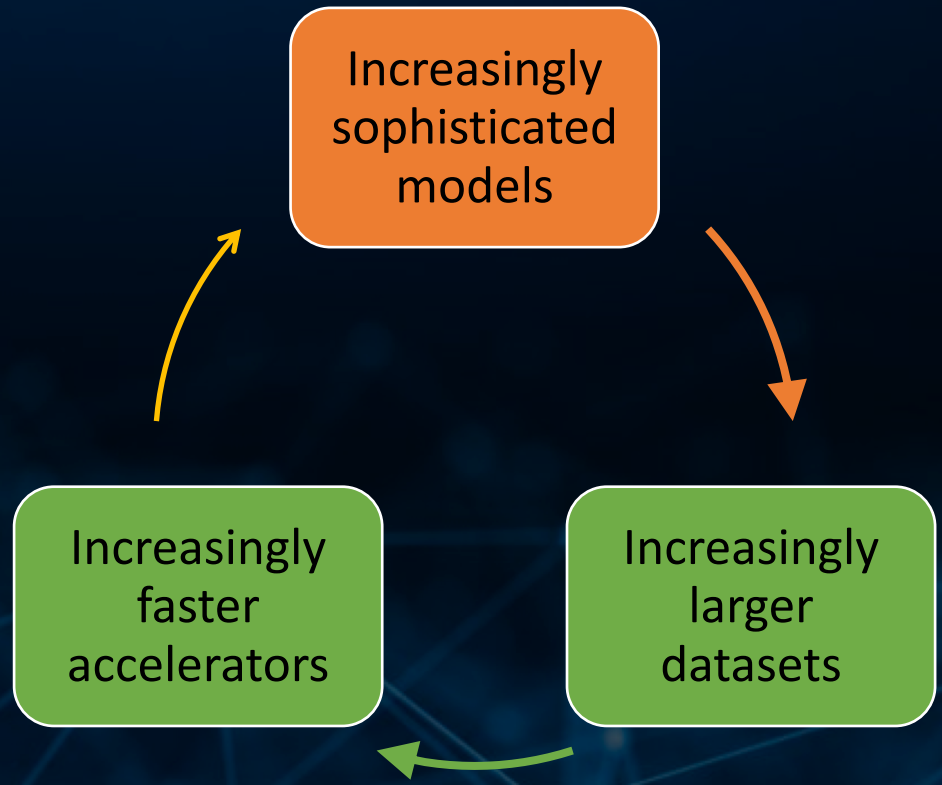
Scaling Distributed Machine Learning with In-Network Aggregation

Jacob Nelson with

Amedeo Sapio[*], Marco Canini[*], Chen-Yu Ho[*], Panos Kalnis[*], Changhoon Kim[‡],
Arvind Krishnamurthy[◊], Masoud Moshref[‡], Dan R. K. Ports, Peter Richtárik[*]

Microsoft Research *KAUST ‡Barefoot Networks ◊University of Washington

Machine Learning



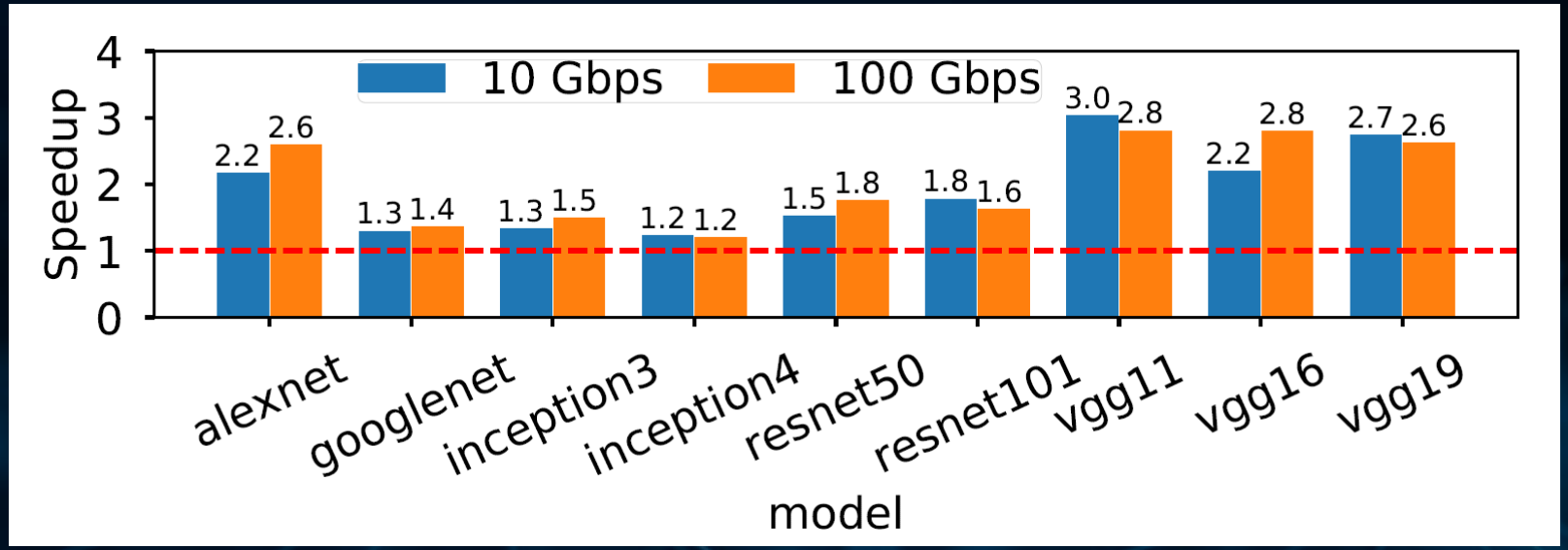
Innovation fueled by leaps in (costly) infrastructure:

Clusters with hundreds of machines,

each with many HW accelerators (GPUs)

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

Scaling Machine Learning



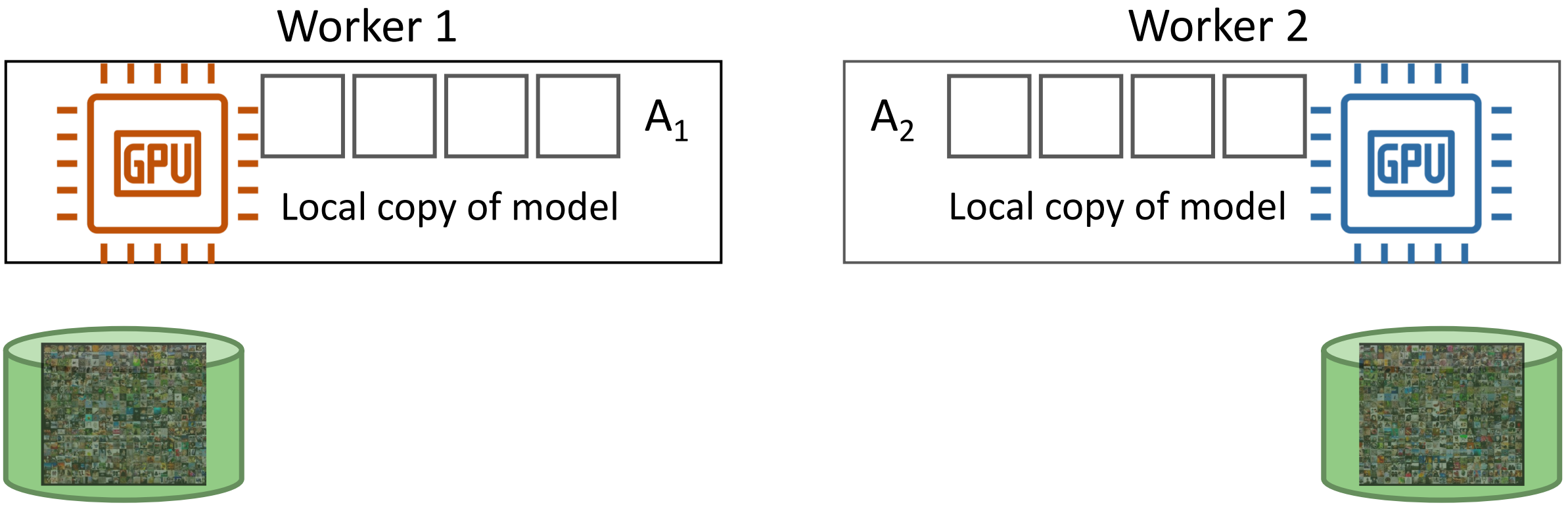
Can the network be the ML accelerator?

A KAUST collaboration with

Outline

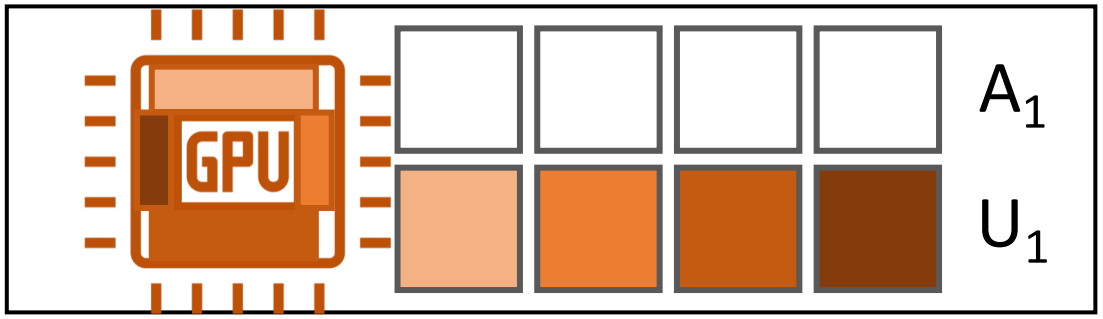
- The distributed training process
- In-network aggregation design
- Evaluation
- Future work and conclusion

Data-parallel distributed training

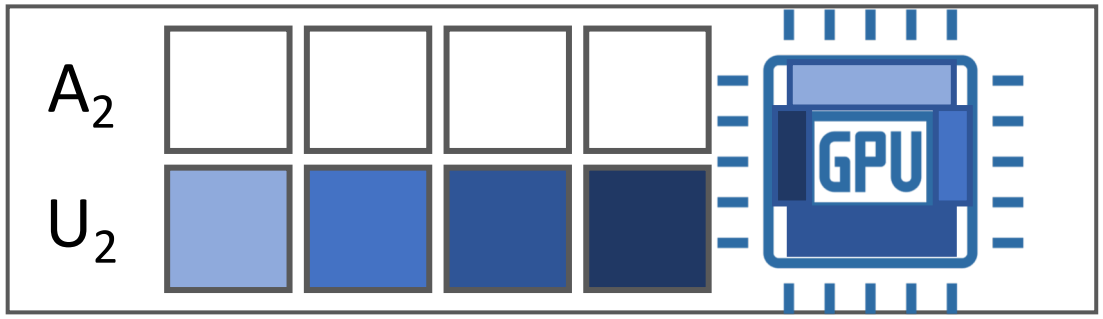


Phase 1: Workers learn independently

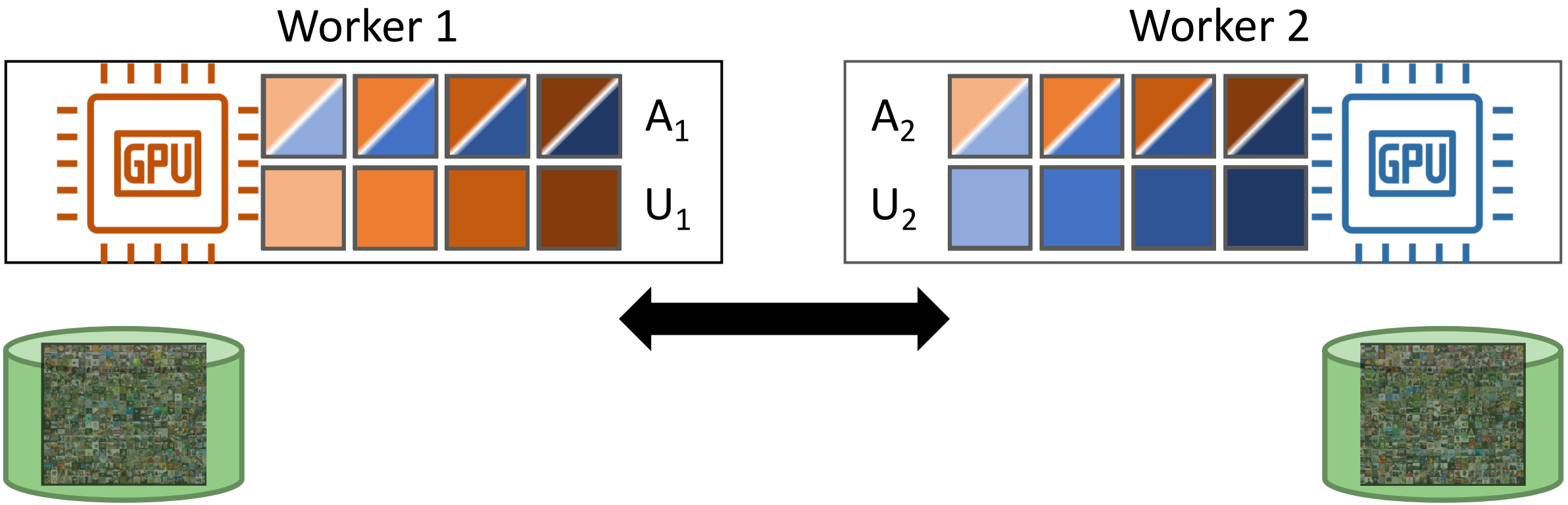
Worker 1



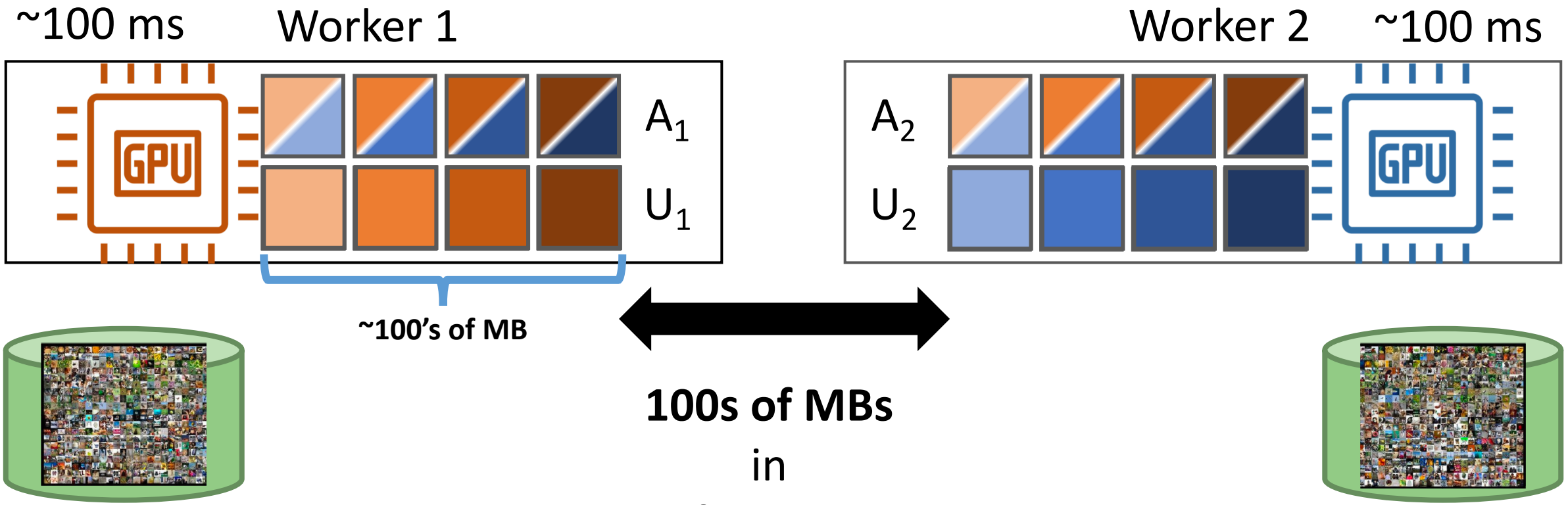
Worker 2



Phase 2: Workers exchange what they've learned



Aggregation is communication-intensive

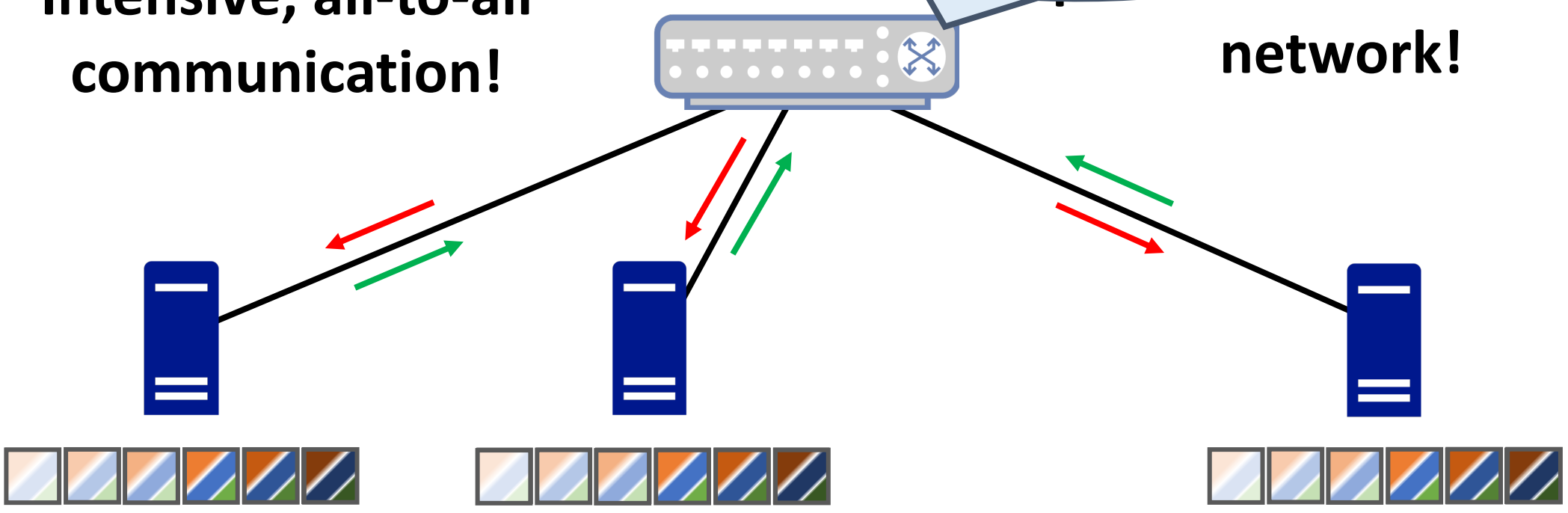


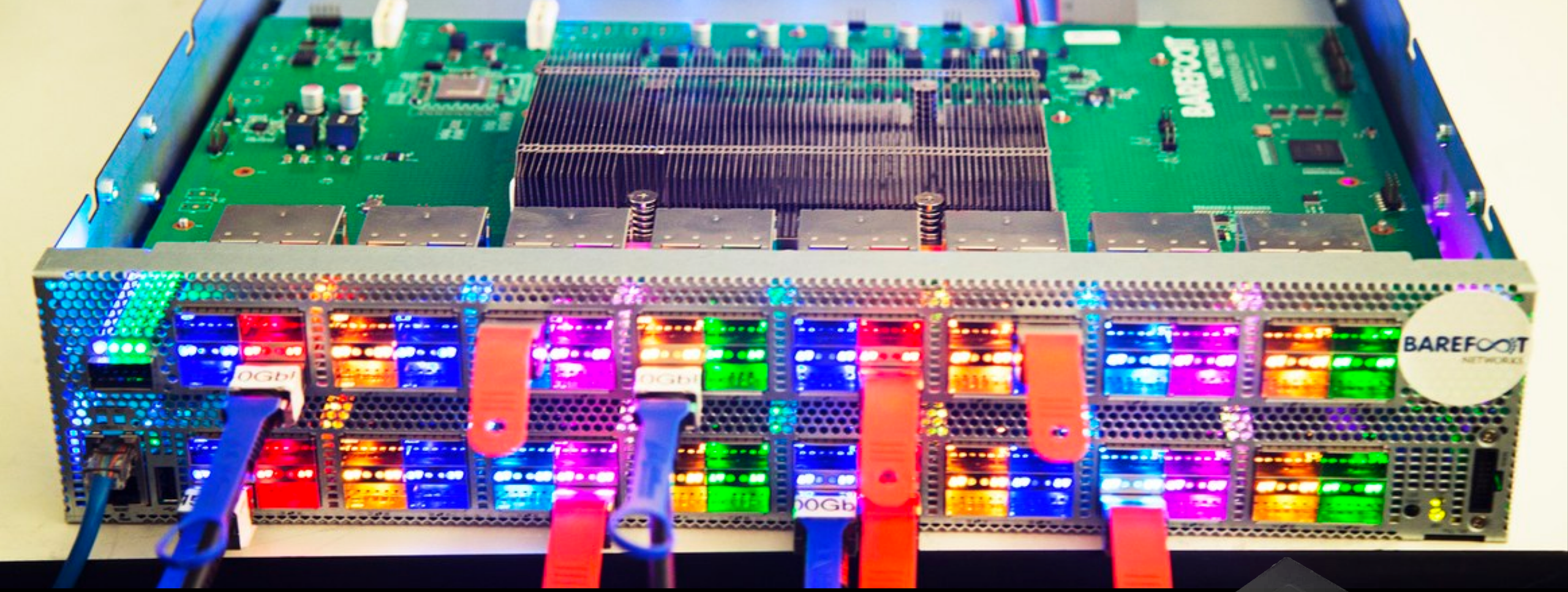
Aggregation is communication-intensive

Problem:
Intensive, all-to-all communication!

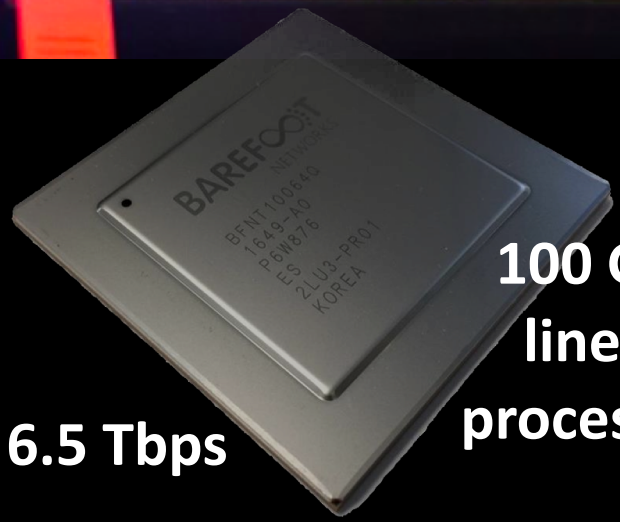
If only I could help…

…sh training …ottleneck to the network!





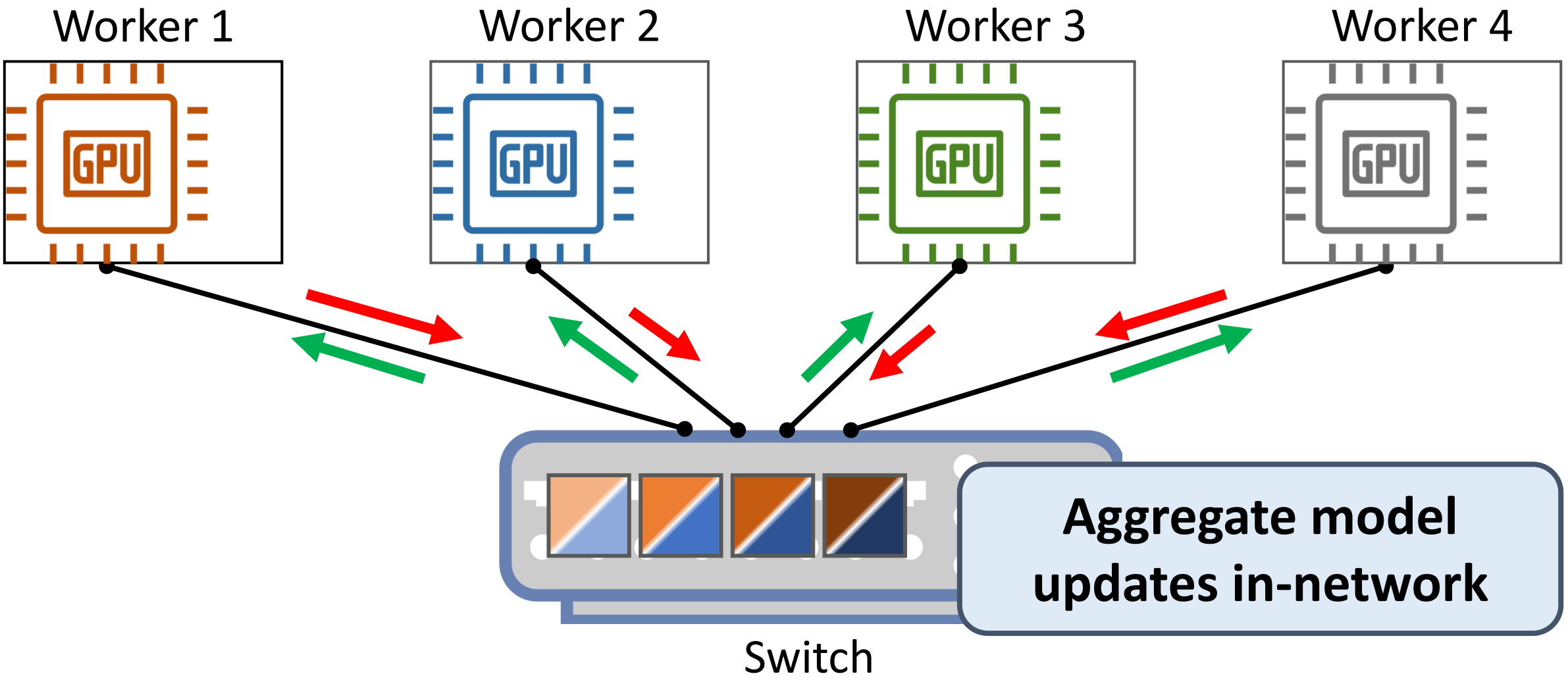
Programmable data plane switches to the rescue!



6.5 Tbps

100 Gbps line rate processing

SwitchML: the network is the ML accelerator



Co-design ML and networking for efficiency

Challenges

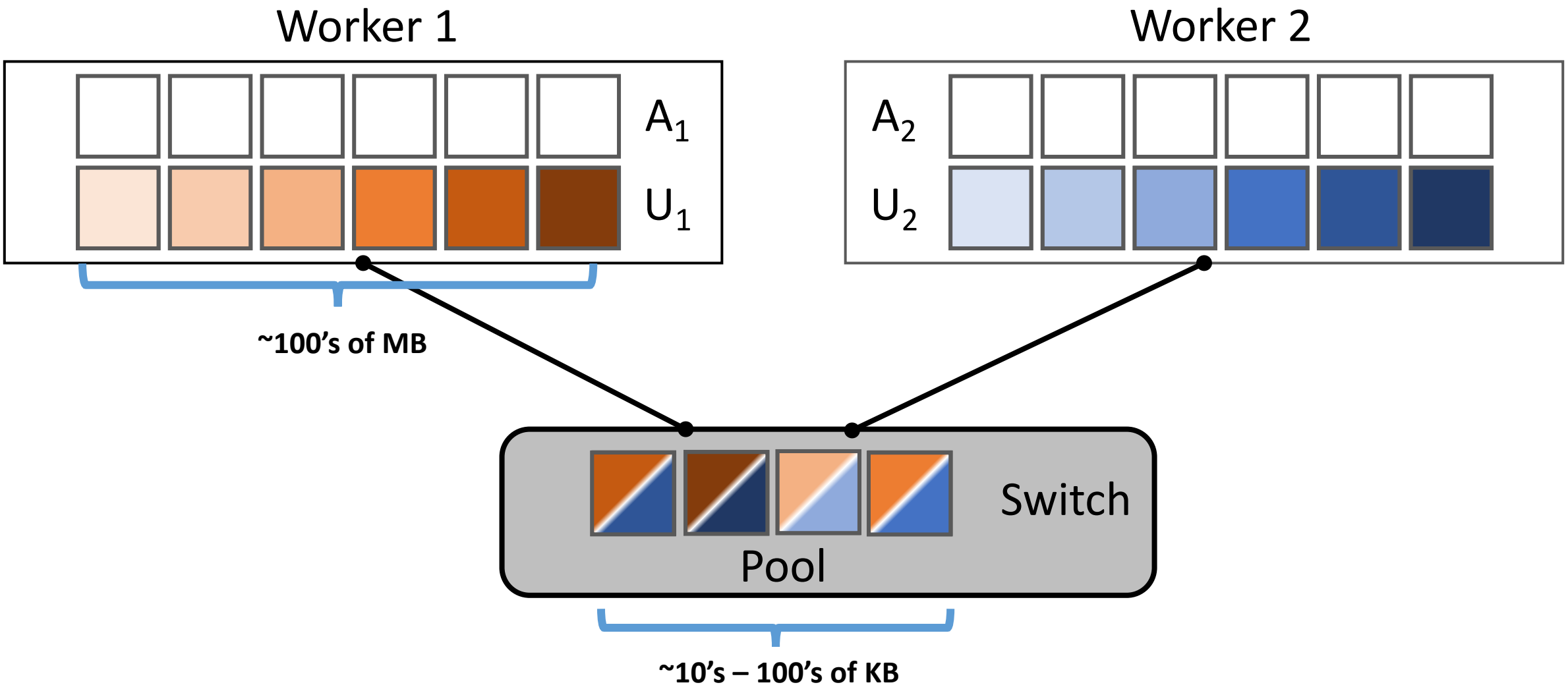
- Limited storage
- Limited computation
- No floating point
- Packet loss

Design

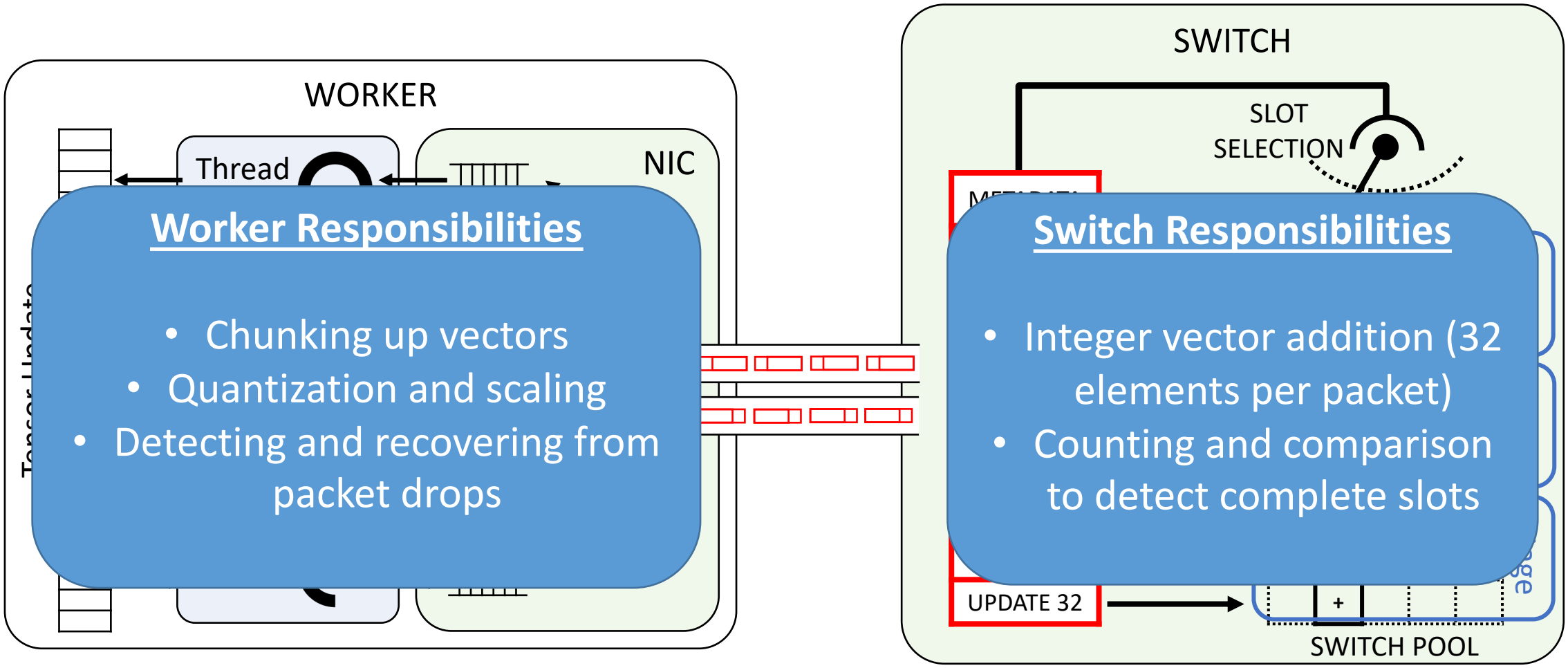
- Pool-based streaming aggregation
- Combined switch-host architecture
- Quantized integer operations
- Failure-recovery protocol

6.5 Tbps
programmable
data plane

Streaming aggregation with a pool



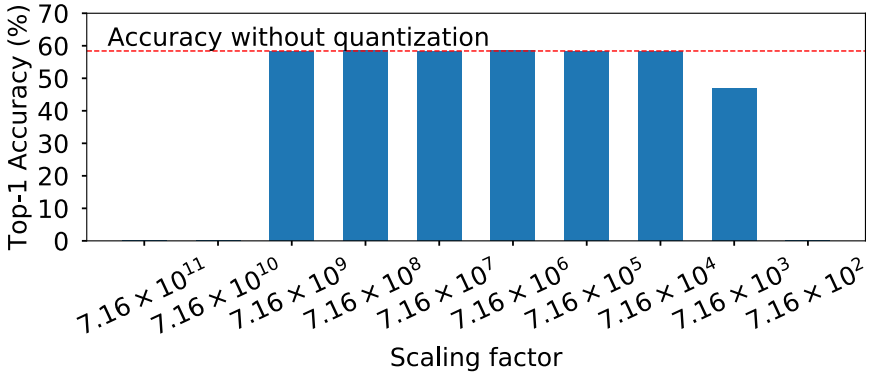
Combined switch-host architecture



Quantization

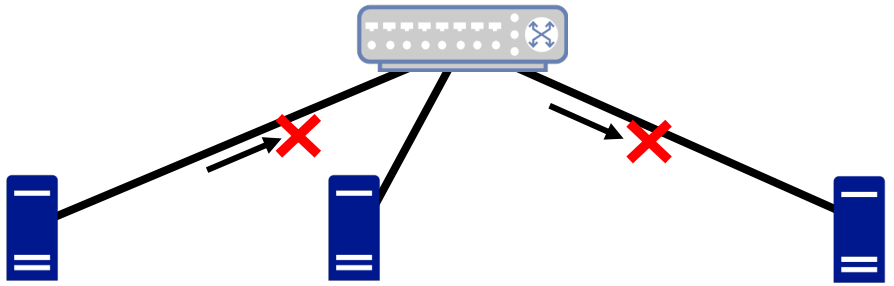
- Convert floating point to 32-bit fixed-point values $\widetilde{U_j^i} = round(sf * U_j^i)$
- Updates are scaled by multiplying for a scaling factor sf $\widetilde{A_j^i} = A_j^i / sf$
- Approach 1: (restricted) 16-bit floating point ↔ 32-bit fixed point conversion
  → Directly in the switch
- Approach 2: 32-bit floating point ↔ 32-bit fixed point conversion
  → At workers with AVX instructions
  With single scaling factor obtained by profiling

This quantization allows training to similar accuracy in a similar number of iterations as an unquantized network for a large range of scaling factors



Packet loss tolerance

- Packet loss can happen in two directions
- Workers detect losses using timers
- Lost packets are retransmitted
  - A model update must not be applied twice
  - A model update must not be applied to a “full” slot



- Workers’ per-slot contributions tracked with a bitmap
  - Ignores duplicates
- Shadow copy of the previous result for a slot
  - Retransmits a dropped result packet

Implementation

- Switch program written in P4 for Barefoot Tofino

- End-host C++ library providing a familiar all-reduce API
  - Kernel bypass

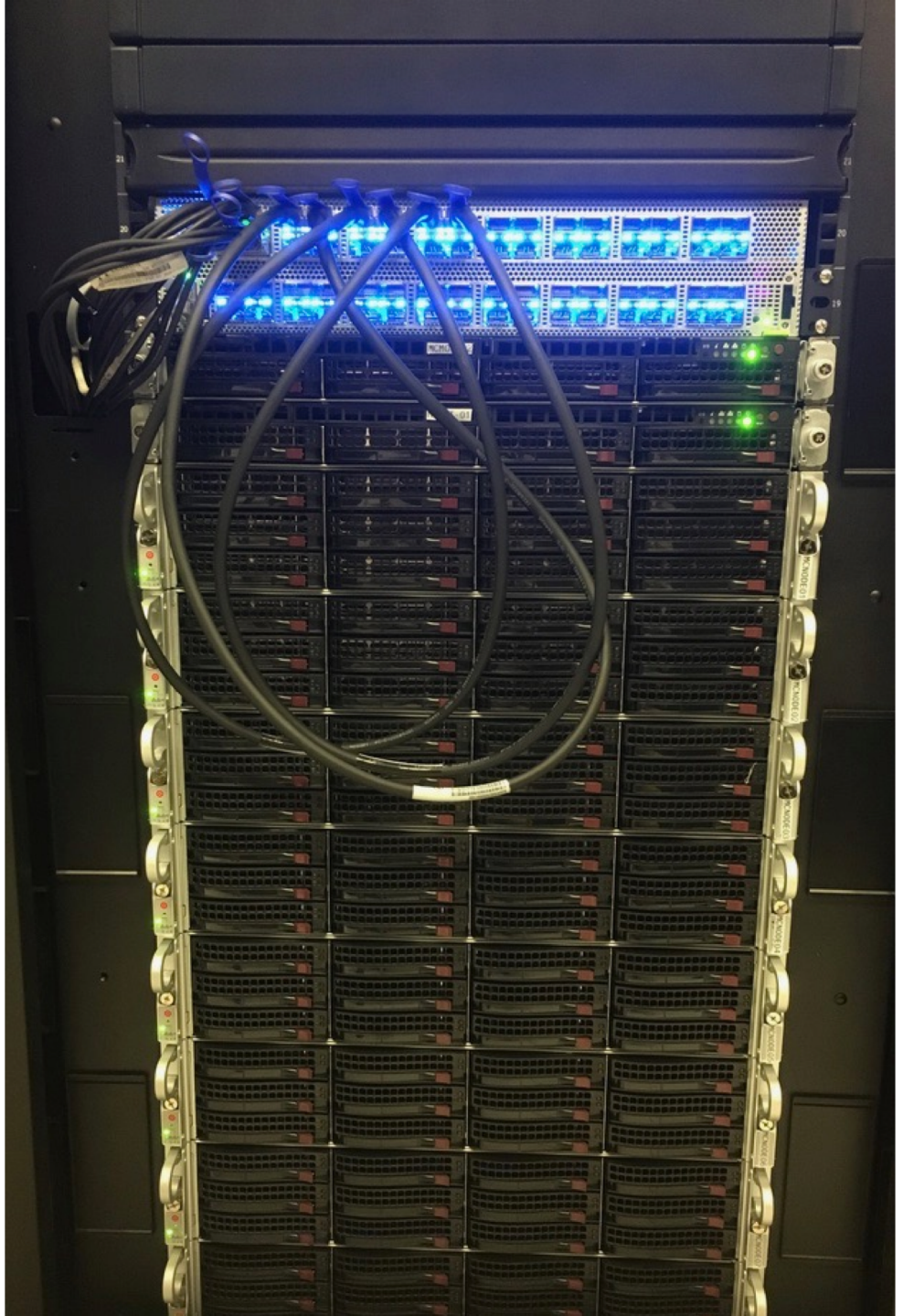
- We have integrated SwitchML with:
  - TensorFlow using Horovod,
  - PyTorch/Caffe2 using Gloo

Evaluation

Testbed:

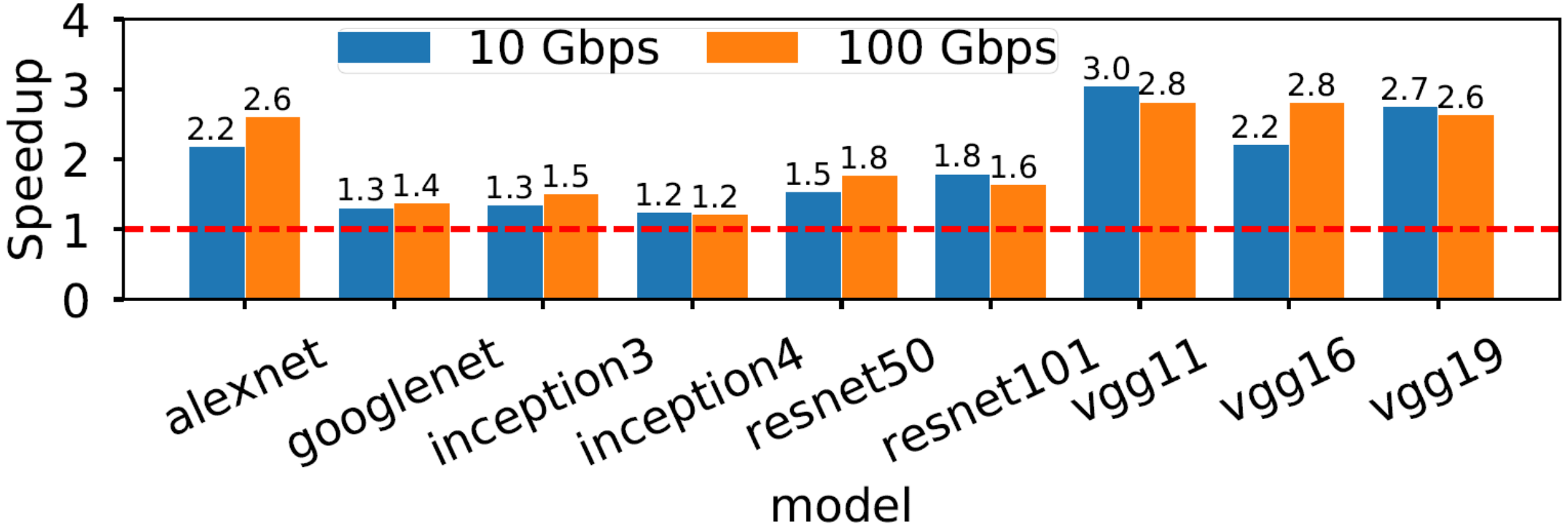
- 16 servers (8 w/ P100 GPUs)
  - 10 Gbps (Intel 82599ES)
  - 100 Gbps (Mellanox Connect-X 5)
- 64 x 100 Gbps switch (Barefoot Tofino)

- Models:
  - 9 standard CNN benchmarks
  - Training on ImageNet
    (except synthetic data with AlexNet)
  - Compared with TensorFlow using the Nvidia Collective Comm. Library (NCCL)



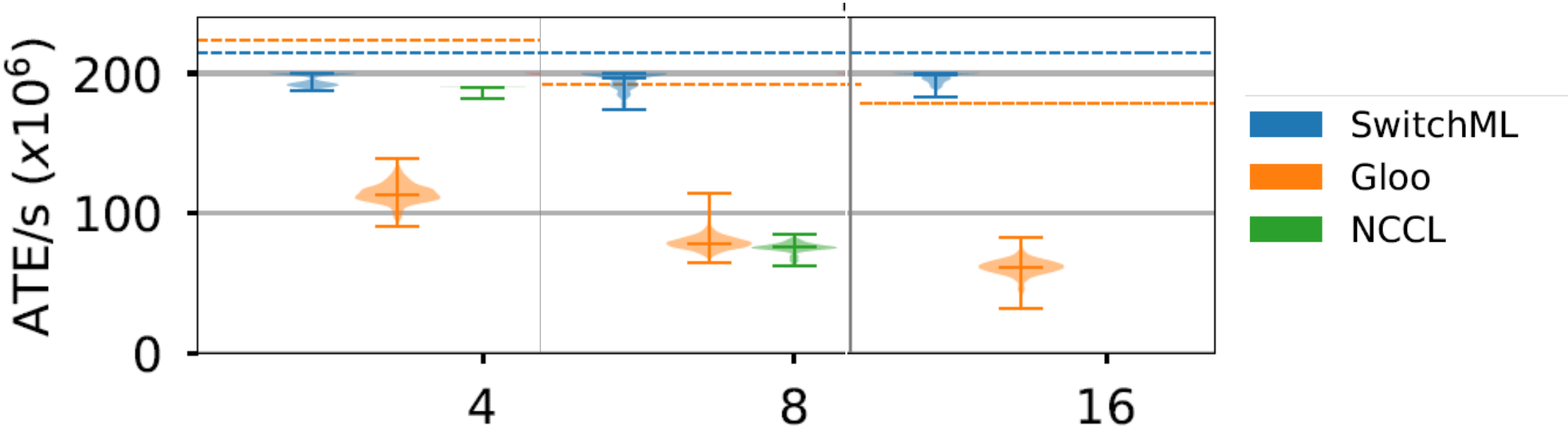
How much faster is SwitchML?

SwitchML provides a speedup from 20% to 300%
compared to Tensorflow/NCCL (with direct GPU memory access)



How does SwitchML scale with the number of workers?

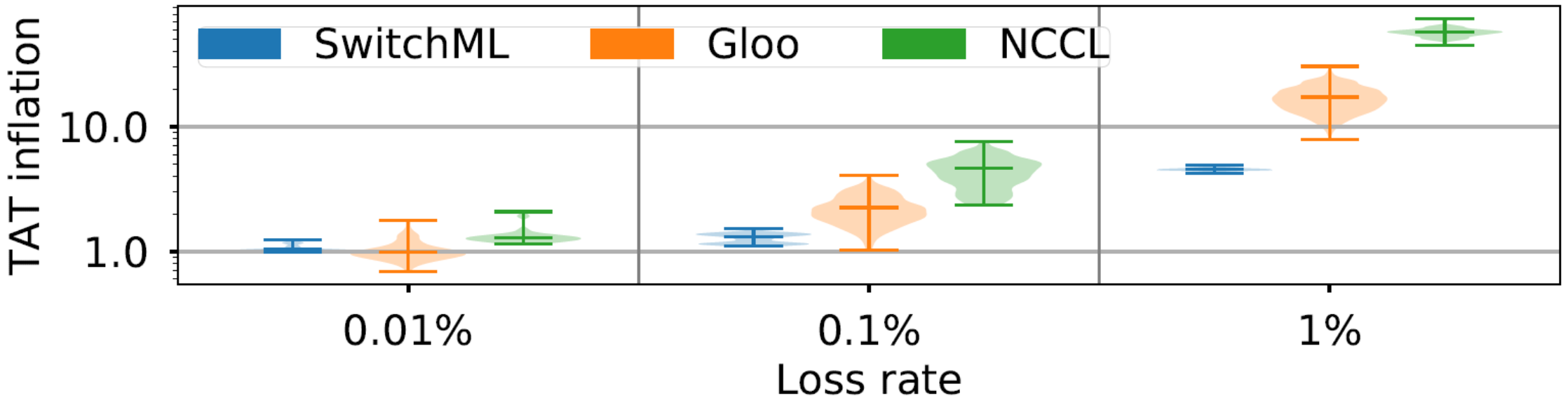
SwitchML performance does not depend on the number of workers



How does SwitchML perform with packet losses?

SwitchML has a lower inflation than TCP

Reasonable packet loss rates have no impact on performance

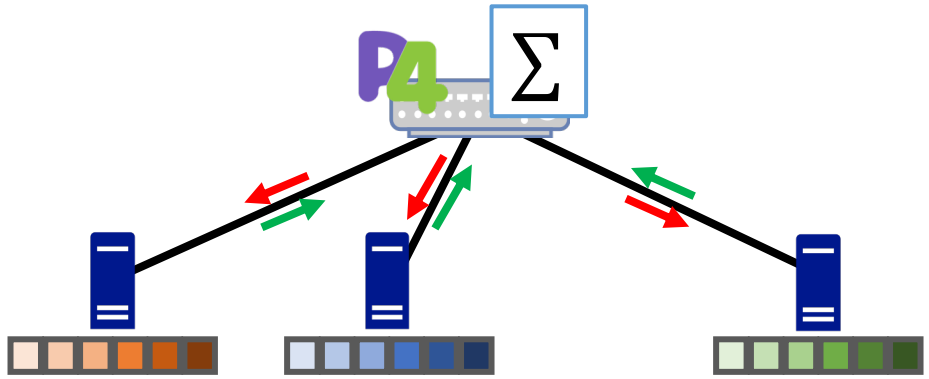


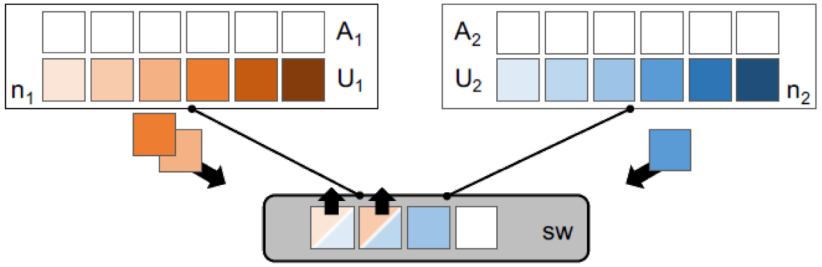
Future work

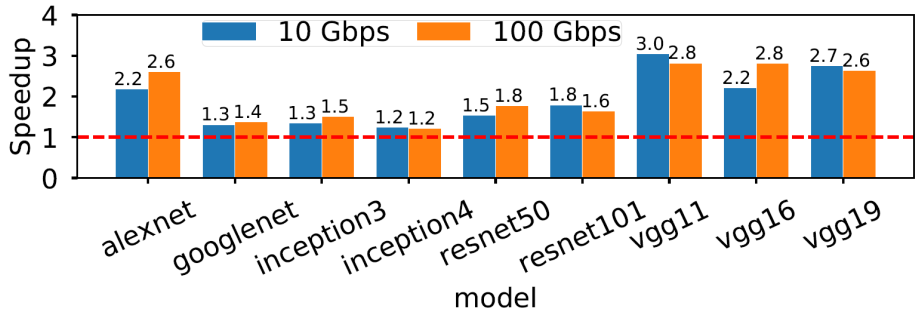
- Multi-rack
  - Can we use multiple switches to implement hierarchical SwithML?
- Multiple jobs, multiple tenants
  - Can we support the multiple jobs in the same rack by partitioning slots?
- Better numeric representations
  - Can we quantize without having to choose a scaling factor?
- More data per packet
  - Full MTU packets would provide ~31% better performance.

Summary

- SwitchML uses **in-network aggregation** to synchronize model updates
  - Reduce network traffic volume and latency
- SwitchML speeds up training up to 300% with real-world DNN benchmarks
- Aggregation time does not depend on the number of workers
- Preprint on arXiv: https://aka.ms/switchml

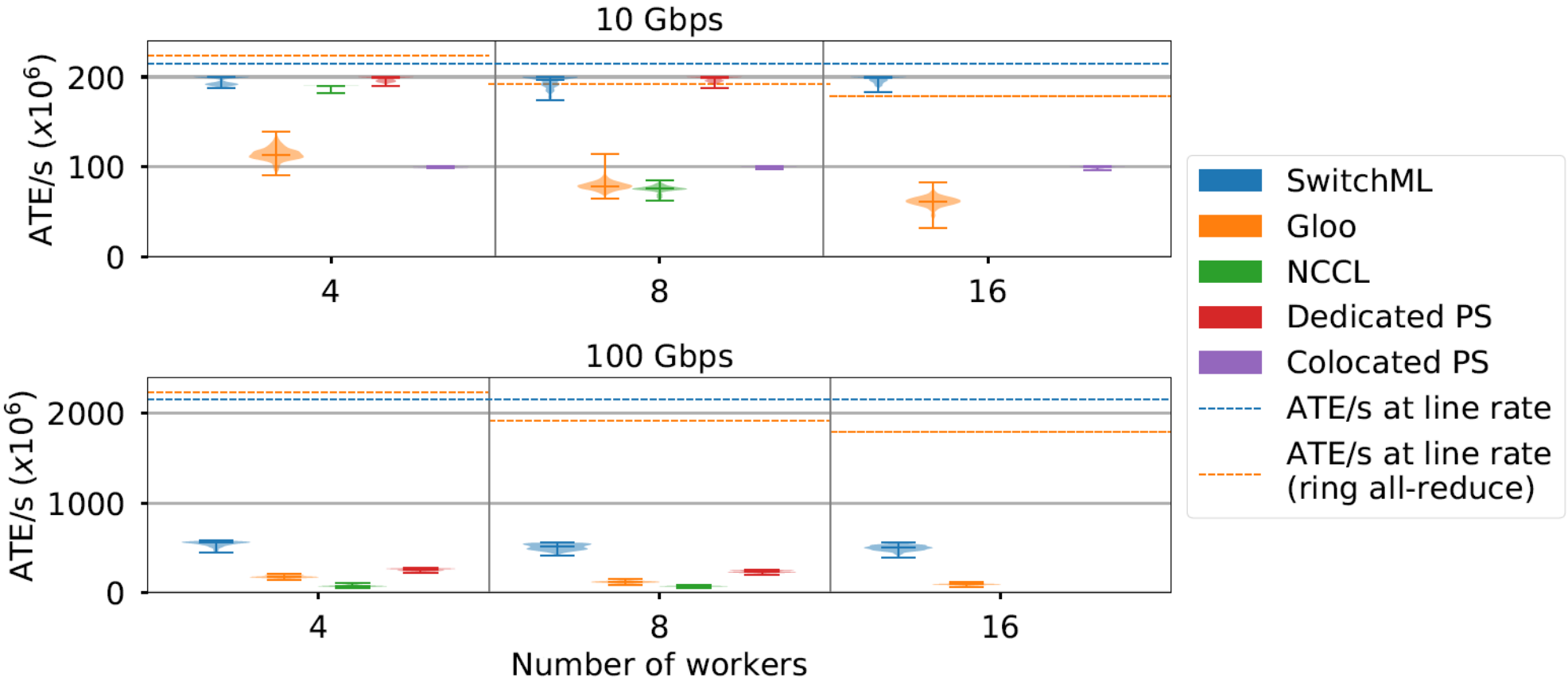




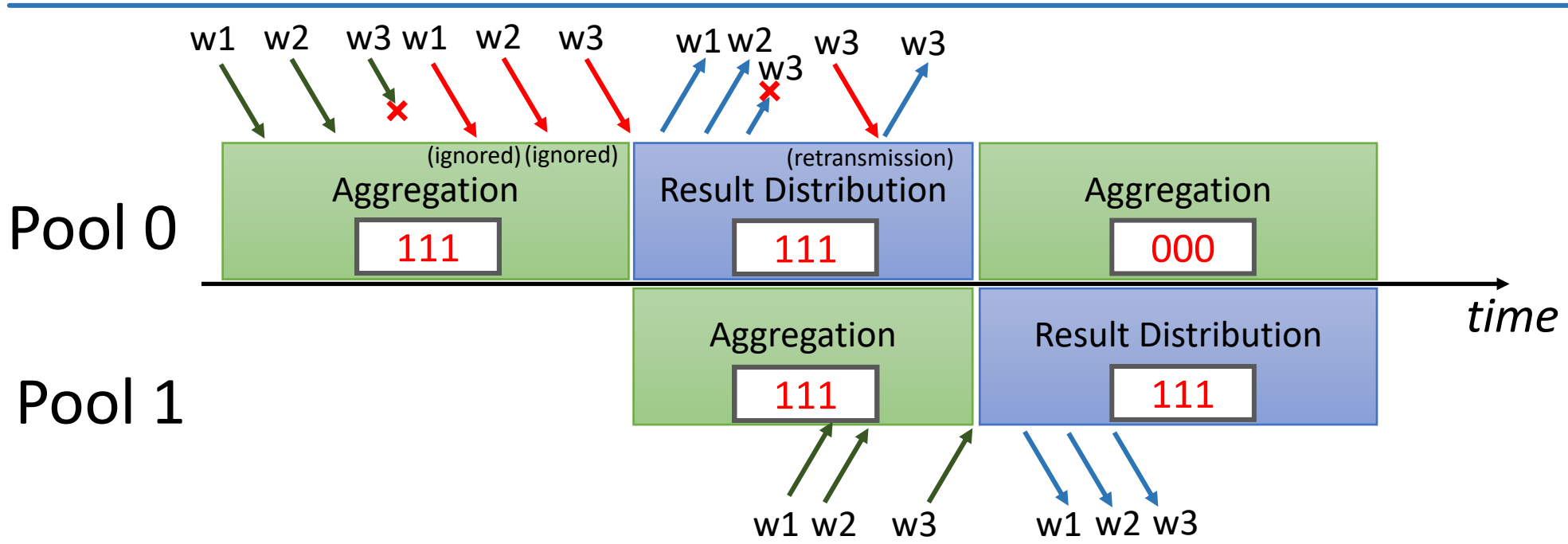


How does SwitchML scale with the number of workers?

SwitchML performance does not depend on the number of workers

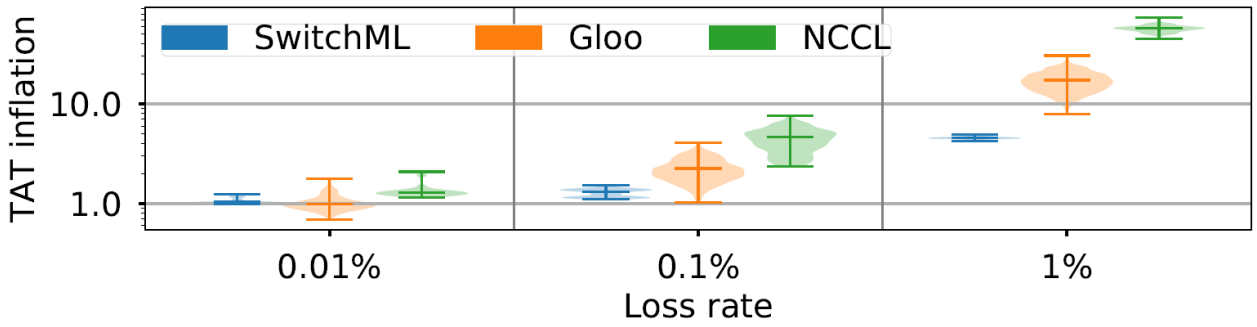


Packet loss tolerance



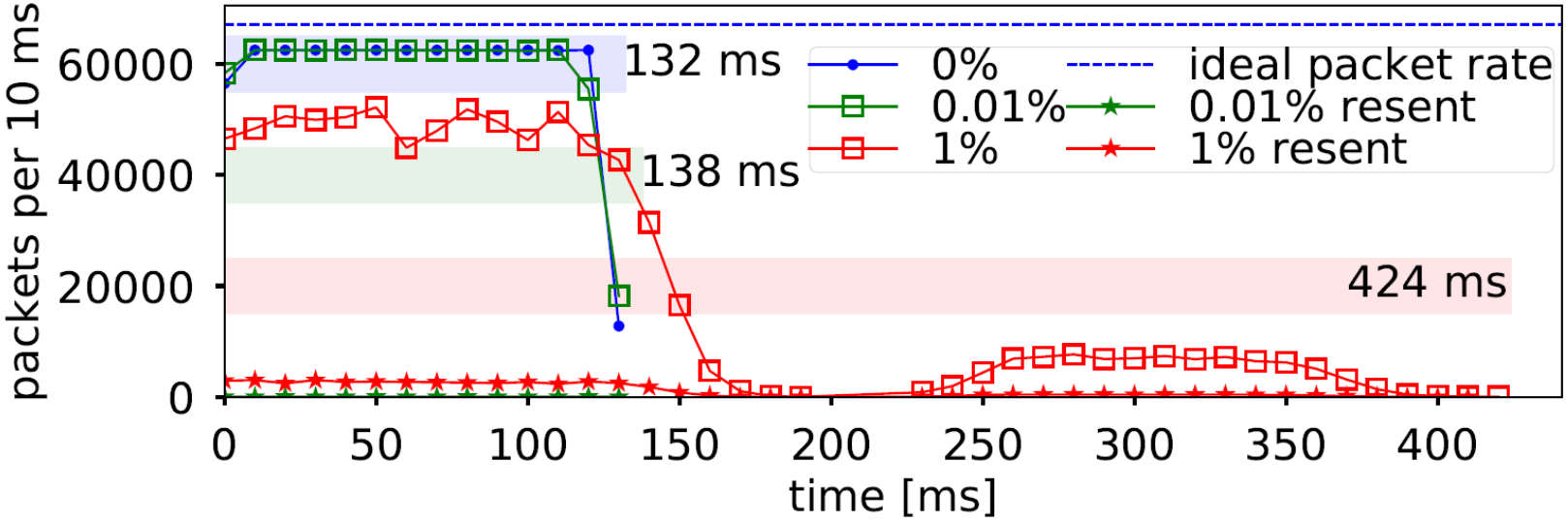
- Workers contribution per-slot tracked with a bitmap
  - Ignores duplicates
- Shadow copy of the previous result for a slot
  - Retransmits a dropped result packet

How does SwitchML perform with packet losses?



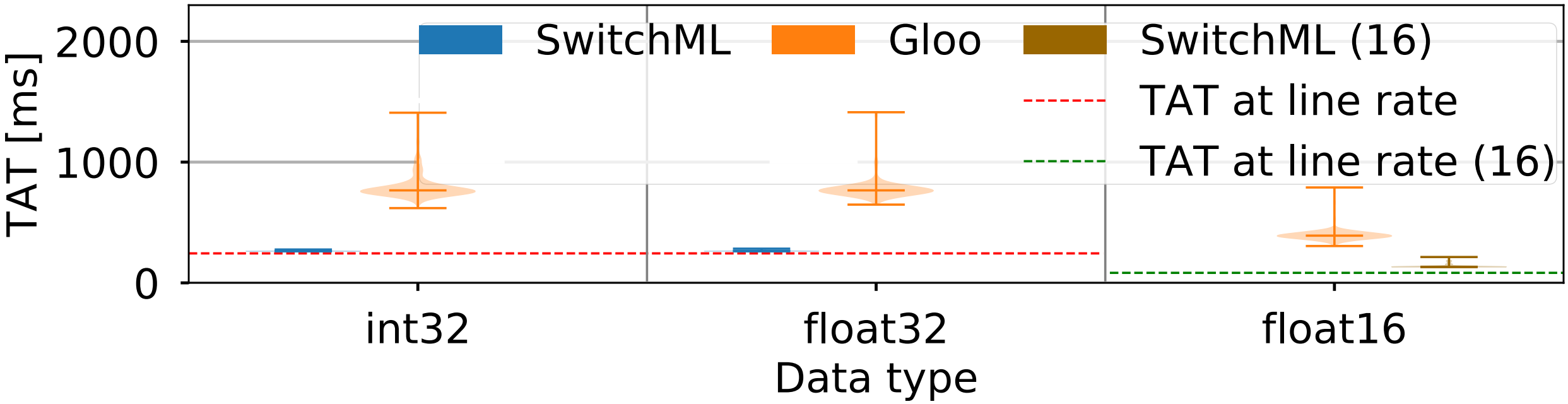
SwitchML has a lower inflation than TCP

Reasonable packet loss rates have no impact on performance



Does quantization affects aggregation speed?

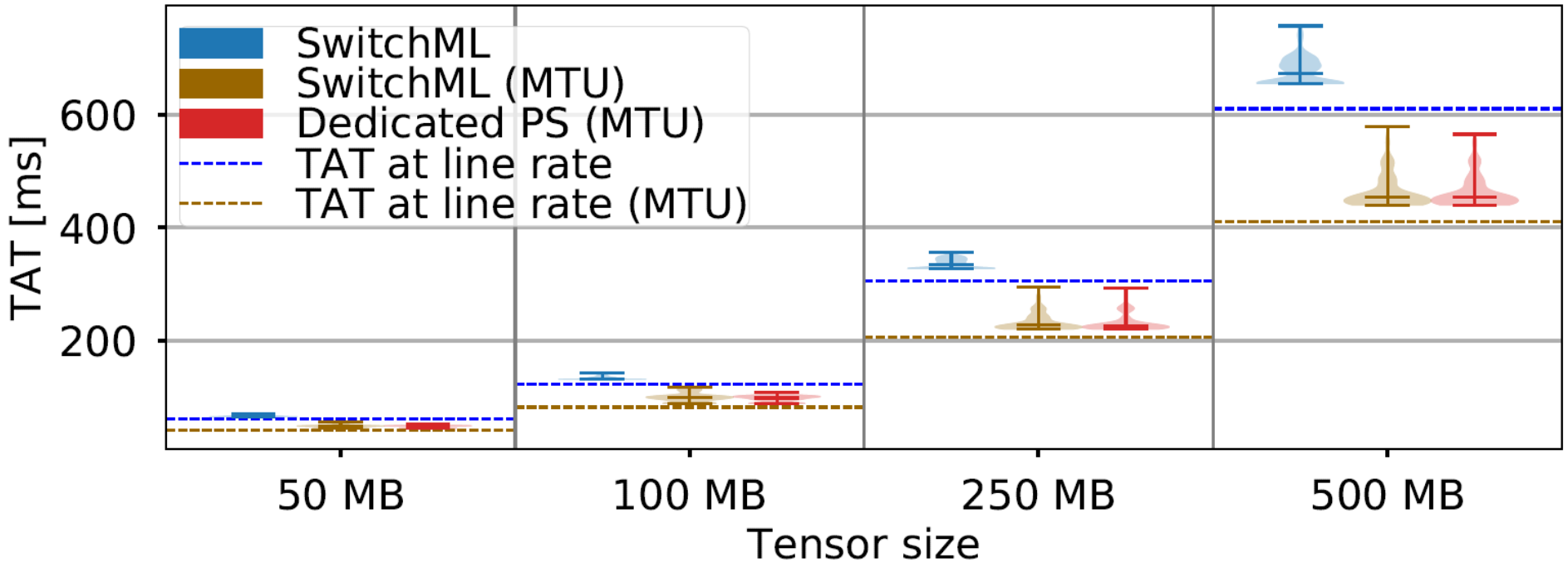
Tensor Aggregation Time unaffected by quantization thanks to AVX instructions



How much does packet size affect performance?

SwitchML reaches line rate with small packets

Would have ~30% better performance if the switch could support MTU-sized packets



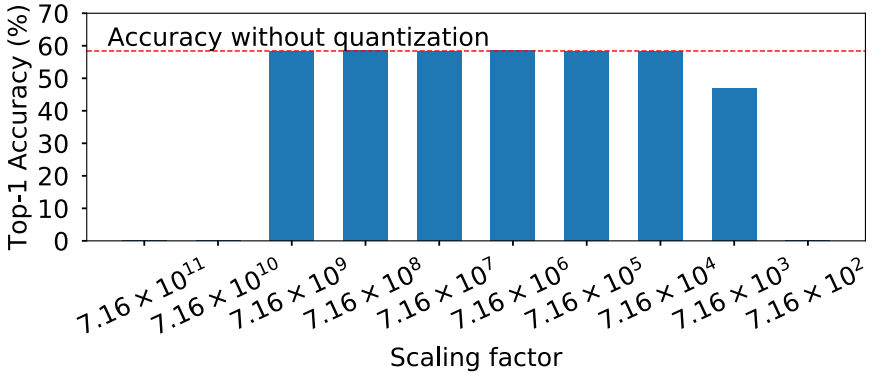
Quantization

- Convert floating point to 32-bit fixed-point values
- Updates are scaled by multiplying for a scaling factor sf

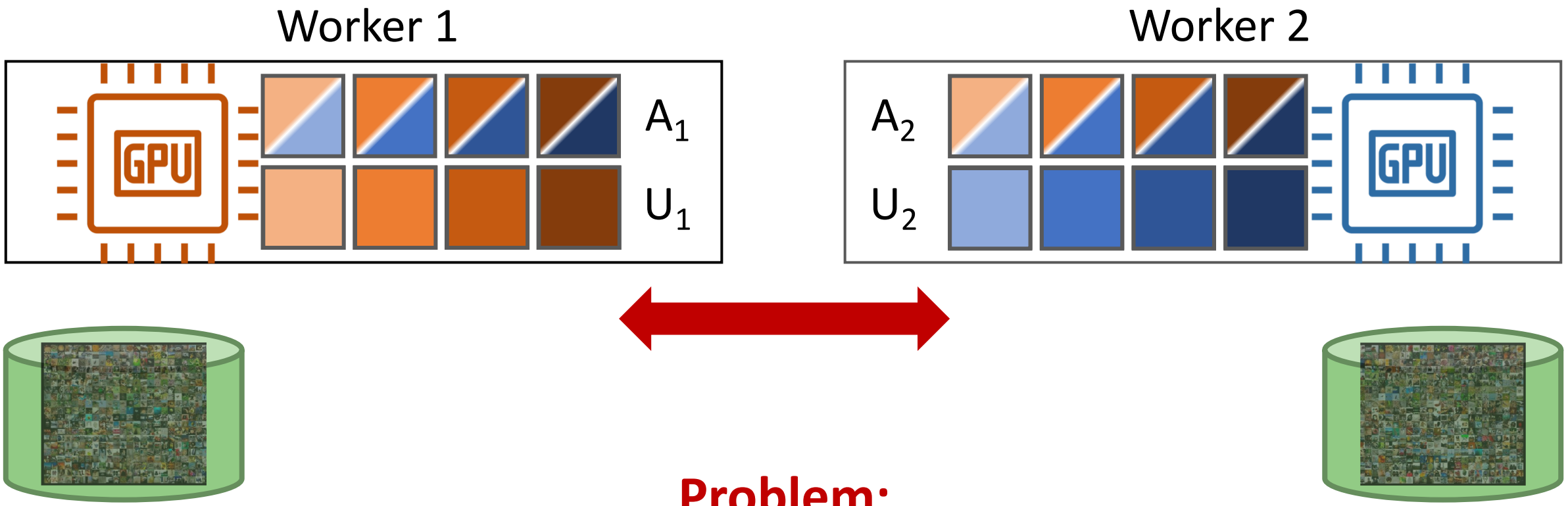
$$\widetilde{U_j^i} = round\left(sf * U_j^i\right) \qquad \widetilde{A_j^i} = A_j^i / sf$$

- 32-bit floating point ↔ 32-bit fixed point conversion ⟶ At workers with AVX instructions
- 16-bit floating point ↔ 32-bit fixed point conversion ⟶ Directly in the switch
  - Scaling is still done by the worker using AVX instructions

This quantization allows training to similar accuracy in a similar number of iterations as an unquantized network for a large range of scaling factors



Aggregation is communication-intensive



Problem:

Very intensive communication in all-to-all fashion!

Network increasingly the bottleneck to training speed