Scaling Deep Learning and Datacenter Applications with Programmable Networks

Marco Canini

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

In-Network Computation is a Dumb Idea Whose Time Has Come

HotNets '17

Amedeo Sapio‡, Ibrahim Abdelaziz, Abdulla Aldilaijan,
Marco Canini, Panos Kalnis
KAUST

ABSTRACT

Programmable data plane hardware creates new opportunities for infusing intelligence into the network. This raises a fundamental question: what kinds of computation should be delegated to the network?

In this paper, we discuss the opportunities and challenges for co-designing data center distributed systems with their network layer. We believe that the time has finally come for offloading part of their computation to execute in-network. However, in-network computation tasks must be judiciously crafted to match the limitations of the network machine architecture of programmable devices. With the help of our experiments on machine learning and graph analytics workloads, we identify that aggregation functions raise opportunities to exploit the limited computation power of networking hardware to lessen network congestion and improve the overall application performance. Moreover, as a proof-of-concept, we propose DAIET, a system that performs in-network data aggregation. Experimental results with an initial prototype show a large data reduction ratio (86.9%-89.3%) and a similar decrease in the workers' computation time.

Programmable networks create the opportunity for in-

Is this a dumb idea?

- increased complexity
- new kinds of failure modes
- could affect correctness
- will put application-specific logic in the network…

ing incarnation in the Barefoot Networks' Tofino [3] switch chip has a flexible parser and a customizable match-action engine. To process packets at high speed, this architecture has a multi-stage pipeline where packets flow at line rate. Each stage has a fixed amount of time to process every packet, allowing for lookups in memory (SRAM and TCAM), manip-

In-Network Computation is a Dumb Idea Whose Time Has Come

HotNets '17

Amedeo Sapio‡, Ibrahim Abdelaziz, Abdulla Aldilaijan,
Marco Canini, Panos Kalnis
KAUST

ABSTRACT

Programmable data plane hardware creates new opportunities for infusing intelligence into the network. This raises a fundamental question: what kinds of computation should be delegated to the network?

In this paper, we discuss the opportunities and challenges for co-designing data center distributed systems with their network layer. We believe that the time has finally come for offloading part of their computation to execute in-network. However, in-network computation tasks must be judiciously crafted to match the limitations of the network machine architecture of programmable devices. With the help of our experiments on machine learning and graph analytics workloads, we identify that aggregation functions raise opportunities to exploit the limited computation power of networking hardware to lessen network congestion and improve the overall application performance. Moreover, as a proof-of-concept, we propose DAIET, a system that performs in-network data aggregation. Experimental results with an initial prototype show a large data reduction ratio (86.9%-89.3%) and a similar decrease in the workers' computation time.

Programmable networks create the opportunity for in-

What to compute in network?
When, where and how to do it?

… do it judiciously:

1. network traffic is significantly reduced; *application benefits significantly*
2. only a minimal change at the application level is required
3. the correctness of the overall computation is not affected

allowing for lookups in memory (SRAM and TCAM), manip-

This talk

Will focus on two common DC workloads:

1. Distributed Deep Learning

2. Key-Value Storage Systems

Deep Learning

Innovation fueled by leaps in (costly) infrastructure:

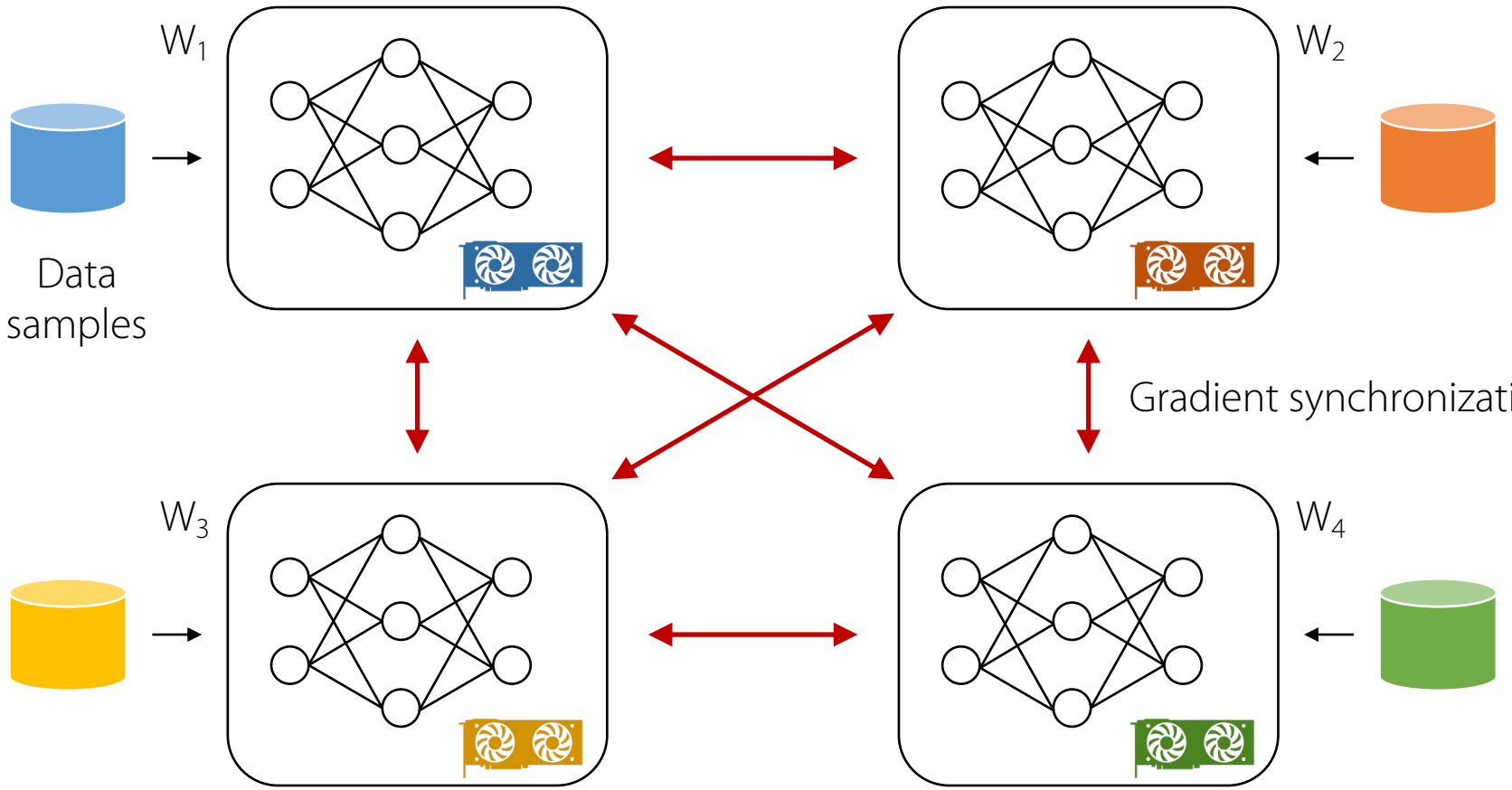
Clusters with hundreds of machines,

each with many HW accelerators (GPUs, TPUs, etc.)

Compute requirements **doubling every 3 months!**

Training models is still **very time-consuming**: days or even weeks!

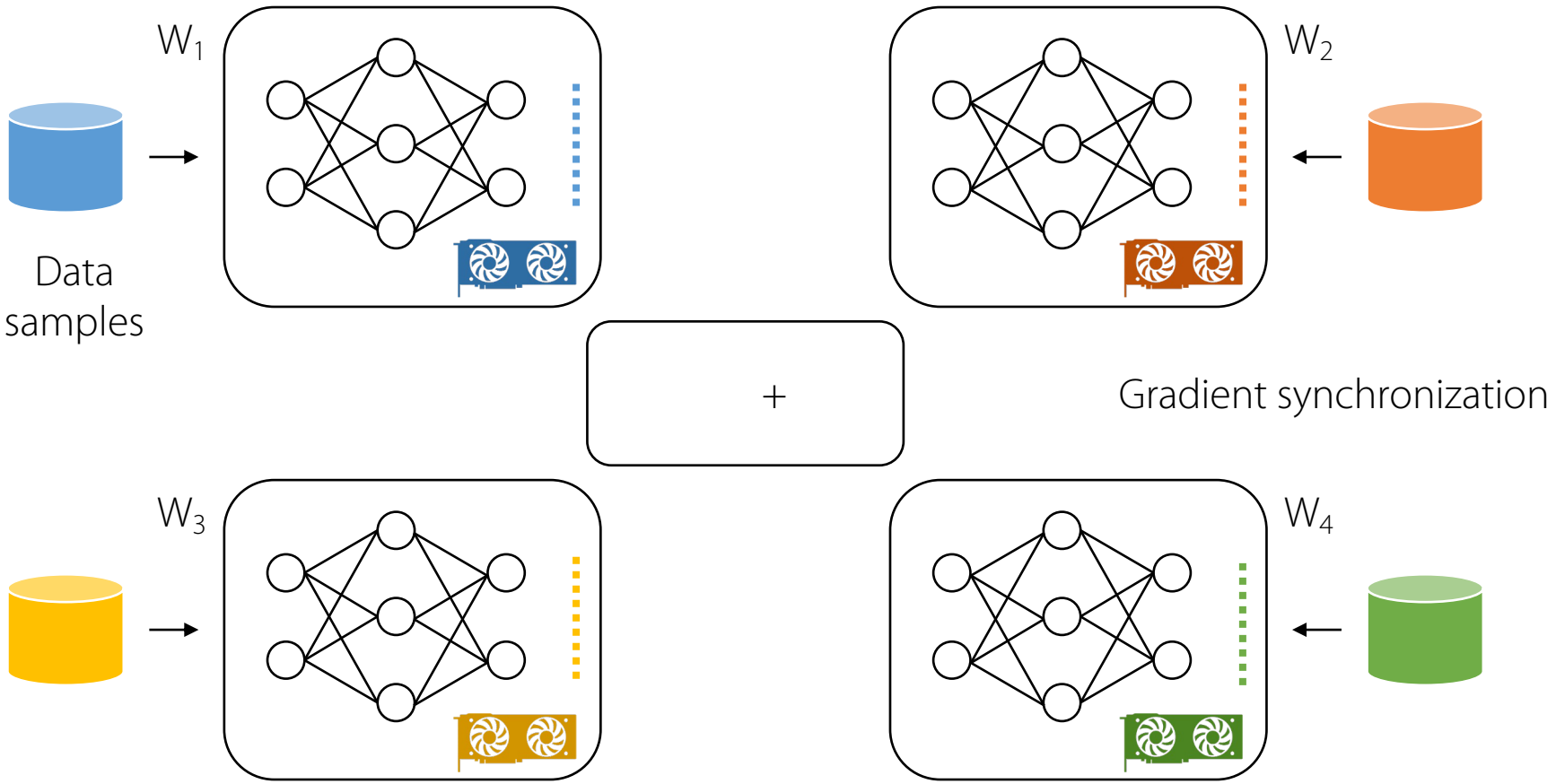
Data-parallel distributed DNN training



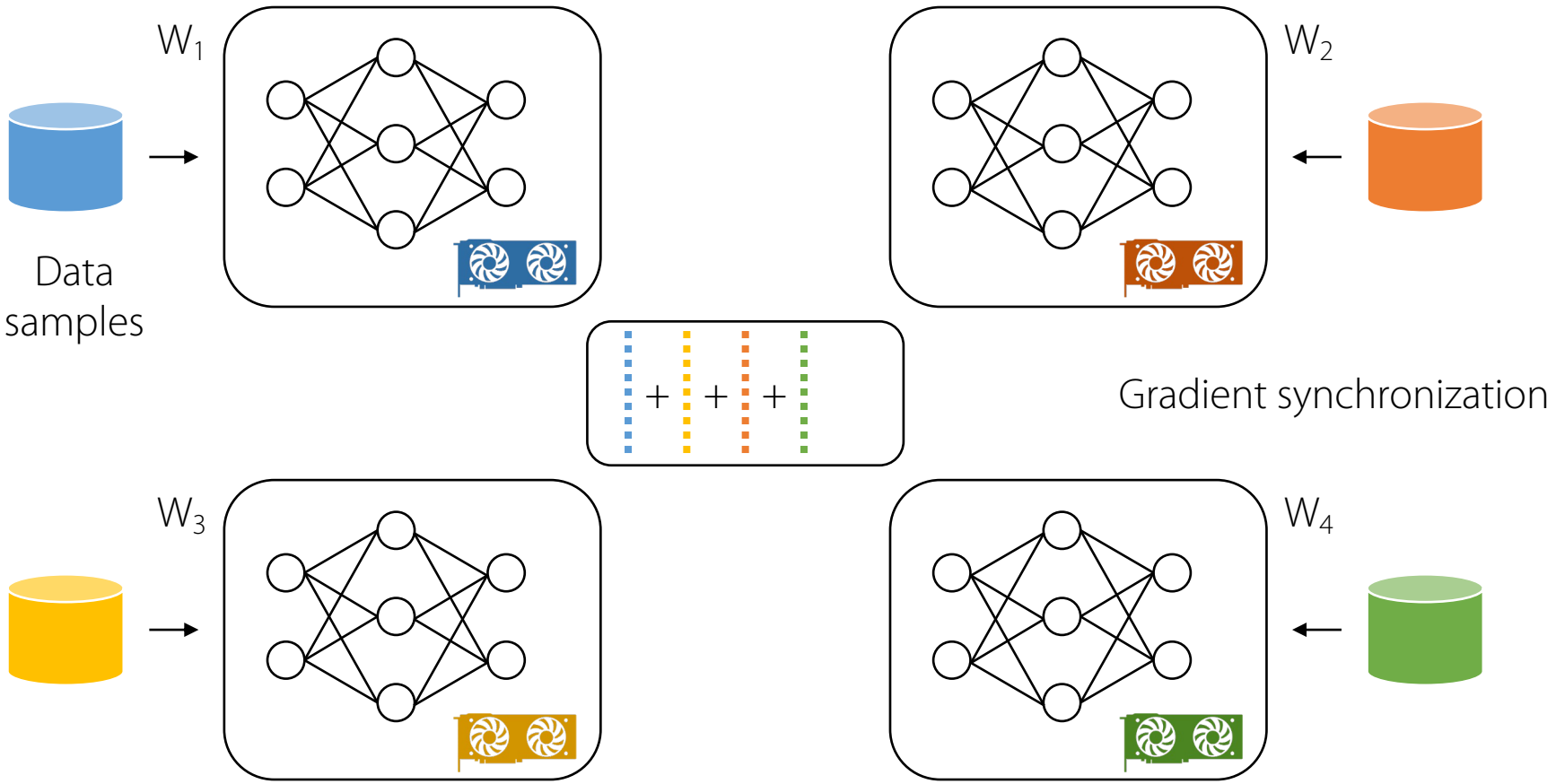
All-to-all intensive communication pattern

100s of MBs to GBs gradient synchronization in **each iteration**

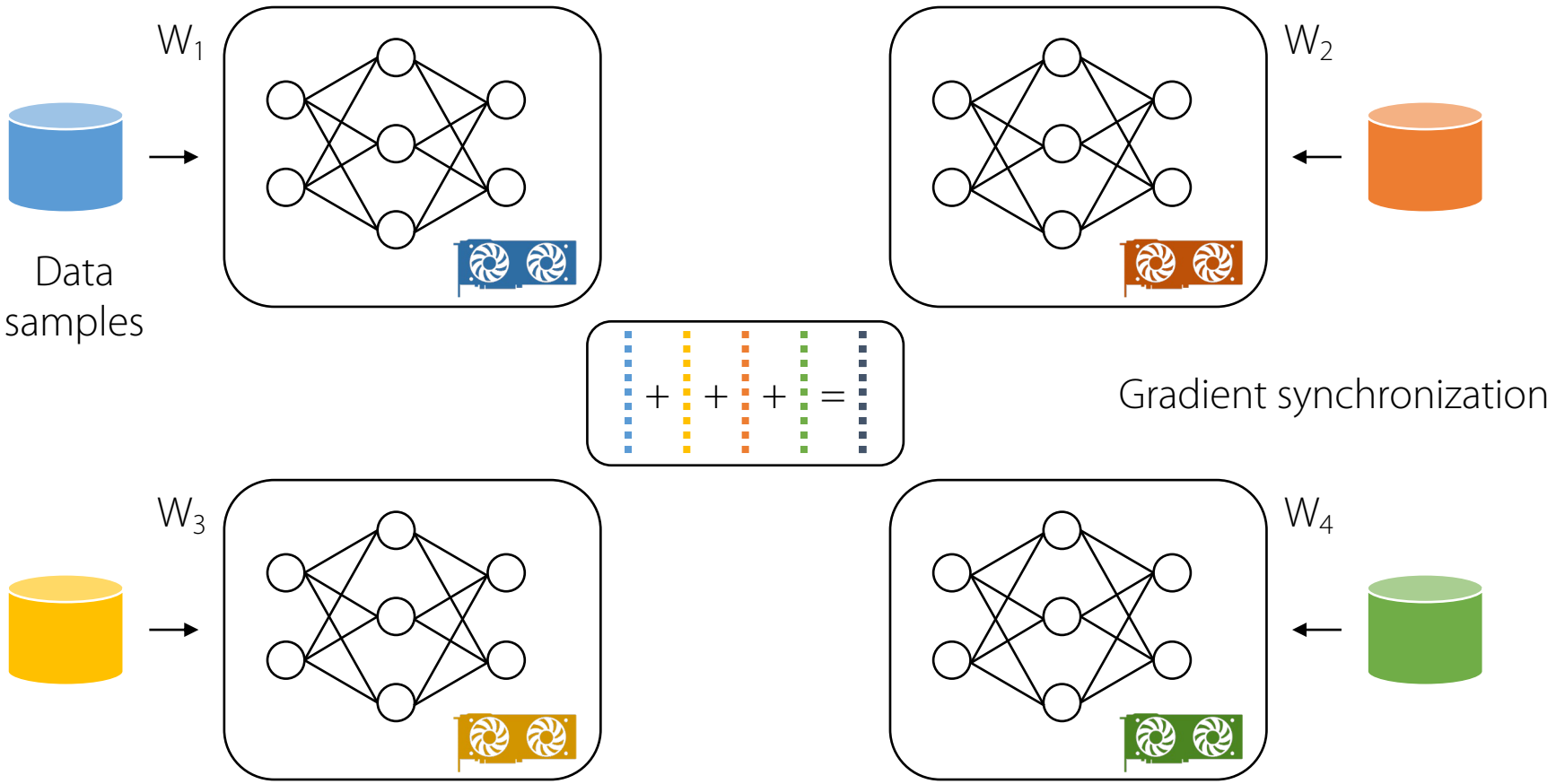
All-to-all reduction



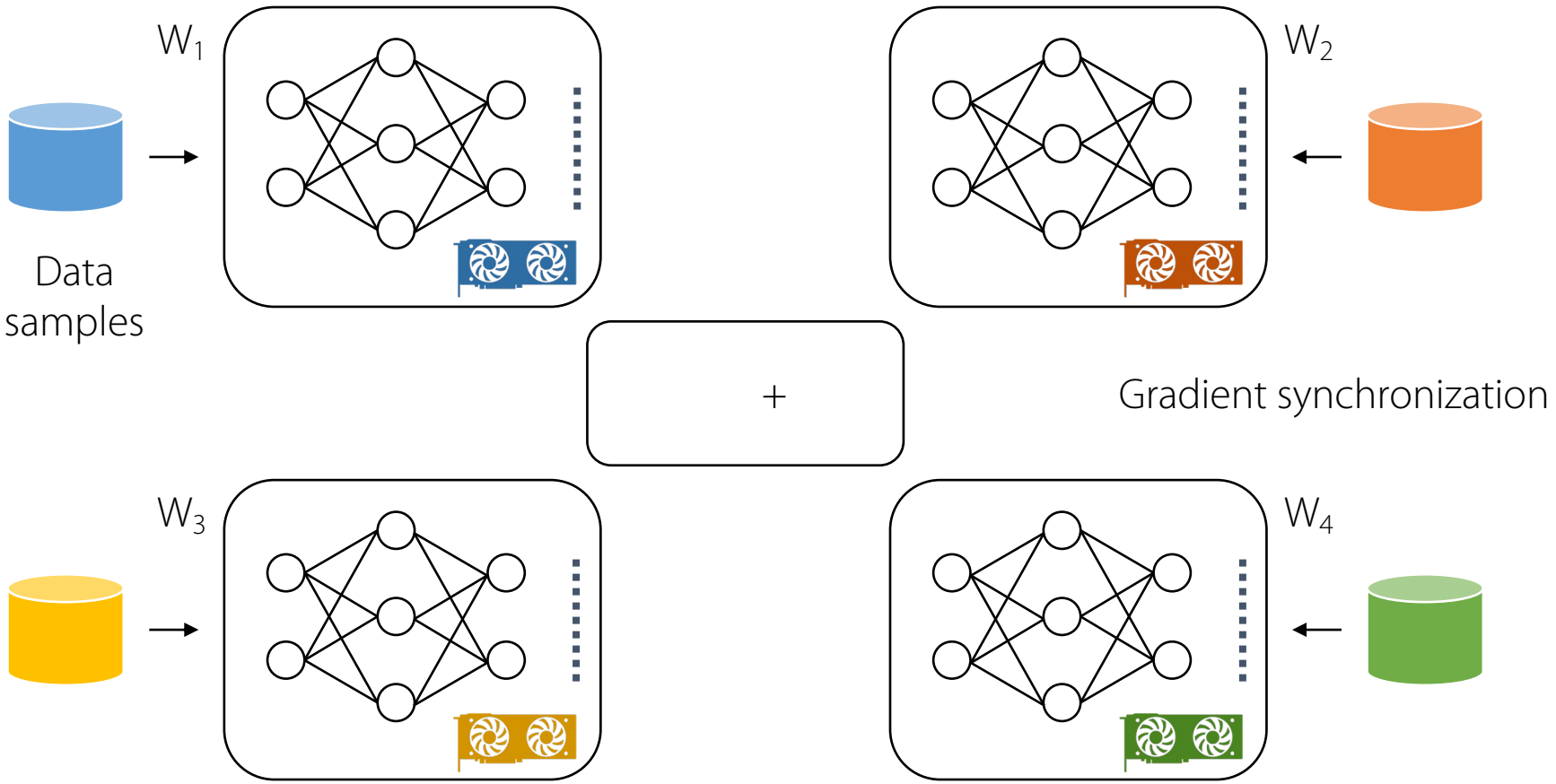
All-to-all reduction



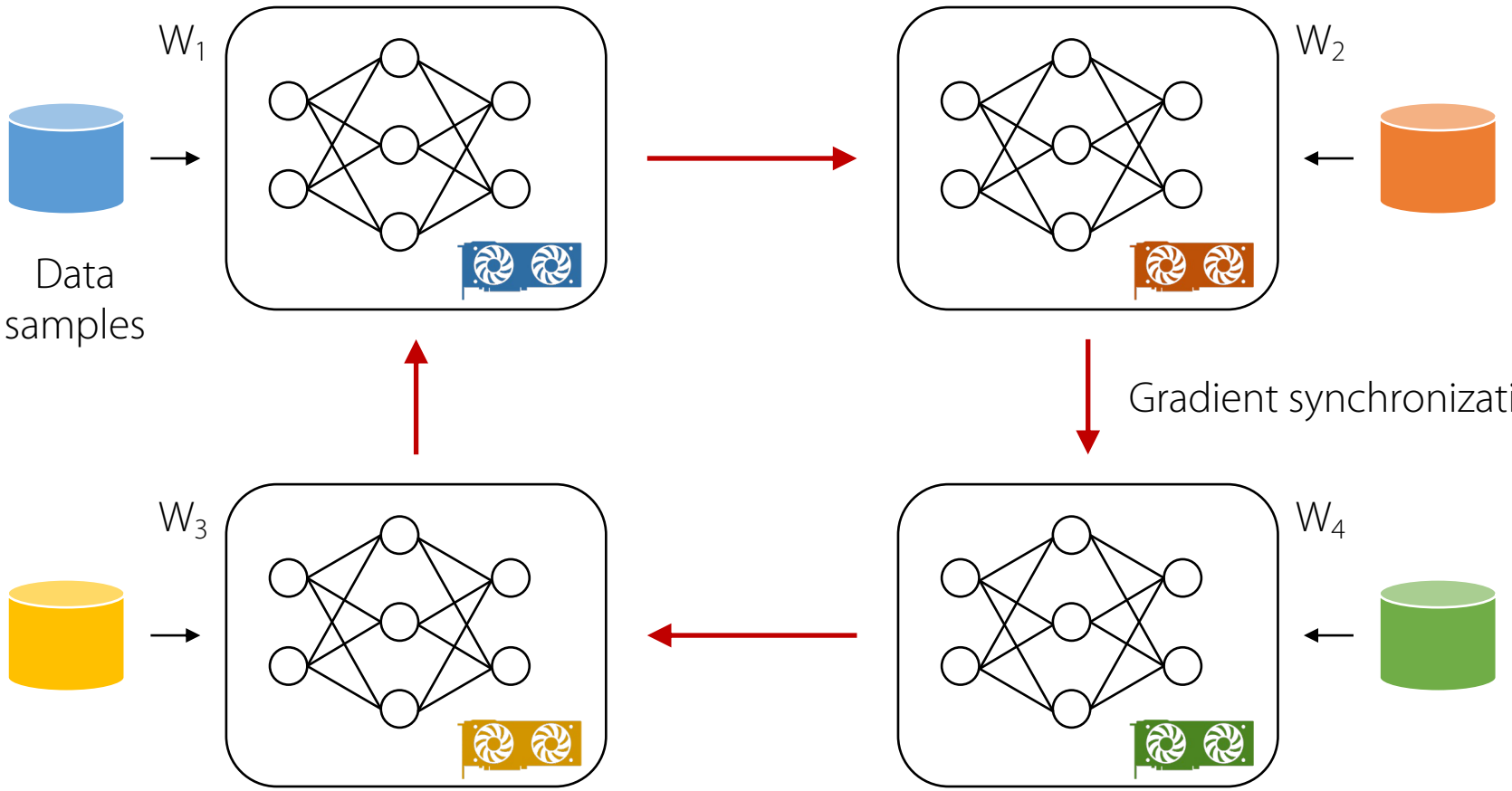
All-to-all reduction



All-to-all reduction

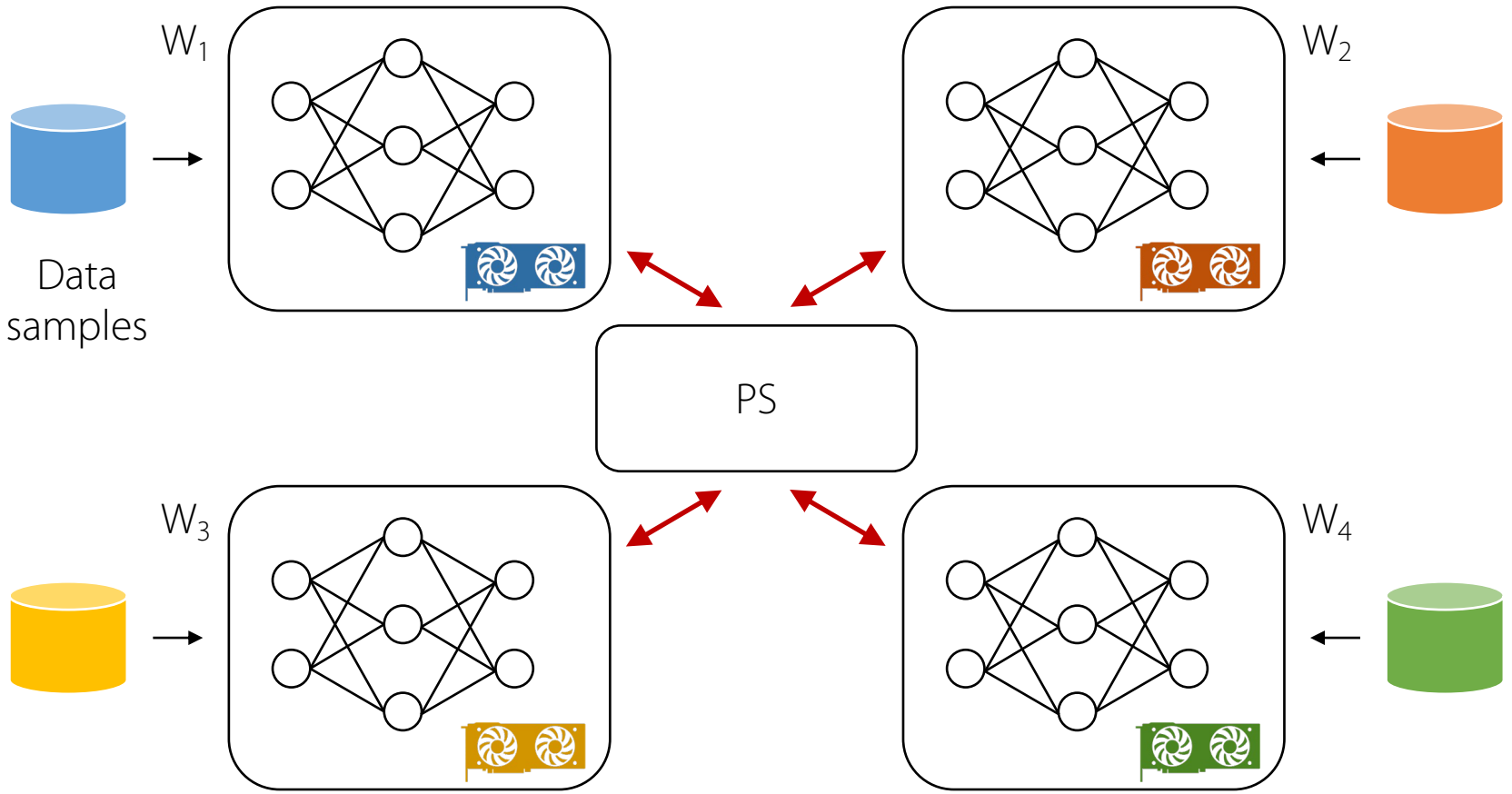


AllReduce



Form a logical ring (or tree, etc.) and run peer-to-peer communication

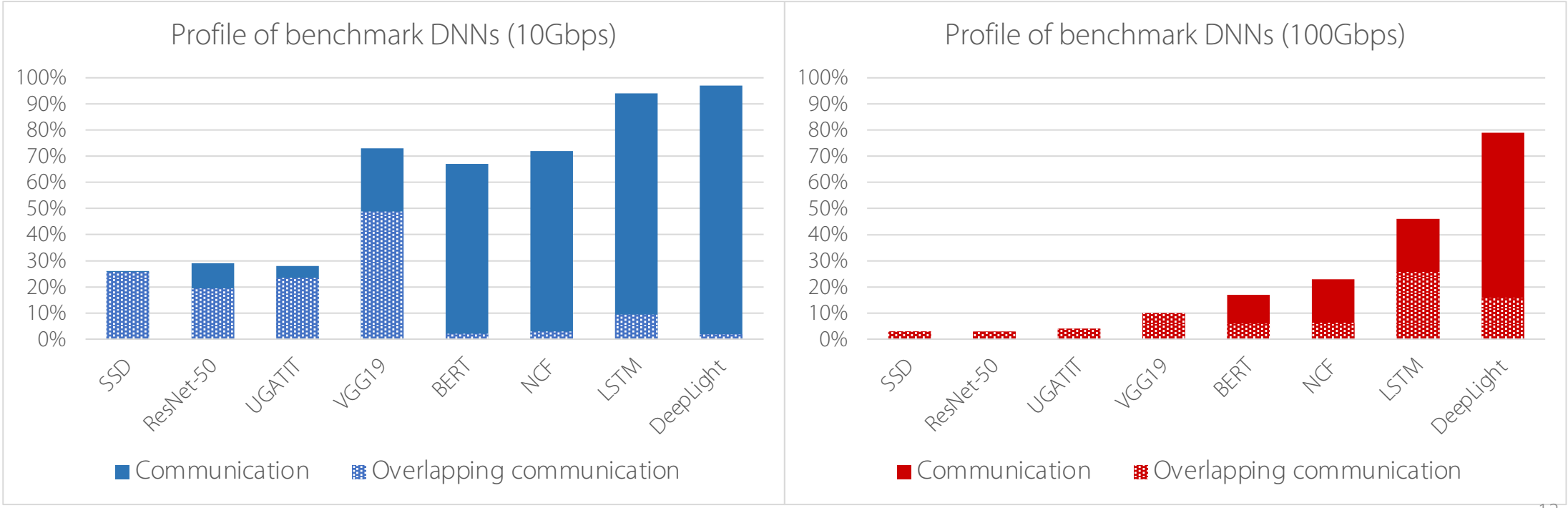
Parameter server (PS)



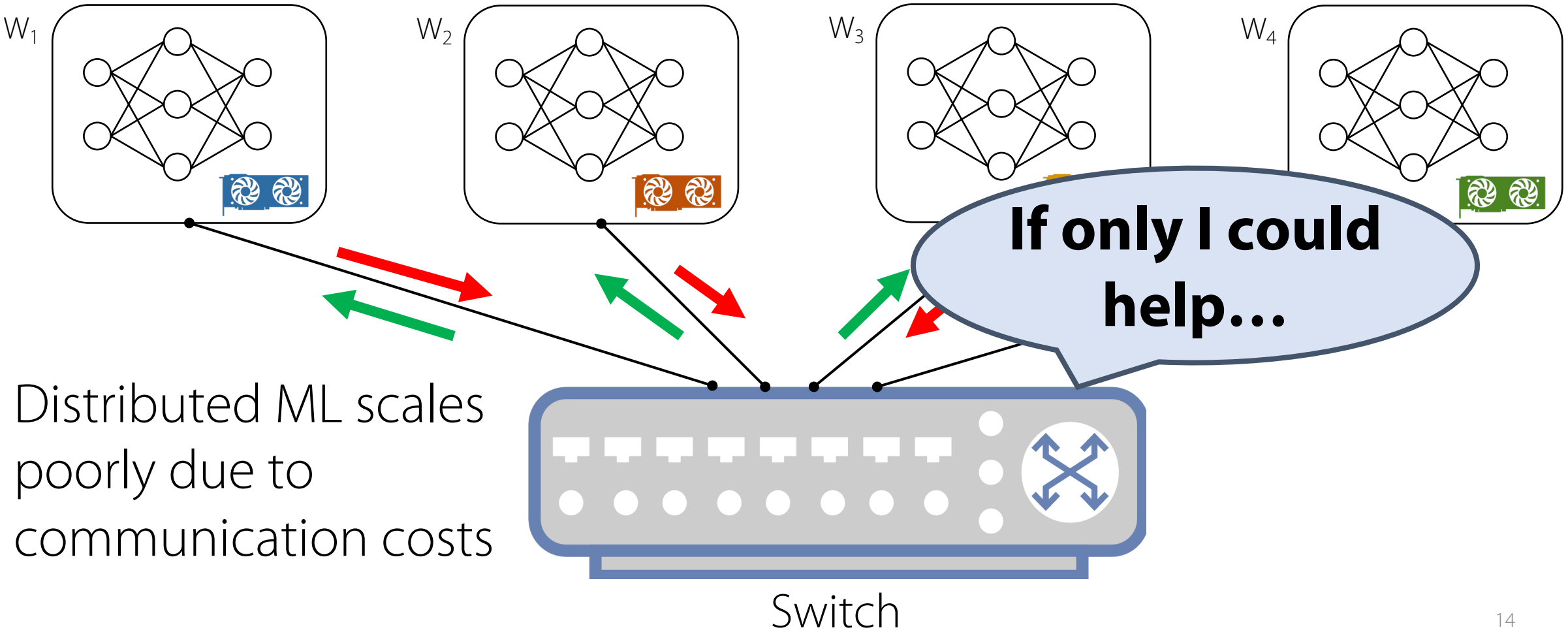
1. Push gradients
2. Aggregate updates at PS
3. Pull aggregated gradients
   - (or updated model parameters)

The network bottleneck

- Compute accelerators performance improvements have so far outpaced network bandwidth increases
- Newer, larger DNN models spend more time on communication



A closer look at model synchronization



Distributed ML scales poorly due to communication costs

SwitchML: Co-design ML and networking

Challenges

Limited computation

Limited storage

No floating points

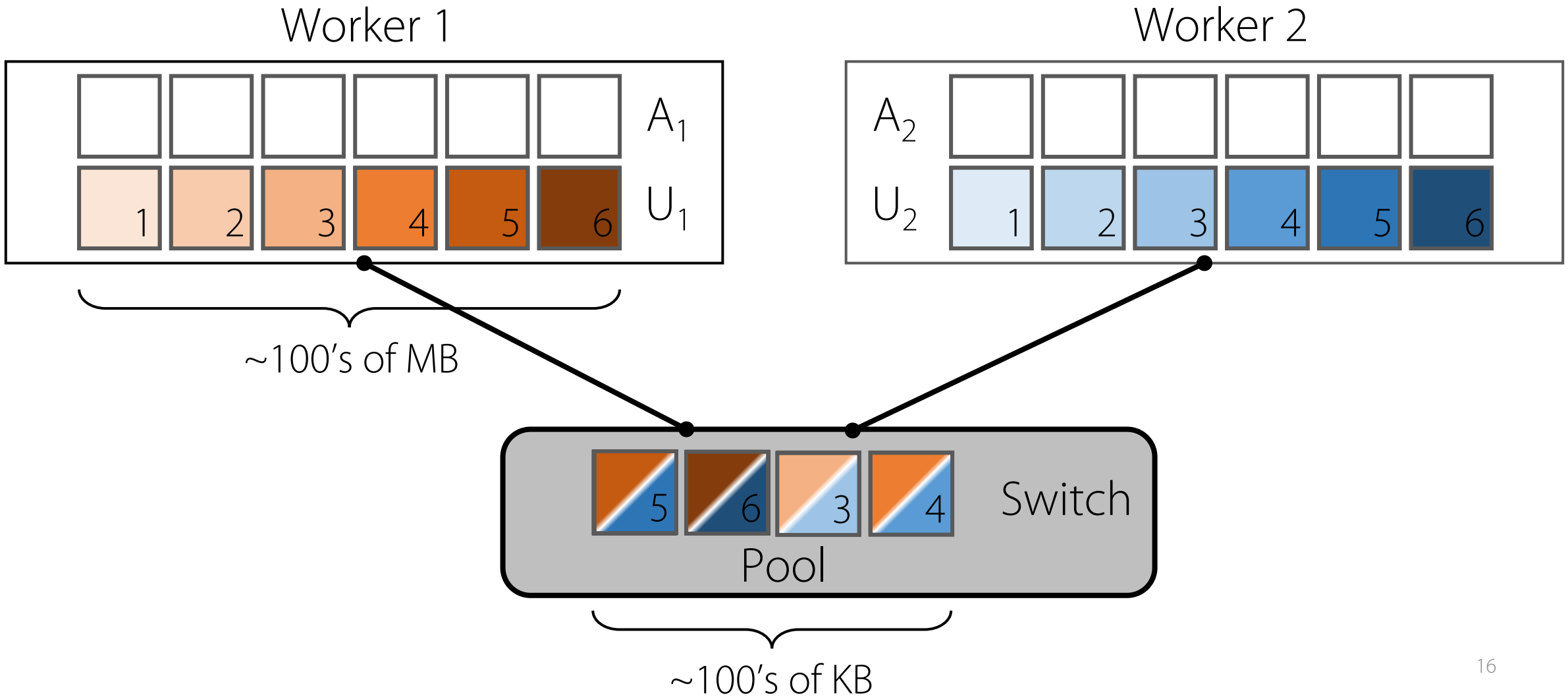
Packet loss

6.5 Tbps
programmable
data plane

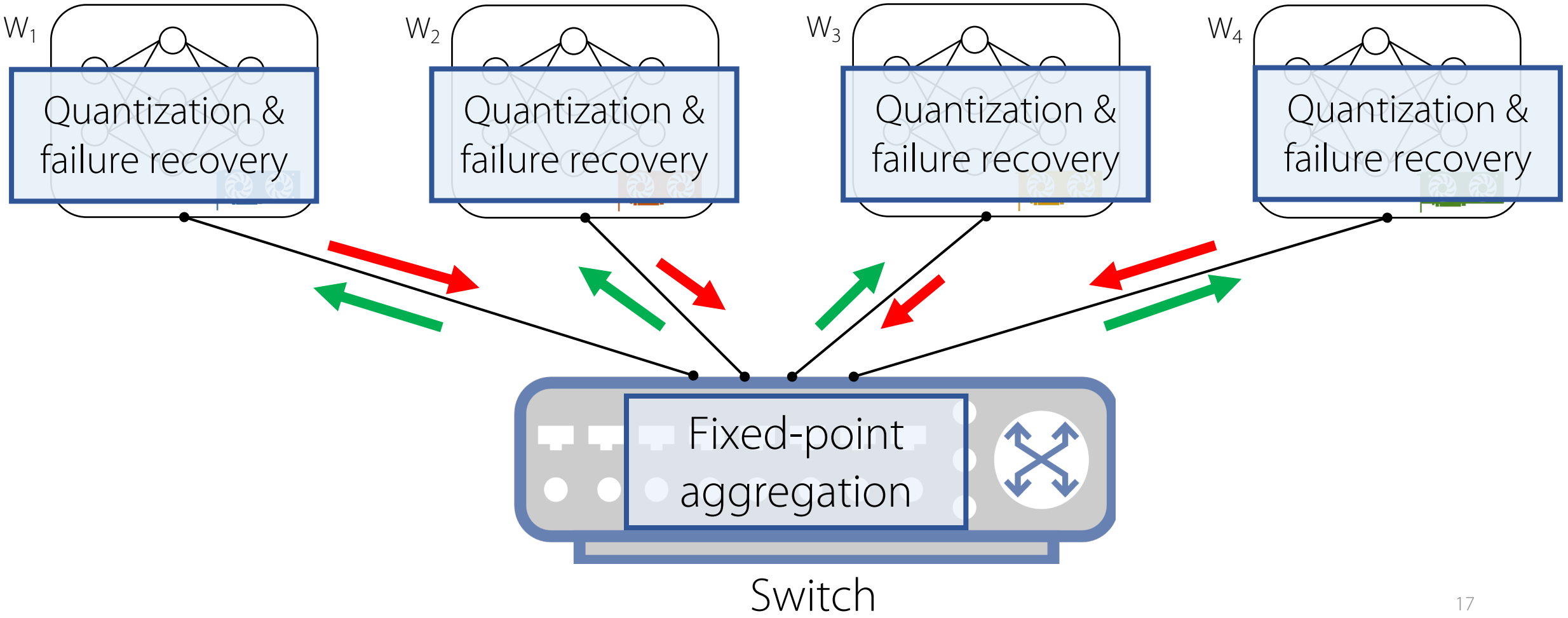
Design

- Pool-based streaming aggregation
- Combined switch-host architecture
- Quantized integer operations
- Failure-recovery protocol
- In-switch RDMA implementation

Streaming aggregation

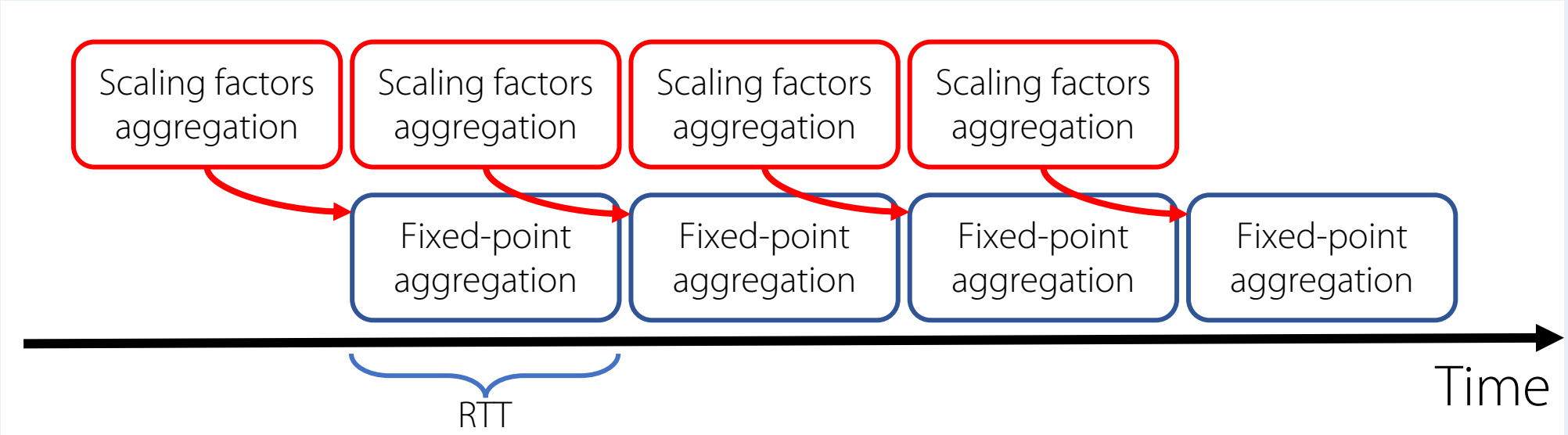


Combined switch-host architecture



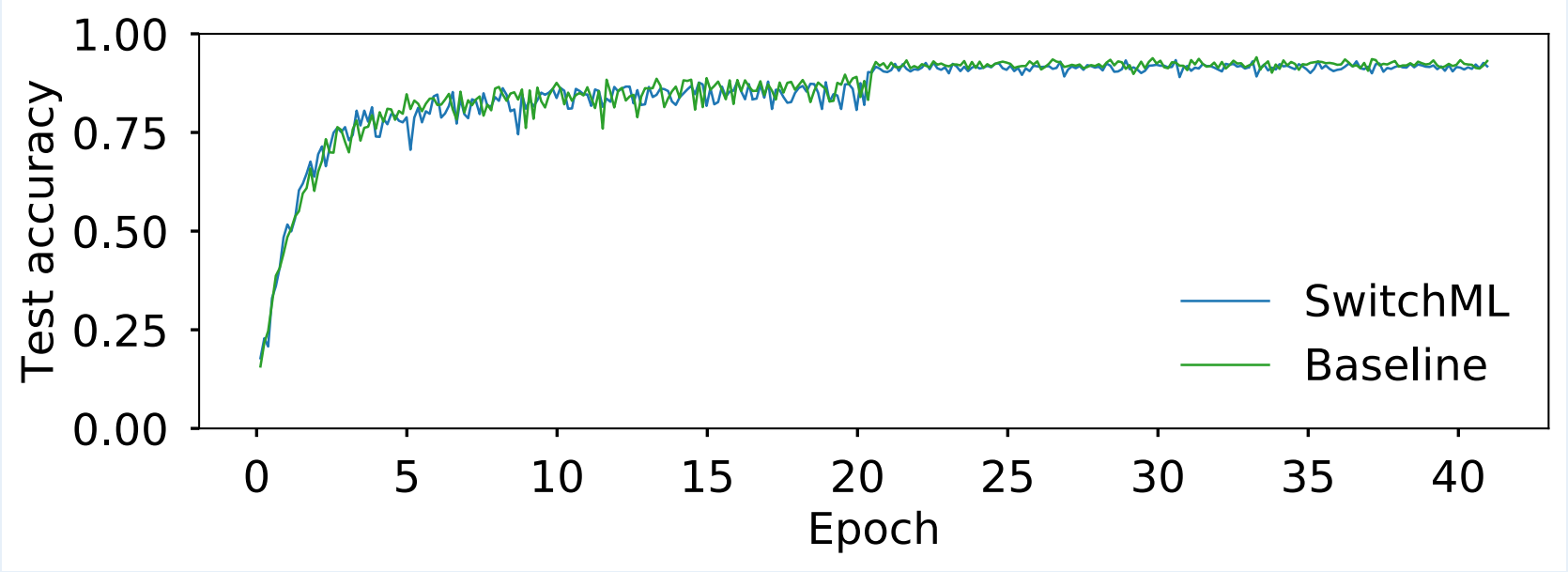
Combined switch-host architecture

Block quantization



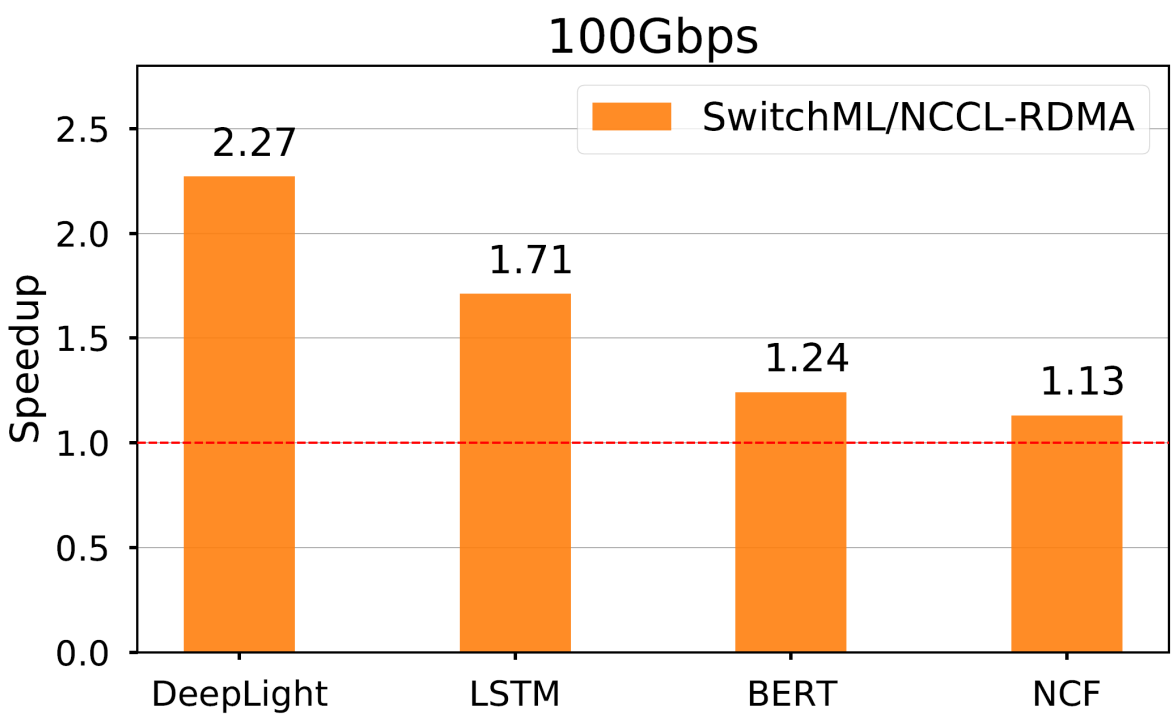
Combined switch-host architecture

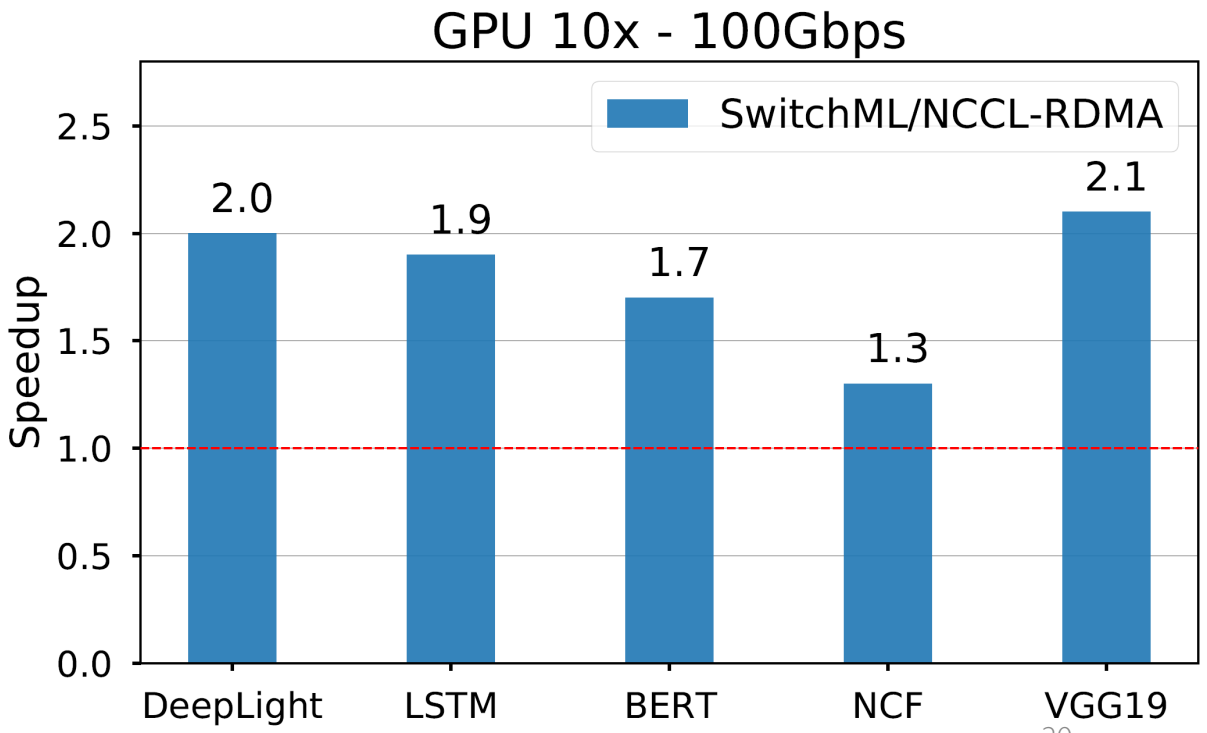
Quantization allows training to similar accuracy in a similar number of iterations as an unquantized network



How much faster is SwitchML?

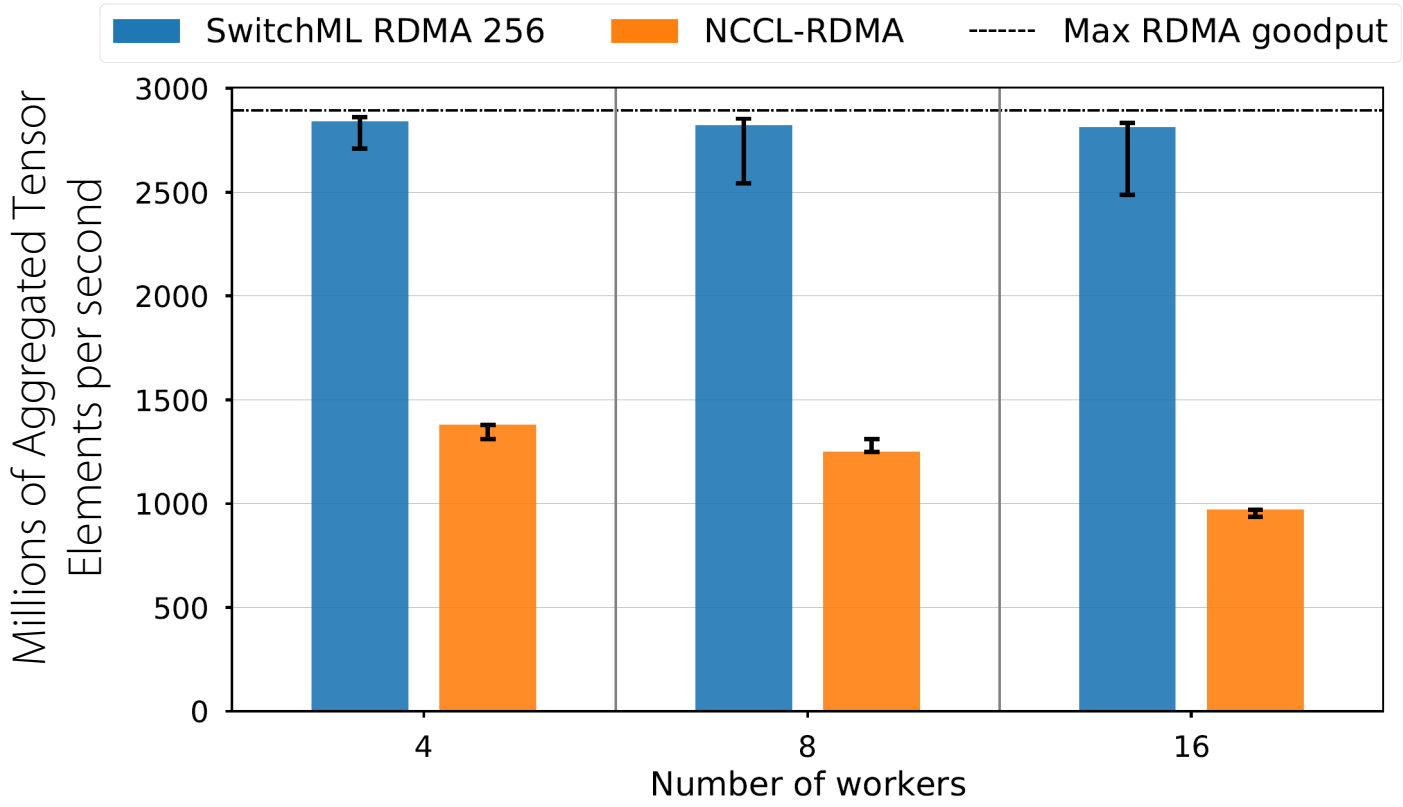
SwitchML provides a speedup in training throughput up to 2.27x on 100Gbps networks
Speedup is higher with faster GPUs that reduce the computation/communication ratio





How does SwitchML scale with the number of workers?

SwitchML performance does not depend on the number of workers



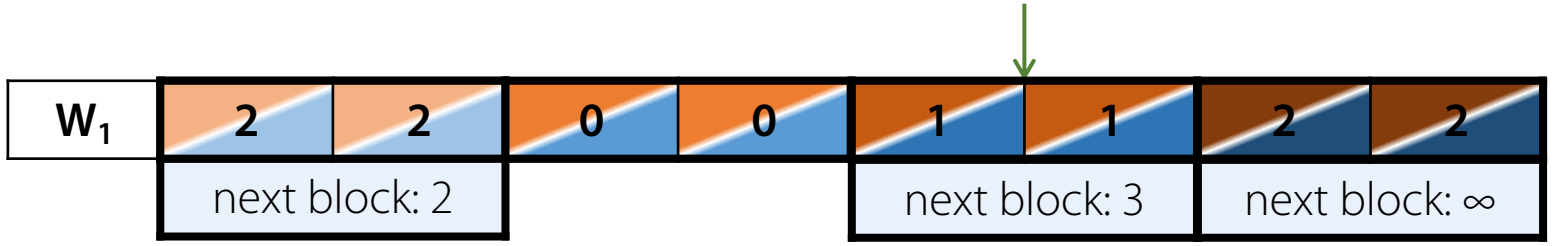
Sparse Collective Communication

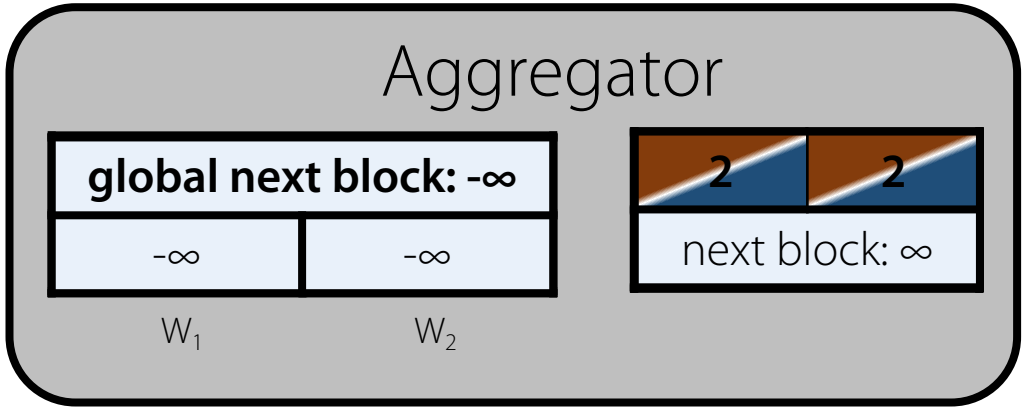
Many gradients in huge models are highly sparse

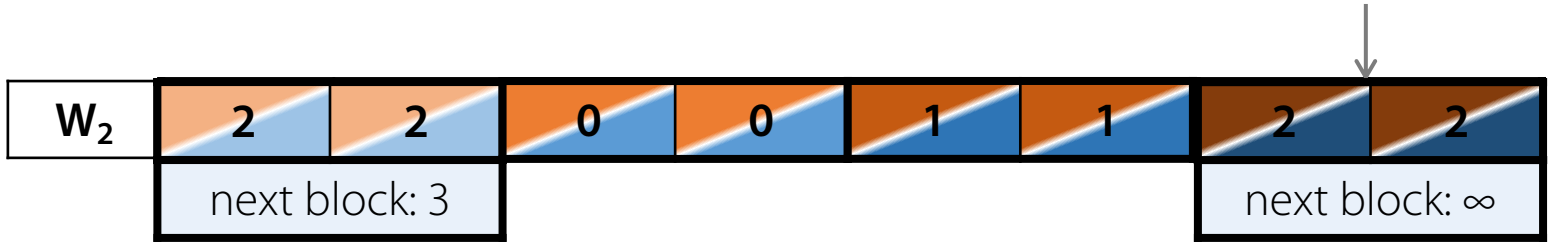
Model	Task	Model size	Sparsity
DeepLight	CTR prediction	2.3 GB	**99%**
LSTM	Language modeling	1.5 GB	**94%**
BERT	Qs answering	1.3 GB	9%
NCF	Recommendation	680 MB	**84%**
VGG19	Image classification	548 MB	32%
ResNet152	Image classification	230 MB	21%

How to efficiently aggregate sparse gradients?

OmniReduce: sparse streaming aggregation







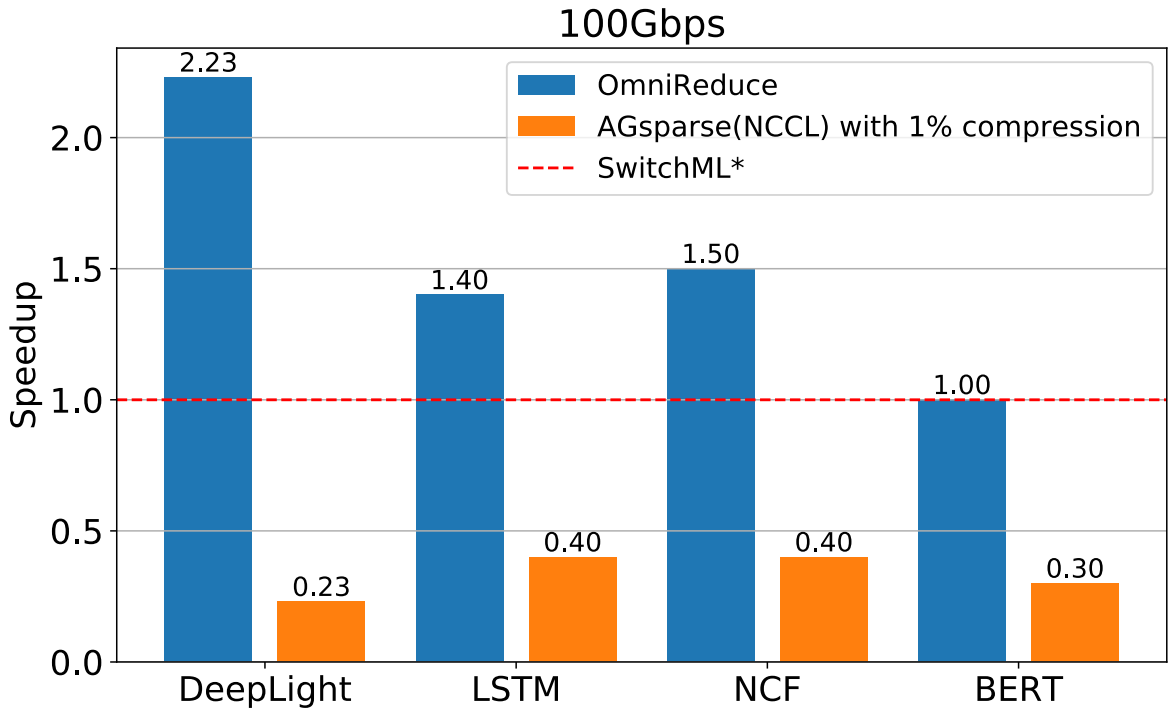
- Split data into blocks
- Stream non-zero blocks to aggregator
- Keep global view of next block

High performance through fine-grained parallelization (*pool of aggregation slots*) and pipelining to saturate network bandwidth

Does OmniReduce speed up training?

OmniReduce is up to 2.23× faster than SwitchML* on 100Gbps networks

Models with higher sparsity gain more from efficient sparse collective communication



- SwitchML* is a software-based implementation of SwitchML
  (fair comparison with software aggregator)
- AGsparse is allgather-based sparse allreduce method
  (compression overheads are not considered)

OmniReduce is in trial deployment at 美团 Meituan

This talk

Will focus on two common DC workloads:

1. ~~Distributed Deep Learning~~

2. Key-Value Storage Systems

CPUs are busy!

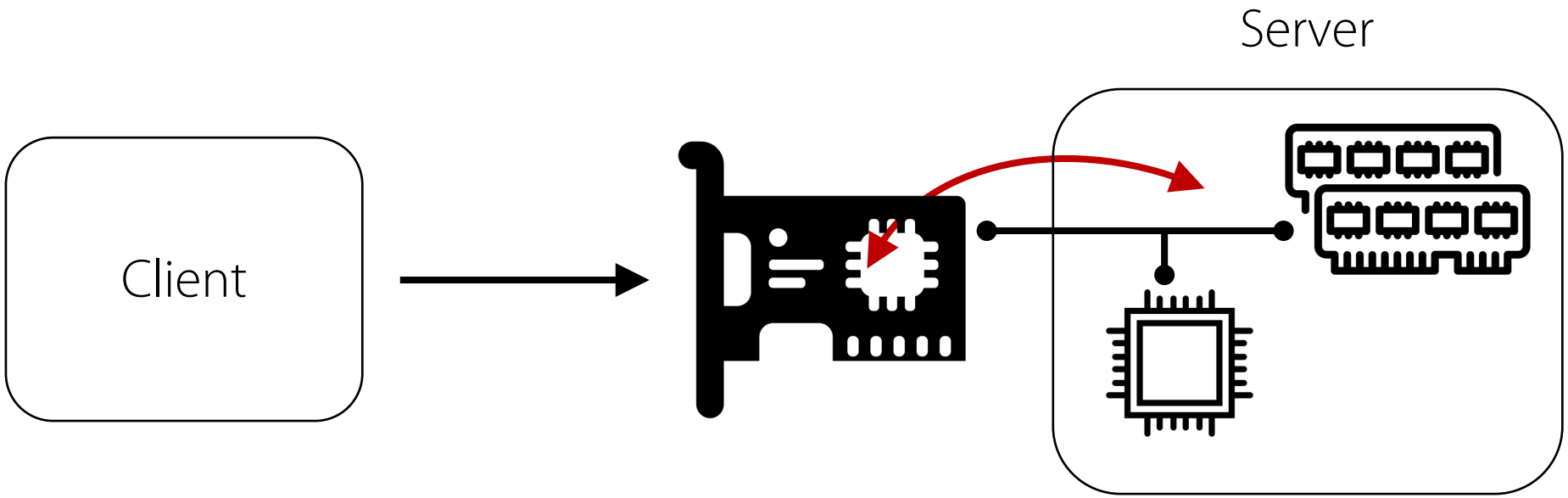
As server CPU cycles are increasingly scarce resource, offload is gaining in popularity:

especially for common, often-repeated operations

NICs are in the network data path and can carry out operations on in-flight data with low latency; **RDMA NICs** are ubiquitous

Can we enable complex offloads on RDMA NICs?

Basic NIC offloads



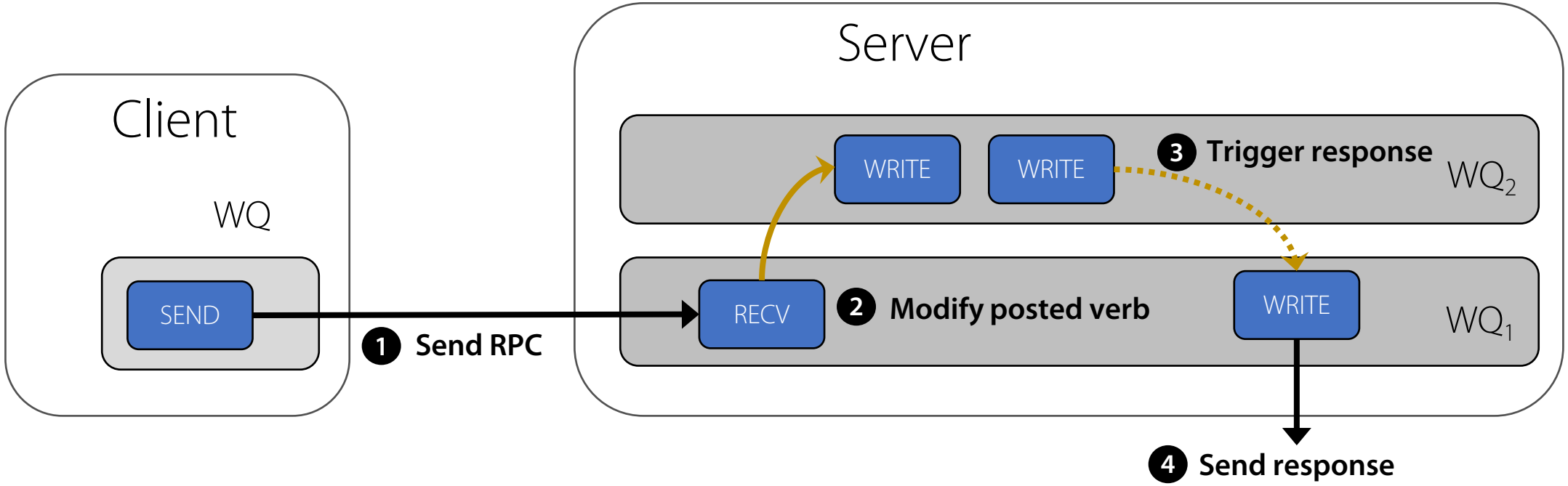
RDMA design oriented to data-movement operations SEND RECV WRITE READ CAS

Lends well to accelerate data IO, shown effective for locking and even consensus

But **lacks flexibility** to offload "arbitrary" logic onto the NIC

Complex offloads require data-dependent execution

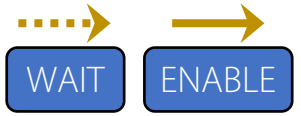
RedN offloads



RedN realizes **self-modifying RDMA programs**

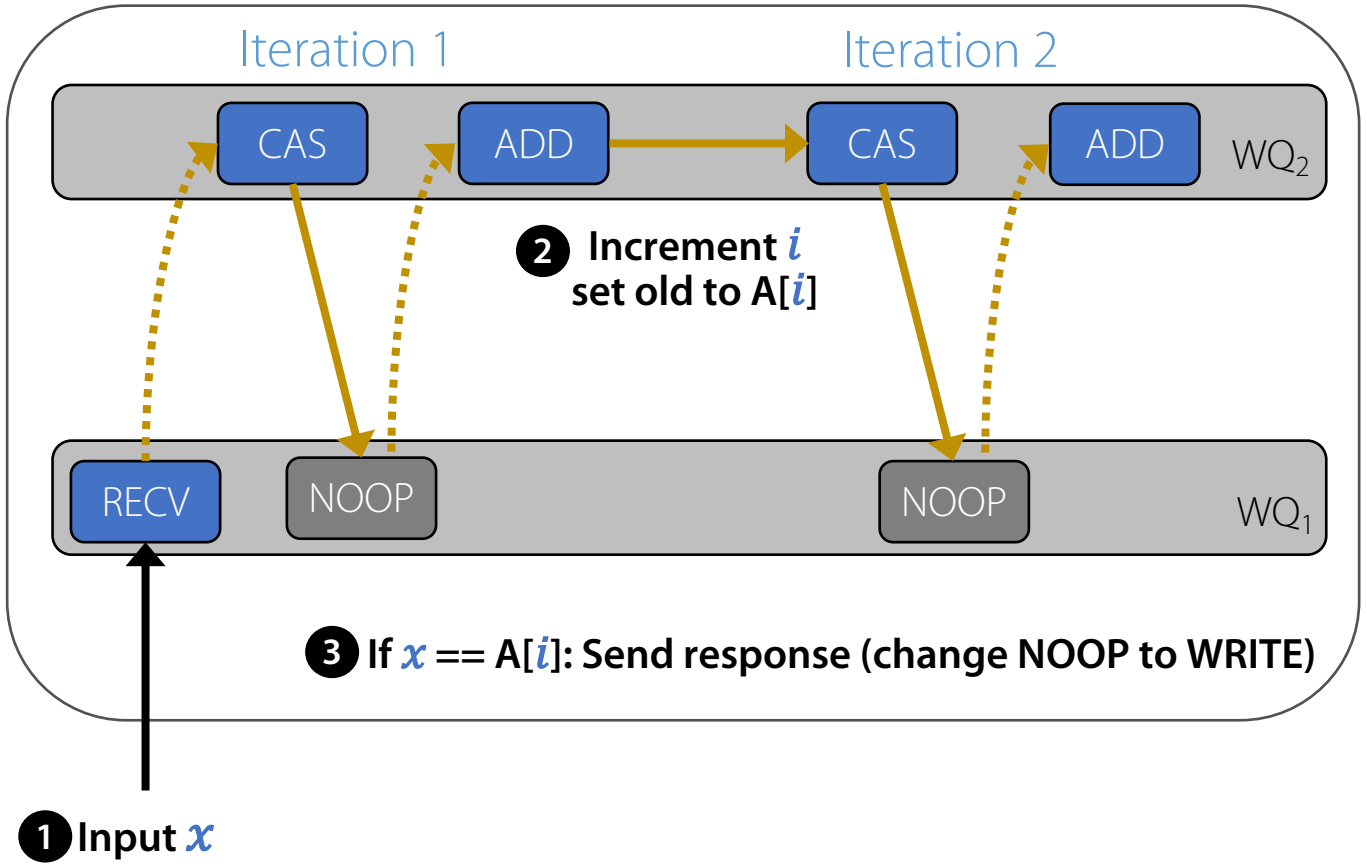
RDMA ops (*verbs*) are pre-post on "idle" Work Queues

Key insight: A verb can modify subsequent ones; a couple of obscure verbs can pace their execution



Loops and conditionals with self-modifying

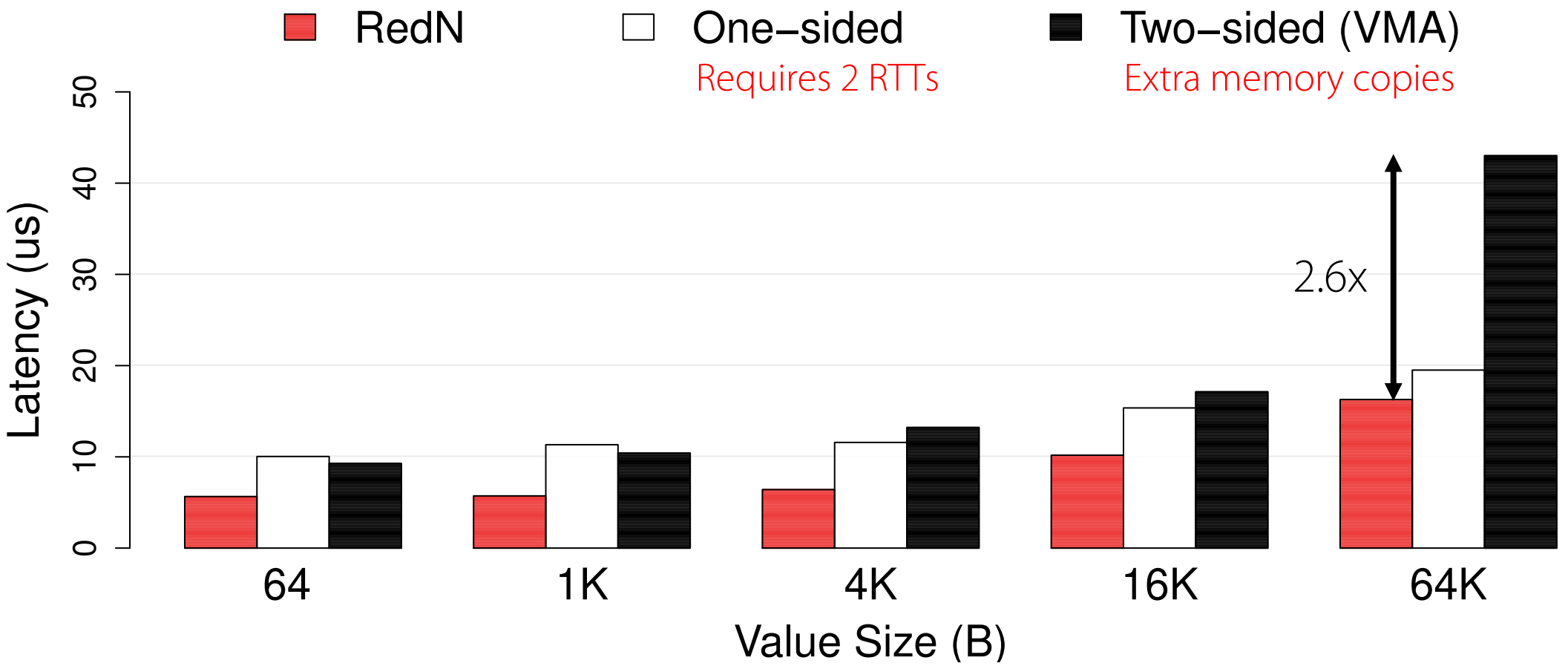
```
Input x
i = 0;
while (i < 2)
    if(x == A[i])
      send(i)
    i++;
```



In theory, sufficient for Turing completeness

In practice, useful for Hash Lookups, List Traversal, and more…

Memcached *get* acceleration



Summary

Lots of pressure for efficiently handling DC workloads with intensive communication, low-latency reqs

Our research shows that **in-networking computing is an effective tool** …

… **especially** when used **judiciously**

What can be computed in-network? The answer is, it's general

But **careful design to balance when, where, how**

Examples: Deep Learning (in-network aggregation), KV Store (complex offloads)

References

- [**SwitchML**, NSDI '21]
  Scaling Distributed Machine Learning with In-Network Aggregation
  A. Sapio, M. Canini, C.-Y. Ho, J. Nelson, P. Kalnis, C. Kim, A. Krishnamurthy, M. Moshref, D. R. K. Ports, P. Richtarik
- [**OmniReduce**, SIGCOMM '21]
  Efficient Sparse Collective Communication and its application to Accelerate Distributed Deep Learning
  J. Fei, C.-Y. Ho, A. N. Sahu, M. Canini, A. Sapio
- [**RedN**, NSDI '22]
  RDMA is Turing complete, we just did not know it yet!
  W. Reda, M. Canini, D. Kostic, S. Peter

Summary

Contact: marco@kaust.edu.sa

Lots of pressure for efficiently handling DC workloads with intensive communication, low-latency reqs

Our research shows that **in-networking computing is an effective tool** …

… **especially** when used **judiciously**

What can be computed in-network? The answer is, it's general

But **careful design to balance when, where, how**

Examples: Deep Learning (in-network aggregation), KV Store (complex offloads)