NCCL: ACCELERATED MULTI-GPU COLLECTIVE COMMUNICATIONS

Cliff Woolley, Sr. Manager, Developer Technology Software, NVIDIA

NVIDIA.

BACKGROUND

What limits the scalability of parallel applications?

Efficiency of parallel computation tasks

- Amount of exposed parallelism
- Amount of work assigned to each processor

Expense of communications among tasks

- Amount of communication
- Degree of overlap of communication with computation

COMMON COMMUNICATION PATTERNS

COMMUNICATION AMONG TASKS

What are common communication patterns?

Point-to-point communication

- Single sender, single receiver
- Relatively easy to implement efficiently

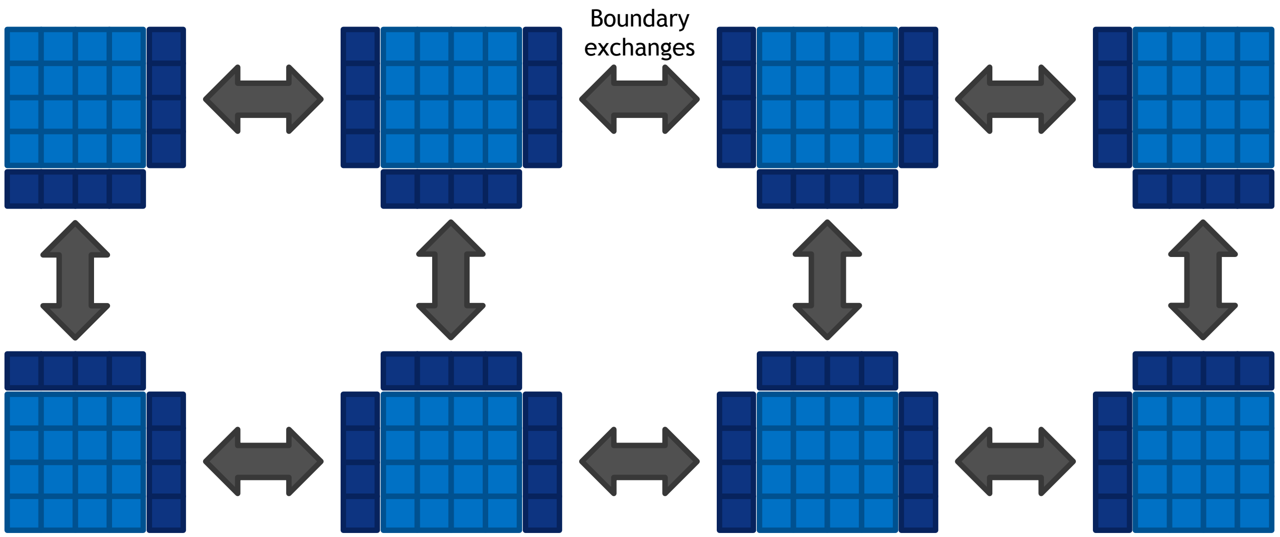
Collective communication

- Multiple senders and/or receivers
- Patterns include broadcast, scatter, gather, reduce, all-to-all, …
- Difficult to implement efficiently

POINT-TO-POINT COMMUNICATION

Single-sender, single-receiver per instance

Most common pattern in HPC, where communication is usually to nearest neighbors

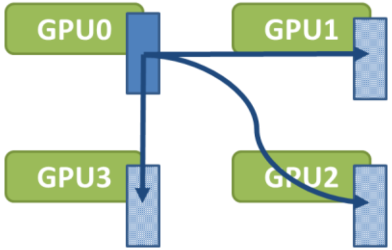


2D Decomposition

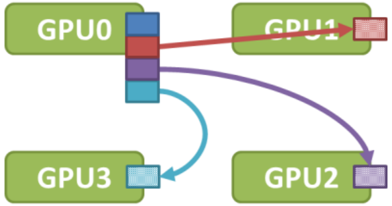
COLLECTIVE COMMUNICATION

Multiple senders and/or receivers

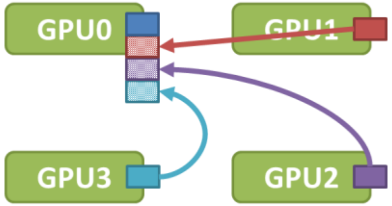
Broadcast



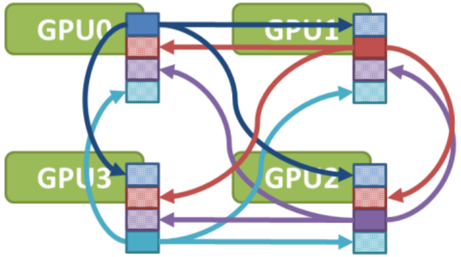
Scatter



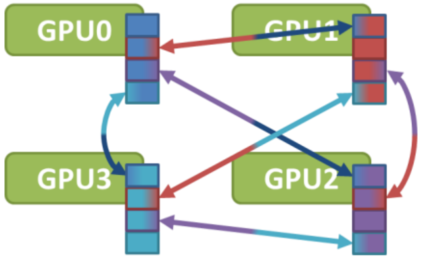
Gather



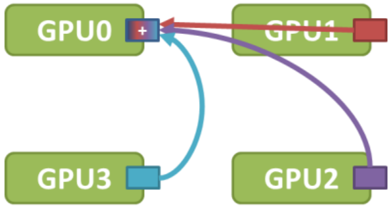
All-Gather



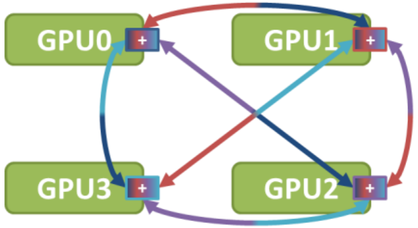
All-to-All



Reduce

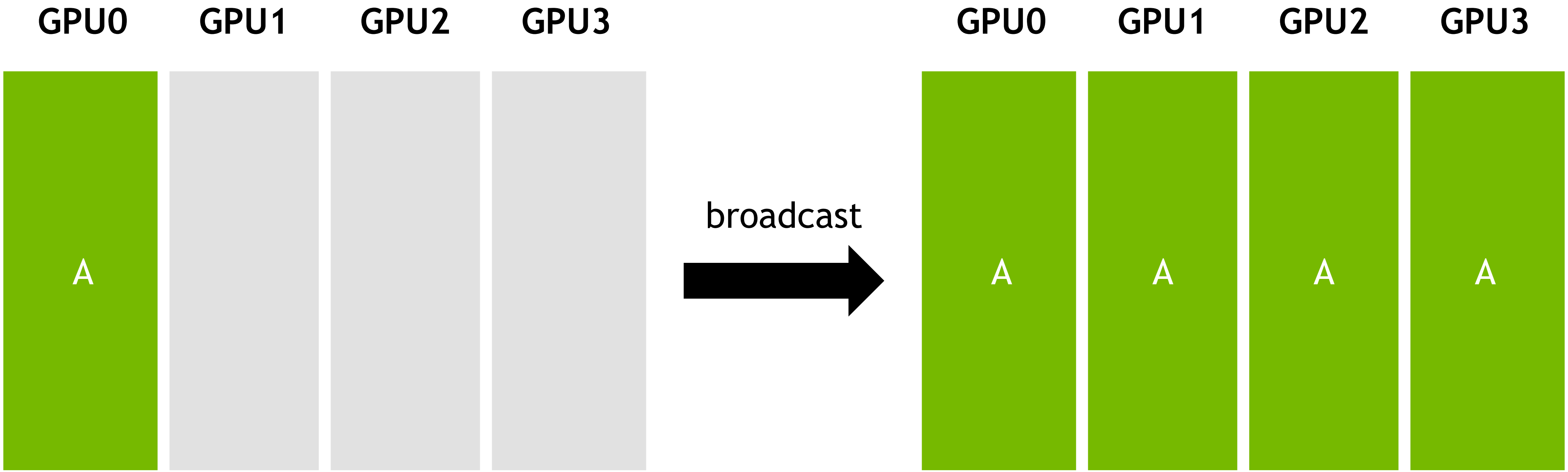


All-Reduce



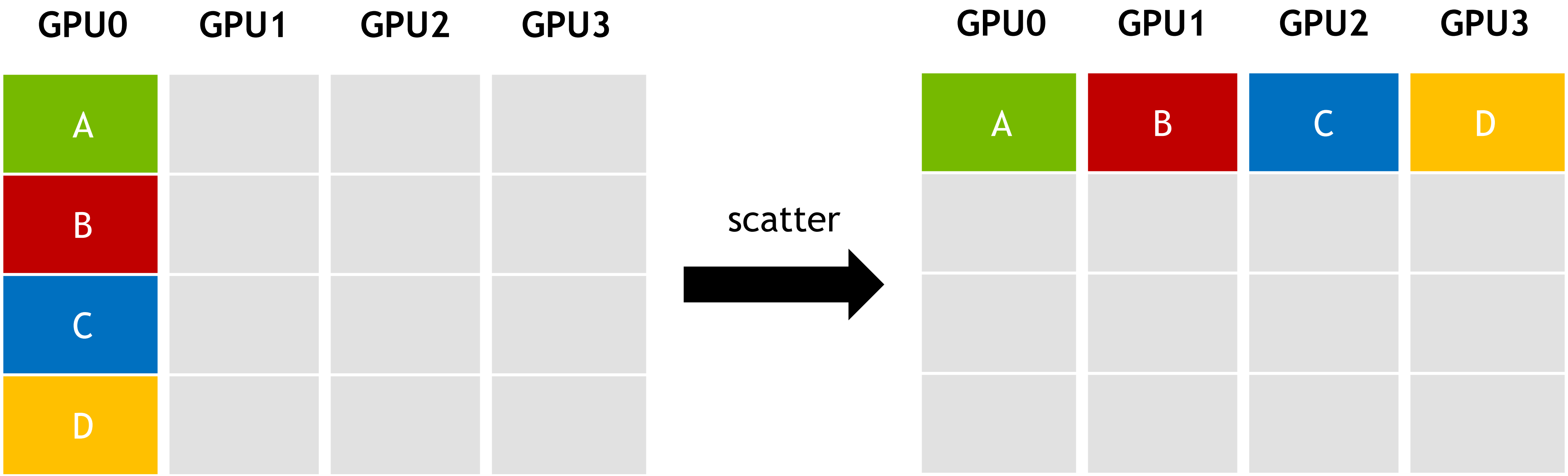
BROADCAST

One sender, multiple receivers



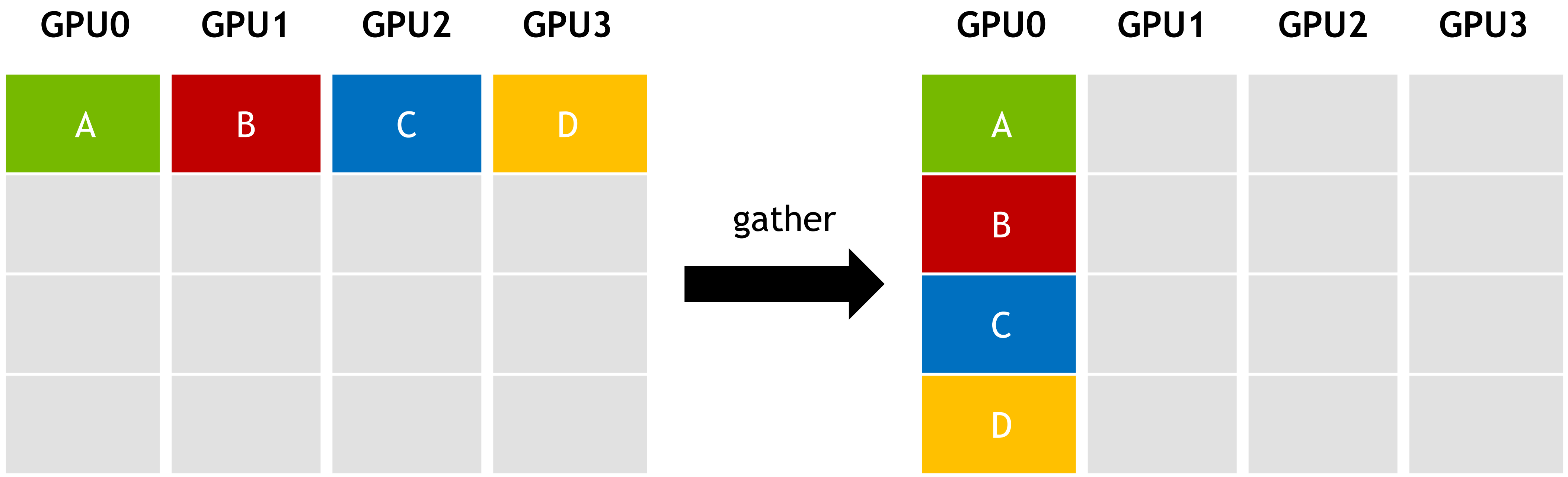
SCATTER

One sender; data is distributed among multiple receivers



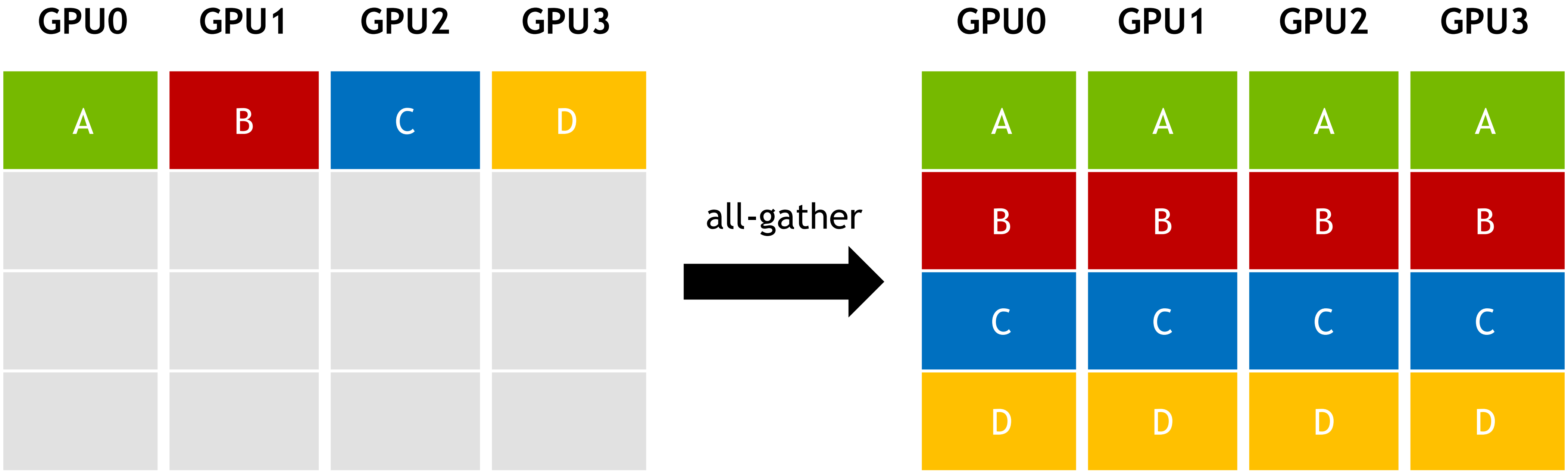
GATHER

Multiple senders, one receiver



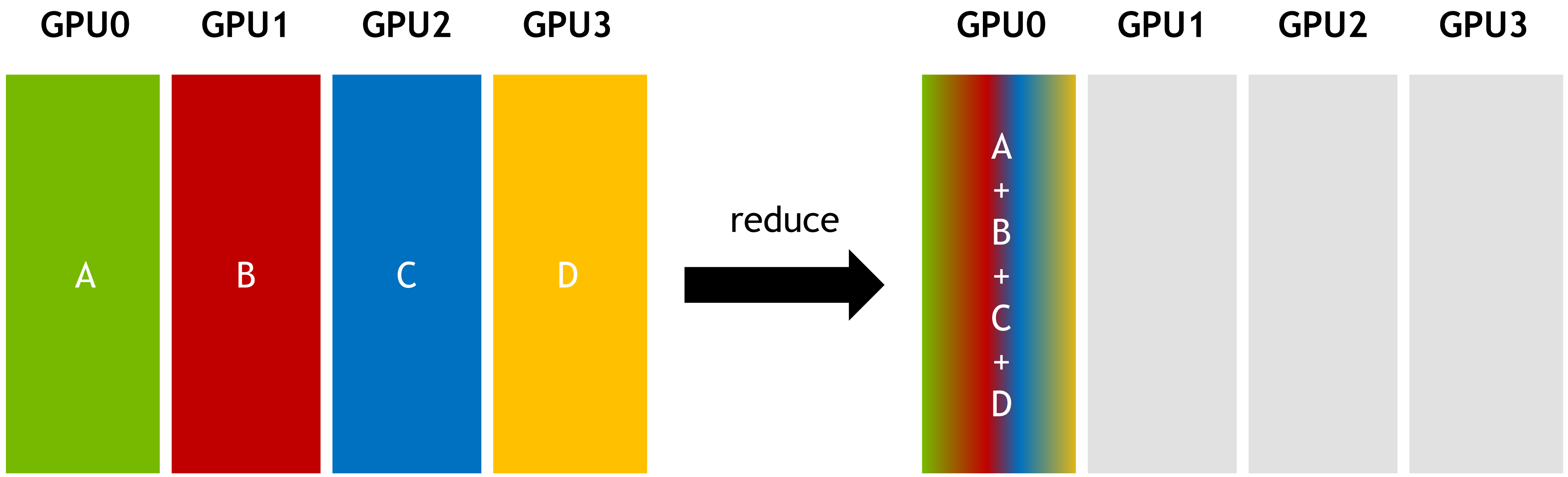
ALL-GATHER

Gather messages from all; deliver gathered data to all participants



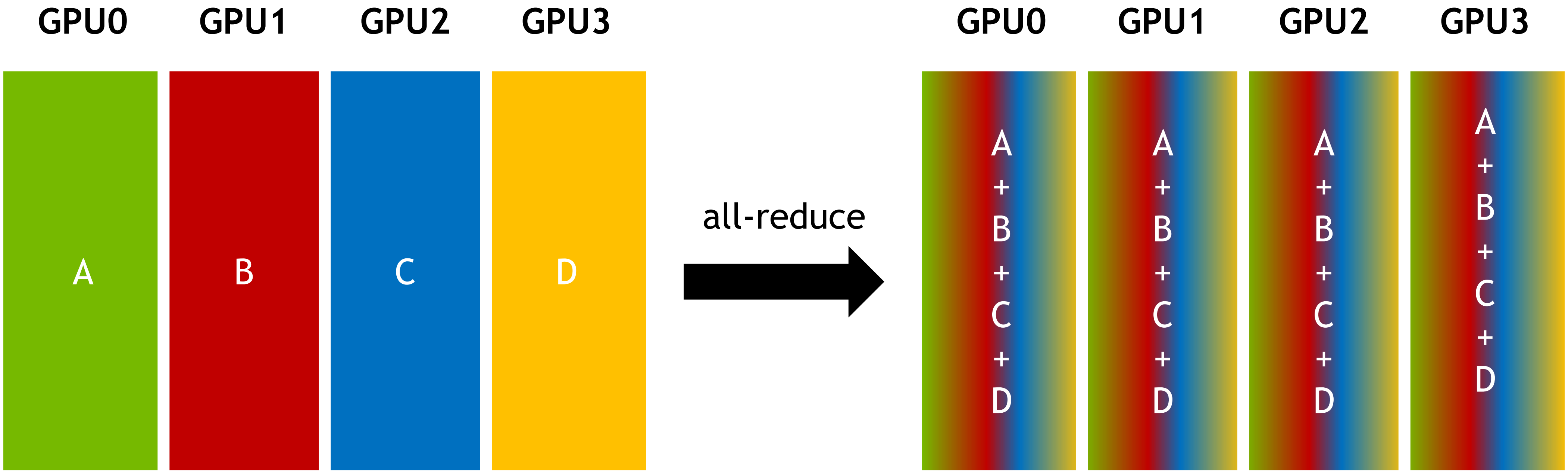
REDUCE

Combine data from all senders; deliver the result to one receiver



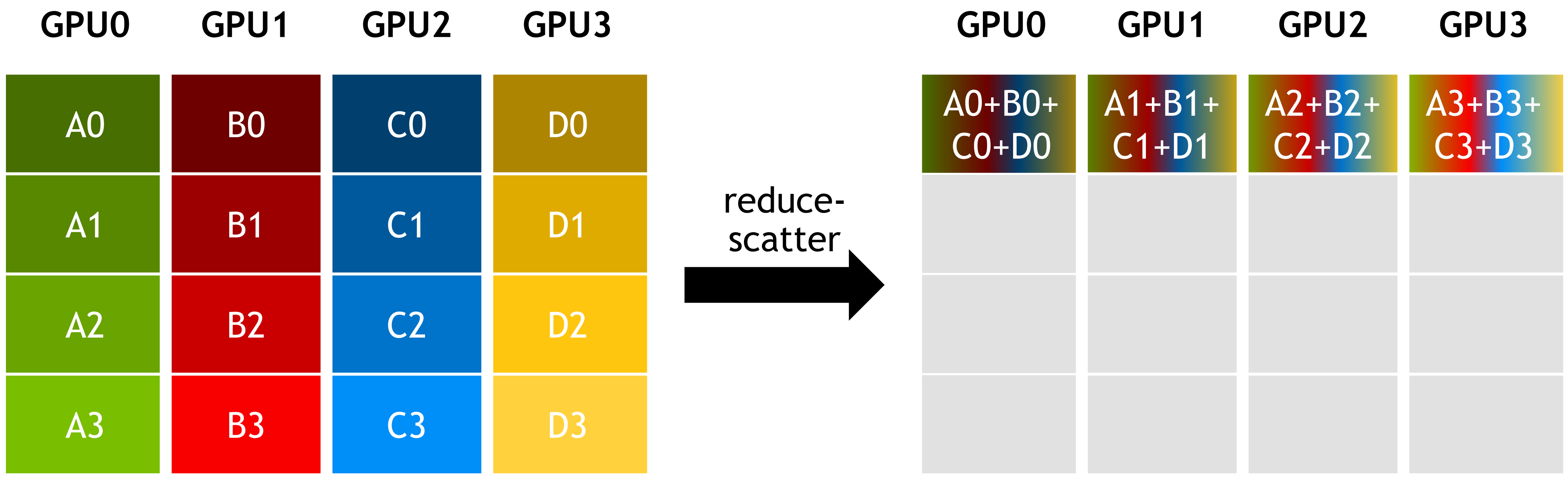
ALL-REDUCE

Combine data from all senders; deliver the result to all participants



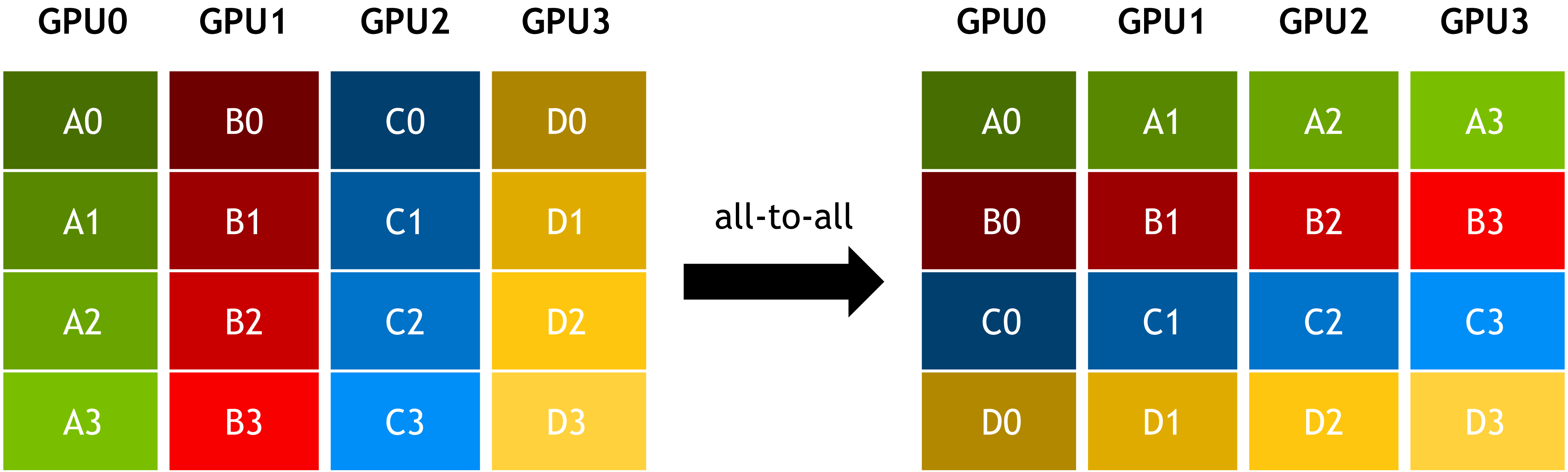
REDUCE-SCATTER

Combine data from all senders; distribute result across participants



ALL-TO-ALL

Scatter/Gather distinct messages from each participant to every other



THE CHALLENGE OF COLLECTIVES

THE CHALLENGE OF COLLECTIVES

Collectives are often avoided because they are expensive. Why?

Having multiple senders and/or receivers compounds communication inefficiencies

- For small transfers, latencies dominate; more participants increase latency
- For large transfers, bandwidth is key; bottlenecks are easily exposed
- May require topology-aware implementation for high performance
- Collectives are often blocking/non-overlapped

THE CHALLENGE OF COLLECTIVES

If collectives are so expensive, do they actually get used? YES!

Collectives are central to scalability in a variety of key applications:

- Deep Learning (All-reduce, broadcast, gather)
- Parallel FFT (Transposition is all-to-all)
- Molecular Dynamics (All-reduce)
- Graph Analytics (All-to-all)
- …

THE CHALLENGE OF COLLECTIVES

Many implementations seen in the wild are suboptimal

Scaling requires efficient communication algorithms and careful implementation

Communication algorithms are topology-dependent

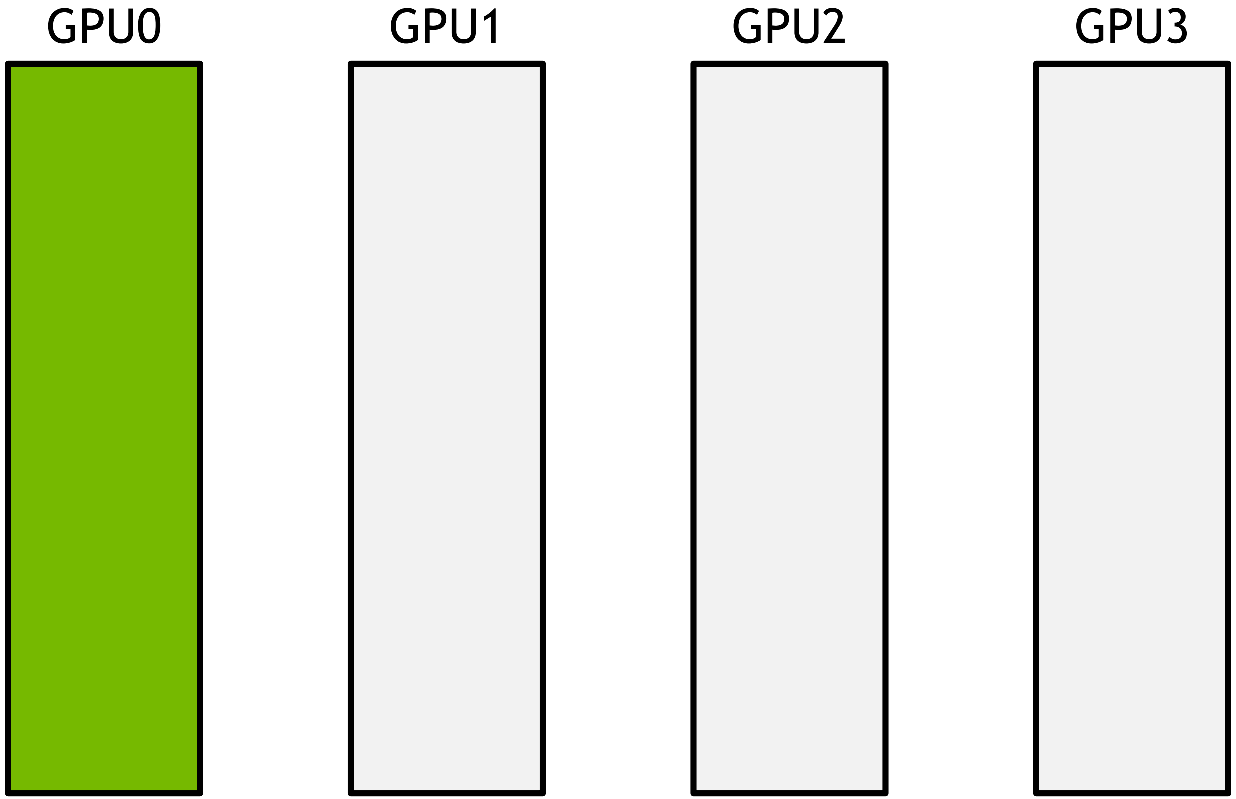
Topologies can be complex – not every system is a fat tree

Most collectives amenable to bandwidth-optimal implementation on rings, and many topologies can be interpreted as one or more rings [P. Patarasuk and X. Yuan]

RING-BASED COLLECTIVES: A PRIMER

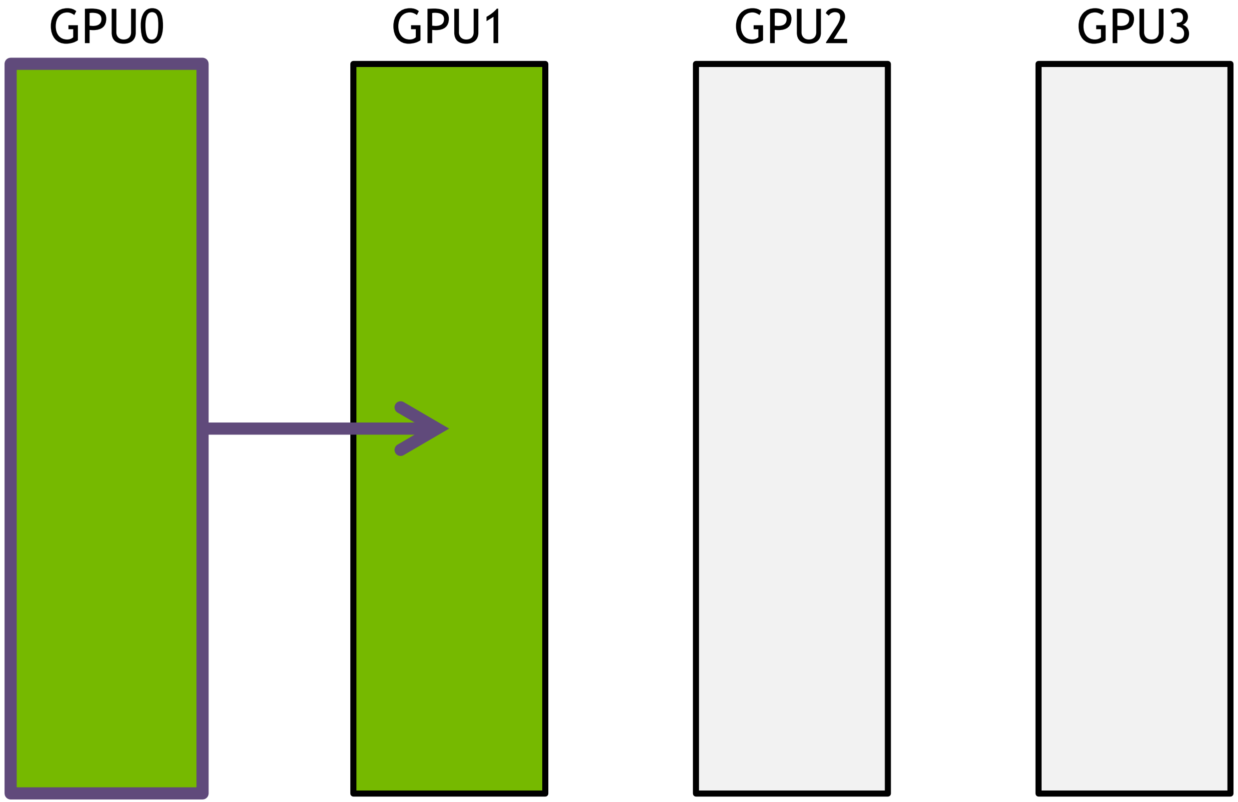
BROADCAST

with unidirectional ring



BROADCAST

with unidirectional ring



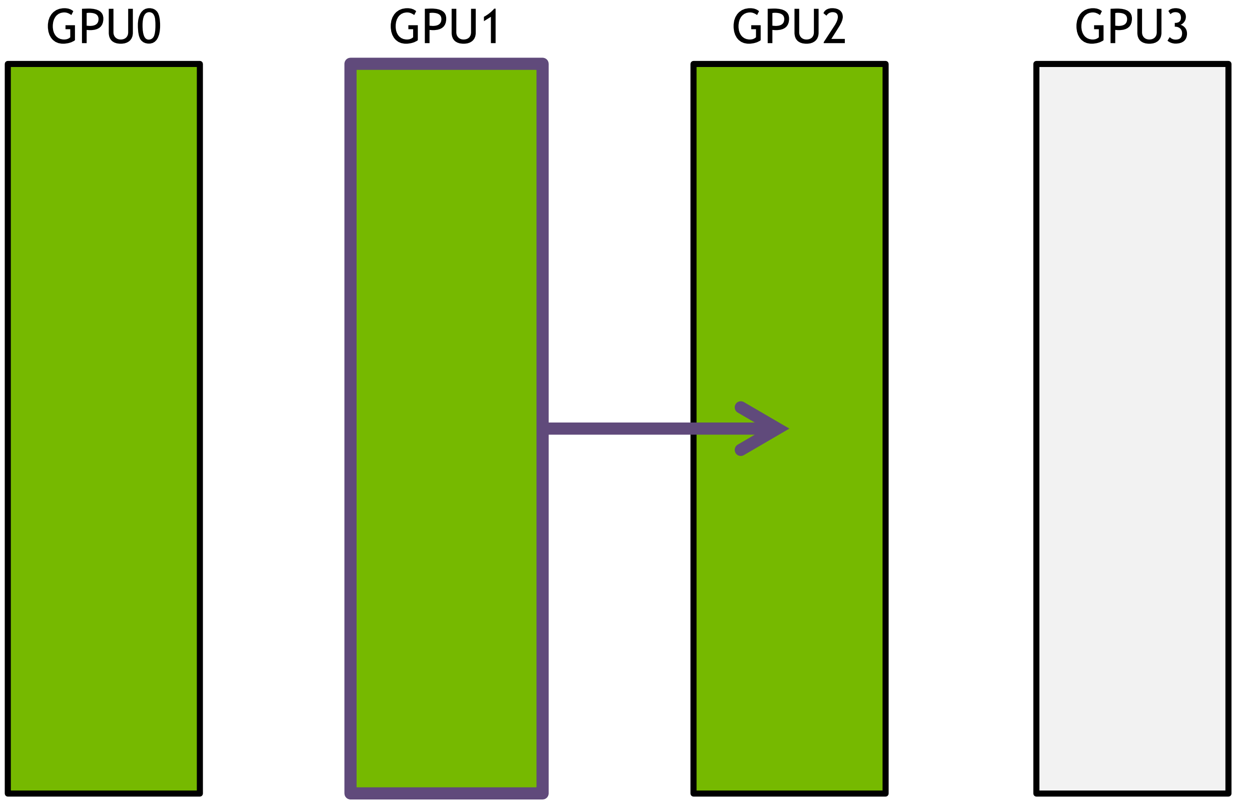
Step 1: $\Delta t = N/B$

N: bytes to broadcast

B: bandwidth of each link

BROADCAST

with unidirectional ring



Step 1: $\Delta t = N/B$

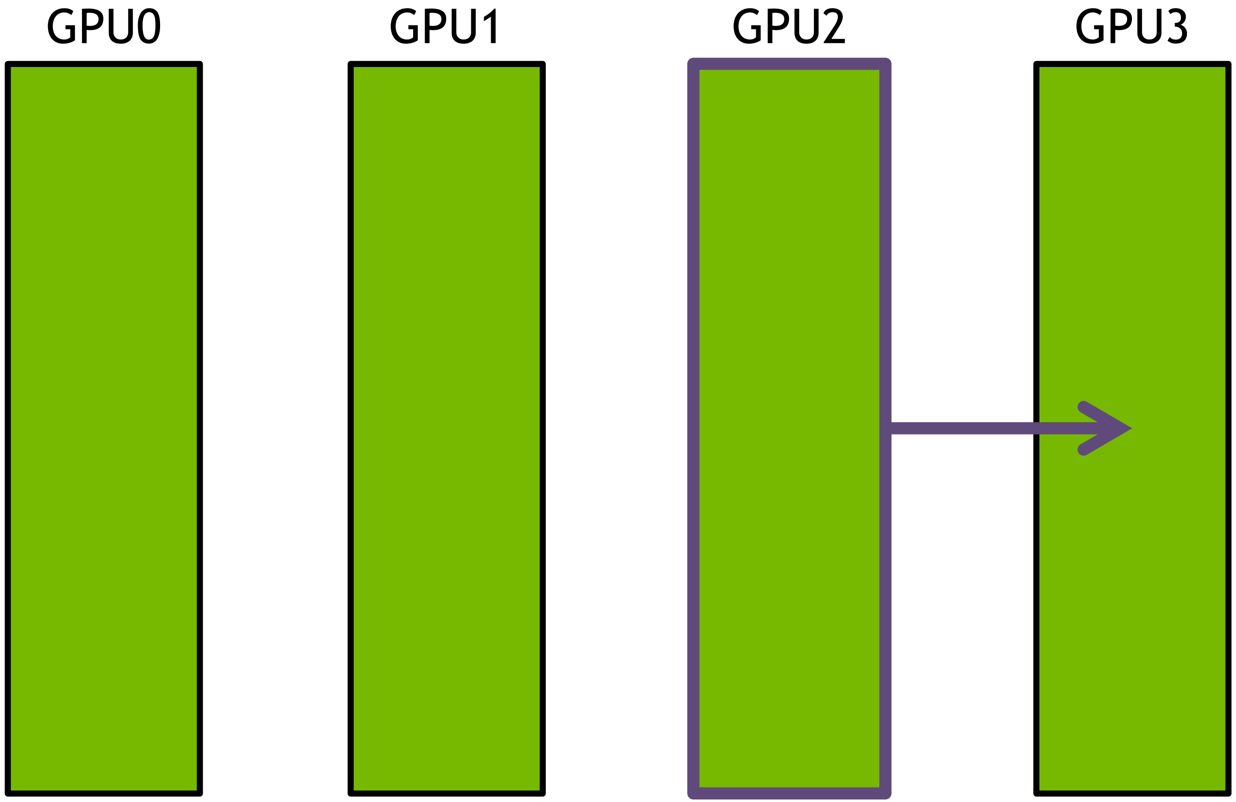
Step 2: $\Delta t = N/B$

N: bytes to broadcast

B: bandwidth of each link

BROADCAST

with unidirectional ring



Step 1: $\Delta t = N/B$

Step 2: $\Delta t = N/B$

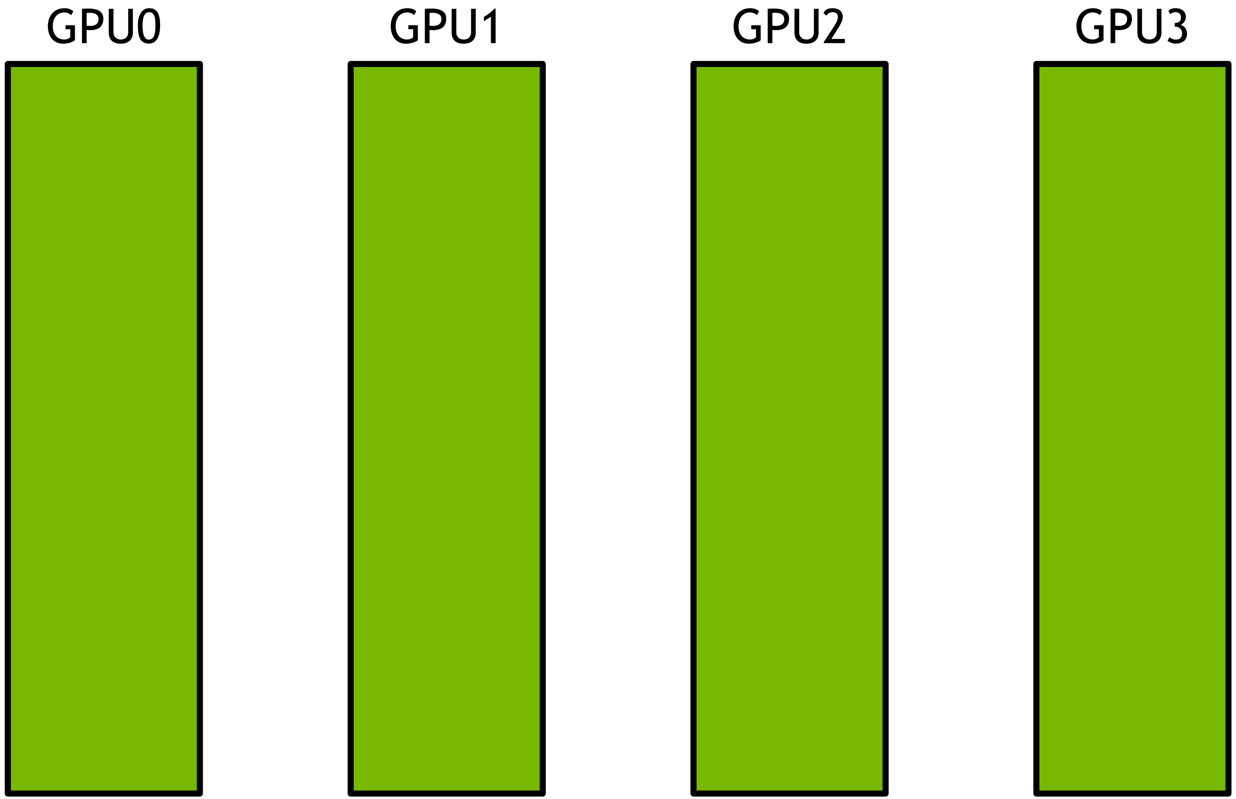
Step 3: $\Delta t = N/B$

N: bytes to broadcast

B: bandwidth of each link

BROADCAST

with unidirectional ring



Step 1: $\Delta t = N/B$

Step 2: $\Delta t = N/B$

Step 3: $\Delta t = N/B$

Total time: $(k-1)N/B$

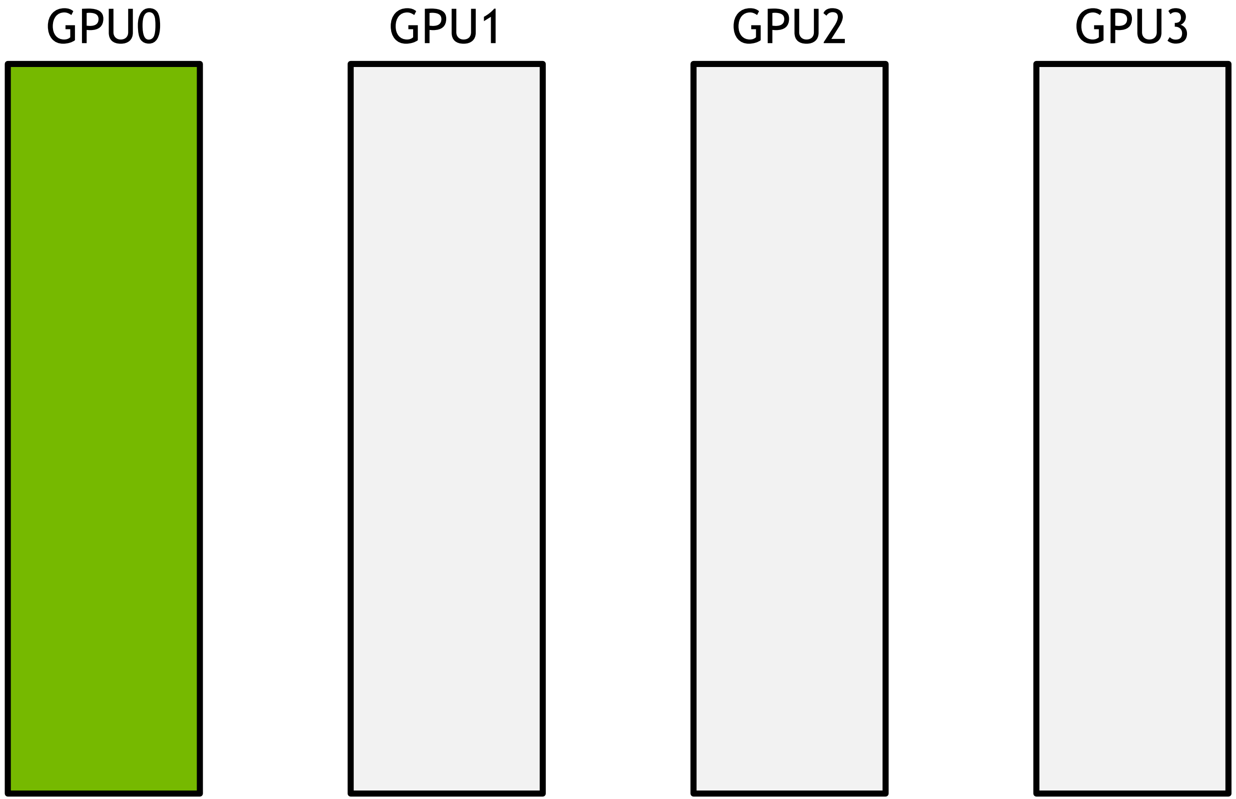
N: bytes to broadcast

B: bandwidth of each link

k: number of GPUs

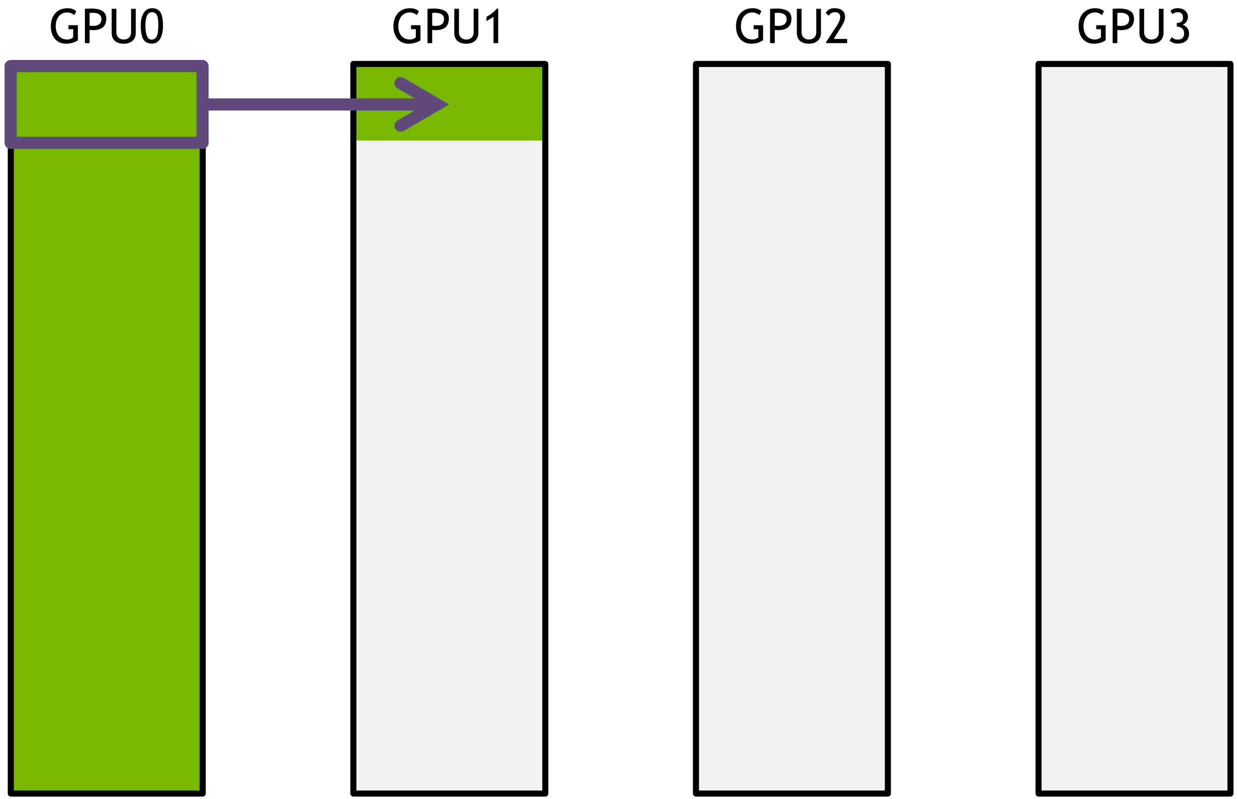
BROADCAST

with unidirectional ring



BROADCAST

with unidirectional ring

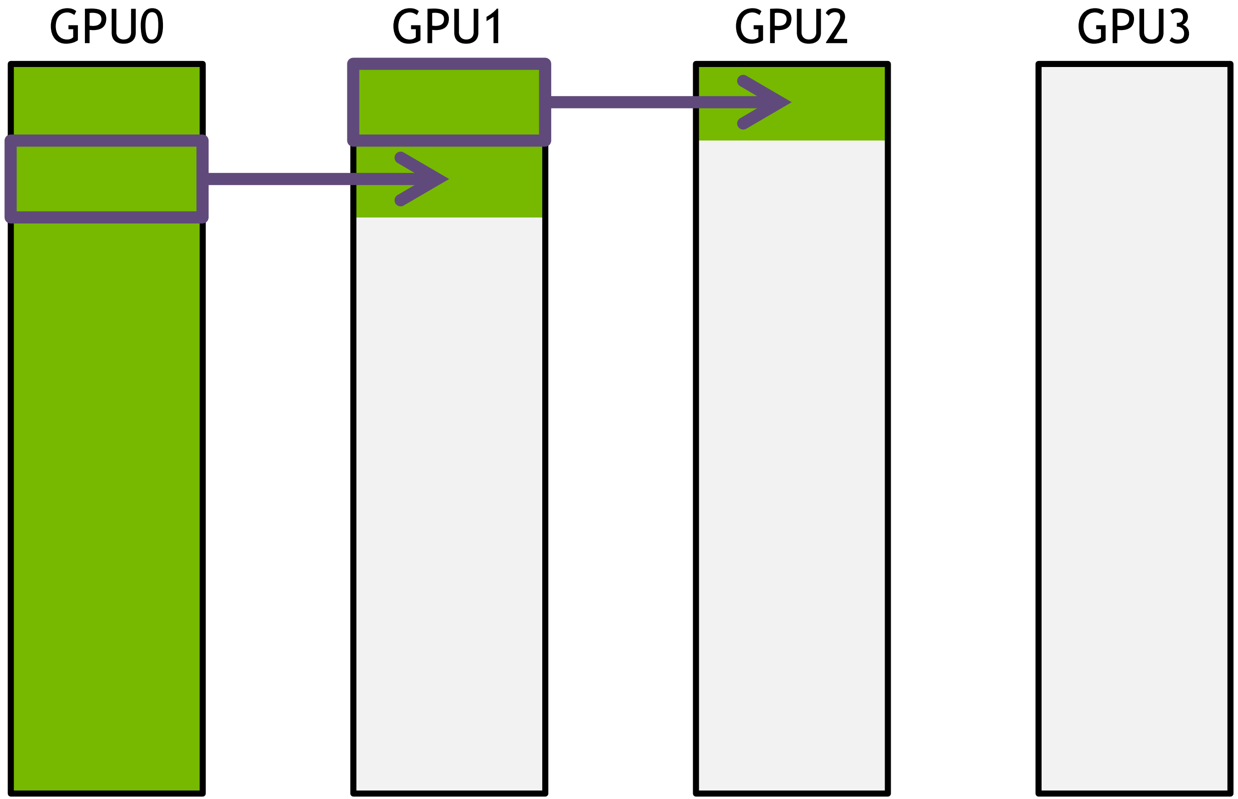


Split data into S messages

Step 1: $\Delta t = N/(SB)$

BROADCAST

with unidirectional ring



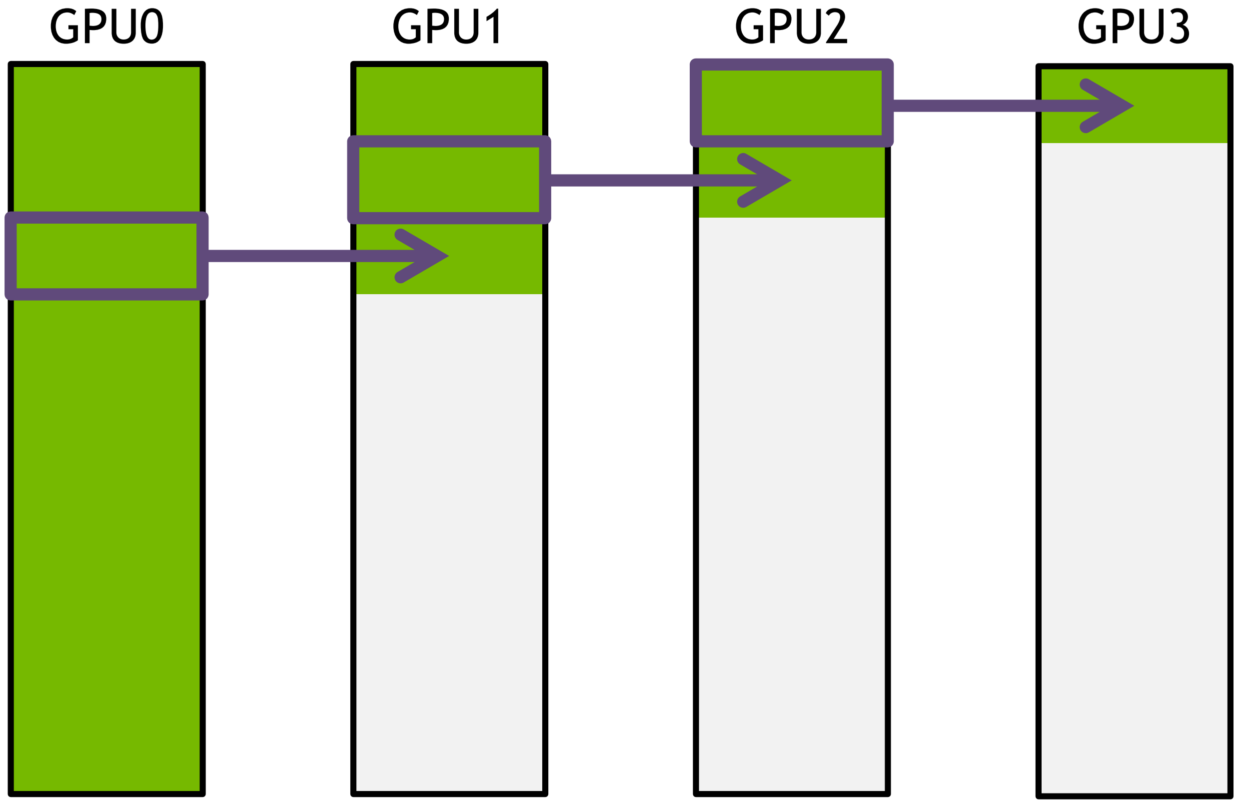
Split data into S messages

Step 1: $\Delta t = N/(SB)$

Step 2: $\Delta t = N/(SB)$

BROADCAST

with unidirectional ring



Split data into S messages

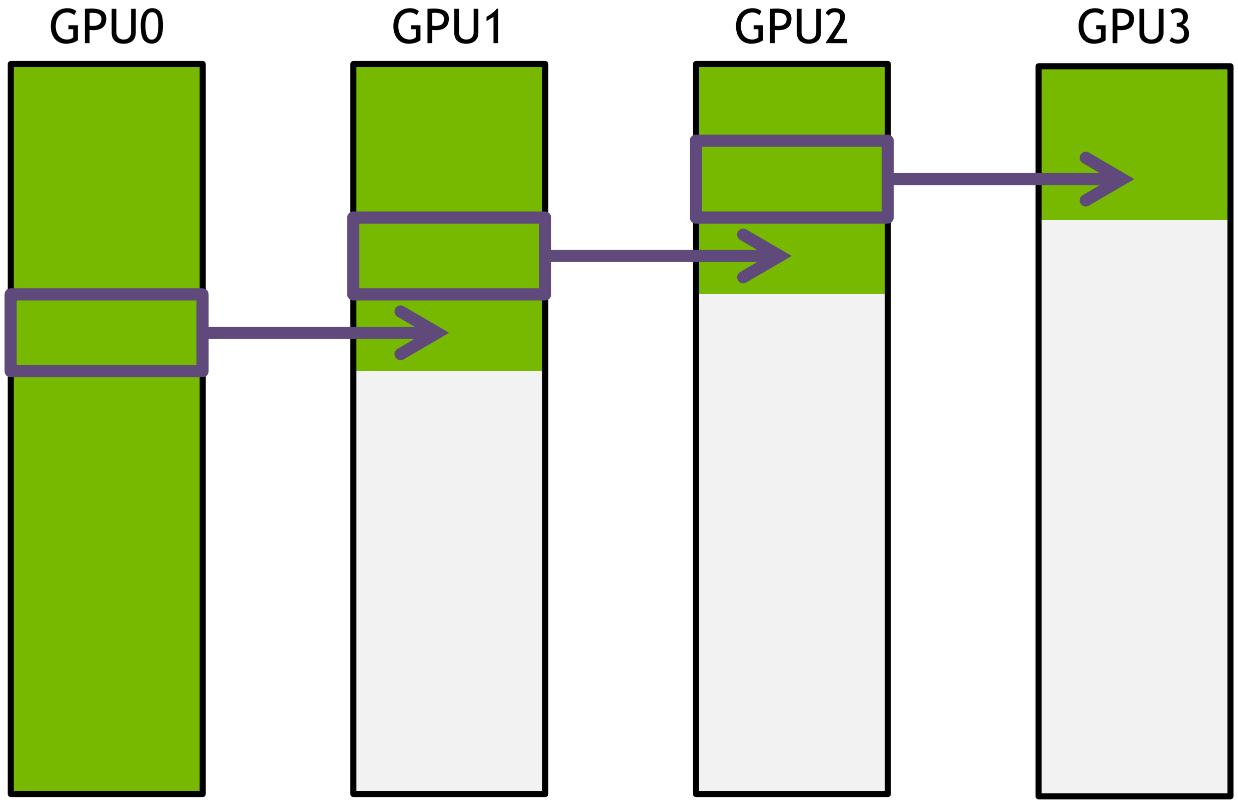
Step 1: $\Delta t = N/(SB)$

Step 2: $\Delta t = N/(SB)$

Step 3: $\Delta t = N/(SB)$

BROADCAST

with unidirectional ring



Split data into S messages

Step 1: $\Delta t = N/(SB)$

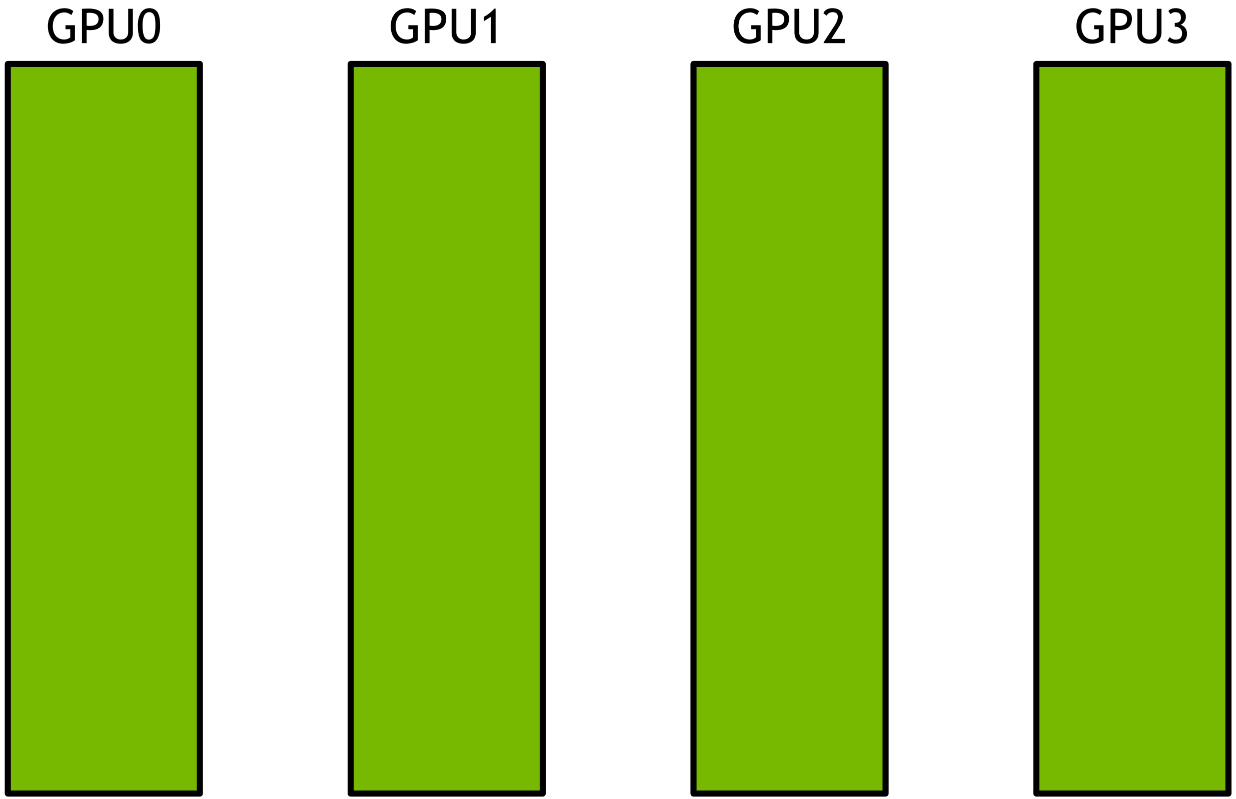
Step 2: $\Delta t = N/(SB)$

Step 3: $\Delta t = N/(SB)$

Step 4: $\Delta t = N/(SB)$

BROADCAST

with unidirectional ring



Split data into S messages

Step 1: $\Delta t = N/(SB)$

Step 2: $\Delta t = N/(SB)$

Step 3: $\Delta t = N/(SB)$

Step 4: $\Delta t = N/(SB)$

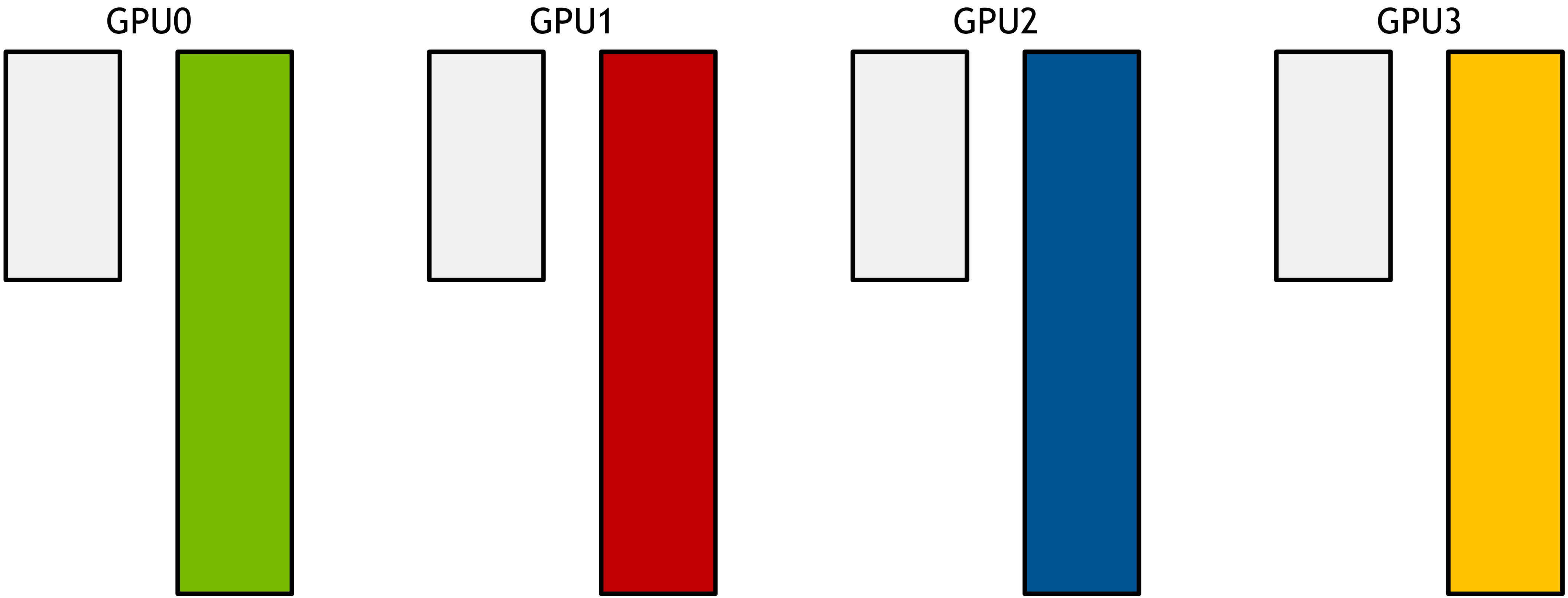
...

Total time:
$SN/(SB) + (k-2)\,N/(SB)$
$= N(S+k-2)/(SB) \rightarrow N/B$

ALL-REDUCE

with unidirectional ring

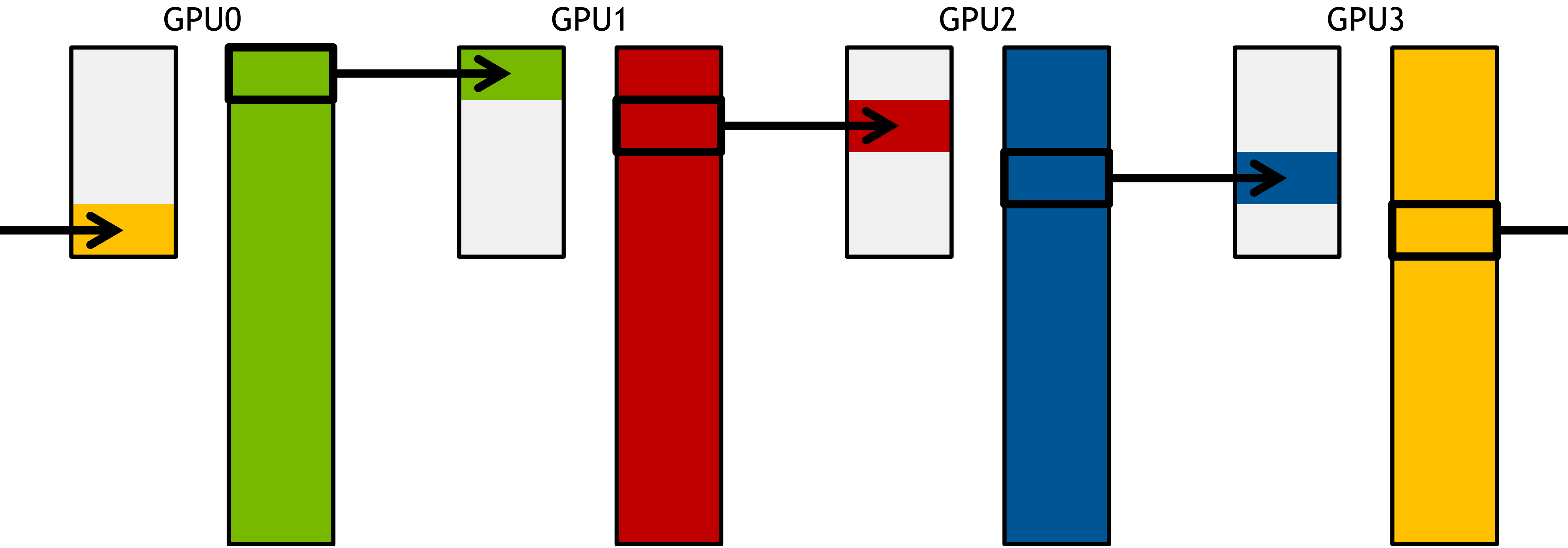
Chunk: 1
Step:



ALL-REDUCE

with unidirectional ring

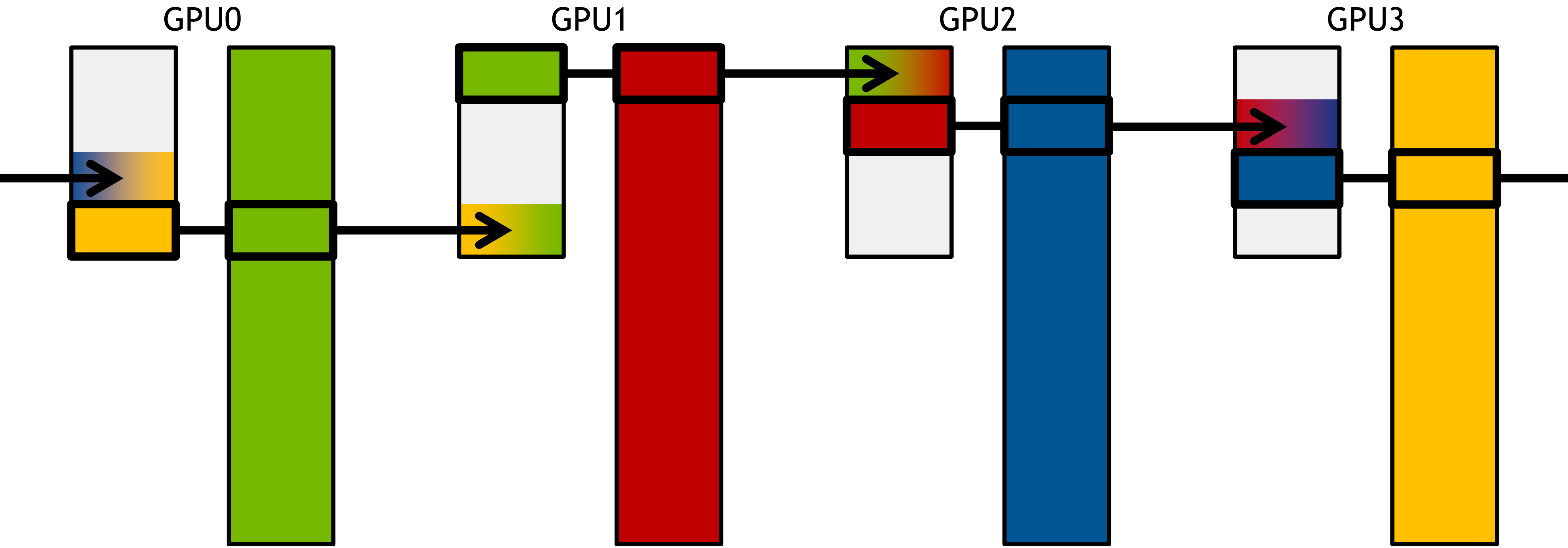
Chunk: 1
Step: 1



ALL-REDUCE

with unidirectional ring

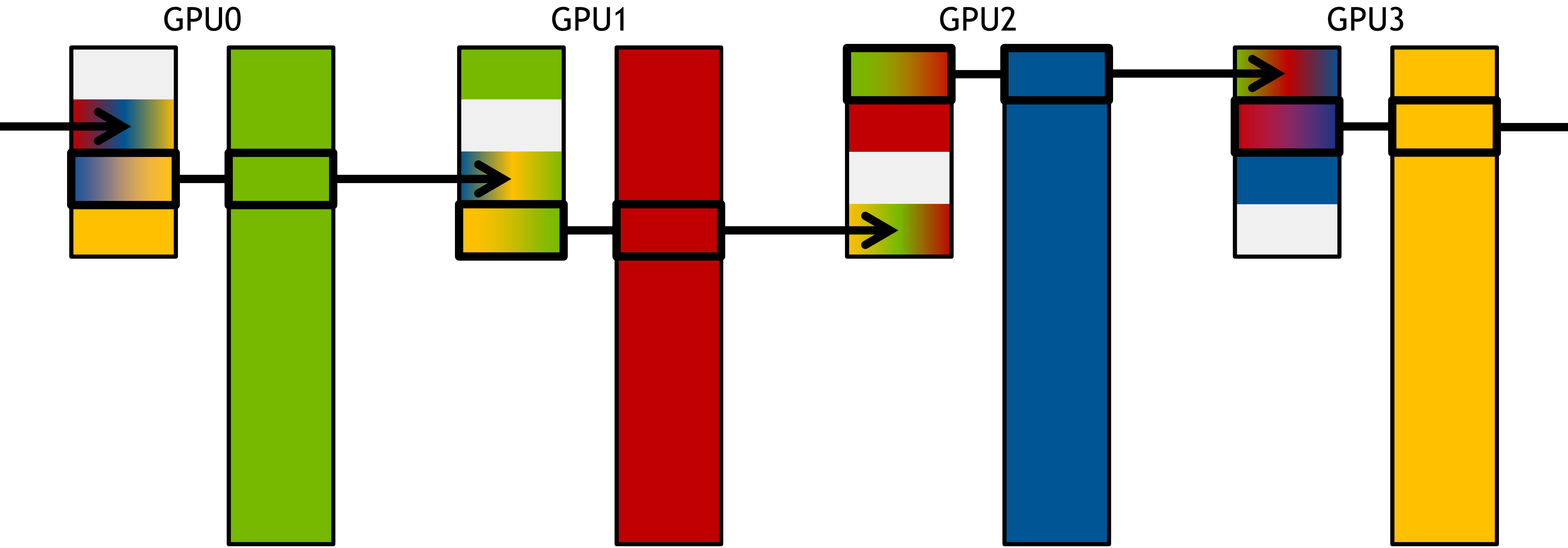
Chunk: 1
Step: 2



ALL-REDUCE

with unidirectional ring

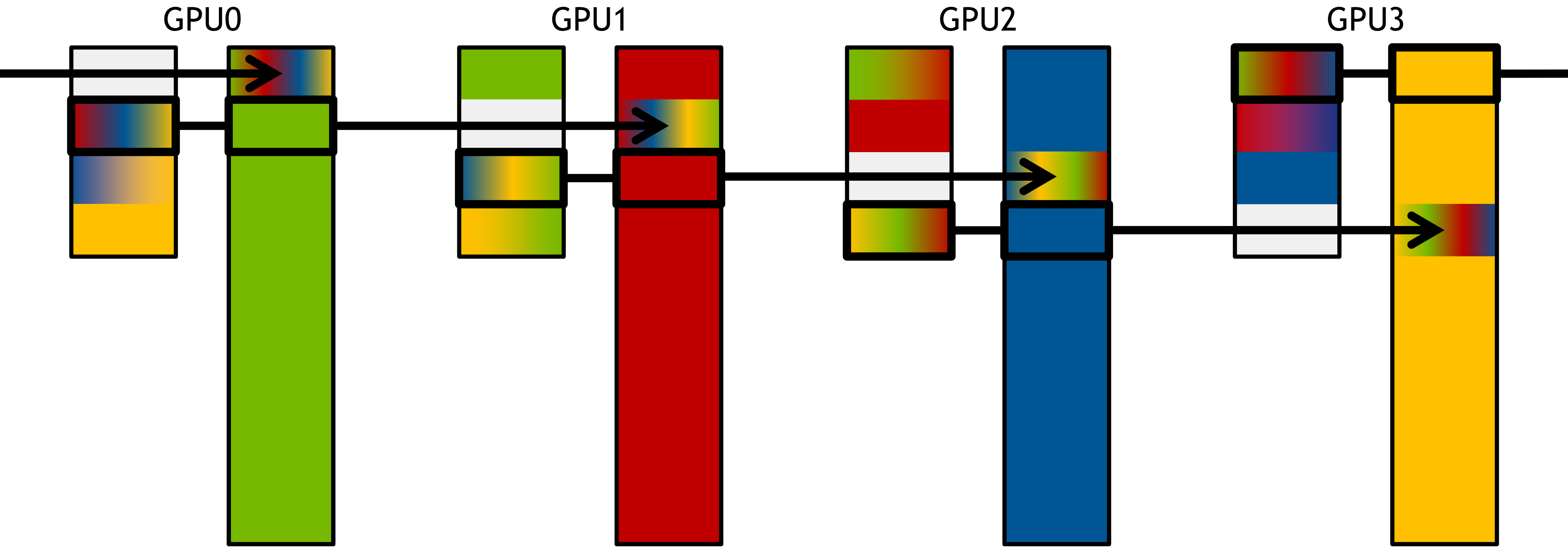
Chunk: 1
Step: 3



ALL-REDUCE

with unidirectional ring

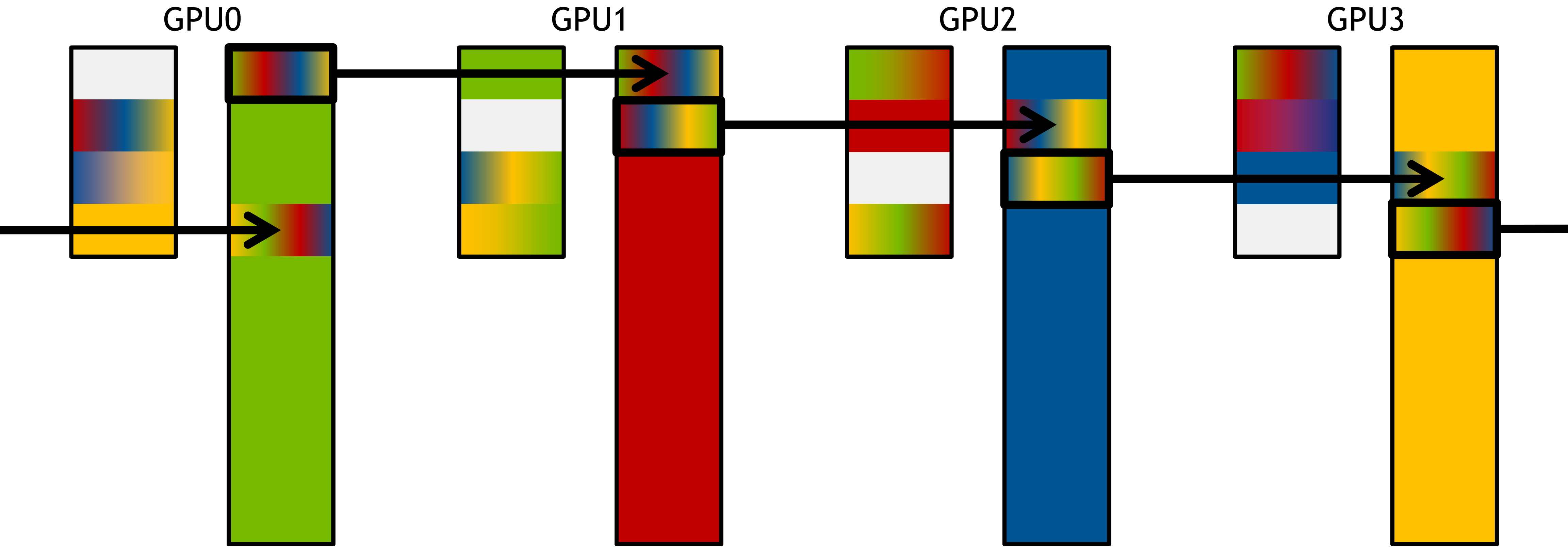
Chunk: 1
Step: 4



ALL-REDUCE

with unidirectional ring

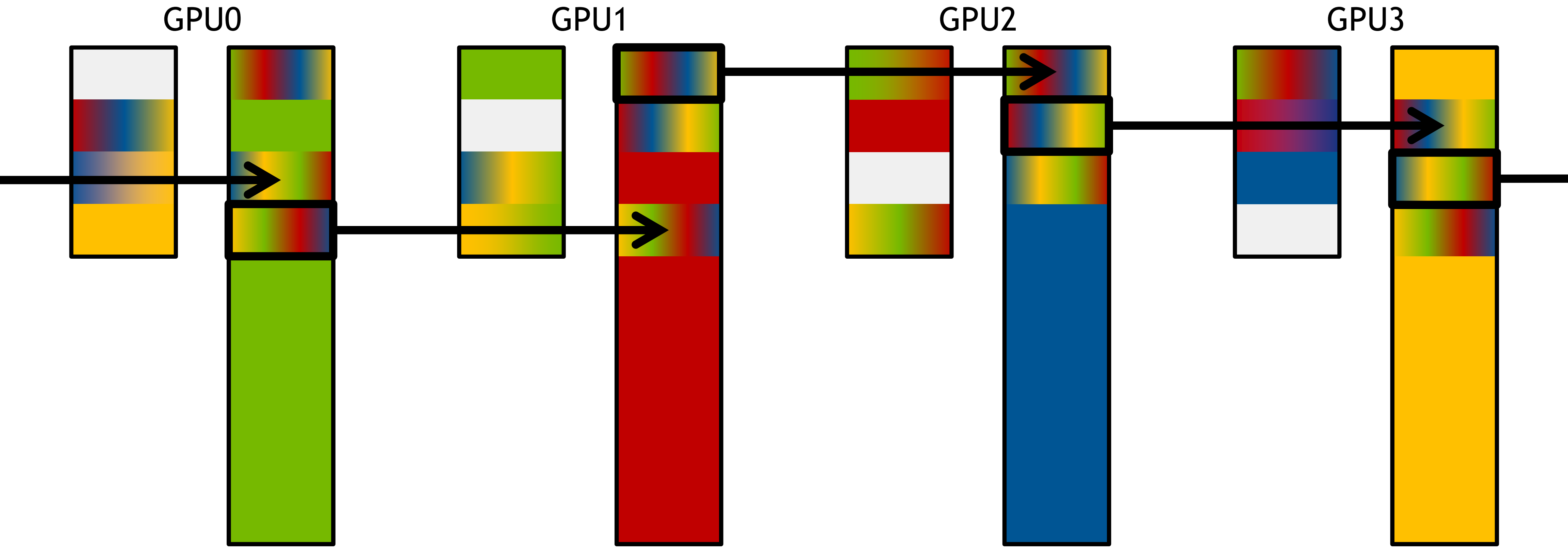
Chunk: 1
Step: 5



ALL-REDUCE

with unidirectional ring

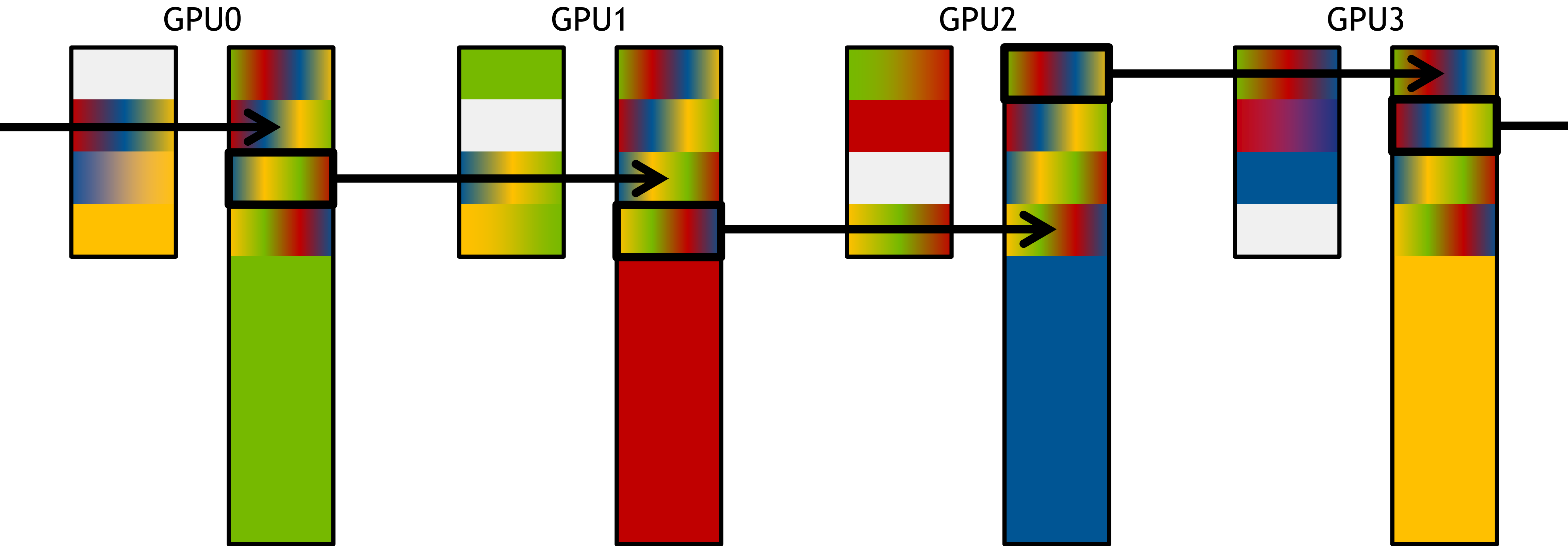
Chunk: 1
Step: 6



ALL-REDUCE

with unidirectional ring

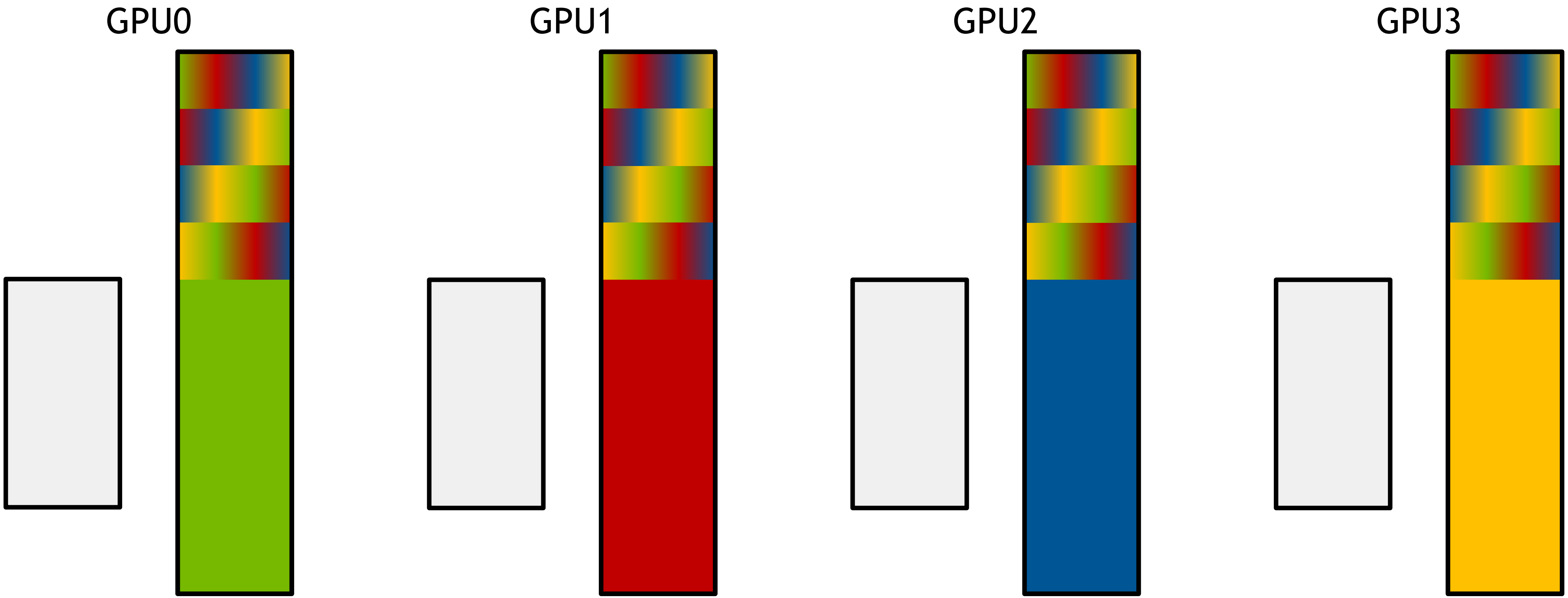
Chunk: 1
Step: 7



ALL-REDUCE

with unidirectional ring

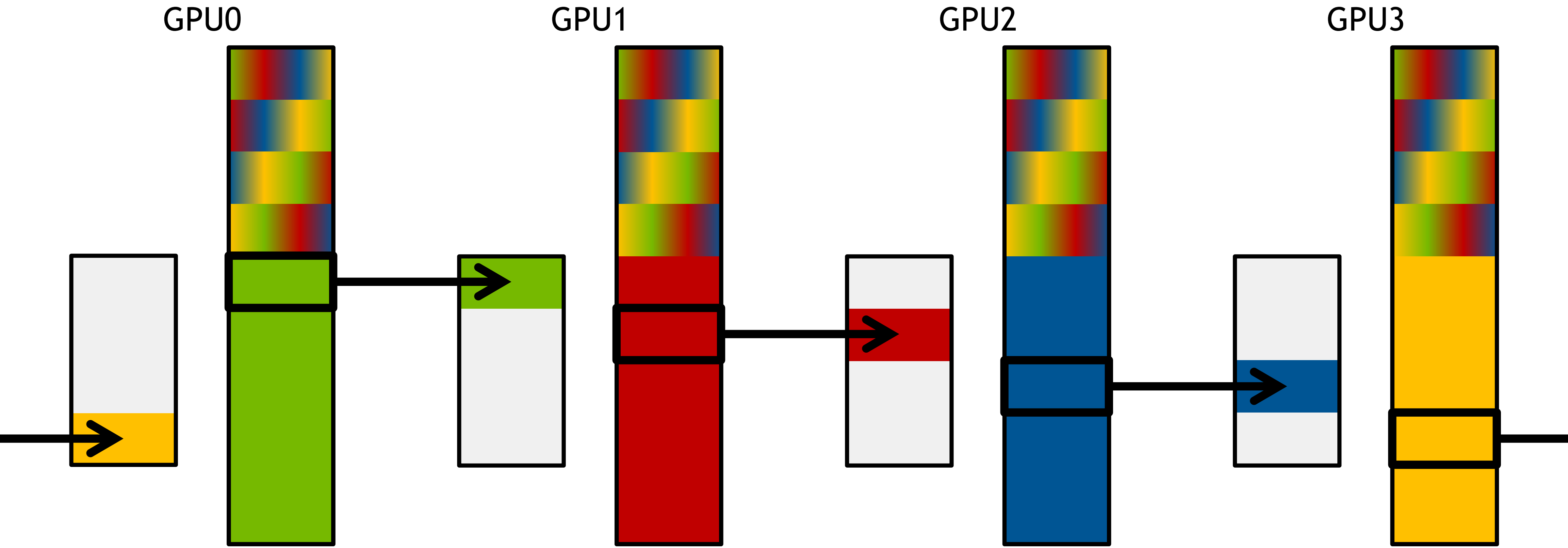
Chunk: 2
Step:



ALL-REDUCE

with unidirectional ring

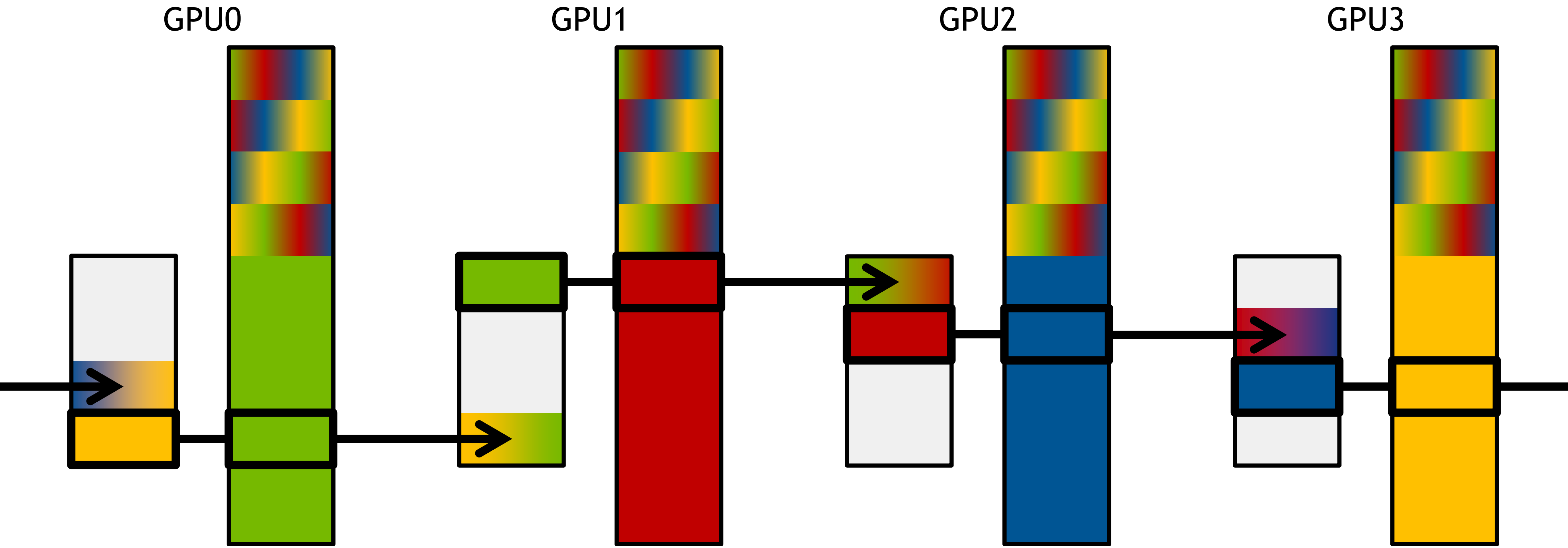
Chunk: 2
Step: 1



ALL-REDUCE

with unidirectional ring

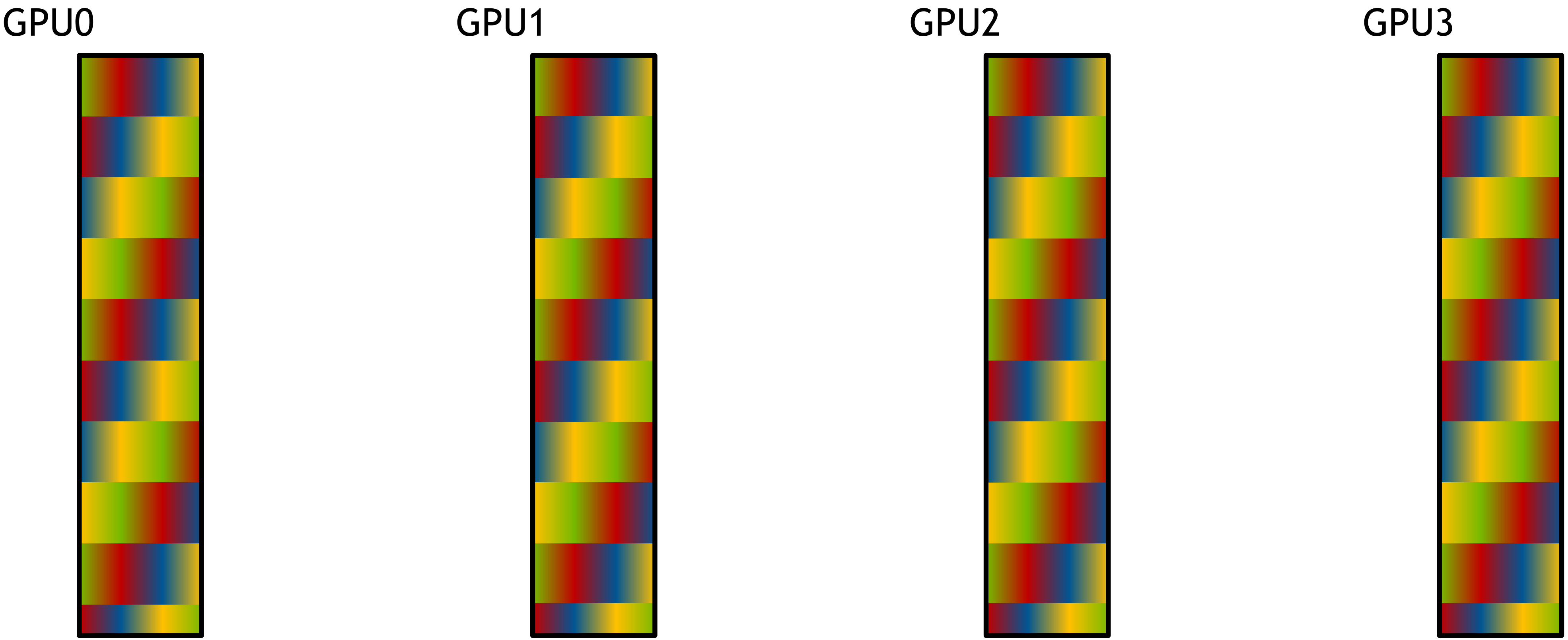
Chunk: 2
Step: 2



ALL-REDUCE

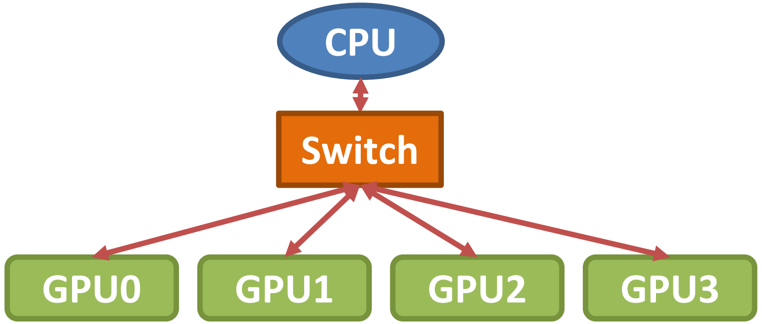
with unidirectional ring

done

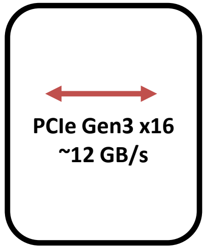


RING-BASED COLLECTIVES

A primer

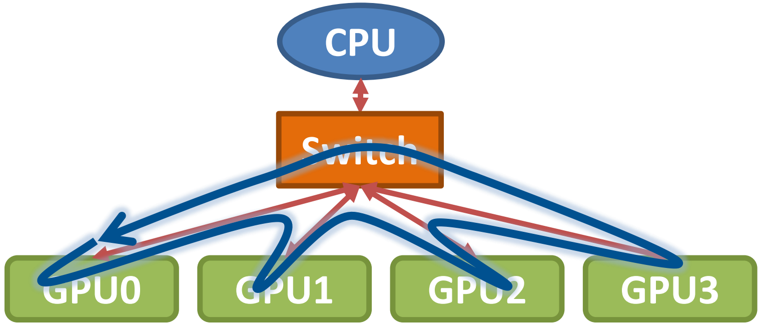


4-GPU-PCIe

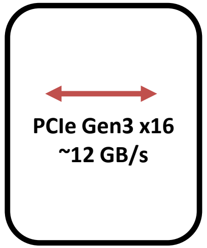


RING-BASED COLLECTIVES

A primer

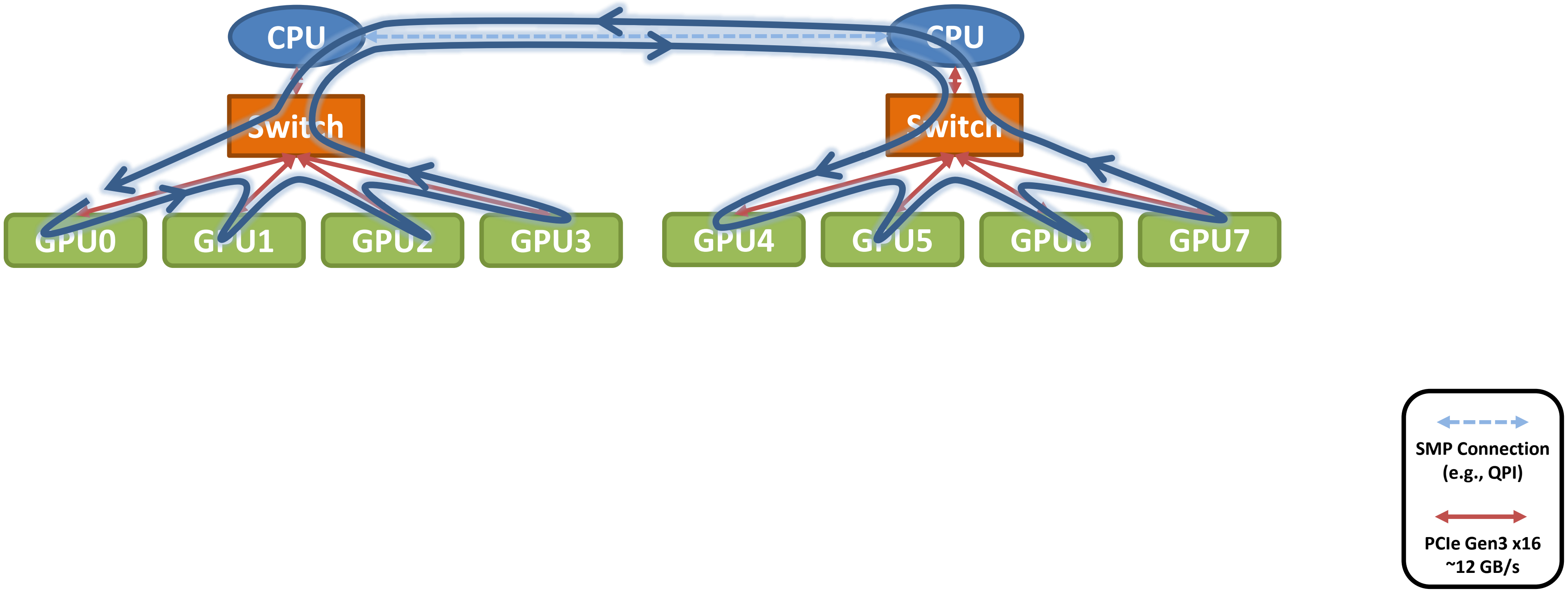


4-GPU-PCIe



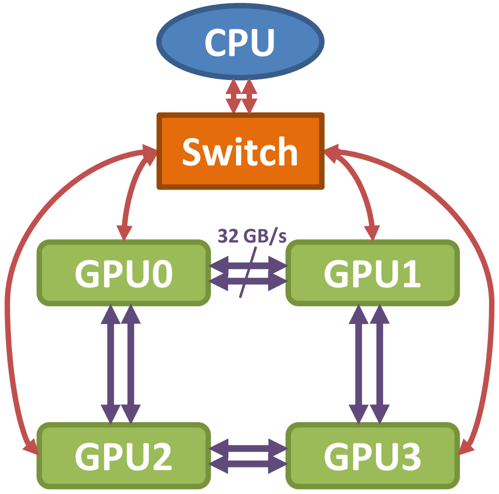
RING-BASED COLLECTIVES

…apply to lots of possible topologies

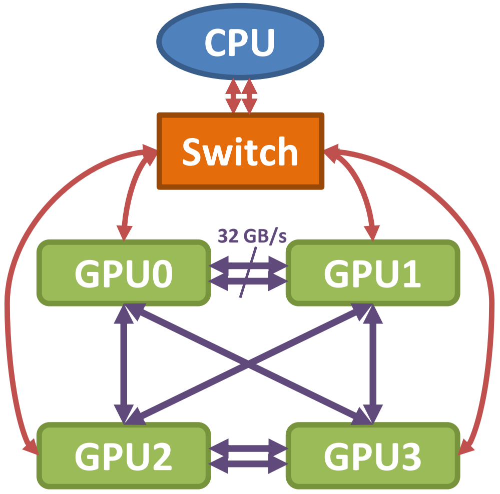


RING-BASED COLLECTIVES

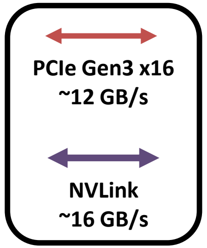
…apply to lots of possible topologies



4-GPU-Ring

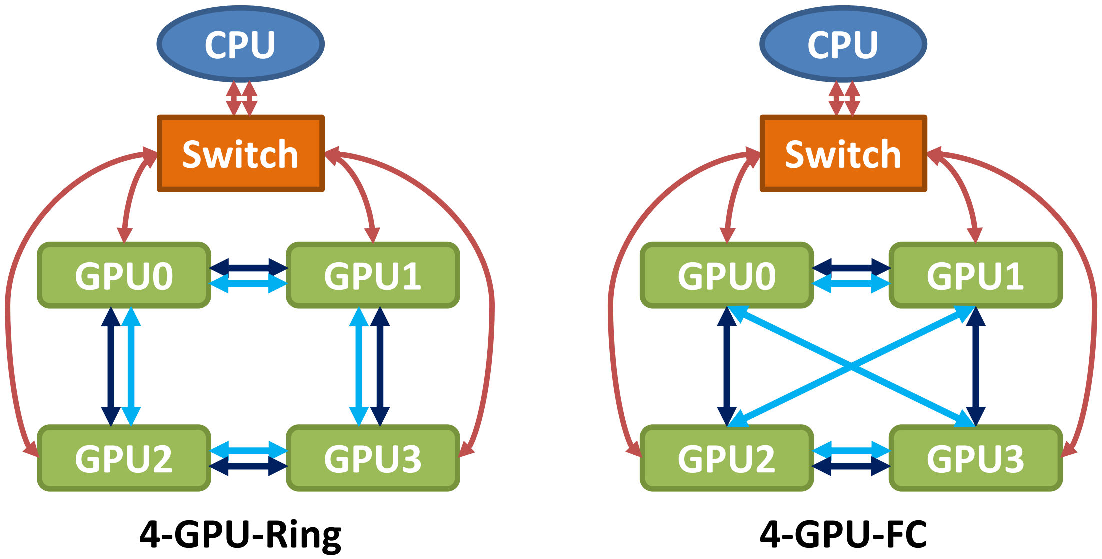


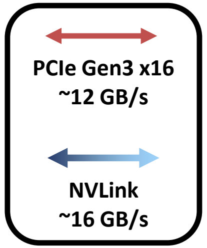
4-GPU-FC



RING-BASED COLLECTIVES

…apply to lots of possible topologies





INTRODUCING NCCL (“NICKEL”): ACCELERATED COLLECTIVES FOR MULTI-GPU SYSTEMS

INTRODUCING NCCL

Accelerating multi-GPU collective communications

GOAL:

- Build a research library of accelerated collectives that is easily integrated and topology-aware so as to improve the scalability of multi-GPU applications

APPROACH:

- Pattern the library after MPI's collectives
- Handle the intra-node communication in an optimal way
- Provide the necessary functionality for MPI to build on top to handle inter-node

NCCL FEATURES AND FUTURES

(Green = Currently available)

Collectives

- Broadcast
- All-Gather
- Reduce
- All-Reduce
- Reduce-Scatter
- Scatter
- Gather
- All-To-All
- Neighborhood

Key Features

- Single-node, up to 8 GPUs
- Host-side API
- Asynchronous/non-blocking interface
- Multi-thread, multi-process support
- In-place and out-of-place operation
- Integration with MPI
- Topology Detection
- NVLink & PCIe/QPI* support

NCCL IMPLEMENTATION

Implemented as monolithic CUDA C++ kernels combining the following:

- GPUDirect P2P Direct Access
- Three primitive operations: Copy, Reduce, ReduceAndCopy
- Intra-kernel synchronization between GPUs
- One CUDA thread block per ring-direction

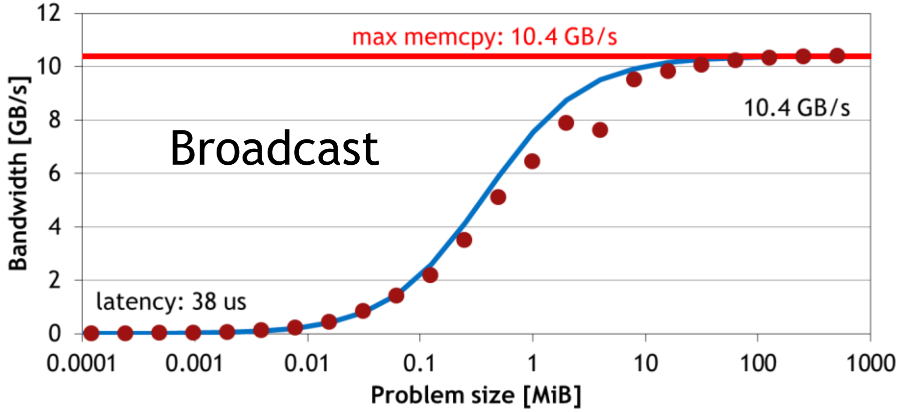
NCCL EXAMPLE

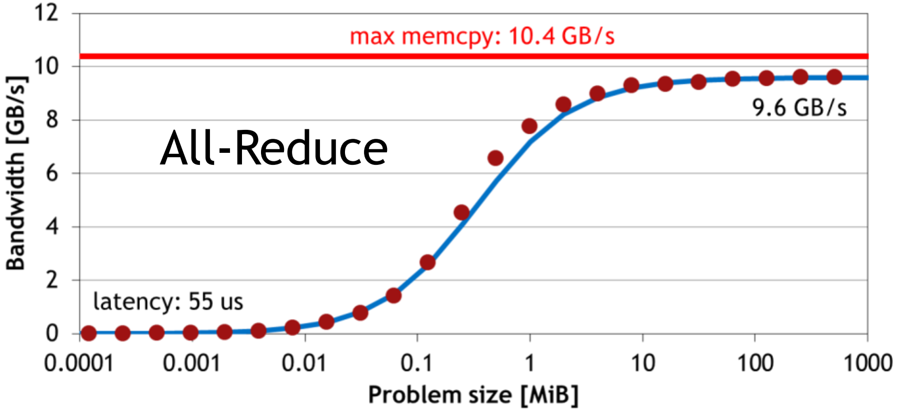
All-reduce

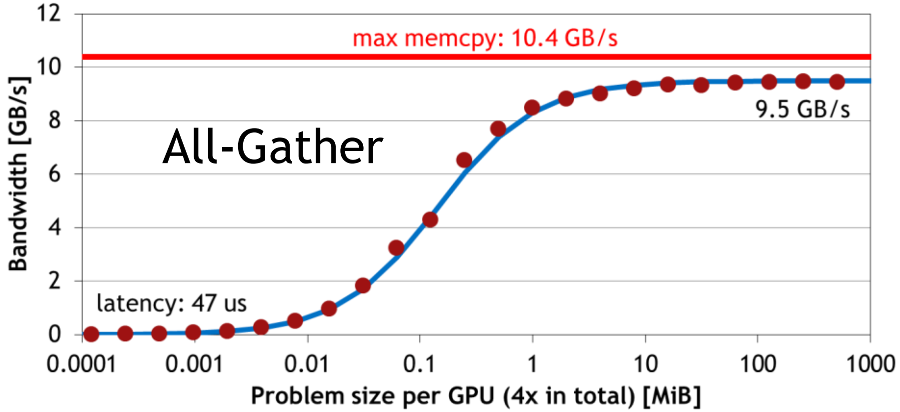
```
#include <nccl.h>
ncclComm_t comm[4];
ncclCommInitAll(comm, 4, {0, 1, 2, 3});
foreach g in (GPUs) { // or foreach thread
  cudaSetDevice(g);
  double *d_send, *d_recv;
  // allocate d_send, d_recv; fill d_send with data
  ncclAllReduce(d_send, d_recv, N, ncclDouble, ncclSum, comm[g], stream[g]);
  // consume d_recv
}
```

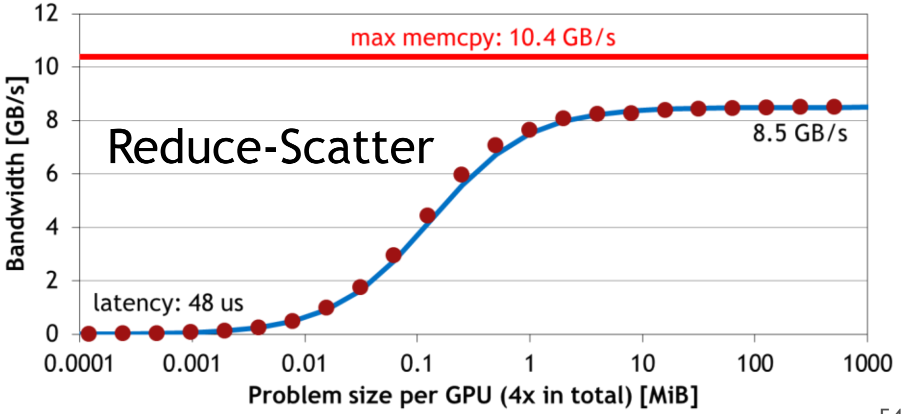
NCCL PERFORMANCE

Bandwidth at different problem sizes (4 Maxwell GPUs)









AVAILABLE NOW
github.com/NVIDIA/nccl

THANKS TO MY COLLABORATORS

Nathan Luehr

Jonas Lippuner

Przemek Tredak

Sylvain Jeaugey

Natalia Gimelshein

Simon Layton

This research is funded in part by the U.S. DOE DesignForward program

NVIDIA