Efficient Direct-Connect Topologies for Collective Communications

Liangyu Zhao[1] Siddharth Pal[2] Tapan Chugh[1] Weiyang Wang[3] Jason Fantl[2]
Prithwish Basu[2] Joud Khoury[2] Arvind Krishnamurthy[1]

[1]University of Washington

[2]Raytheon BBN

[3]MIT CSAIL

FOCI Talk, October 2023

Collective Communication

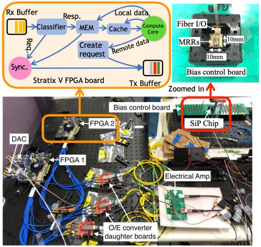
- **Collective Communication** refers to communication patterns in which a group of nodes in a parallel computing system exchange information.
  - e.g. broadcast, reduce, allreduce, all-to-all, etc.
- Originally a topic in high-performance computing, it is now extensively used for parameter synchronization in distributed ML training/inferencing, becoming a significant overhead.

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
$S_0^{(0)}$	$S_0^{(1)}$	$S_0^{(2)}$	$\bigoplus_i S_0^{(i)}$	$\bigoplus_i S_0^{(i)}$	$\bigoplus_i S_0^{(i)}$
$S_1^{(0)}$	$S_1^{(1)}$	$S_1^{(2)}$	$\bigoplus_i S_1^{(i)}$	$\bigoplus_i S_1^{(i)}$	$\bigoplus_i S_1^{(i)}$
$S_2^{(0)}$	$S_2^{(1)}$	$S_2^{(2)}$	$\bigoplus_i S_2^{(i)}$	$\bigoplus_i S_2^{(i)}$	$\bigoplus_i S_2^{(i)}$

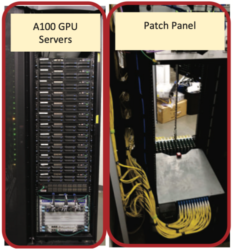
Optical Circuit Network

An emerging approach is to use **optical circuit network** to achieve higher bandwidth at reasonable capital expenditure and energy cost.

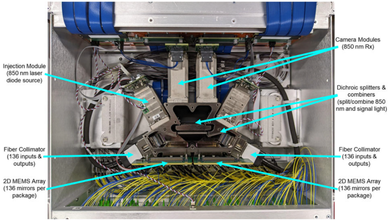
- In optical network, a node is **directly connected** to another node via optical circuit instead of electrical switch. Unconnected pair of nodes cannot communicate directly.
- Optical circuit has **high reconfiguration/rewiring latency**, necessitating a fixed topology during collective communication.



(a) SiP-ML (SIGCOMM '21)



(b) TopoOpt (NSDI '23)



(c) TPU v4 (Google)

Problem Statement

Given hardware and workload specifications, how to find a **topology** and a corresponding **communication schedule** that achieve the best collective communication performance?

Hardware Specifications:

- d: degree of topology (# of ports)
- b: bandwidth of link
- α: latency of send/recv

Workload Specifications:

- N: # of nodes
- M: size of data

Motivation

Observations:

- Coming up with a topology and communication schedule is hard at large scale.
- Direct search for either topology or schedule can easily be an intractable optimization problem.

Question

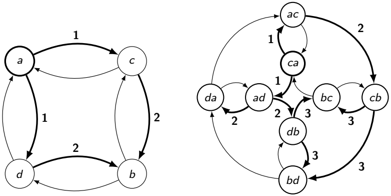
Can we design efficient topology and schedule at small scale first and then expand them to large scale?

Expansion Techniques

Given base topology and communication schedule,

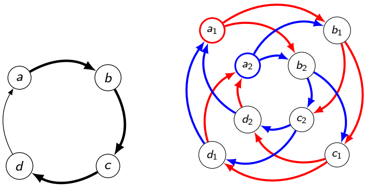
- We have graph transformations to expand the **base topology** into larger ones.
- The **base schedule** is also expanded to match the expanded topology.
- The sacrifice in **overall performance** is mathematically bounded during the process.

Line Graph Expansion:



Expanding N while maintaining the same d.

Degree Expansion:



Expanding N and d at the same time.

Observations:

- Different expansion techniques expand N and d differently and offer different performance trade-off (latency vs. bandwidth).
- We also have various base topologies and schedules for expansion.

Question

Given the target hardware and workload, how to derive the best topology and schedule?

Topology Finder

- Given a target topology size, the topology finder explores **possible base topologies and combinations of expansion techniques** to reach the target size.
- The resulting candidate topologies and schedules form a **Pareto-frontier**. The best one is then decided by hardware/workload specifications.

Expansion Techniques	# of Nodes	Deg	Moore	BW
Line Graph Exp $L^n(G)$	$d^n N$	d	✓	×
Degree Exp $G * n$	nN	nd	×	✓
Cartesian Power $G^{\square n}$	N^n	nd	×	✓
Cartesian Prod $G_1 \square \ldots \square G_n$	$\prod_i N_i$	$\sum_i d_i$	×	✓

Table: Summary of Expansion Techniques

Topology	T_L	T_B	$T_L + T_B$
$\Pi_{4,1024}$	10α	$2.664M/B$	323.5us
$L^3(C(16, \{3,4\}))$	12α	$2.039M/B$	291.0us
$L^2(\text{Diamond}^{\square 2})$	16α	$2.008M/B$	328.4us
$L(\text{DBJMod}(2,4)^{\square 2})$	22α	$2.000M/B$	387.8us
$(\text{UniRing}(1,4) \square \text{UniRing}(1,8))^{\square 2}$	40α	$1.998M/B$	567.6us
Theoretical Lower Bound	$\mathbf{10\alpha}$	$\mathbf{1.998M/B}$	**267.6us**

Table: Pareto-frontier for $N=1024$, $d=4$. The allreduce time T_L+T_B is computed with $\alpha=10\mu$s and $M/B=1$MB/100Gbps.

Motivation

Observations:

- Expansion techniques have huge gaps in the coverage of topology sizes.
  - Given a base topology with $N = 4, d = 2$, line graph expansion can only generate topologies of $8, 16, 32, \ldots$ ($d^n N$) number of nodes.
- There exist off-the-shelf topologies from graph theory with favorable characteristics (e.g. the low diameter of expander graphs).

Question

Given a topology, can we *efficiently* construct an *efficient* schedule for it?

Existing Solutions

Earlier work has explored ways to generate communication schedule for a given topology.

- SCCL (PPoPP '21) uses *satisfiability modulo theories* (SMT).
- TACCL (NSDI '23) uses *mixed integer linear program* (MILP).
- **Poor Scalability:** both involve *NP-hard* optimization.

Conclusion: At large sizes, existing solutions either take too long to generate schedule or fail to generate one.

# of nodes	4	8	16	32	64
SCCL	0.59s	0.86s	21.4s	$>10^4$s	$>10^4$s
TACCL	0.50s	7.39s	1801s	1802s	n/a

Table: Generation Time on Hypercube

# of nodes	4	9	16	25	36
SCCL	0.61s	1.00s	60s	3286s	$>10^4$s
TACCL	0.45s	67.8s	1801s	1802s	n/a

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

Breadth-First-Broadcast (BFB) Schedule

We enforce *Breadth-First-Broadcast* (BFB) for allgather schedule generation. We aim to find the best schedule **among all BFB schedules instead of all possible schedules.**

- **Advantage:** The scheduling problem can be formulated as a *linear program*, which can be efficiently solved in *polynomial time*.
- Although BFB does not guarantee optimality in an arbitrary topology, it is proven to generate optimal schedules for many topologies with inherent symmetry.
  - e.g. torus, hypercube, and twisted torus used by TPU v4.

$$
\begin{aligned}
\text{minimize} \quad & U_{u,t} \\
\text{subject to} \quad & \sum_{v} x_{v,(w,u),t} \le U_{u,t}, && \forall w \in N^{-}(u) \\
& \sum_{w} x_{v,(w,u),t} = 1, && \forall v \in N_t^{-}(u) \\
& 0 \le x_{v,(w,u),t} \le 1. && \forall w, v
\end{aligned}
$$

Figure: BFB Linear Program Formulation

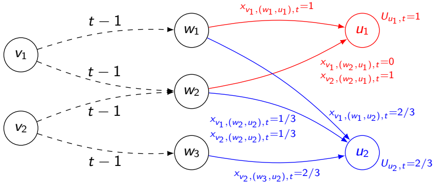


Figure: BFB Example

BFB vs Existing Solutions

Conclusion: BFB schedule generation is orders of magnitude faster than previous work.

# of nodes	4	8	16	32	64	**1024**
SCCL	0.59s	0.86s	21.4s	$>10^4$s	$>10^4$s	$>10^4$s
TACCL	0.50s	7.39s	1801s	1802s	n/a	n/a
BFB	**<0.01s**	**<0.01s**	**<0.01s**	**0.03s**	**0.17s**	**52.7s**

Table: Generation Time on Hypercube

# of nodes	4	9	16	25	36	**2500**
SCCL	0.61s	1.00s	60s	3286s	$>10^4$s	$>10^4$s
TACCL	0.45s	67.8s	1801s	1802s	n/a	n/a
BFB	**<0.01s**	**<0.01s**	**<0.01s**	**0.01s**	**0.03s**	**61.1s**

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

Direct-Connect Optical Testbed

- 12 servers, each with an NVIDIA A100 GPU.
- 100 Gbps HP NIC, configured as 4x25Gbps breakout interfaces.
- Topology is reconfigurable via a *Telescent* optical patch panel.

(a) A100 Servers

(b) Optical Patch Panel

Conclusion: Our topologies consistently outperform baselines across all topology sizes N and allreduce data sizes M.

N	**Topology**
5	Complete Graph: K_5
6	Degree Expansion of Complete graph: $K_3 * 2$
7	Circulant Graph: $C(7, \{2, 3\})$
8	Complete Bipartite Graph: $K_{4,4}$
9	Hamming Graph: $H(2, 3)$
10	Degree Exp of BFB Bidirectional Ring: BiRing$(2, 5) * 2$
11	Circulant Graph: $C(11, \{2, 3\})$
12	Circulant Graph: $C(12, \{2, 3\})$

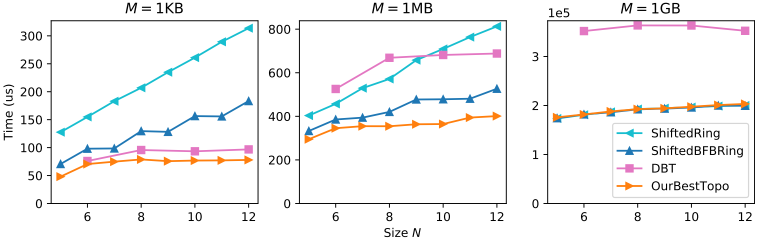
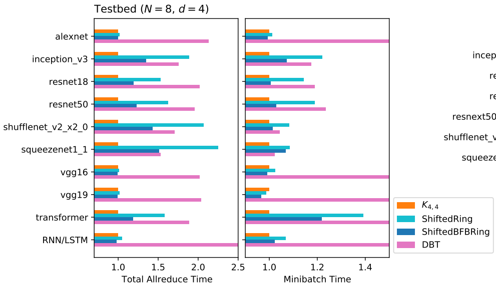


Figure: Comparing allreduce performance of shifted rings, double binary trees (DBT), and our best bidirectional topologies from Pareto-frontier at degree 4.

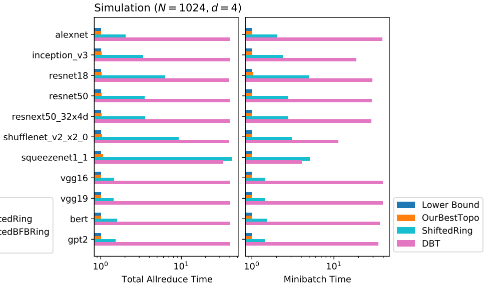
Conclusion: Our topologies speed up DNN training, especially at large scale.

Average improvements over the closest baseline:

	8-node Experiment	1024-node Simulation
Total Allreduce Time	30%	6.7×
Minibatch Time	10%	2.6×



(a) 8-node optical testbed training results.



(b) 1024-node simulated training results.

Frontera Supercomputer

- Located at the *Texas Advanced Computing Center* (TACC).
- 396 Intel Xeon CPU nodes in a 6D torus topology. We used up to 54-node 4D torus.
- Each with a Rockport NC1225 network card, capable of 25 Gbps per link.

Figure: Frontera Supercomputer

Conclusion: BFB torus schedules achieve top performance in all torus constructions. Traditional torus schedule from HPC performs well only in torus with equal dimensions.

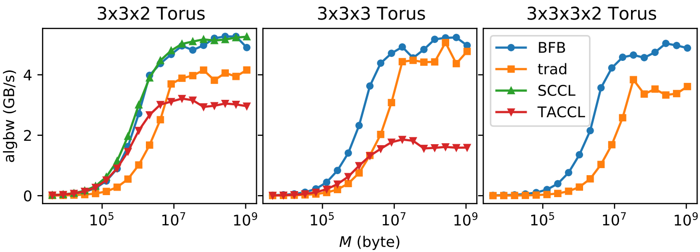


Figure: Comparing allreduce performance of torus schedules generated by BFB, traditional torus scheduling, SCCL, and TACCL.

Contributions

- **Expansion techniques** for synthesizing large-scale collective communication topologies and schedules.
- A polynomial-time **schedule generation** for large-scale network topologies.
- A **topology finder** to generate Pareto-efficient topologies and schedules for target hardware and workload.
- A **compiler** for lowering communication schedules to runtime.

Thank you

arXiv: https://arxiv.org/abs/2202.03356