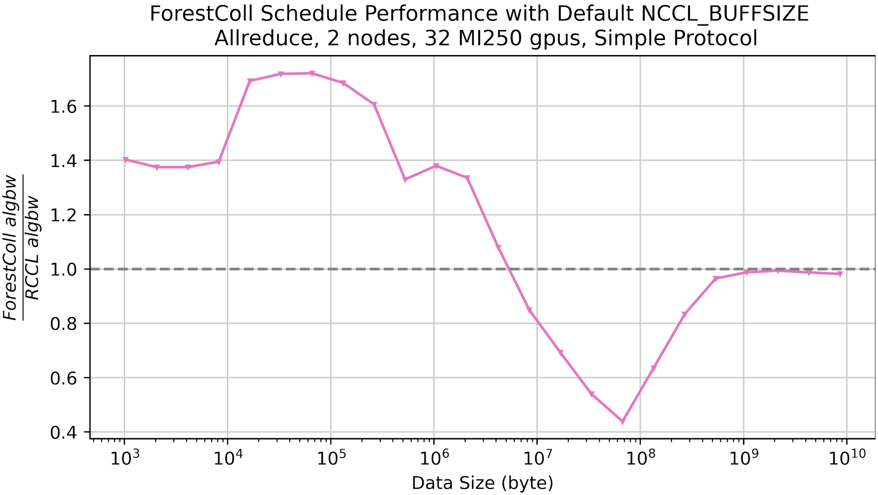


Time Cost: 4/3

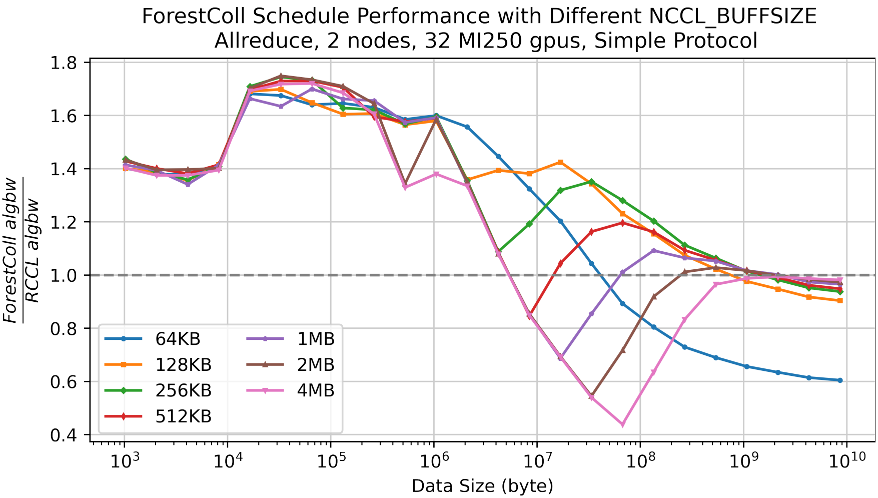
Chunk Size

- ForestColl schedule assumes that data is transmitted as **flows** along the trees rather than through discrete send/recv steps.
- Ideally, **chunk size** should be as small as possible to enhance bandwidth utilization; however, send/recv has **overhead** in practice.

Overhead-dominated $\xLeftrightarrow[\text{Chunk Size}]{\text{small} \qquad\qquad\qquad \text{large}}$ Pipeline bubble, Idle links


ForestColl Schedule Performance with Default NCCL_BUFFSIZE
Allreduce, 2 nodes, 32 MI250 gpus, Simple Protocol
ForestColl algbw / RCCL algbw
1.6
1.4
1.2
1.0
0.8
0.6
0.4
10^3
10^4
10^5
10^6
10^7
10^8
10^9
10^{10}
Data Size (byte)

Chunk Size Experiment



Thank you

Paper: https://arxiv.org/abs/2402.06787
GitHub: https://github.com/liangyuRain/ForestColl