ForestColl: Efficient Collective Communications on Heterogeneous Network Fabrics

Presented by **Liangyu Zhao**

University of Washington, Microsoft Research

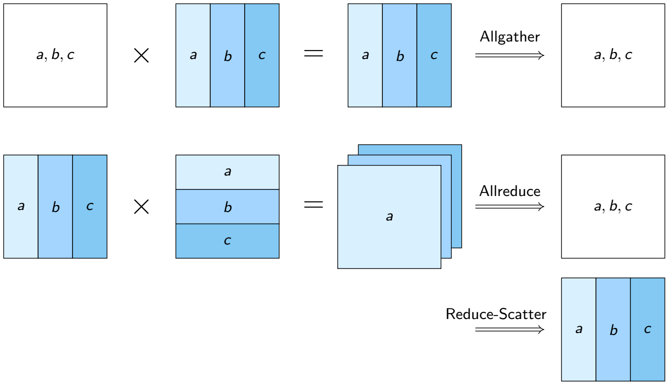
Bottom Line Up Front

- Problem Statement:
  - Network topologies of ML hardware platforms are highly **diverse** and **heterogeneous**.
  - Existing communication libraries cannot fully unlock their performance potential.

- ForestColl: a high-performance solution for collective communications on any network topology.
  - **Collective Communication:** up to 3x faster than vendor-provided libraries.
  - **Improved Training Efficiency:** 20% speedup in large language model (LLM) training.
  - **Schedule Generation:** orders of magnitude ($>10^4$x) faster than previous methods.

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 a collective where every node/GPU broadcasts a distinct shard of data.
  - reduce-scatter = *reversed* allgather
  - allreduce = reduce-scatter + allgather



Challenges

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

- **Diversity & Heterogeneity:** today's ML network topologies are highly *diverse* across hardware platforms and *heterogeneous* within individual networks.
- **Scalability:** optimizing aggregation and multicast traffic requires strict data dependency, often resulting in NP-hard discrete optimization.

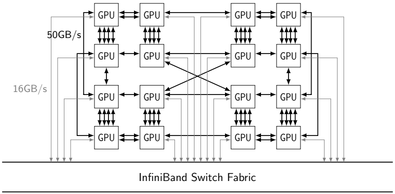


Figure: AMD MI250 Box Topology

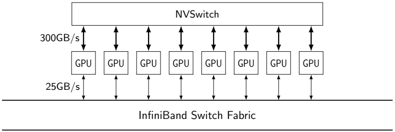


Figure: NVIDIA DGX A100 Box Topology

# 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

ForestColl: construct spanning trees (forest☺) with k trees rooted at each node/GPU.

- In allgather, every tree *simultaneously* broadcasts $1/k$ of the data from its root.
- **Performance:** the trees achieve mathematically **minimum overlap/congestion**.
- **Scalability:** computation is in **strongly polynomial time**.

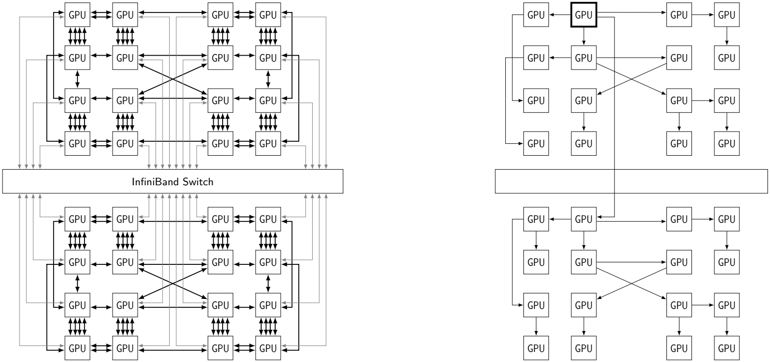


Figure: 2-Box AMD MI250

Previous Work

	SCCL [PPoPP '21]	TACCL [NSDI '23]	BFB [NSDI '25]	Blink [MLSys '20]	TE-CCL [SIGCOMM '24]	**ForestColl**
Switch-based Network	×	✓	×	×	✓	✓
Optimal Schedule	✓	×	×	×	×	✓
Scalable Runtime	×	×	✓	✓	×	✓

Previous schedule generation methods either

- focus on switchless direct-connect networks only;
- lack theoretical performance guarantees for generated schedules;
- rely on NP-hard optimization methods.

ForestColl Optimality

Q: What is the optimal allgather throughput given a network topology?

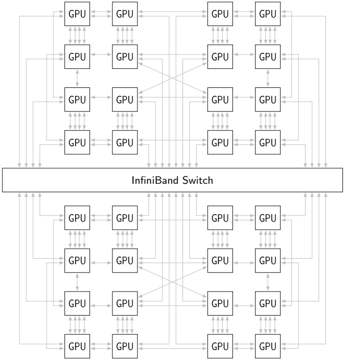


Figure: 2-Box AMD MI250

Q: What is the optimal allgather throughput given a network topology?

- Previous works often look at **the amount of data received vs bandwidth** at a single node. The allgather time lower bound is:

$$\frac{M}{B} \cdot \frac{N-1}{N} = \underbrace{\frac{M}{N}}_{\text{shard size}} \cdot \underbrace{(N-1)}_{\#\text{ of shards}} \Big/ \underbrace{B}_{\text{node bandwidth}}$$

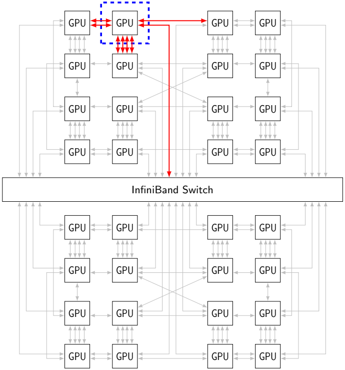


Figure: 2-Box AMD MI250

Q: What is the optimal allgather throughput given a network topology?

- Previous works often look at **the amount of data received vs bandwidth** at a single node. The allgather time lower bound is:

$$\frac{M}{B}\cdot\frac{N-1}{N}=\underbrace{\frac{M}{N}}_{\text{shard size}}\cdot\underbrace{(N-1)}_{\#\text{ of shards}}\;/\;\underbrace{B}_{\text{node bandwidth}}$$

- What if the throughput is not bounded by the bandwidth of a single node?

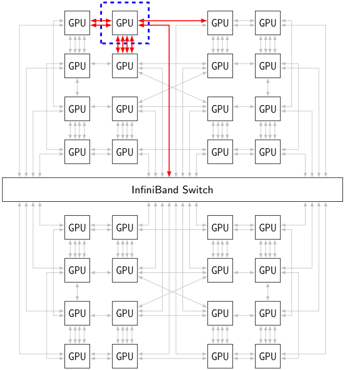


Figure: 2-Box AMD MI250

Q: What is the optimal allgather throughput given a network topology?

- Consider an arbitrary network cut S.

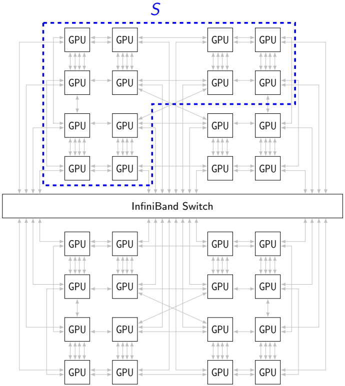


Figure: 2-Box AMD MI250

ForestColl Optimality

Q: What is the optimal allgather throughput given a network topology?

- Consider an arbitrary network cut S.
- Cut S implies an allgather time lower bound:

$$\frac{\text{min data exiting } S}{\text{available bandwidth}} = \frac{\text{shard size } \times \text{ num of GPUs in } S}{\text{exiting bandwidth of } S}$$

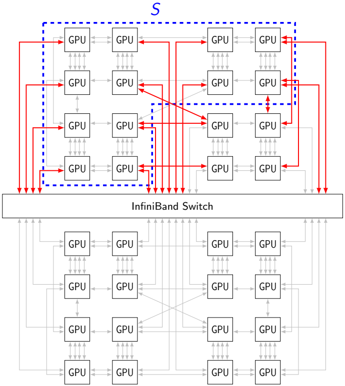


Figure: 2-Box AMD MI250

Q: What is the optimal allgather throughput given a network topology?

- Consider an arbitrary network cut S.
- Cut S implies an allgather time lower bound:

$$\frac{\text{min data exiting } S}{\text{available bandwidth}} = \frac{\text{shard size} \times \text{num of GPUs in } S}{\text{exiting bandwidth of } S}$$

- The optimal allgather throughput is determined by a **bottleneck cut** S^*, where

$$\text{shard size} \times \frac{\text{num of GPUs in } S^*}{\text{exiting bandwidth of } S^*}$$

  is maximized across all possible network cuts.

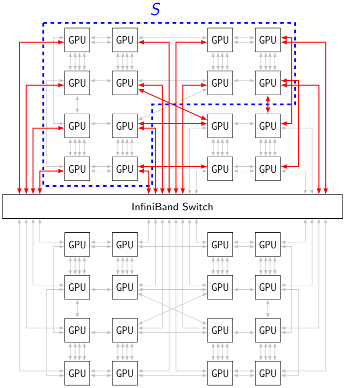


Figure: 2-Box AMD MI250

ForestColl Optimality

Q: What is the optimal allgather throughput given a network topology?

- Consider an arbitrary network cut S.
- Cut S implies an allgather time lower bound:

$$\frac{\text{min data exiting } S}{\text{available bandwidth}} = \frac{\text{shard size } \times \text{ num of GPUs in } S}{\text{exiting bandwidth of } S}$$

- The optimal allgather throughput is determined by a **bottleneck cut** S^*, where

$$\text{shard size} \times \frac{\text{num of GPUs in } S^*}{\text{exiting bandwidth of } S^*} = \boxed{\frac{M}{N} \max_{S \subset V, S \not\supseteq V_c} \frac{|S \cap V_c|}{B^+(S)}}$$

  is maximized across all possible network cuts.

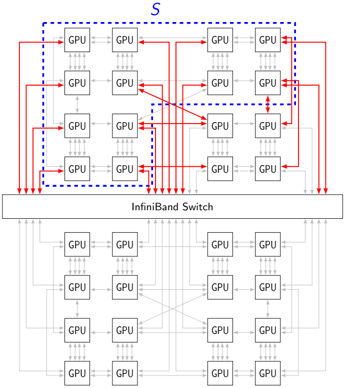


Figure: 2-Box AMD MI250

ForestColl Optimality

Q: What is the optimal allgather throughput given a network topology?

- Consider an arbitrary network cut S.
- Cut S implies an allgather time lower bound:

$$\frac{\text{min data exiting } S}{\text{available bandwidth}} = \frac{\text{shard size} \times \text{num of GPUs in } S}{\text{exiting bandwidth of } S}$$

- The optimal allgather throughput is determined by a **bottleneck cut** S^*, where

$$\text{shard size} \times \frac{\text{num of GPUs in } S^*}{\text{exiting bandwidth of } S^*} = \boxed{\frac{M}{N} \max_{S \subset V, S \not\supseteq V_c} \frac{|S \cap V_c|}{B^+(S)}}$$

  is maximized across all possible network cuts.

1. The spanning trees generated by ForestColl achieve the above optimality.
2. ForestColl can efficiently compute the above optimality.

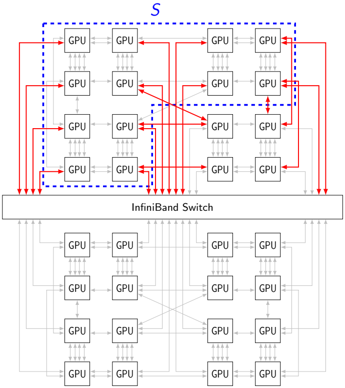


Figure: 2-Box AMD MI250

NVIDIA DGX A100:

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

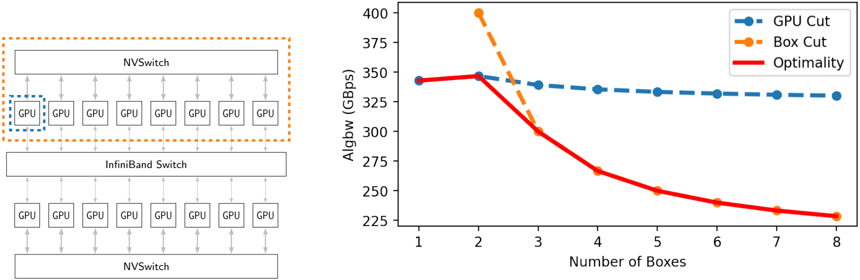
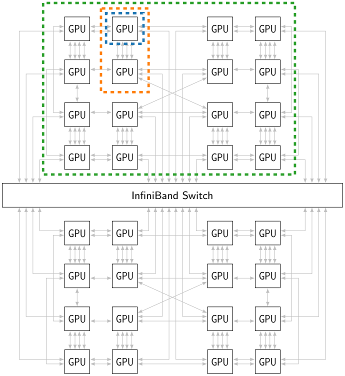


Figure: Optimality and performance bounds from different cuts of NVIDIA DGX A100 topologies

ForestColl Optimality

AMD MI250:

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



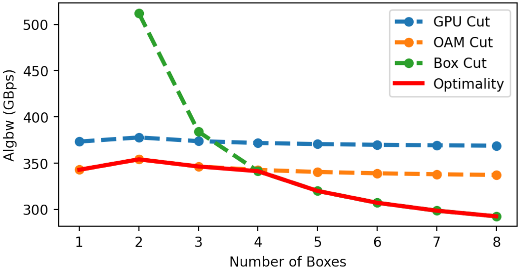


Figure: Optimality and performance bounds from different cuts of AMD MI250 topologies

Q: Why not just use rings?

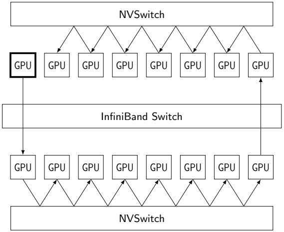


Figure: NCCL Ring

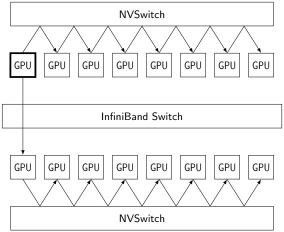


Figure: ForestColl

Q: Why not just use rings?

- **Bottleneck:** *inter-box* bandwidth is significantly less than *intra-box* bandwidth.

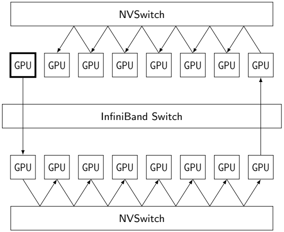


Figure: NCCL Ring

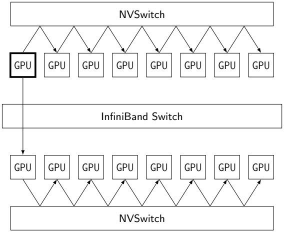


Figure: ForestColl

Ring vs ForestColl

Q: Why not just use rings?

- **Bottleneck:** *inter-box* bandwidth is significantly less than *intra-box* bandwidth.
- Rings often overuse inter-box bandwidth, even though data could be sent intra-box.

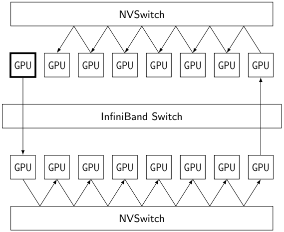


Figure: NCCL Ring

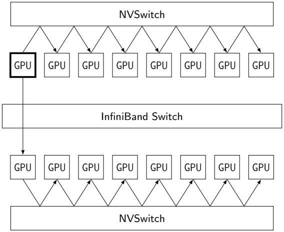


Figure: ForestColl

Q: Why not just use rings?

- **Bottleneck:** *inter-box* bandwidth is significantly less than *intra-box* bandwidth.
- Rings often overuse inter-box bandwidth, even though data could be sent intra-box.

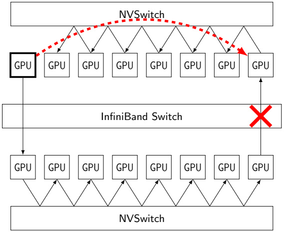


Figure: NCCL Ring

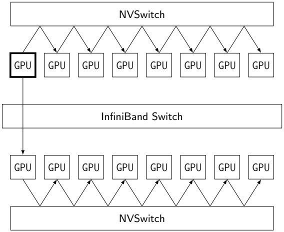


Figure: ForestColl

Q: Why not just use rings?

- **Bottleneck:** *inter-box* bandwidth is significantly less than *intra-box* bandwidth.
- Rings often overuse inter-box bandwidth, even though data could be sent intra-box.
  - When all GPUs broadcast simultaneously, ring allgather generates nearly 2x amount of inter-box traffic compared to ForestColl.

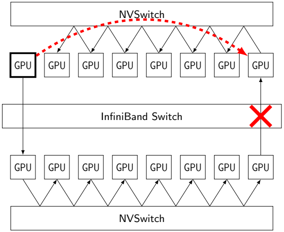


Figure: NCCL Ring

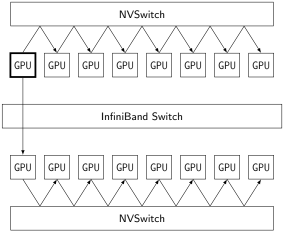
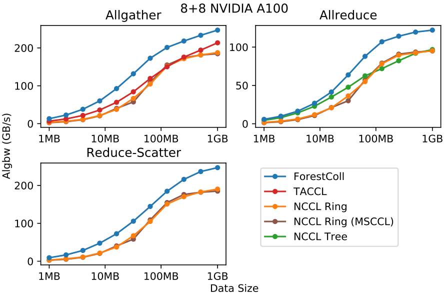


Figure: ForestColl

Comparison against NCCL on 2x NVIDIA DGX A100 boxes:

- **From 1MB to 1GB data sizes**, ForestColl is, on average, **130%, 85%, and 27%** faster than NCCL in allgather, reduce-scatter, and allreduce.

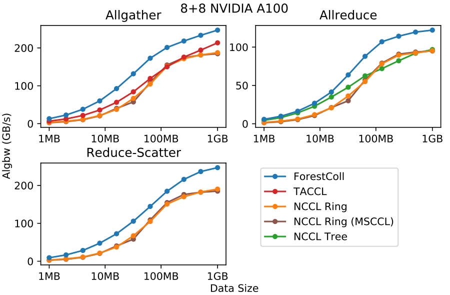


Allgather	Algbw (GB/s)					ForestColl / Baseline				
	1M	**16M**	**128M**	**1G**	**Avg**	**1M**	**16M**	**128M**	**1G**	**Avg**
ForestColl	13.1	92.6	201	247	130	-	-	-	-	-
TACCL	6.67	56.4	150	213	97.3	2.0x	1.6x	1.3x	1.2x	1.5x
NCCL Ring	3.17	37.6	152	187	85.8	4.1x	2.5x	1.3x	1.3x	2.3x
Reduce-Scatter	Algbw (GB/s)					ForestColl / Baseline				
	1M	**16M**	**128M**	**1G**	**Avg**	**1M**	**16M**	**128M**	**1G**	**Avg**
ForestColl	9.24	72.5	185	247	119	-	-	-	-	-
NCCL Ring	3.17	37.5	151	190	86.0	2.9x	1.9x	1.2x	1.3x	1.8x
Allreduce	Algbw (GB/s)					ForestColl / Baseline				
	1M	**16M**	**128M**	**1G**	**Avg**	**1M**	**16M**	**128M**	**1G**	**Avg**
ForestColl	5.75	41.4	107	122	65.0	-	-	-	-	-
NCCL Tree	4.47	34.8	71.9	96.8	48.8	1.3x	1.2x	1.5x	1.3x	1.3x
NCCL Ring	1.75	20.8	78.3	95.3	44.6	3.3x	2.0x	1.4x	1.3x	2.0x
NCCL Best	4.47	34.8	78.3	96.8	50.1	1.3x	1.2x	1.4x	1.3x	1.3x

Figure: ForestColl vs NCCL on 2-box NVIDIA DGX A100.

Comparison against NCCL on 2x NVIDIA DGX A100 boxes:

- **From 1MB to 1GB data sizes**, ForestColl is, on average, **130%, 85%, and 27%** faster than NCCL in allgather, reduce-scatter, and allreduce.
- **At 1GB data size**, ForestColl is **32%, 30%, and 26%** faster than NCCL in allgather, reduce-scatter, and allreduce.



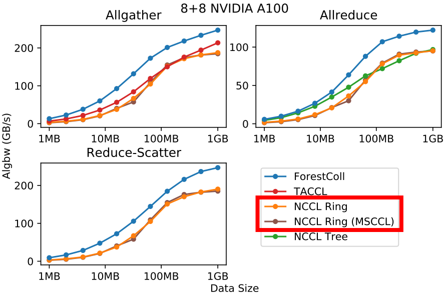
Allgather	Algbw (GB/s)					ForestColl / Baseline				
	1M	**16M**	**128M**	**1G**	**Avg**	**1M**	**16M**	**128M**	**1G**	**Avg**
ForestColl	13.1	92.6	201	247	130	-	-	-	-	-
TACCL	6.67	56.4	150	213	97.3	2.0x	1.6x	1.3x	1.2x	1.5x
NCCL Ring	3.17	37.6	152	187	85.8	4.1x	2.5x	1.3x	1.3x	2.3x
Reduce-Scatter	Algbw (GB/s)					ForestColl / Baseline				
	1M	**16M**	**128M**	**1G**	**Avg**	**1M**	**16M**	**128M**	**1G**	**Avg**
ForestColl	9.24	72.5	185	247	119	-	-	-	-	-
NCCL Ring	3.17	37.5	151	190	86.0	2.9x	1.9x	1.2x	1.3x	1.8x
Allreduce	Algbw (GB/s)					ForestColl / Baseline				
	1M	**16M**	**128M**	**1G**	**Avg**	**1M**	**16M**	**128M**	**1G**	**Avg**
ForestColl	5.75	41.4	107	122	65.0	-	-	-	-	-
NCCL Tree	4.47	34.8	71.9	96.8	48.8	1.3x	1.2x	1.5x	1.3x	1.3x
NCCL Ring	1.75	20.8	78.3	95.3	44.6	3.3x	2.0x	1.4x	1.3x	2.0x
NCCL Best	4.47	34.8	78.3	96.8	50.1	1.3x	1.2x	1.4x	1.3x	1.3x

Figure: ForestColl vs NCCL on 2-box NVIDIA DGX A100.

Collective Operation Evaluation

Comparison against NCCL on 2x NVIDIA DGX A100 boxes:

- **From 1MB to 1GB data sizes**, ForestColl is, on average, **130%, 85%, and 27%** faster than NCCL in allgather, reduce-scatter, and allreduce.
- **At 1GB data size**, ForestColl is **32%, 30%, and 26%** faster than NCCL in allgather, reduce-scatter, and allreduce.
- We use MSCCL library for schedule implementation and execution.
  - Implementing NCCL's ring algorithms in MSCCL yields identical performance to NCCL, proving that **ForestColl's speedups stem solely from scheduling optimizations.**



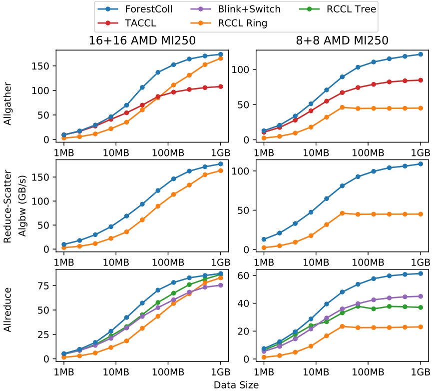
Allgather	Algbw (GB/s)					ForestColl / Baseline				
	1M	16M	128M	1G	Avg	1M	16M	128M	1G	Avg
ForestColl	13.1	92.6	201	247	130	-	-	-	-	-
TACCL	6.67	56.4	150	213	97.3	2.0x	1.6x	1.3x	1.2x	1.5x
NCCL Ring	3.17	37.6	152	187	85.8	4.1x	2.5x	1.3x	1.3x	2.3x
Reduce-Scatter	Algbw (GB/s)					ForestColl / Baseline				
	1M	16M	128M	1G	Avg	1M	16M	128M	1G	Avg
ForestColl	9.24	72.5	185	247	119	-	-	-	-	-
NCCL Ring	3.17	37.5	151	190	86.0	2.9x	1.9x	1.2x	1.3x	1.8x
Allreduce	Algbw (GB/s)					ForestColl / Baseline				
	1M	16M	128M	1G	Avg	1M	16M	128M	1G	Avg
ForestColl	5.75	41.4	107	122	65.0	-	-	-	-	-
NCCL Tree	4.47	34.8	71.9	96.8	48.8	1.3x	1.2x	1.5x	1.3x	1.3x
NCCL Ring	1.75	20.8	78.3	95.3	44.6	3.3x	2.0x	1.4x	1.3x	2.0x
NCCL Best	4.47	34.8	78.3	96.8	50.1	1.3x	1.2x	1.4x	1.3x	1.3x

Figure: ForestColl vs NCCL on 2-box NVIDIA DGX A100.

Collective Operation Evaluation

Comparison against RCCL on 2x AMD MI250 boxes:

- **16+16 Setting:** ForestColl is, on average, 91%, 87%, and 15% faster in allgather, reduce-scatter, and allreduce.
- **8+8 Setting** (half of the GPUs per node)**:** ForestColl is, on average, 2.98x, 2.86x, and 1.40x faster in allgather, reduce-scatter, and allreduce.



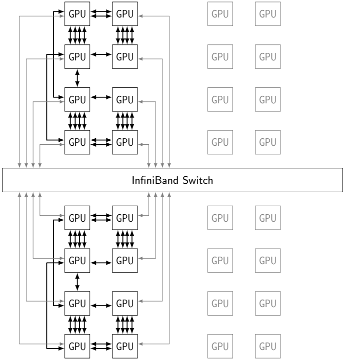
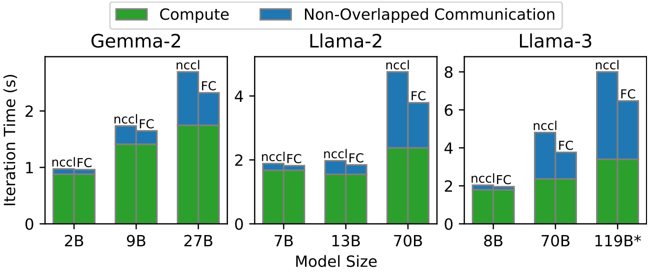


Figure: 8+8 Topology

ML Training Evaluation

In PyTorch FSDP training of state-of-the-art LLMs across 2x DGX A100,

- The communication speedup offered by ForestColl reduces training iteration times by 14% for Gemma 27B and 20% for Llama 70B and 119B* compared to NCCL.

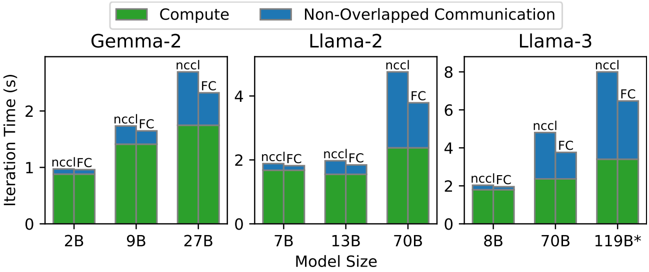


Llama-3-119B* is our reduced version of Llama-3-405B, with fewer hidden layers.

ML Training Evaluation

In PyTorch FSDP training of state-of-the-art LLMs across 2x DGX A100,

- The communication speedup offered by ForestColl reduces training iteration times by 14% for Gemma 27B and 20% for Llama 70B and 119B* compared to NCCL.
- Larger models are more communication-bound, leading to greater improvements with ForestColl.

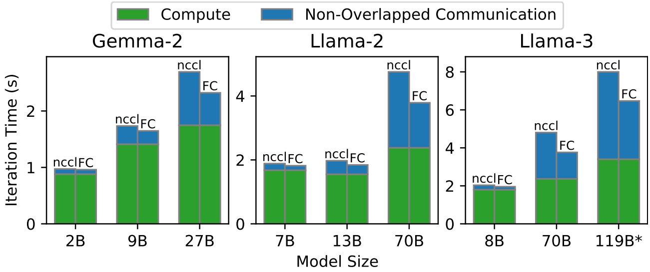


Llama-3-119B* is our reduced version of Llama-3-405B, with fewer hidden layers.

ML Training Evaluation

In PyTorch FSDP training of state-of-the-art LLMs across 2x DGX A100,

- The communication speedup offered by ForestColl reduces training iteration times by 14% for Gemma 27B and 20% for Llama 70B and 119B* compared to NCCL.
- Larger models are more communication-bound, leading to greater improvements with ForestColl.
  - Forced to use **smaller batch sizes** to avoid GPU out of memory.
  - Less compute-communication overlap due to GPU **resource contention** (e.g., SM, memory) between compute and communication kernels.

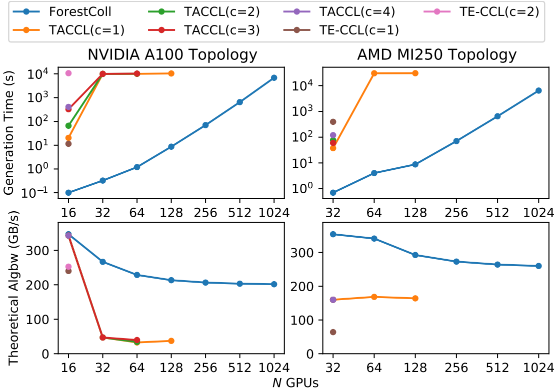


Llama-3-119B* is our reduced version of Llama-3-405B, with fewer hidden layers.

Schedule Generation Evaluation

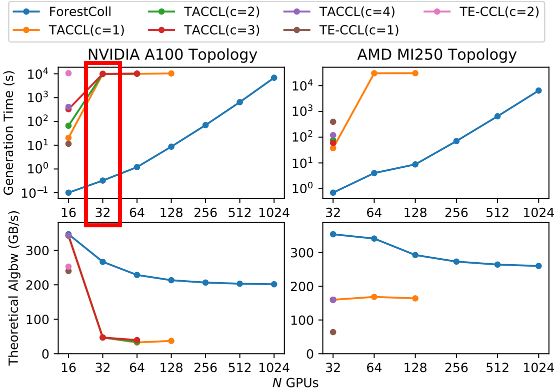
Comparison against TACCL [NSDI '23] and TE-CCL [SIGCOMM '24]:

- **Speed:** ForestColl is orders of magnitude faster in schedule generation time.
- **Quality:** ForestColl's schedules always achieve theoretically optimal algorithmic bandwidth.
- **Easy to Use:** ForestColl requires no parameter sweep.



Comparison against TACCL [NSDI '23] and TE-CCL [SIGCOMM '24]:

- **Speed:** ForestColl is orders of magnitude faster in schedule generation time.
- **Quality:** ForestColl's schedules always achieve theoretically optimal algorithmic bandwidth.
- **Easy to Use:** ForestColl requires no parameter sweep.



Discussion

In-Network Collective Communications

- Tree representation is compatible with in-network reduce/multicast.
- NVLink SHARP simplifies intra-box reduce/multicast for ForestColl.

Drawbacks

- ForestColl prioritizes throughput over latency.
  - Large data transfers are more performance-critical for LLM training.
  - CCLs support switching to low-latency algorithms based on data size at runtime.
- ForestColl has high implementation complexity.
  - Ongoing Work: Transition from MSCCL (domain-specific language) to MSCCL++ (CUDA kernel implementation).

Summary

ForestColl is a schedule generation algorithm for collective communications that

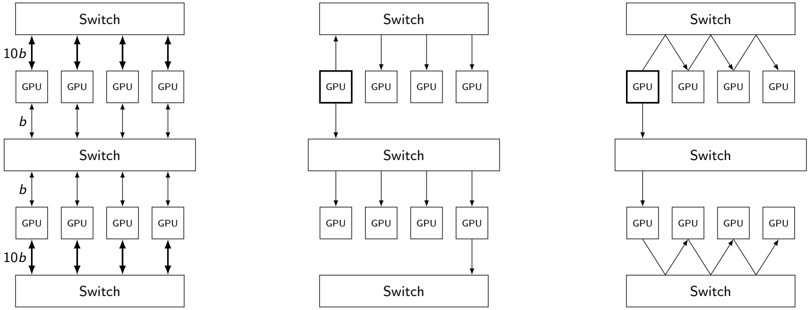
- provides **provably optimal** schedule;
- works on **any network topology** (direct-connect or switch topology);
- runs in **strongly polynomial time** (scalable to large number of nodes);
- outperforms state-of-the-art solutions in collective communication performance, ML training, and schedule generation speed.

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

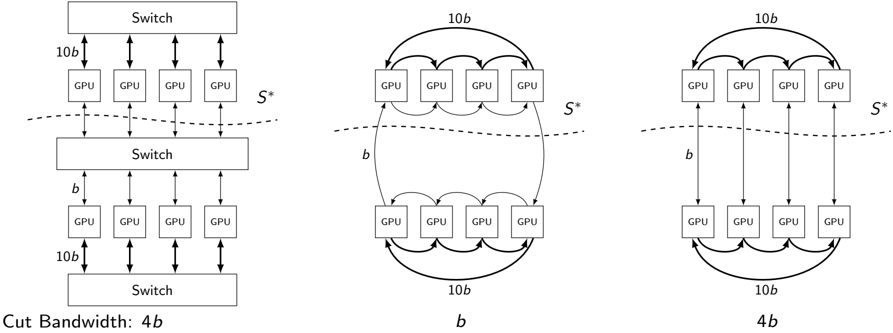
Switch Topology

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.

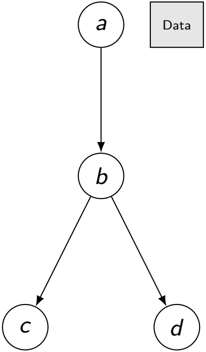


- **Previous work** proposed ways such as unwinding a switch into a ring.
- **Edge Splitting:** for each switch node w, iteratively choose edges $(u, w), (w, t)$ and replace them by (u, t) without sacrificing connectivity.
  - Originally used to prove connectivity properties of Eulerian graph. (Jackson,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



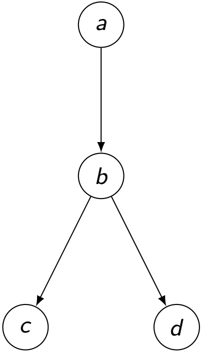
Pipeline Schedule

Non-Pipeline Schedule

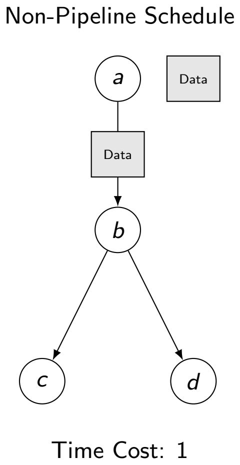


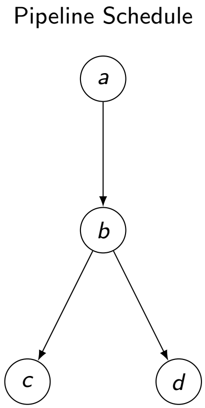
Time Cost: 0

Pipeline Schedule



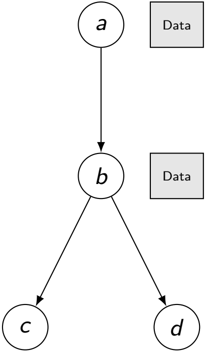
Pipeline Schedule





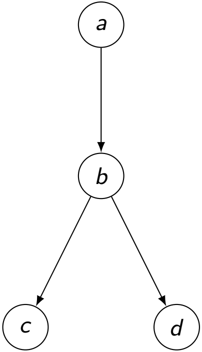
Pipeline Schedule

Non-Pipeline Schedule



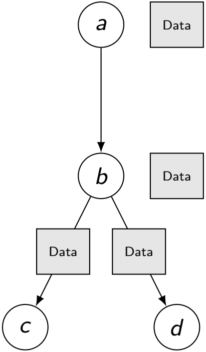
Time Cost: 1

Pipeline Schedule



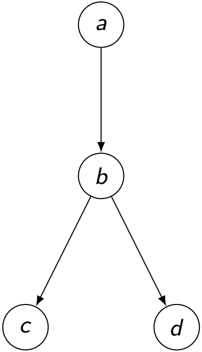
Pipeline Schedule

Non-Pipeline Schedule

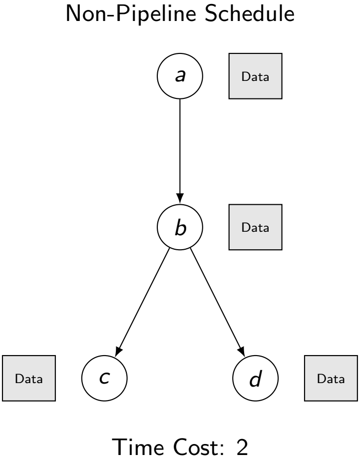


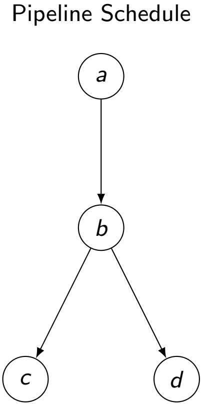
Time Cost: 2

Pipeline Schedule



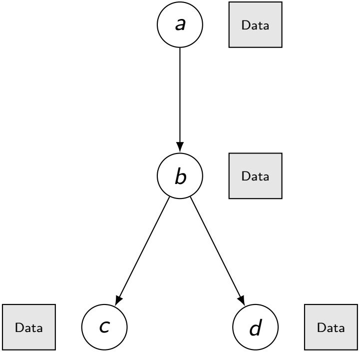
Pipeline Schedule





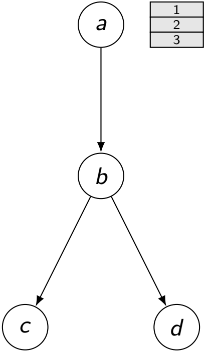
Pipeline Schedule

Non-Pipeline Schedule



Time Cost: 2

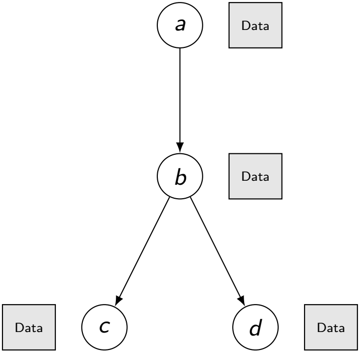
Pipeline Schedule



Time Cost: 0

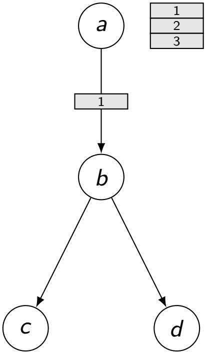
Pipeline Schedule

Non-Pipeline Schedule



Time Cost: 2

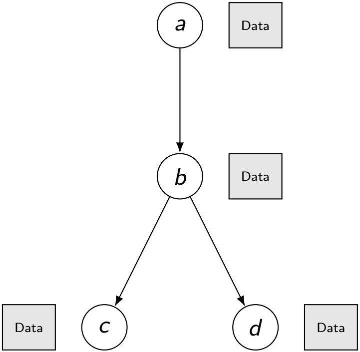
Pipeline Schedule



Time Cost: 1/3

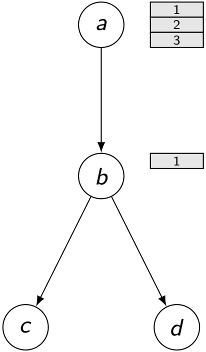
Pipeline Schedule

Non-Pipeline Schedule



Time Cost: 2

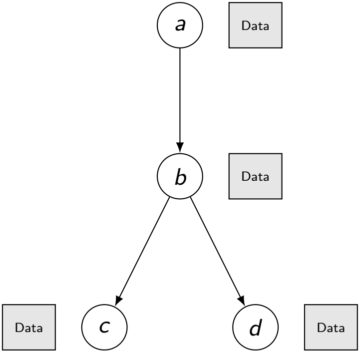
Pipeline Schedule



Time Cost: 1/3

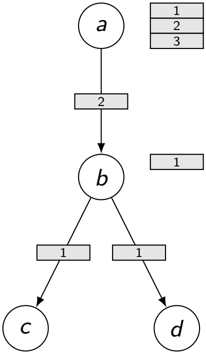
Pipeline Schedule

Non-Pipeline Schedule



Time Cost: 2

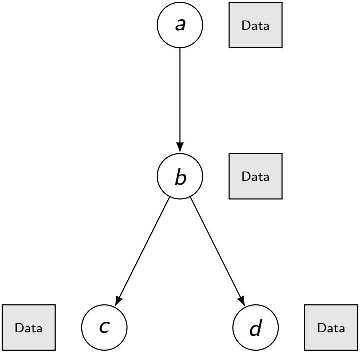
Pipeline Schedule



Time Cost: 2/3

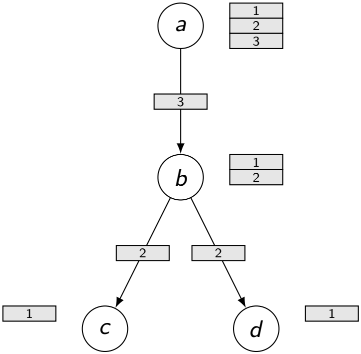
Pipeline Schedule

Non-Pipeline Schedule



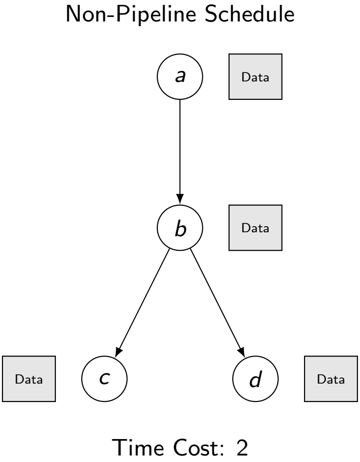
Time Cost: 2

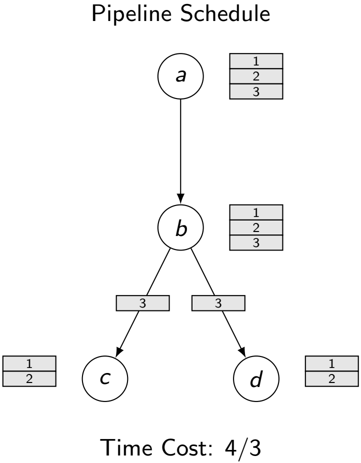
Pipeline Schedule



Time Cost: 3/3

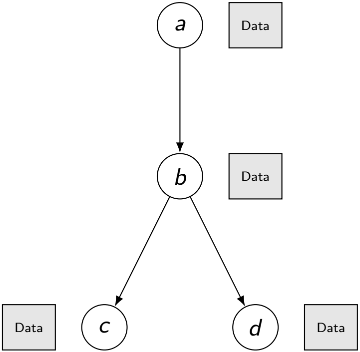
Pipeline Schedule





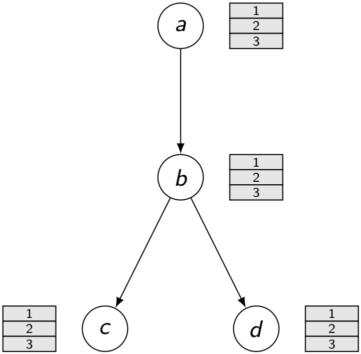
Pipeline Schedule

Non-Pipeline Schedule



Time Cost: 2

Pipeline Schedule

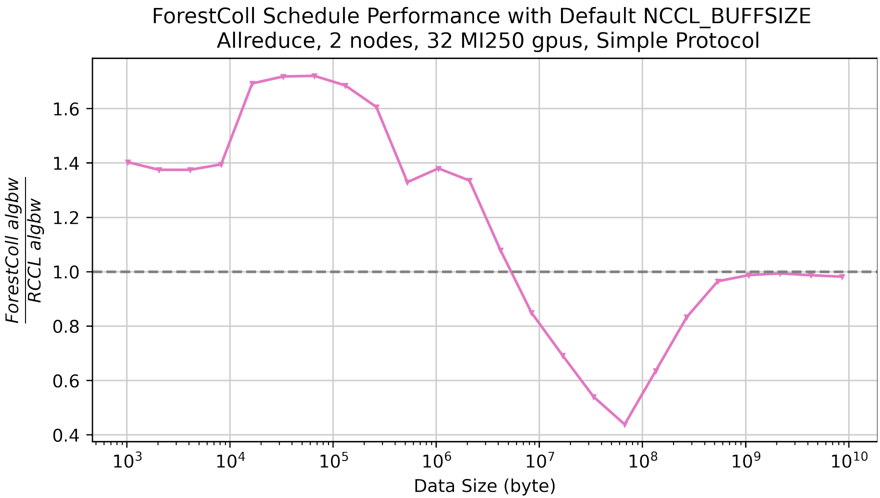


Time Cost: $4/3$

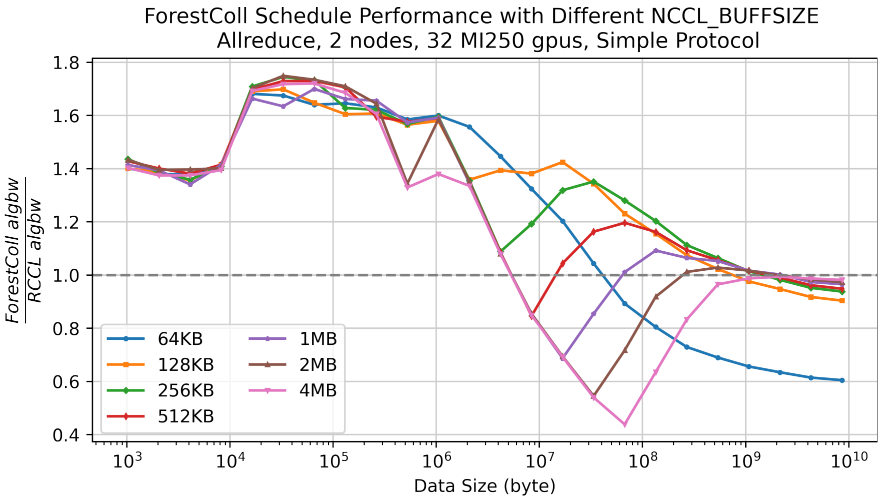
Chunk Size

- 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.

Chunk Size Experiment



Chunk Size Experiment



Thank you

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