Data Parallelism in LLM Training

Spring 2026

Lecturer: Yuedong (Steven) Xu

Fudan University

ydxu@fudan.edu.cn

Disclaimer

Machine learning systems is a broad and rapidly evolving field. The course material has been developed using a broad spectrum of resources, including research papers, lecture slides, blogposts, research talks, tutorial videos, and other materials shared by the research community. We sincerely appreciate their efforts and assistance, and try our best to cite the sources of the materials used in this course.

Distributed LLM Training: Outline

- Data Parallelism
  - Parameter-Server
  - **All-Reduce**
  - Memory optimization

Collective Communication Basics

- Collective Communication

> *"**Collective communication** is communication that involves a group of processing elements (termed nodes in this entry) and affects a data transfer between all or some of these processing elements. Data transfer may include the application of a reduction operator or other transformation of the data." 《Encyclopedia of Parallel Computing》*

集合通信是指一个进程组的所有进程都参与的全局通信操作。

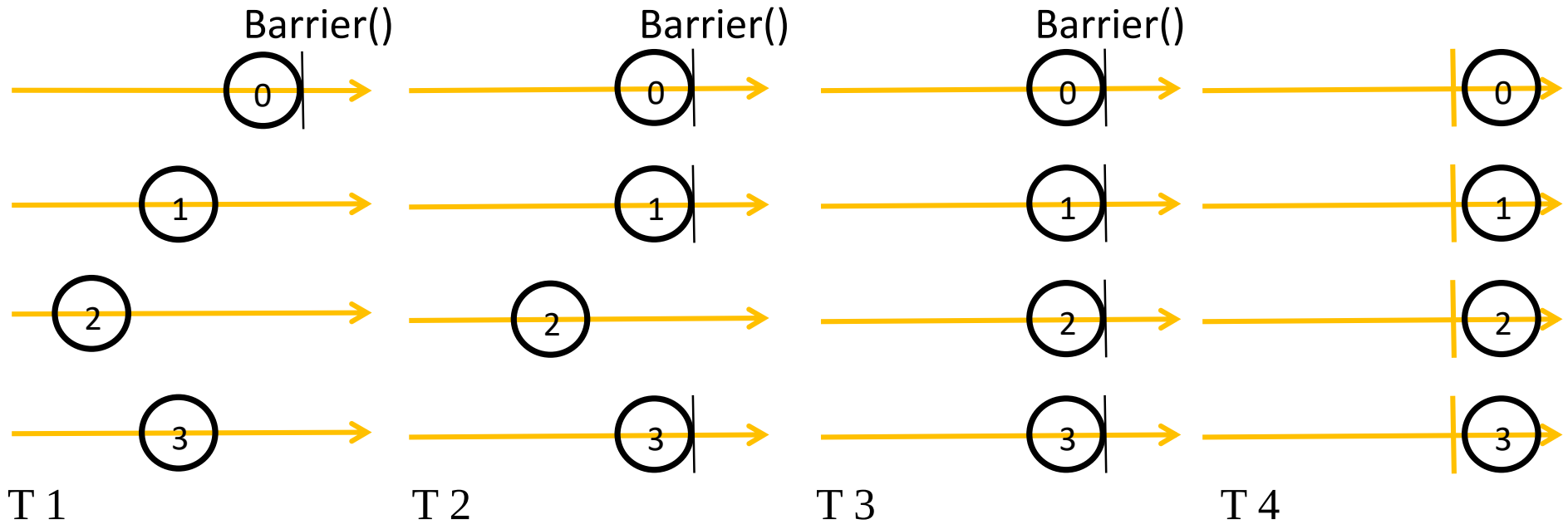
全部点对点通信完成才算集合通信完成

Collective Communication Basics

- Why Collective Communication
  - Simplified Programming Interface
    - Developers don't need to manually code complex synchronization and data distribution logic for every scenario.
  - Scalability for Large-Scale Systems
    - As systems grow to thousands of nodes or GPUs (e.g., in supercomputers or cloud clusters), managing communication becomes exponentially complex. These libraries support sparse data handling, fault tolerance, and observability features to ensure reliable operation at scale.
  - Decompose Compute and Communication
    - Allowing machine learning researchers and system engineers to work on their own.

Collective Communication Basics

- Collective Communication: Basic Operations
  - **SEND**, **RECEIVE**, COPY, BARRIER, SIGNAL+WAIL (in Message Passing Interface, i.e. MPI)

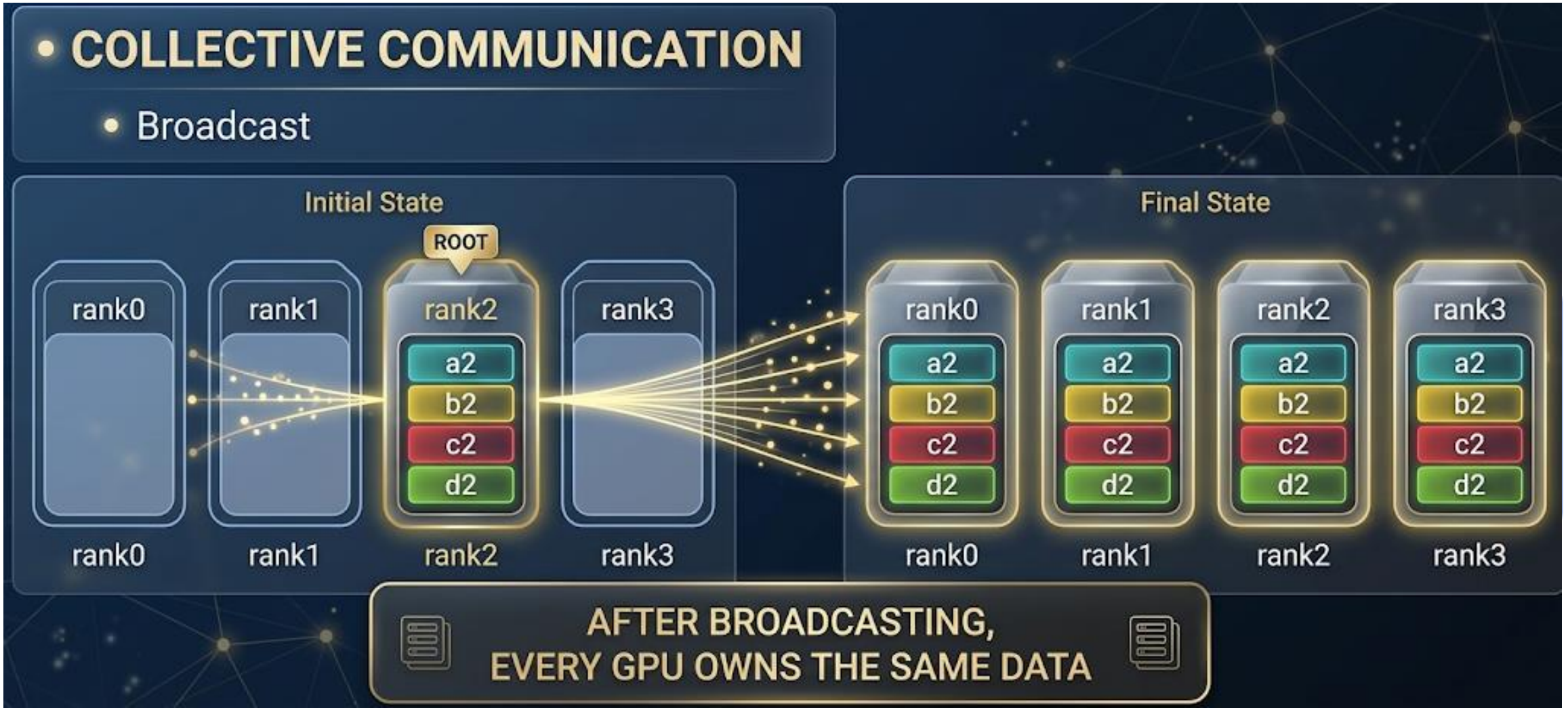


* Pictures replotted, but originated from Mengyuan @ Zhihu

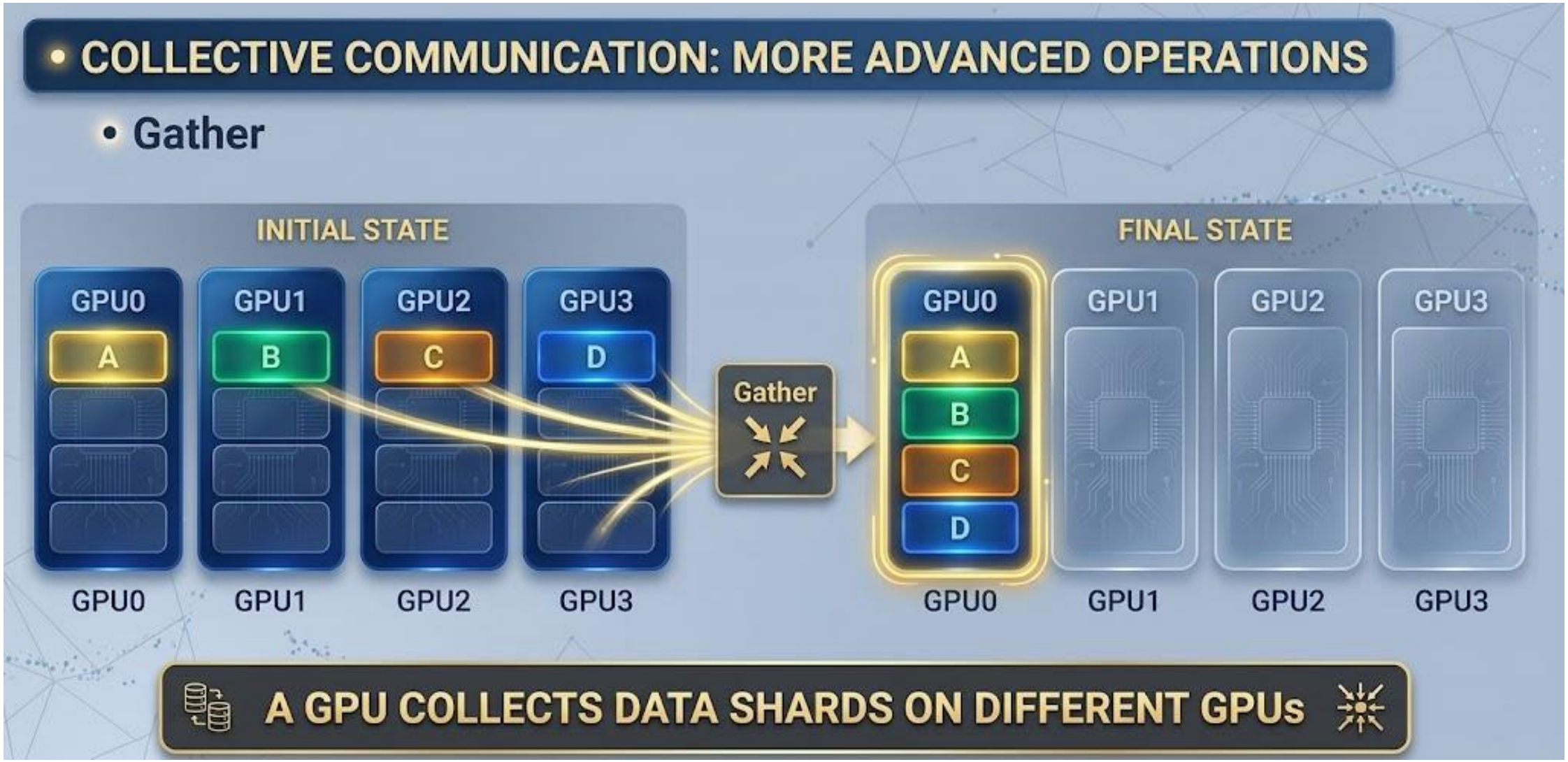
Collective Communication Basics

- Collective Communication: More Advanced Operations
  - Broadcast: one-to-many
  - Gather: many-to-one, and All-Gather: many-to-many
  - Scatter: one-to-many
  - Reduce: many-to-one, and All-Reduce: many-to-many
  - Reduce-Scatter: aggregate data and then transmit
  - All-to-All: many-to-many
  - AllReduce: ?

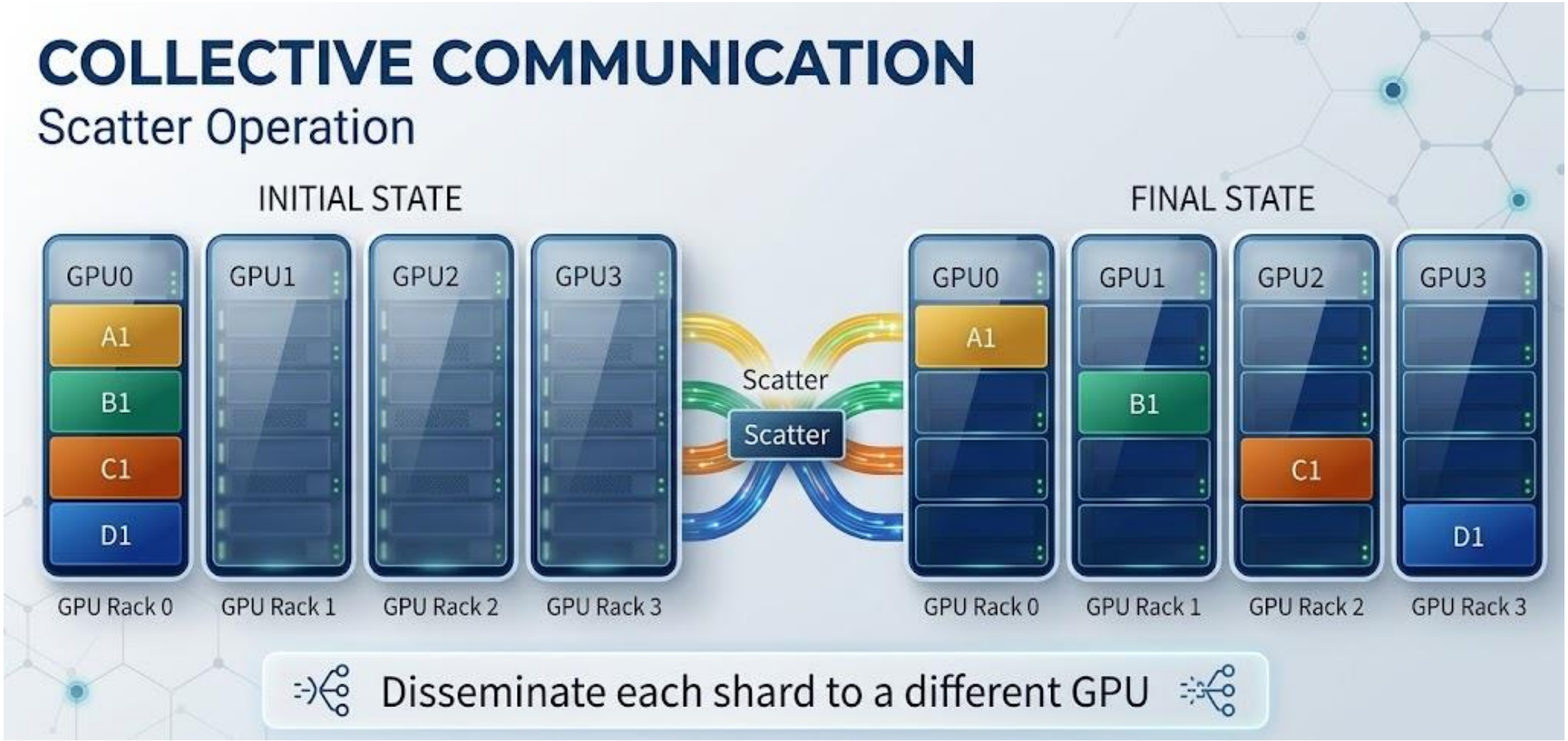
Collective Communication Basics



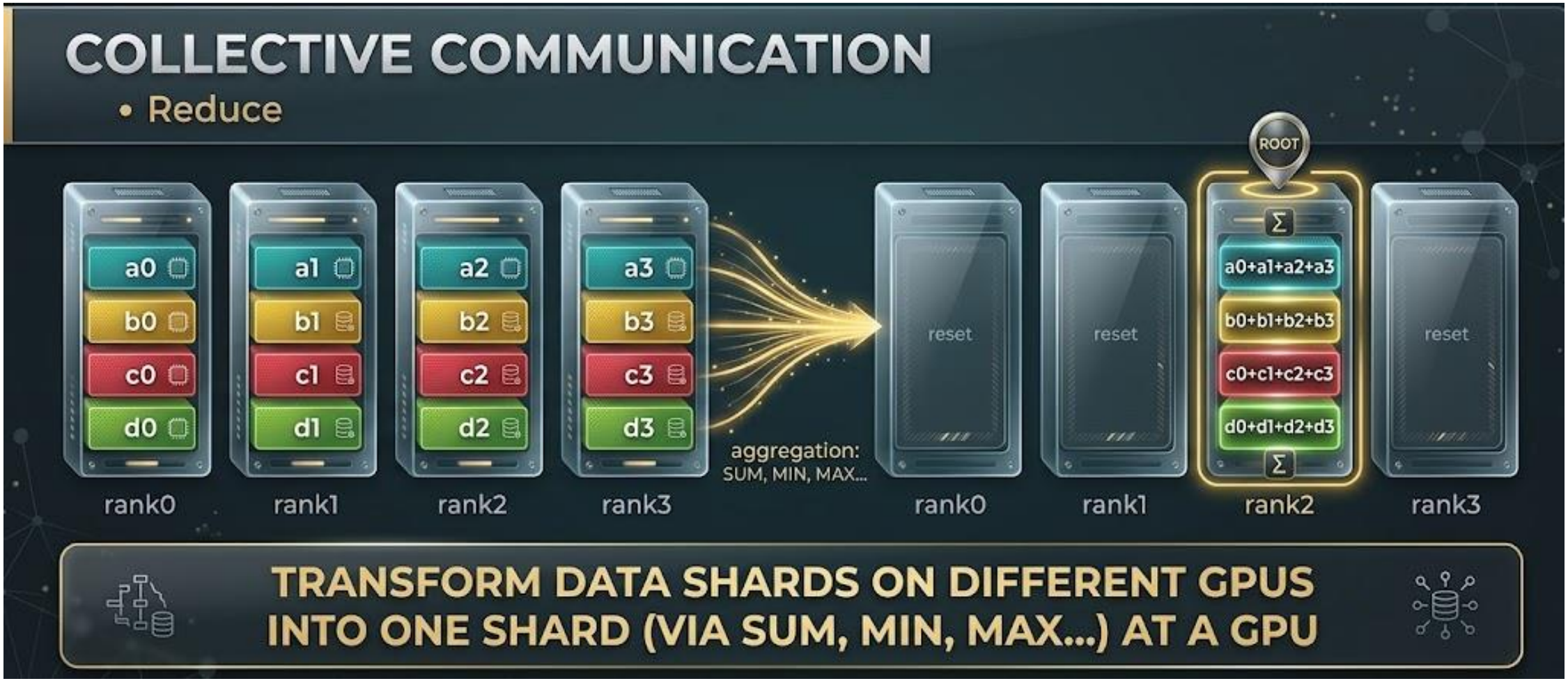
Collective Communication Basics



Collective Communication Basics



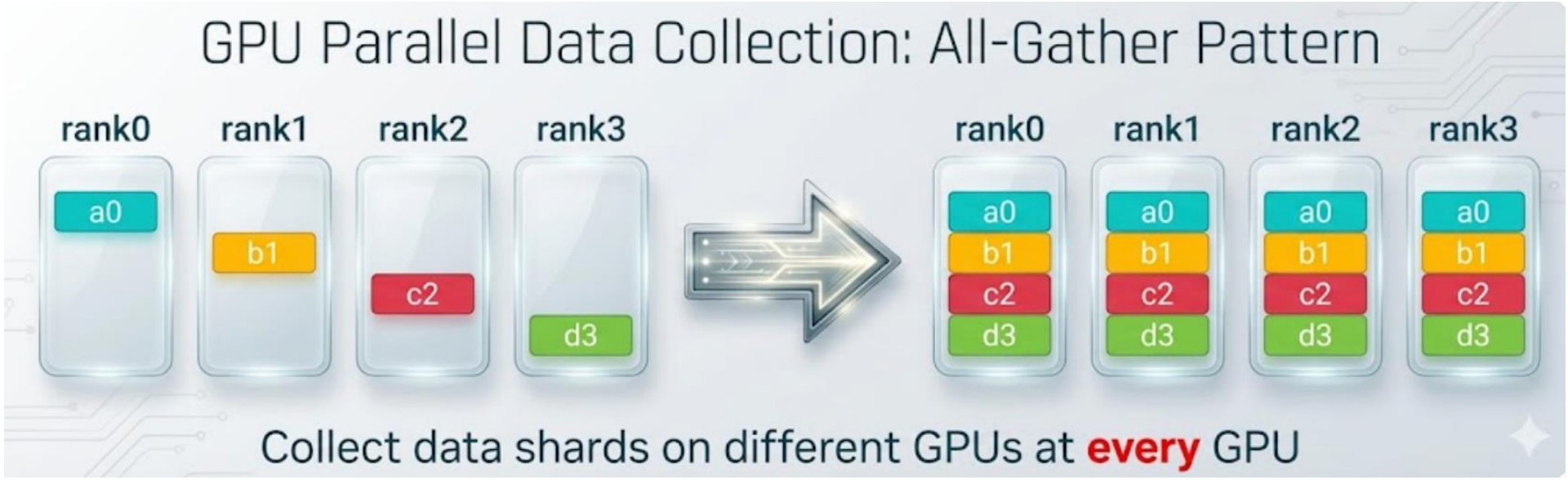
Collective Communication Basics



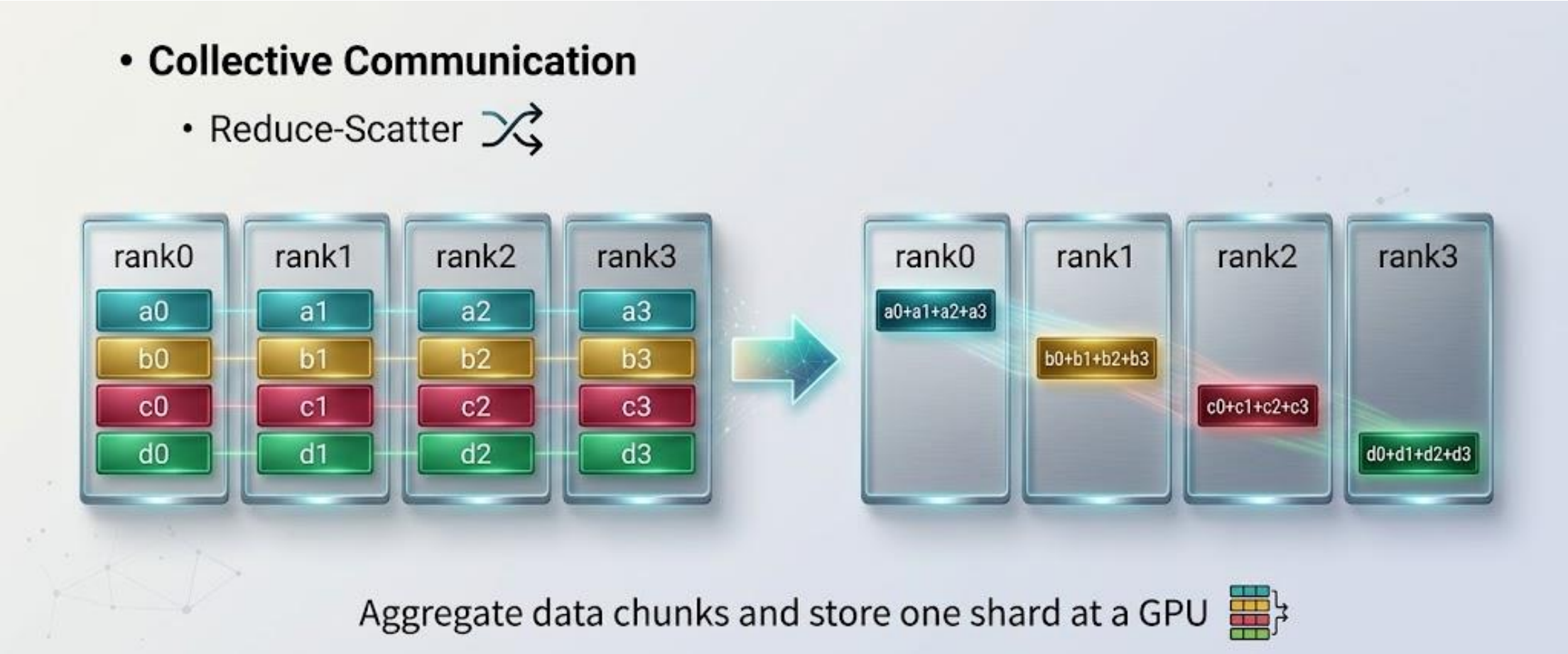
Collective Communication Basics

Collective Communication Basics

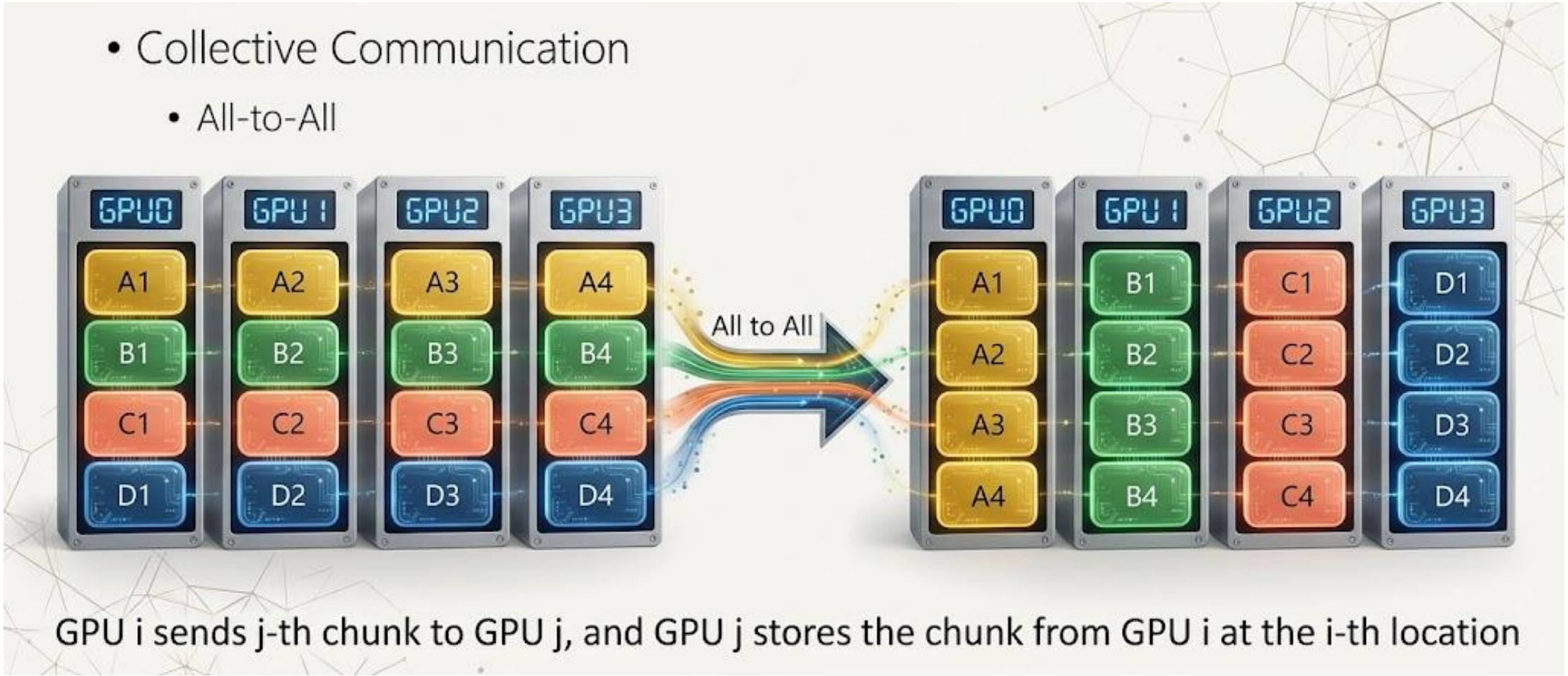
- Collective Communication
  - All-Gather



Collective Communication Basics



Collective Communication Basics



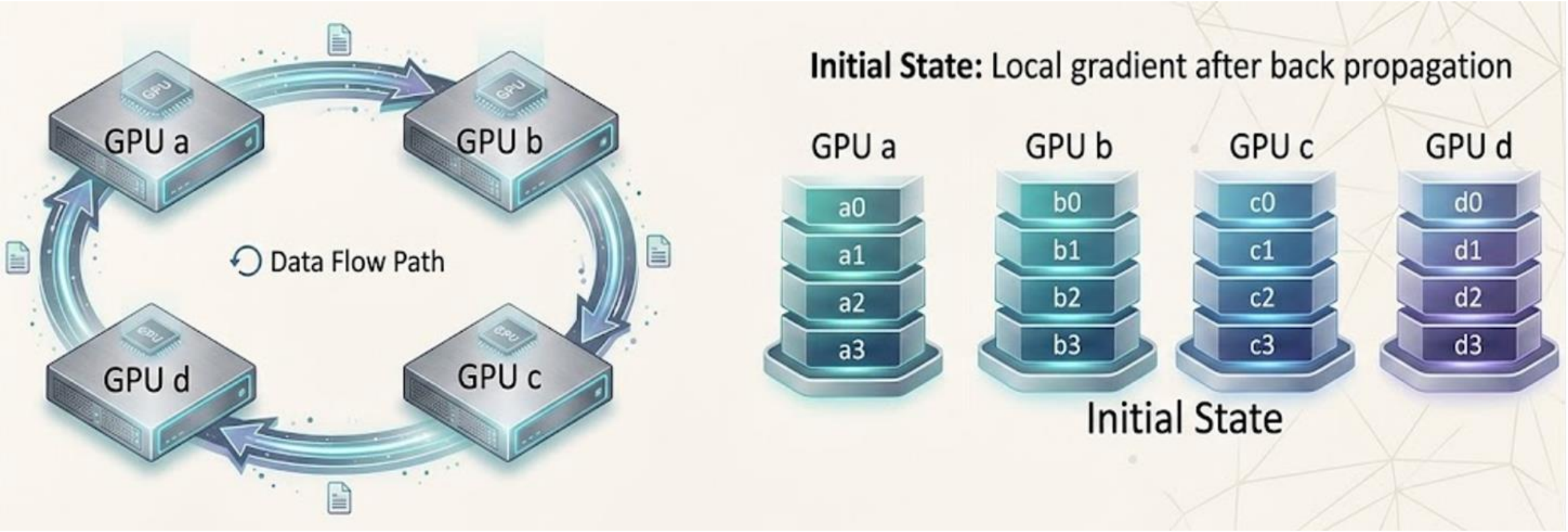
All-to-All is purely a data routing and memory transposition operation

Starting from the most important “All-Reduce”

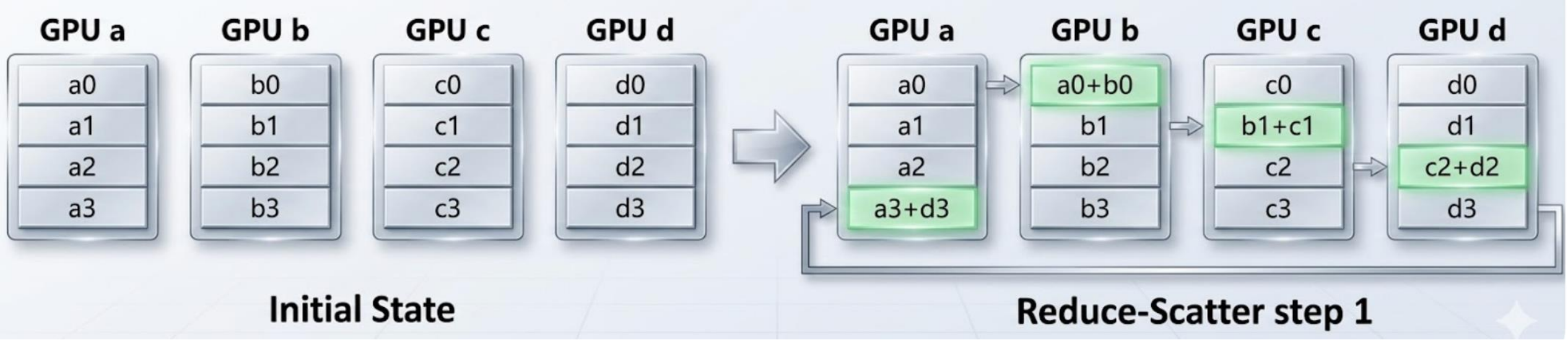
Ring All-Reduce

- Ring All-Reduce
- Tree All-Reduce
- Topology-aware All-Reduce

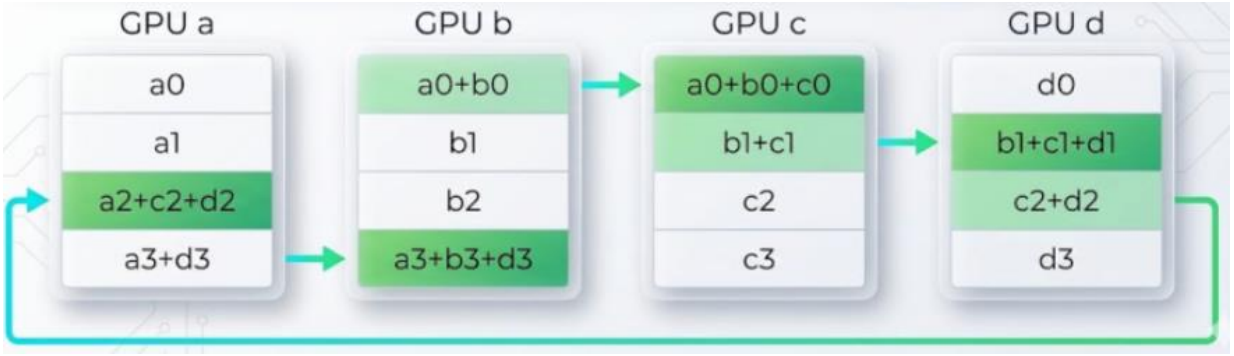
Ring All-Reduce



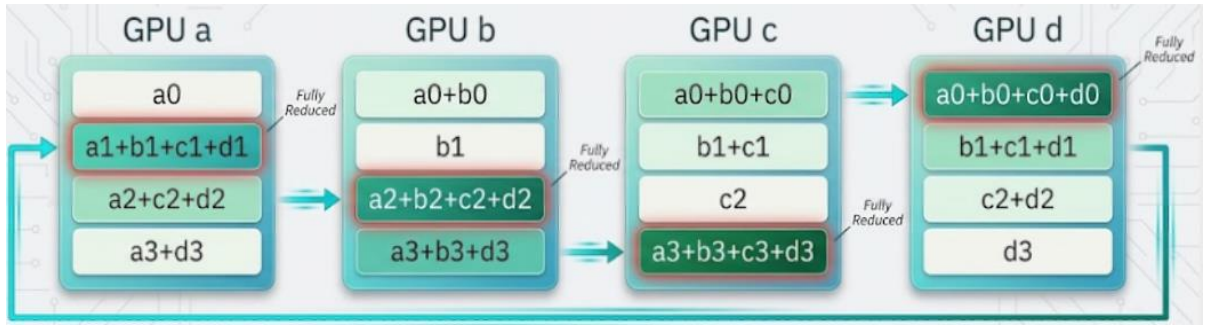
Ring All-Reduce



Ring All-Reduce



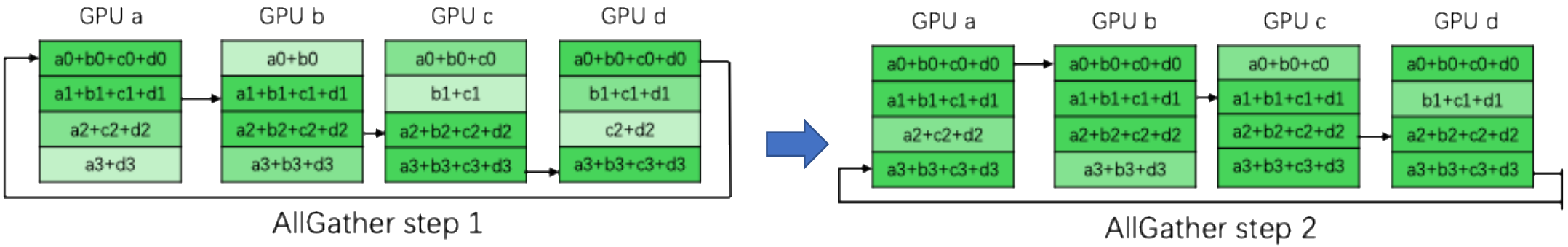
Reduce-Scatter Step 2



Reduce-Scatter Step 3

Ring All-Reduce

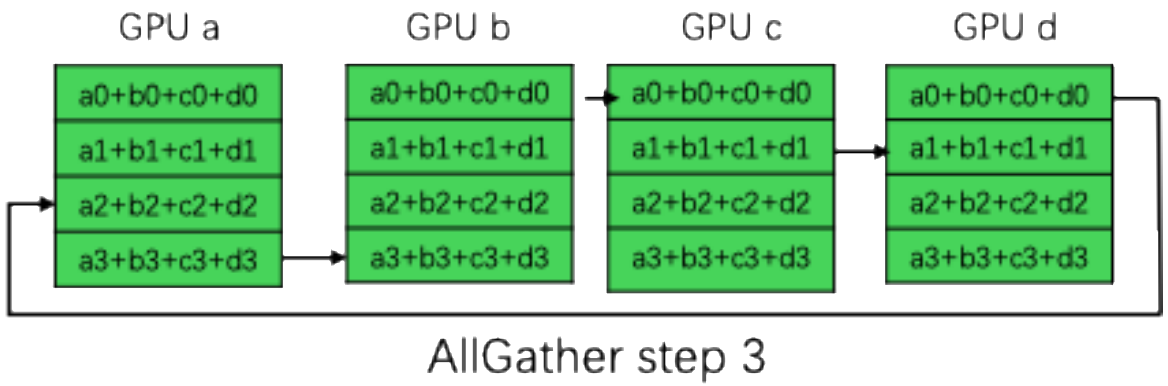
- Ring All-Reduce



All-Gather

Ring All-Reduce

- Ring All-Reduce



AllGather

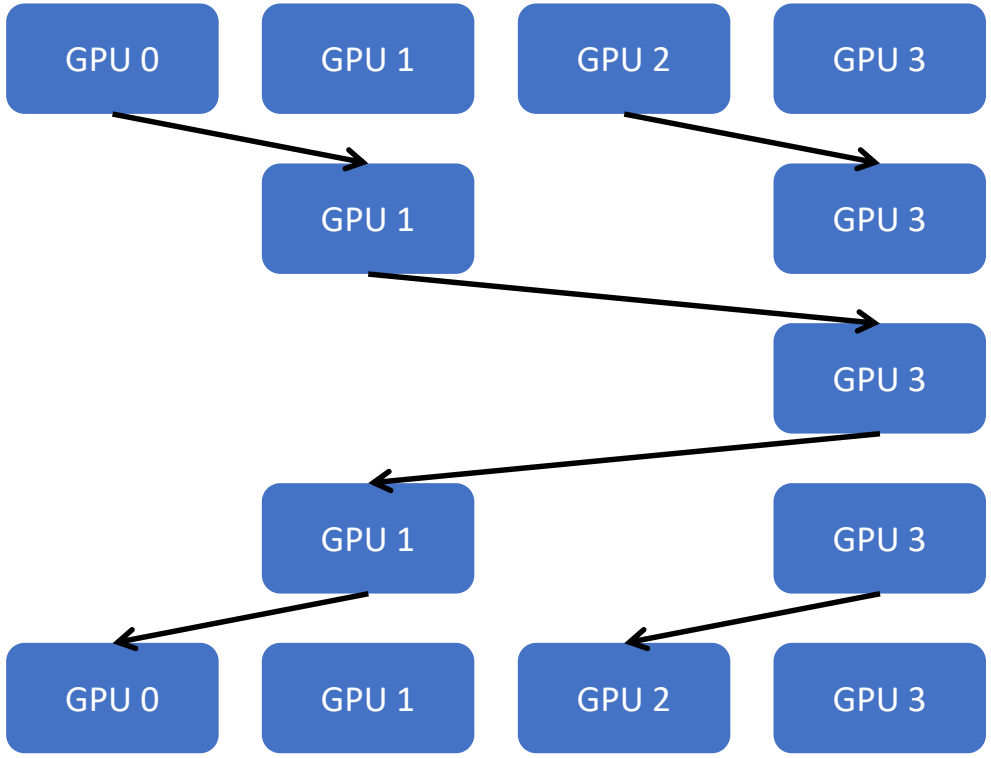
- Reduce-Scatter
  - N: # of GPUs, S: per-GPU data volume
  - $N-1$ rounds
  - S/N data transmission per round
- All-Gather
  - One round boradcasting or $N-1$ clockwise rounds
  - Data volume: $(N-1)*S/N$
- Total traffic per GPU
  - $\boldsymbol{2*(N-1)*S/N}$

Ring All-Reduce

“All-Reduce = Reduce-Scatter + All-Gather”

Tree All-Reduce

- Tree All-Reduce

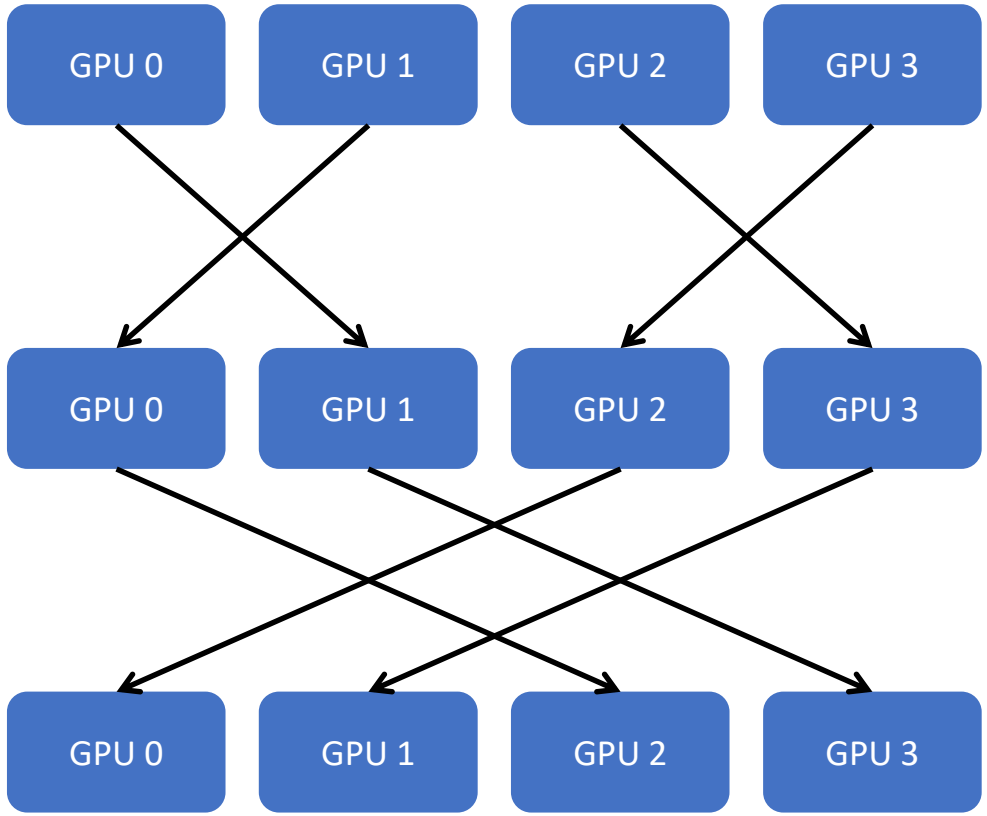


Data communication

- Upward
  - $\log_2 N$ rounds
  - At most S data per round
- Downward
  - $\log_2 N$ rounds
  - At most S data per round
- Total traffic per GPU
  - $2 * S * \log_2 N$ (naïve transmission without overlapping)

Butterfly All-Reduce

- Butterfly All-Reduce

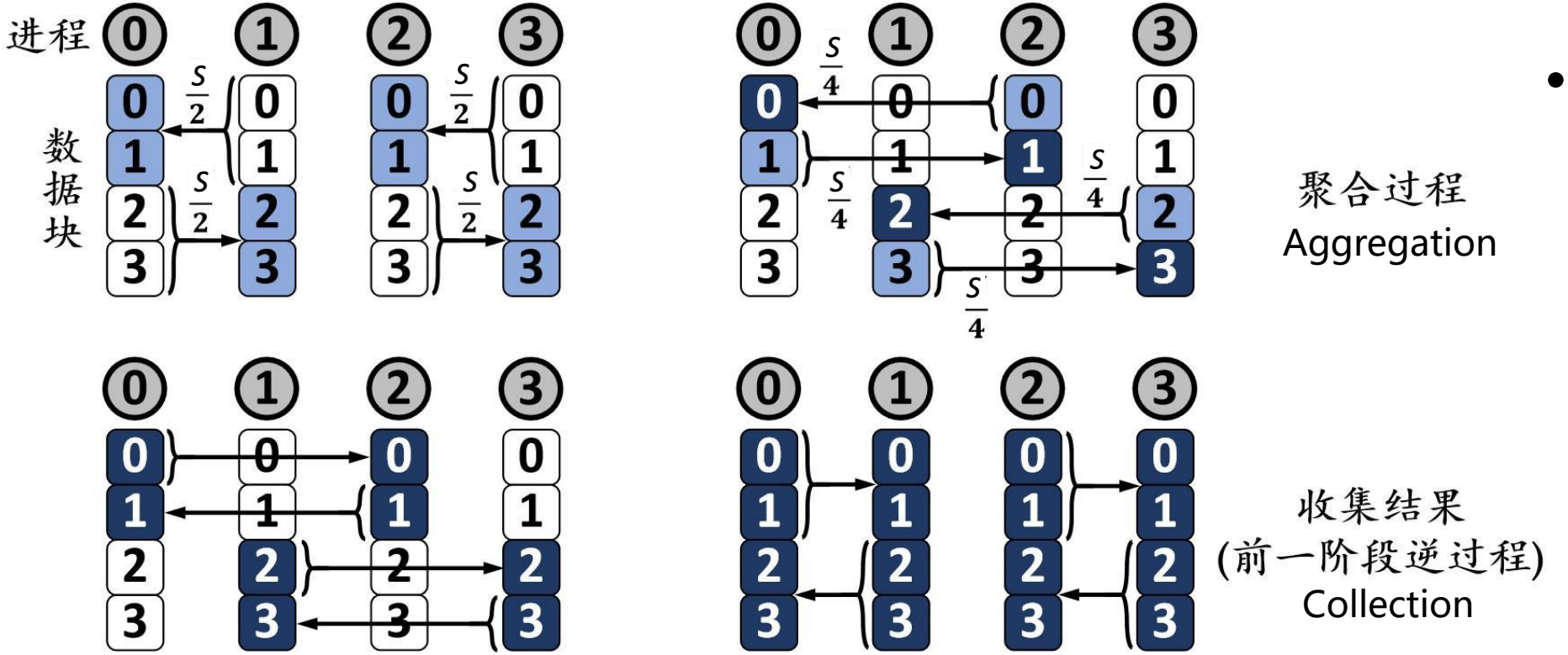


Butterfly Reduce: Utilizing bidirectional bandwidth

- Butterfly Reduce
  - $\log_2 N$ rounds
  - S data per round
- Total Traffic per-GPU
  - $\log_2 N * S$

Rabenseifner All-Reduce

- Rabenseifner Reduce



Rabenseifner Algorithm (~ recursive halving-doubling)

- Total Traffic per-GPU
  - $2\log_2 N$ rounds
  - Different traffic load at every round
  - Total traffic volume: $2\left(\frac{S}{2}+\frac{S}{4}+\cdots+\frac{S}{N}\right) \approx 2\frac{N-1}{N}S$

《Allreduce算法及其硬件加速方法介绍》

Cross-Comparison

- Communication Cost
  - Assumption: each GPU can send and receive data simultaneously
  - Classical α-β model: latency = $\alpha + \beta \cdot \frac{S}{B}$

"start-up" delay

transmission efficiency

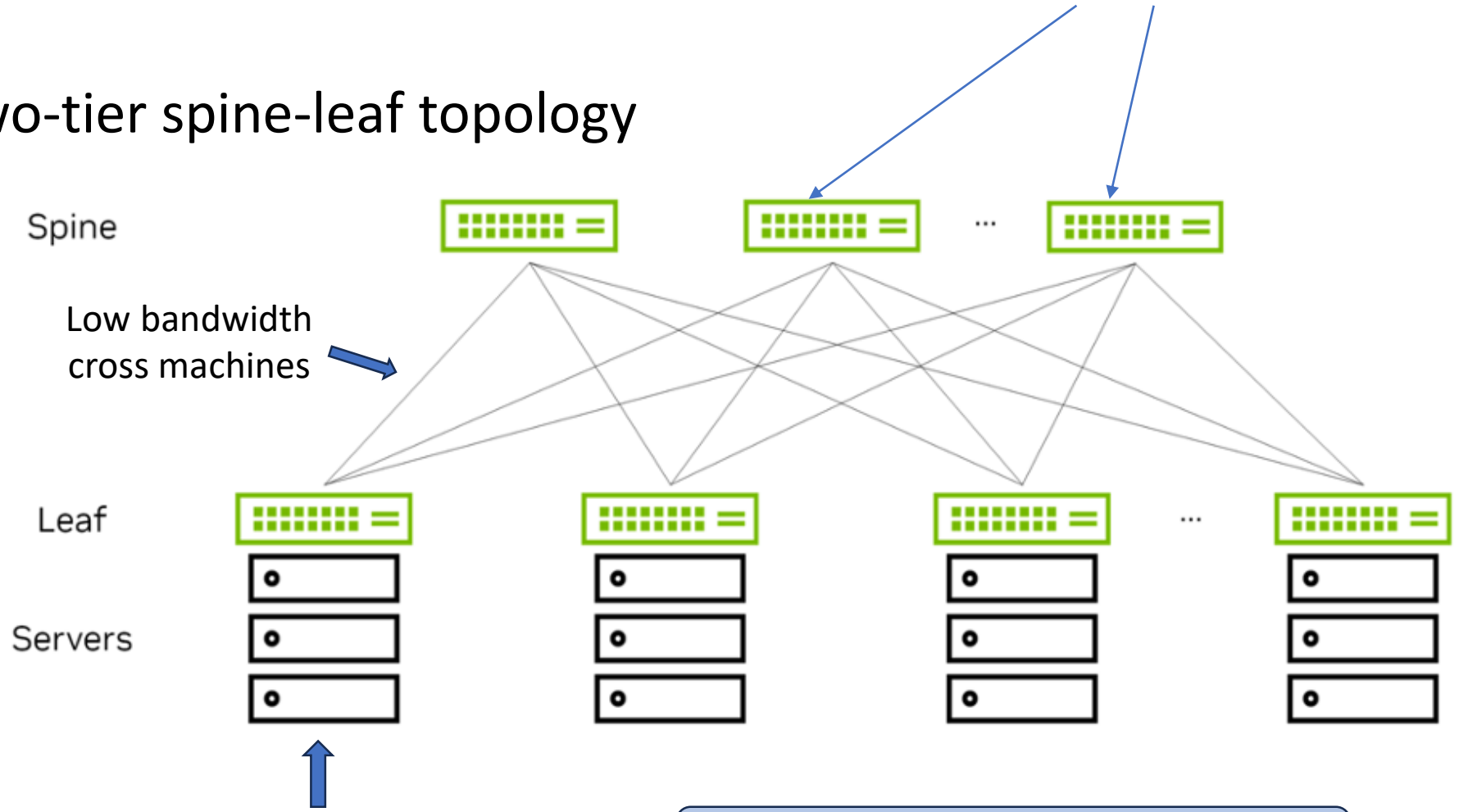
Message size/bandwidth

Algorithm	Rounds	Traffic Load	Cost	Pros and Cons
Ring	2(N-1)	2S(N-1)/N	*2(N-1)*(α + β∗S/N/B)*	▪ Large data chunk aggregation ▪ Relatively small # of GPUs ▪ Not suitable for short chunks ▪ Ease of implementation ▪ Utilizating bidirectional links
Rabenseifner	2 $\lceil \log N \rceil$	2S(N-1)/N	2$\lceil \log N \rceil\alpha$ + 2$(N-1)$ $\beta * S/N$/B	▪ Relatively smaller data chunk ▪ Large # of GPUs ▪ Periodically changing communication pairs

Challenge of Scaling Out

Network congestion, load balancing

- Two-tier spine-leaf topology



Topology-aware CCL

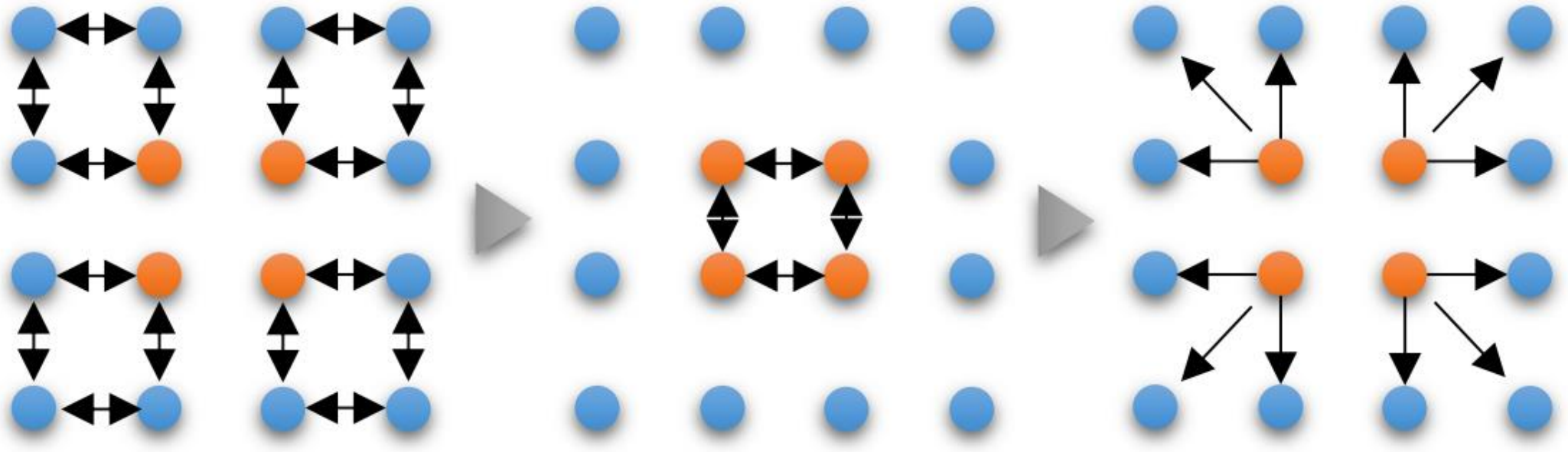
All-Reduce for LLM training

- Each company develops its own collective communication libraries
  - NCCL (NVIDIA CCL)
  - MSCCL (Microsoft CCL)
  - Gloo
  - HCCL (Huawei CCL)
  - ACCL (Alibaba CCL)
  - TCCL (Tencent CCL)
  - Many to be added

Tailored for their own harware, datacenter network topology, etc.

All-Reduce for LLM training

- Hierarchical All-Reduce (for reference)



《Highly Scalable Deep Learning Training System with Mixed-Precision: Training ImageNet in Four Minutes》

Thanks!