Collective Communnication Algorithms (Part-2)
CS 422 / CS 536

Vamsi Addanki

Purdue University

vaddank@purdue.edu

March 31, 2026

Overview

Allgather

Setting:

- n nodes (ranks). Nodes can represent processes, GPUs, etc.
- Node i initially holds block ii, each block is of size $\frac{m}{n}$, where m is the message size.

Goal: Each node obtains $\{00, 11, \ldots, (n-1)(n-1)\}$

Node	Blocks
0	00
1	11
2	22
3	33
4	44
5	55
6	66
7	77

$\Longrightarrow$

Node	Blocks
0	00 11 22 33 44 55 66 77
1	00 11 22 33 44 55 66 77
2	00 11 22 33 44 55 66 77
3	00 11 22 33 44 55 66 77
4	00 11 22 33 44 55 66 77
5	00 11 22 33 44 55 66 77
6	00 11 22 33 44 55 66 77
7	00 11 22 33 44 55 66 77

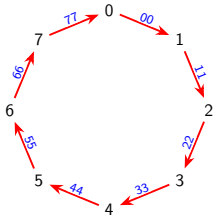
Ring Allgather

Algorithm:

- For $s = 0, 1, \ldots, n-2$:
  - Node $i \rightarrow (i+1) \bmod n$
  - Sends block $((i-s) \bmod n)\,((i-s) \bmod n)$
- After step s, each node has $s+2$ blocks

Completion: $n-1$ steps

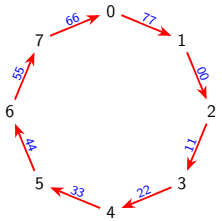
Ring Allgather: Step 0



Distance: 1
Data: 1 blocks
Cost: $\alpha + \beta \cdot \frac{m}{n}$

Node	Blocks
0	00 77
1	11 00
2	22 11
3	33 22
4	44 33
5	55 44
6	66 55
7	77 66

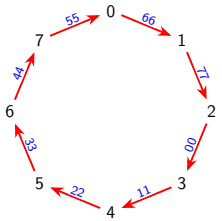
Ring Allgather: Step 1



Distance: 1
Data: 1 blocks
Cost: $\alpha + \beta \cdot \frac{m}{n}$

Node	Blocks
0	00 77 66
1	11 00 77
2	22 11 00
3	33 22 11
4	44 33 22
5	55 44 33
6	66 55 44
7	77 66 55

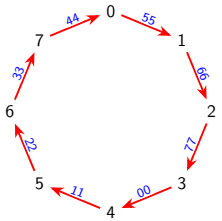
Ring Allgather: Step 2



Distance: 1
Data: 1 blocks
Cost: $\alpha + \beta \cdot \frac{m}{n}$

Node	Blocks
0	00 77 66 55
1	11 00 77 66
2	22 11 00 77
3	33 22 11 00
4	44 33 22 11
5	55 44 33 22
6	66 55 44 33
7	77 66 55 44

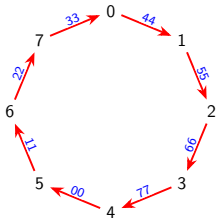
Ring Allgather: Step 3



Distance: 1
Data: 1 blocks
Cost: $\alpha + \beta \cdot \frac{m}{n}$

Node	Blocks
0	00 77 66 55 44
1	11 00 77 66 55
2	22 11 00 77 66
3	33 22 11 00 77
4	44 33 22 11 00
5	55 44 33 22 11
6	66 55 44 33 22
7	77 66 55 44 33

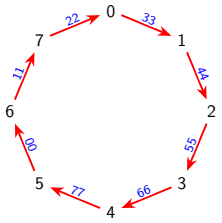
Ring Allgather: Step 4



Distance: 1
Data: 1 blocks
Cost: $\alpha + \beta \cdot \frac{m}{n}$

Node	Blocks
0	00 77 66 55 44 33
1	11 00 77 66 55 44
2	22 11 00 77 66 55
3	33 22 11 00 77 66
4	44 33 22 11 00 77
5	55 44 33 22 11 00
6	66 55 44 33 22 11
7	77 66 55 44 33 22

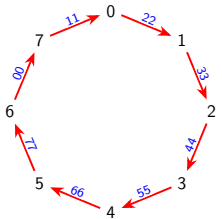
Ring Allgather: Step 5



Distance: 1
Data: 1 blocks
Cost: $\alpha + \beta \cdot \frac{m}{n}$

Node	Blocks
0	00 77 66 55 44 33 22
1	11 00 77 66 55 44 33
2	22 11 00 77 66 55 44
3	33 22 11 00 77 66 55
4	44 33 22 11 00 77 66
5	55 44 33 22 11 00 77
6	66 55 44 33 22 11 00
7	77 66 55 44 33 22 11

Ring Allgather: Step 6



Distance: 1
Data: 1 blocks
Cost: $\alpha + \beta \cdot \frac{m}{n}$

Node	Blocks
0	00 11 22 33 44 55 66 77
1	00 11 22 33 44 55 66 77
2	00 11 22 33 44 55 66 77
3	00 11 22 33 44 55 66 77
4	00 11 22 33 44 55 66 77
5	00 11 22 33 44 55 66 77
6	00 11 22 33 44 55 66 77
7	00 11 22 33 44 55 66 77

Ring AllGather α-β cost

Total cost: $(n-1)\cdot\alpha+\beta\cdot\frac{n-1}{n}\cdot m$

$$\sum_{s=0}^{n-2}\alpha+\beta\cdot\frac{m}{n}=(n-1)\cdot\alpha+\beta\cdot\frac{n-1}{n}\cdot m$$

Recursive Doubling (Original) Allgather

Algorithm:

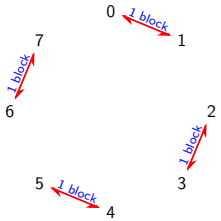
- For $s = 0, 1, \ldots, \log n - 1$:
  - Node $i \rightarrow i \oplus 2^s$
    - Distance doubling
  - Exchange all blocks accumulated so far
    - Data doubling
- After step s, each node has 2^{s+1} blocks

Completion: $\log n$ steps

Visualization: `https://www.youtube.com/watch?v=9KXS5g8M3BQ`

- By Christina Zhang @ Purdue

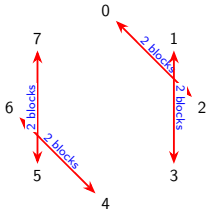
Recursive Doubling: Step 0



Distance: 1
Data: 1 blocks
Cost: $\alpha + \beta \cdot \frac{m}{n}$

Node	Blocks
0	00 11
1	11 00
2	22 33
3	33 22
4	44 55
5	55 44
6	66 77
7	77 66

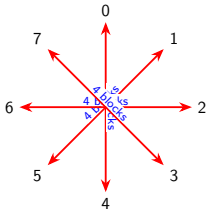
Recursive Doubling: Step 1



Distance: 2
Data: 2 blocks
Cost: $\alpha + \beta \cdot 2 \cdot \frac{m}{n}$

Node	Blocks
0	00 11 22 33
1	11 00 33 22
2	22 33 00 11
3	33 22 11 00
4	44 55 66 77
5	55 44 77 66
6	66 77 44 55
7	77 66 55 44

Recursive Doubling: Step 2



Distance: 4
Data: 4 blocks
Cost: $\alpha + \beta \cdot 4 \cdot \frac{m}{n}$

Node	Blocks
0	00 11 22 33 44 55 66 77
1	11 00 33 22 55 44 77 66
2	22 33 00 11 66 77 44 55
3	33 22 11 00 77 66 55 44
4	44 55 66 77 00 11 22 33
5	55 44 77 66 11 00 33 22
6	66 77 44 55 22 33 00 11
7	77 66 55 44 33 22 11 00

Recursive Doubling (Original) AllGather α-β cost

Total cost: $\log_2(n) \cdot \alpha + \beta \cdot \frac{n-1}{n} \cdot m$

$$\sum_{s=0}^{\log_2(n)-1} \alpha + \beta \cdot 2^s \cdot \frac{m}{n} = \log_2(n) \cdot \alpha + \beta \cdot \frac{n-1}{n} \cdot m$$

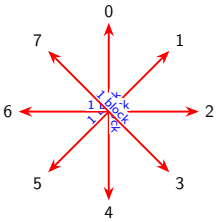
Recursive Doubling (Distance Halving) Allgather

Algorithm:

- For $s = 0, 1, \ldots, \log n - 1$:
  - Node $i \rightarrow i \oplus 2^{(\log n - 1 - s)}$
    - Distance halving
  - Exchange all blocks accumulated so far
    - Data doubling
- After step s, each node has 2^{s+1} blocks

Completion: $\log n$ steps

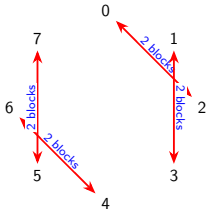
Recursive Doubling (Halving): Step 0



Distance: 4
Data: 1 blocks
Cost: $\alpha + \beta \cdot \frac{m}{n}$

Node	Blocks
0	00 44
1	11 55
2	22 66
3	33 77
4	44 00
5	55 11
6	66 22
7	77 33

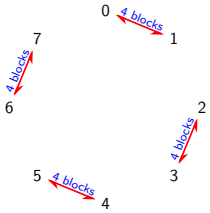
Recursive Doubling (Halving): Step 1



Distance: 2
Data: 2 blocks
Cost: $\alpha + \beta \cdot 2 \cdot \frac{m}{n}$

Node	Blocks
0	00 44 22 66
1	11 55 33 77
2	22 66 00 44
3	33 77 11 55
4	44 00 66 22
5	55 11 77 33
6	66 22 44 00
7	77 33 55 11

Recursive Doubling (Halving): Step 2



Distance: 1
Data: 4 blocks
Cost: $\alpha + \beta \cdot 4 \cdot \frac{m}{n}$

Node	Blocks
0	00 11 22 33 44 55 66 77
1	00 11 22 33 44 55 66 77
2	00 11 22 33 44 55 66 77
3	00 11 22 33 44 55 66 77
4	00 11 22 33 44 55 66 77
5	00 11 22 33 44 55 66 77
6	00 11 22 33 44 55 66 77
7	00 11 22 33 44 55 66 77

Recursive Doubling (Halving/Doubling) AllGather α-β cost

Total cost: $\log_2(n) \cdot \alpha + \beta \cdot \frac{n-1}{n} \cdot m$

$$\sum_{s=0}^{\log_2(n)-1} \alpha + \beta \cdot 2^s \cdot \frac{m}{n} = \log_2(n) \cdot \alpha + \beta \cdot \frac{n-1}{n} \cdot m$$

- **Note:** We have not accounted for the distance of communication, and so the cost resembles the original recursive doubling algorithm.

Observation

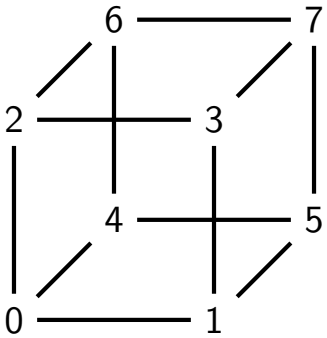
- Recursive doubling communicates with nodes at distances 2^s
- This corresponds to flipping one bit if nodes are labelled as bitstrings
  - **Hypercube!**
- The communication pattern is a rolled-out hypercube: **Butterfly**

Hypercube Topology:

- Represent node i using $\log n$-bit bitstring
- Two nodes are connected if their labels differ in exactly one bit
- Each step of recursive doubling uses one dimension

Optimality:

- All communication happens along **direct edges** (distance $= 1$)
- No multi-hop routing needed


6
7
2
3
4
5
0
1

Cyclic Recursive Doubling

- A cyclic permutation of the original recursive doubling algorithm
- Communicate with node at distances $2^{(\log n - 1 - s)}$ at step s
- Useful in directed network topologies e.g., Photonic interconnects

Visualization: `https://www.youtube.com/watch?v=ZElFv_NN3Vw`

- By Christina Zhang @ Purdue

Bruck's Allgather (Radix r) I

Setting:

- $n = r^k$ nodes, labeled $0, \ldots, n-1$
- Node i initially holds block ii

Algorithm:

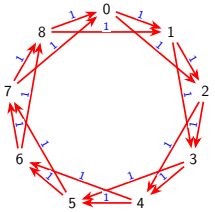
- Perform a cyclic shift so node i holds data starting at index i
- For $s = 0, 1, \ldots, \log_r n - 1$:
  - Node i sends to: $i + j \cdot r^s \mod n, \qquad j = 1, \ldots, r-1$
  - Send r^s contiguous blocks to each neighbor
  - Each step communicates along offsets of size r^s (radix-r expansion)
- Perform inverse cyclic shift to restore order

Bruck's Allgather (Radix r) II

Completion: $\log_r n$ steps
Visualization: `https://www.youtube.com/watch?v=40IUuxmYPaU`

- By Christina Zhang @ Purdue

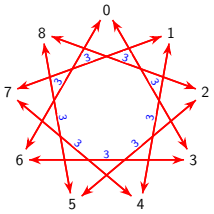
Bruck's Allgather ($r = 3$): Step 0



Offset: 1
Neighbors: 2
Data: 2 blocks
Cost: $\alpha + \beta \cdot 2 \cdot \frac{m}{n}$

Node	Blocks
0	00 77 88
1	11 88 00
2	22 00 11
3	33 11 22
4	44 22 33
5	55 33 44
6	66 44 55
7	77 55 66
8	88 66 77

Bruck's Allgather ($r = 3$): Step 1



Offset: 3
Neighbors: 2
Data: 6 blocks
Cost: $\alpha + \beta \cdot 6 \cdot \frac{m}{n}$

Node	Blocks
0	00 77 88 33 11 22 66 44 55
1	11 88 00 44 22 33 77 55 66
2	22 00 11 55 33 44 88 66 77
3	33 11 22 66 44 55 00 77 88
4	44 22 33 77 55 66 11 88 00
5	55 33 44 88 66 77 22 00 11
6	66 44 55 00 77 88 33 11 22
7	77 55 66 11 88 00 44 22 33
8	88 66 77 22 00 11 55 33 44

Bruck's AllGather α-β cost

Total cost: $\log_r(n) \cdot \alpha + \beta \cdot \frac{n-1}{n} \cdot m$

$$\sum_{s=0}^{\log_r(n)-1} \alpha + \beta \cdot r^s \cdot \frac{m}{n} = \log_r(n) \cdot \alpha + \beta \cdot \frac{n-1}{n} \cdot m$$

- **Note:** We have not accounted for the distance of communication, and so the cost resembles the original recursive doubling algorithm.

Revisiting the Cost Model

- The goal of a cost model for collective communication algorithms is to estimate the **completion time**
- We know from network latencies lecture:
  - Propagation delay
  - Transmission delay
  - Queueing delay or congestion due to bandwidth sharing
  - Store-and-foward, cut-through switching
  - For collectives: Any other setup or synchronization delay

Hockney's α-β Cost Model

- α: Latency or startup time, independent of message size
- β: Time per byte or inverse of bandwidth
- m: Message size in bytes
- Time to complete a step with L bytes of data transmission is estimated as: $\alpha + \beta \cdot L$
- Key assumption: **No congestion and no propagation delay**, equi-distant nodes, and perfect synchronization

Accounting for Congestion and Propagation Delays

- α: Startup or initialization time for each step e.g., kernel launch or send/recv overheads, independent of message size
- β: Time per byte or inverse of bandwidth
- δ: Link propagation delay
- d: Distance of communication in terms of hops or links

Cost of a step: $\underbrace{\alpha}_{\textit{Setup latency}} + \underbrace{d \cdot \delta}_{\textit{Propagation delay}} + \underbrace{\beta \cdot L \cdot d}_{\textit{Transmission delay}}$

- $\beta \cdot L$: Transmission delay to serialize L bytes
- $\beta \cdot L \cdot d$: Tranmission delay when d messages overlap and share bandwidth

Relation to Concurrent Flow

- G: The network topology
- M: A matrix specifying the communication pattern in each step
- $\theta(M, G)$: The maximum concurrent flow or throughput for the communication pattern M on the network G
  - Concurrent tells us the maximum fraction of the traffic matrix that can be *simultaneously* be routed through the network within capacity constraints
  - **Observation**: Concurrent flow is the effective fraction of bandwidth available to each communication step, and thus β should be scaled by $\frac{1}{\theta(M,G)}$ to account for congestion and bandwidth sharing

Cost of a step: $\underbrace{\alpha}_{\text{Setup latency}} + \underbrace{d \cdot \delta}_{\text{Propagation delay}} + \underbrace{\beta \cdot L \cdot \frac{1}{\theta(M, G)}}_{\text{Transmission delay}}$

We are Still Sloppy!

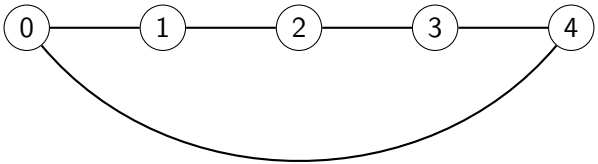
- Packet headers
- Store-and-forward, cut-through switching (we assumed cut-through so far)
- Synchronization overheads
- Reduction computation overheads
- Multiple channels e.g., used in NCCL
- Non-uniform link bandwidths
- Non-uniform link latencies
- ...
- You know the drill: Count every latency along the way! :)

AllGather on Torus

- Motivation: Google's TPUv4 uses 3-D Torus topology
- Data/Model/Pipeline parallelism along each dimension of the physical topology
- Collective communication: Either on a single dimension, or all dimensions

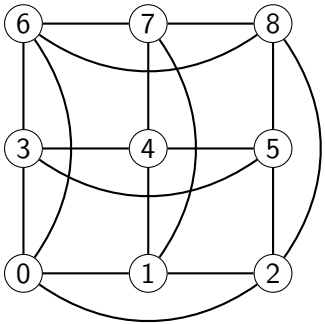
1D Torus

- Bi-directional links
- n nodes in a single dimension



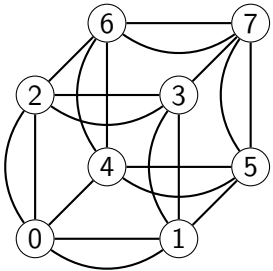
2D Torus

- Bi-directional links
- n nodes arrange in two dimensions: $n = X \times Y$, where X and Y are the number of nodes in each dimension e.g., 3×3



3D Torus

- Bi-directional links
- n nodes arrange in three dimensions: $n = X \times Y \times Z$, where X, Y, and Z are the number of nodes in each dimension e.g., $2 \times 2 \times 2$



Swing Allgather

Setting:

- n nodes arranged in a ring
- Node r initially holds block rr

Algorithm:

- Let $\rho(s)$ be defined as follows:

$$\rho(s) = \sum_{i=0}^{s} (-2)^i$$

- For the reduce-scatter peer selection, a node r communicates with $\pi(r, s)$ in step s

$$\pi(r, s) = \begin{cases} r + \rho(s) \bmod n & \text{if } r \text{ is even} \\ r - \rho(s) \bmod n & \text{if } r \text{ is odd} \end{cases}$$

Swing Allgather

- Swing allgather uses the *same peer function in reverse order*
- For $s = 0, 1, \ldots, \log_2 n - 1$, node r communicates with

$$\pi(r, \log_2 n - 1 - s)$$

- At step s, node r sends the blocks gathered so far

Swing Allgather

Key Property:

- Communication distance is $|\rho(s)|$
- For reduce-scatter, the distances are: 1, 1, 3, 5, ...
- For *allgather*, the order is reversed

For $n = 8$: distances $= 3,\ 1,\ 1$

Completion:

- $\log_2 n$ steps

Visualization: `https://www.youtube.com/watch?v=mu4mS5idI-c`

- By Christina Zhang @ Purdue

Swing Allgather: Block Selection

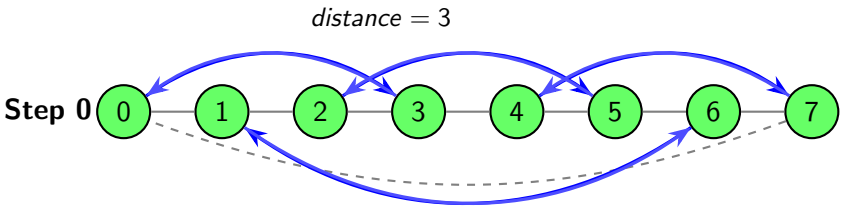
Recursive block selection:

function GetAGIdxs($r, step, n, blocks$)
  if $step \geq \log_2 n$ **then return**
  end if
  for $s = step$ to $\log_2 n - 1$ **do**
    $peer \leftarrow \pi(r, \log_2 n - 1 - s)$
    $blocks[peer] \leftarrow 1$ ▷ direct
    GetAGIdxs($peer, s + 1, n, blocks$) ▷ future
  end for
end function

Allgather:

for $s = 0$ to $\log_2 n - 1$ **do**
  $peer \leftarrow \pi(r, \log_2 n - 1 - s)$
  $send \leftarrow [0]^n$, $recv \leftarrow [0]^n$
  GetAGIdxs($r, s, n, send$)
  GetAGIdxs($peer, s, n, recv$)
  sendrecv($peer, data, send, recv$)
end for

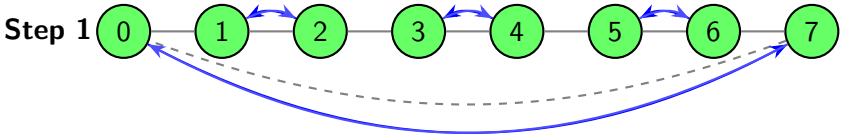
Swing Allgather: Step 0



Node	Blocks
0	00 33
1	11 66
2	22 55
3	33 00
4	44 77
5	55 22
6	66 11
7	77 44

Swing Allgather: Step 1

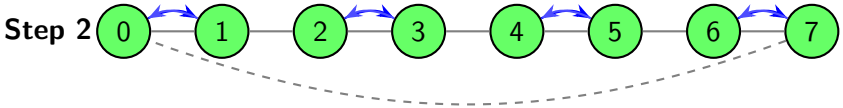
$distance = 1$



Node	Blocks
0	00 33 77 44
1	11 66 22 55
2	22 55 11 66
3	33 00 44 77
4	44 77 33 00
5	55 22 66 11
6	66 11 55 22
7	77 44 00 33

Swing Allgather: Step 2

$distance = 1$



Node	Blocks
0	00 33 77 44 11 66 22 55
1	11 66 22 55 00 33 77 44
2	22 55 11 66 33 00 44 77
3	33 00 44 77 22 55 11 66
4	44 77 33 00 55 22 66 11
5	55 22 66 11 44 77 33 00
6	66 11 55 22 77 44 00 33
7	77 44 00 33 66 11 55 22

Trivance Allgather

Setting:

- n nodes arranged in a bidirectional ring
- Node r initially holds reduced block rr

Algorithm:

- For the Trivance Reduce-Scatter communication pattern, at step k node r communicates with

$$\pi(r, k, n) = (\pi_{\text{left}}, \pi_{\text{right}})$$

where

$$\pi_{\text{left}} = r - 3^k \bmod n, \qquad \pi_{\text{right}} = r + 3^k \bmod n$$

- In the bandwidth-optimal variant, Allgather runs in the *reverse* order of Reduce-Scatter
- At each step, node r broadcasts the subset of reduced blocks needed by its two peers

Trivance Allgather

Key Property:

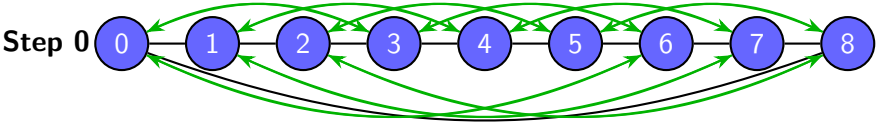
- Reduce-Scatter uses distances: 1, 3, 9, ... i.e., distance tripling
- Allgather uses the reverse order
- For $n = 9$: distances $= 3, 1$
- Data size triples at each step during Allgather
  - Similar in spirit to Recursive Halving/Doubling, instead tripling
  - Communication in both directions simultaneously

Completion:

- $\log_3 n$ steps

Trivance Allgather ($n = 9$): Step 0

$\frac{m}{9}$ bytes

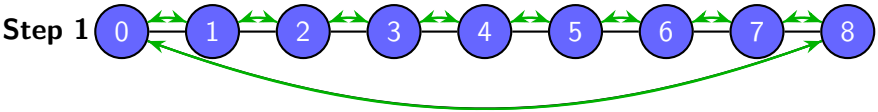


Max. congestion = 3

Node	Blocks
0	00 33 66
1	11 44 77
2	22 55 88
3	33 66 00
4	44 77 11
5	55 88 22
6	66 00 33
7	77 11 44
8	88 22 55

Trivance Allgather ($n = 9$): Step 1

$\frac{m}{3}$ bytes



Max. congestion = 1

Node	Blocks
0	00 33 66 11 44 77 22 55 88
1	11 44 77 22 55 88 00 33 66
2	22 55 88 33 66 00 11 44 77
3	33 66 00 44 77 11 22 55 88
4	44 77 11 55 88 22 33 66 00
5	55 88 22 66 00 33 44 77 11
6	66 00 33 77 11 44 55 88 22
7	77 11 44 88 22 55 66 00 33
8	88 22 55 00 33 66 77 11 44

AllReduce

Recall:

- Reduce-scatter is the reverse of AllGather
- AllReduce is a sequence of Reduce-scatter and AllGather

Latency-optimal versions: Reduce-scatter and then AllGather can be avoided by transmitting the entire message in each step, and performing the reduction on the fly.

- A factor of 2 speedup in terms of the number of steps

Further Reading I

[1] Thakur R, Rabenseifner R, Gropp W. Optimization of collective communication operations in MPICH. The International Journal of High Performance Computing Applications. 2005 Feb;19(1):49-66.

[2] Pješivac-Grbović J, Angskun T, Bosilca G, Fagg GE, Gabriel E, Dongarra JJ. Performance analysis of MPI collective operations. Cluster Computing. 2007 Jun;10(2):127-43.

[3] Chan E, Heimlich M, Purkayastha A, Van De Geijn R. Collective communication: theory, practice, and experience. Concurrency and Computation: Practice and Experience. 2007 Sep 10;19(13):1749-83.

[4] Bruck J, Ho CT, Kipnis S, Weathersby D. Efficient algorithms for all-to-all communications in multi-port message-passing systems. InProceedings of the sixth annual ACM symposium on Parallel algorithms and architectures 1994 Aug 1 (pp. 298-309).

Further Reading II

[5] De Sensi D, Bonato T, Saam D, Hoefler T. Swing: Short-cutting rings for higher bandwidth allreduce. In21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24) 2024 (pp. 1445-1462).

[6] Juerss A, Addanki V, Schmid S. Trivance: Latency-Optimal AllReduce by Shortcutting Multiport Networks. arXiv preprint arXiv:2602.17254. 2026 Feb 19.

[7] Addanki V. When Light Bends to the Collective Will: A Theory and Vision for Adaptive Photonic Scale-up Domains. InProceedings of the 24th ACM Workshop on Hot Topics in Networks 2025 Nov 17 (pp. 326-334).

[8] Rahman M, Joseph S, Kodkani N, Arzani B, Addanki V. Harvest: Adaptive Photonic Switching Schedules for Collective Communication in Scale-up Domains. arXiv preprint arXiv:2602.09188. 2026 Feb 9.

The End