ATP: In-network Aggregation for Multi-tenant Learning

ChonLam Lao, Yanfang Le, Kshiteej Mahajan, Yixi Chen, Wenfei Wu*, Aditya Akella, Michael Swift

NSDI 2021 (Best Paper Award)

*Corresponding Author

Outline

- **Background**
- ATP
- Implementation and Evaluation
- Review

Background

- Parameter Server

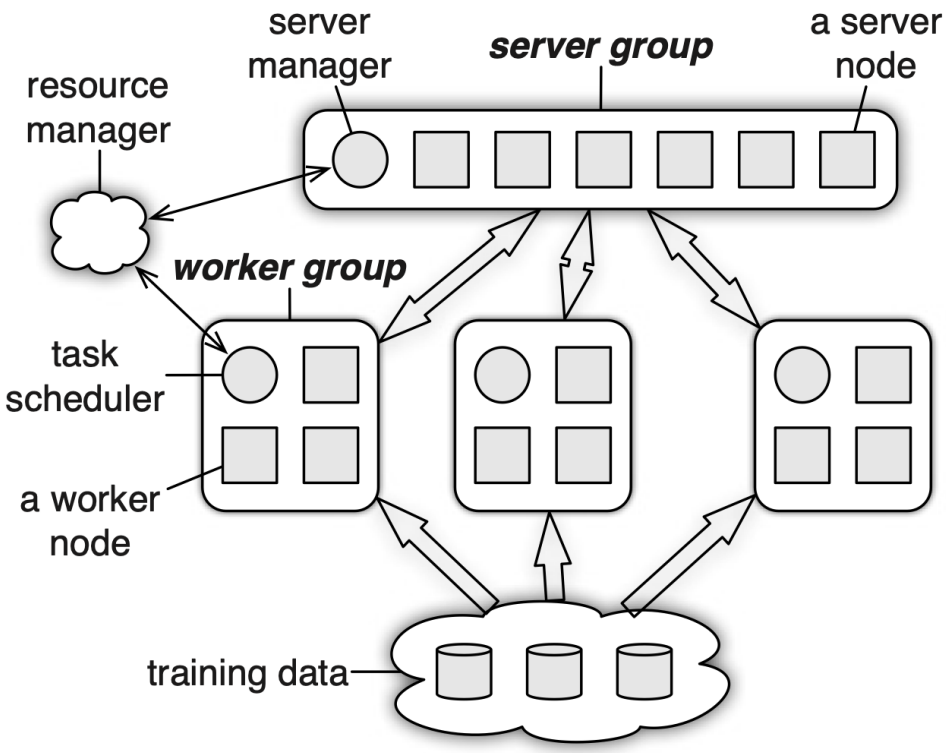
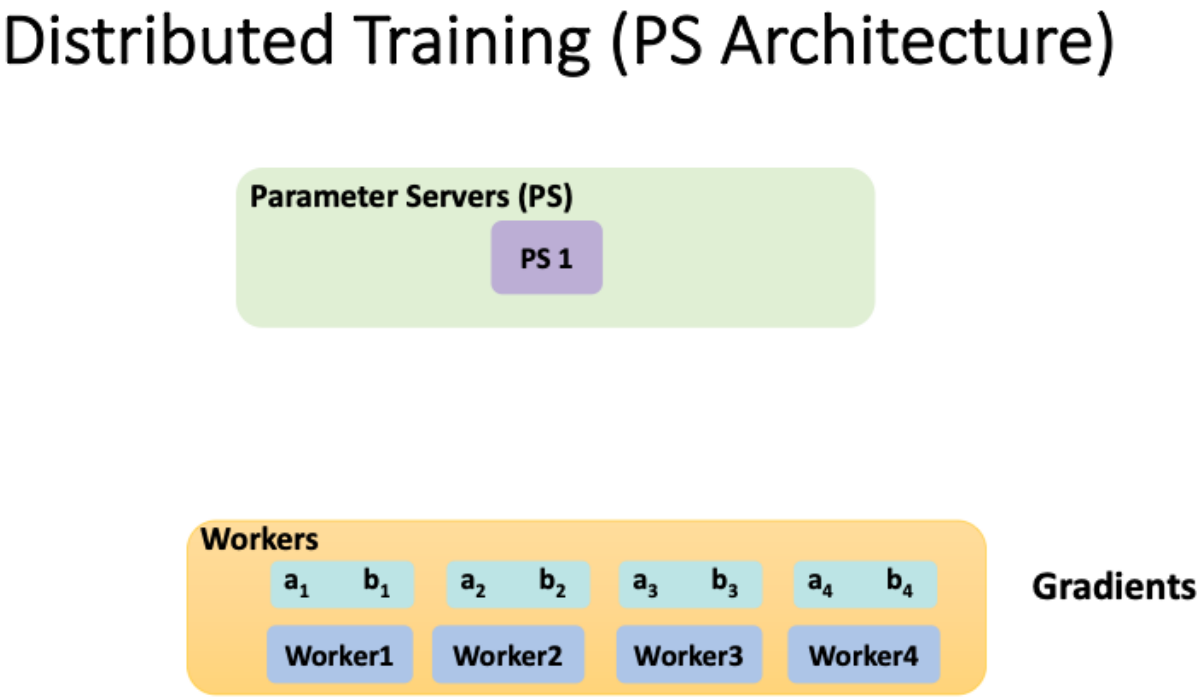


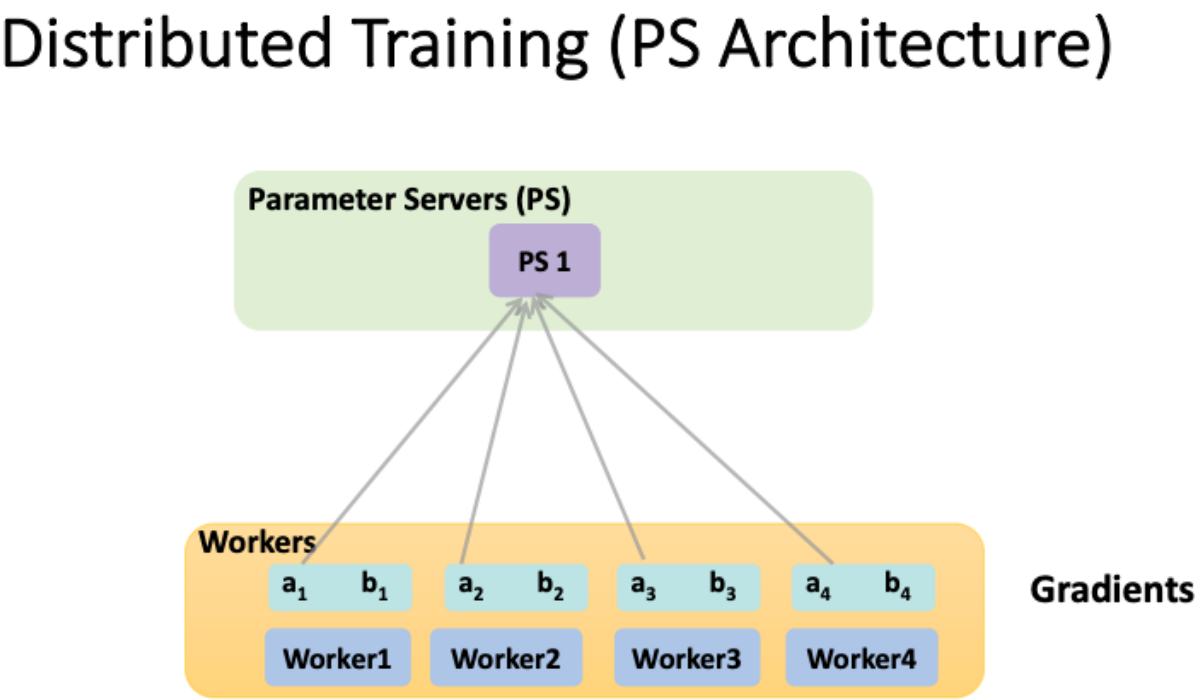
Figure 4: Architecture of a parameter server communicating with several groups of workers.

Li Mu, David G. Andersen, et al., Scaling Distributed Machine Learning with the Parameter Server, **OSDI 2014**

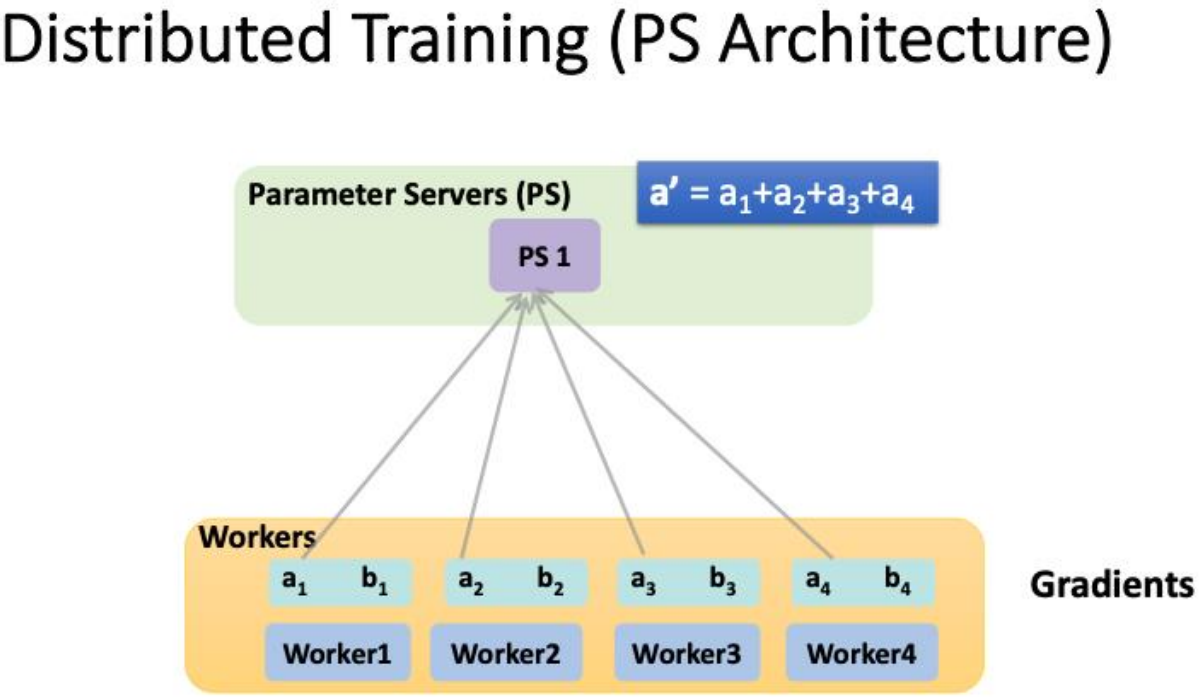
Background


Distributed Training (PS Architecture)
Parameter Servers (PS)
PS 1
Workers
a_1 b_1
a_2 b_2
a_3 b_3
a_4 b_4
Worker1
Worker2
Worker3
Worker4
Gradients
2

Background

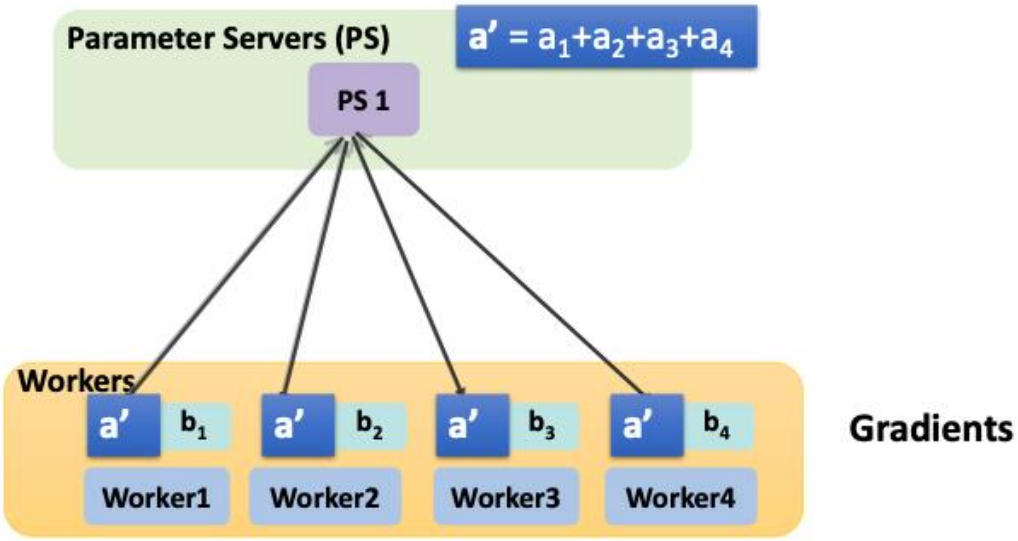

Distributed Training (PS Architecture)
Parameter Servers (PS)
PS 1
Workers
a_1 b_1
a_2 b_2
a_3 b_3
a_4 b_4
Worker1
Worker2
Worker3
Worker4
Gradients
2

Background

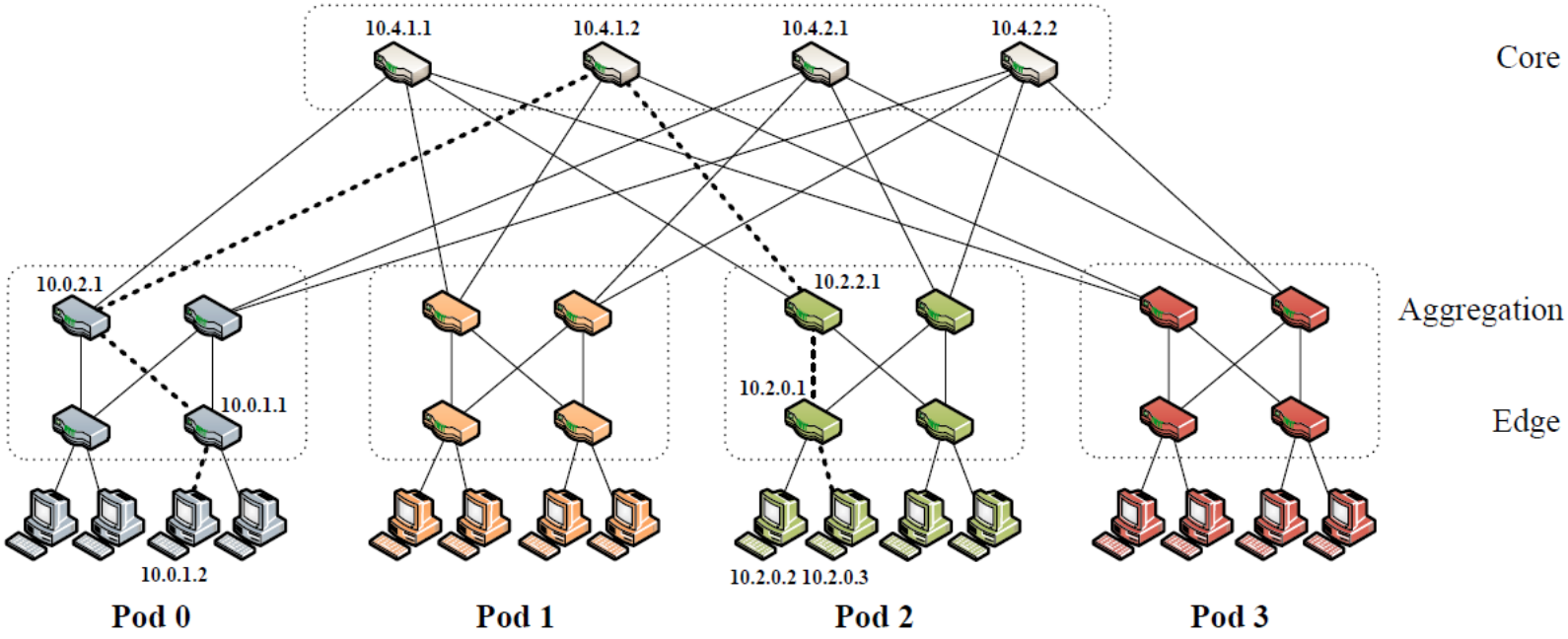

Distributed Training (PS Architecture)
Parameter Servers (PS)
a' = a1+a2+a3+a4
PS 1
Workers
a1 b1
a2 b2
a3 b3
a4 b4
Worker1
Worker2
Worker3
Worker4
Gradients
2

Background

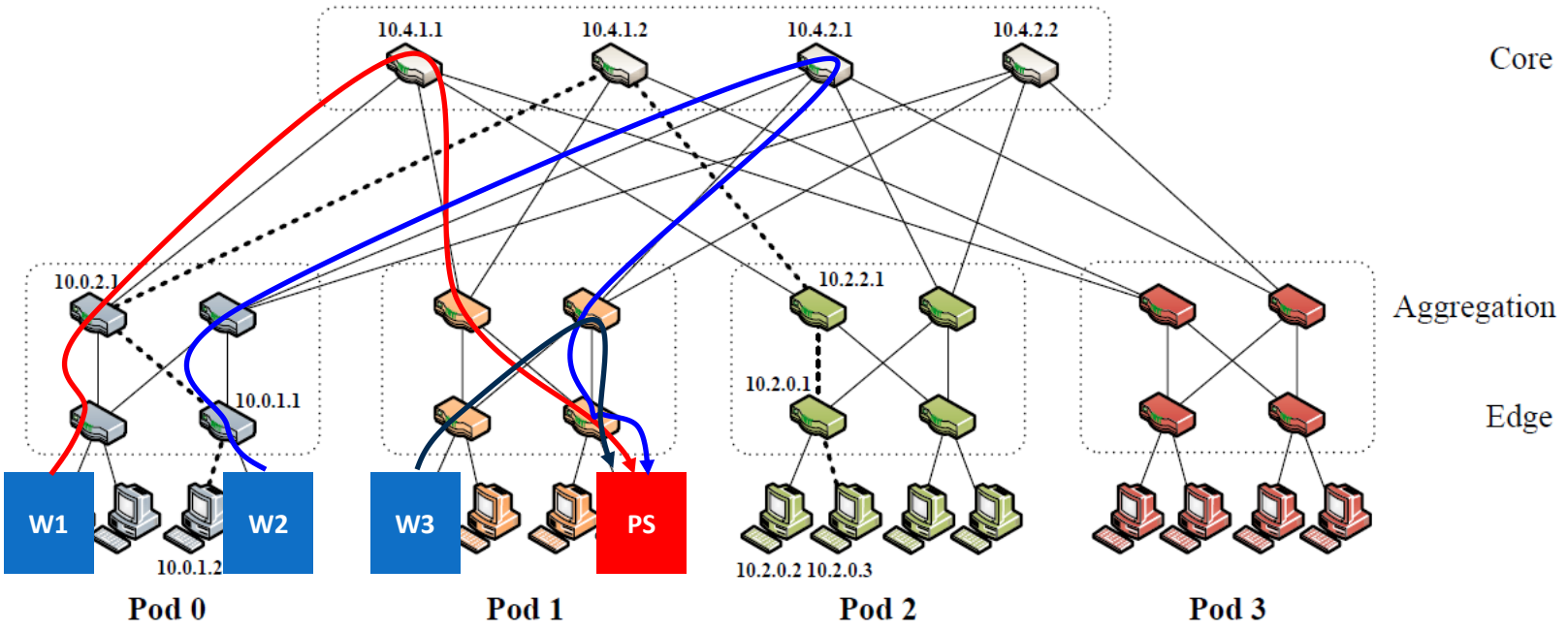
Distributed Training (PS Architecture)



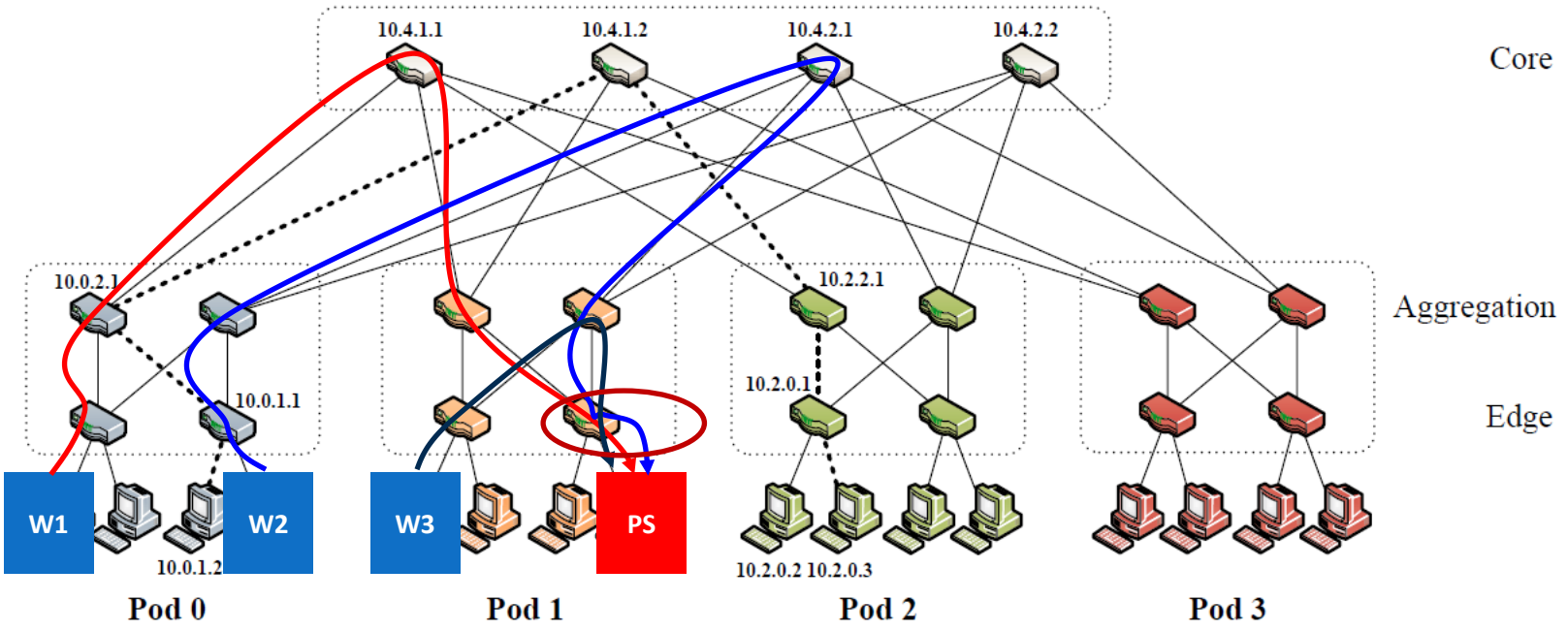
Motivation


10.4.1.1
10.4.1.2
10.4.2.1
10.4.2.2
Core
10.0.2.1
10.0.1.1
10.0.1.2
10.2.2.1
10.2.0.1
10.2.0.2 10.2.0.3
Aggregation
Edge
Pod 0
Pod 1
Pod 2
Pod 3

Motivation

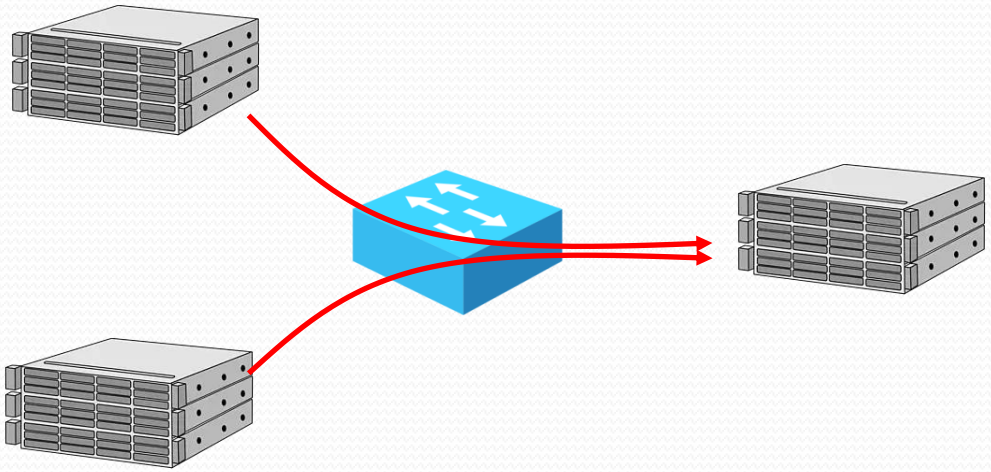

10.4.1.1
10.4.1.2
10.4.2.1
10.4.2.2
Core
10.0.2.1
10.0.1.1
10.0.1.2
10.2.2.1
10.2.0.1
10.2.0.2 10.2.0.3
Aggregation
Edge
W1
W2
W3
PS
Pod 0
Pod 1
Pod 2
Pod 3

Motivation



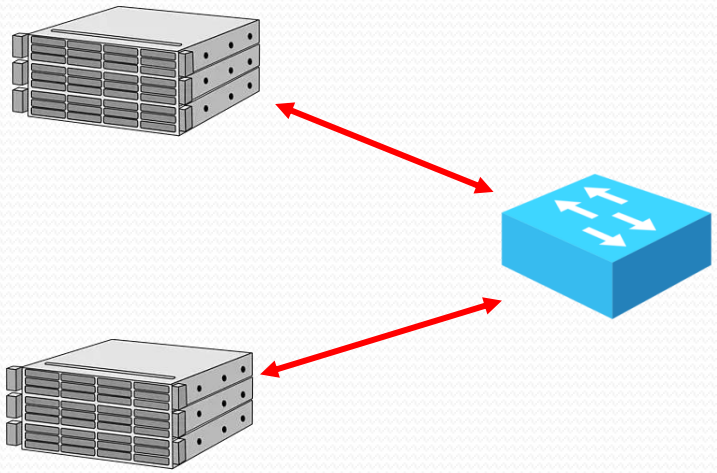
Network becomes a bottleneck!

In-network Aggregation



N to 1 communication pattern causes the congestion?

In-network Aggregation



What if the switch can do the gradient aggregation in the network?

Programmable Switch: A Potential Solution

Trend of In-network Computation

- Programmable switch offers in-transit packet processing and in-network state

3

Programmable Switch: A Potential Solution

Trend of In-network Computation

- Programmable switch offers in-transit packet processing and in-network state

3

Programmable Switch: A Potential Solution

Trend of In-network Computation

- Programmable switch offers in-transit packet processing and in-network state

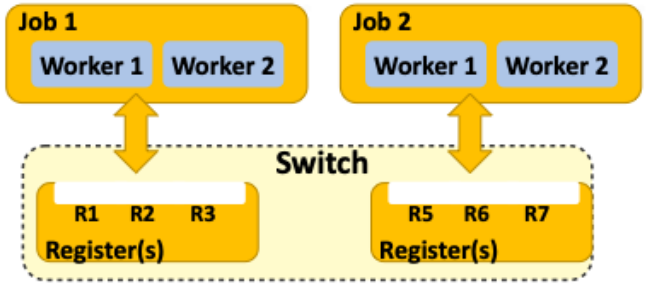
- Reduce training time by moving gradient aggregation into the network

3

Switch ML

State-of-the-art In-network Aggregation

- SwitchML (Sapio et al. NSDI'21)
  - Target single-rack settings
  - Support multiple jobs by static partitioning of switch resources



4

Scaling Distributed Machine Learning with In-Network Aggregation, NSDI 2021

Limitations of a Programmable Switch

SwitchML: Co-design ML and networking

Challenges

- Limited computation
- Limited storage
- **No floating points**
- Packet loss

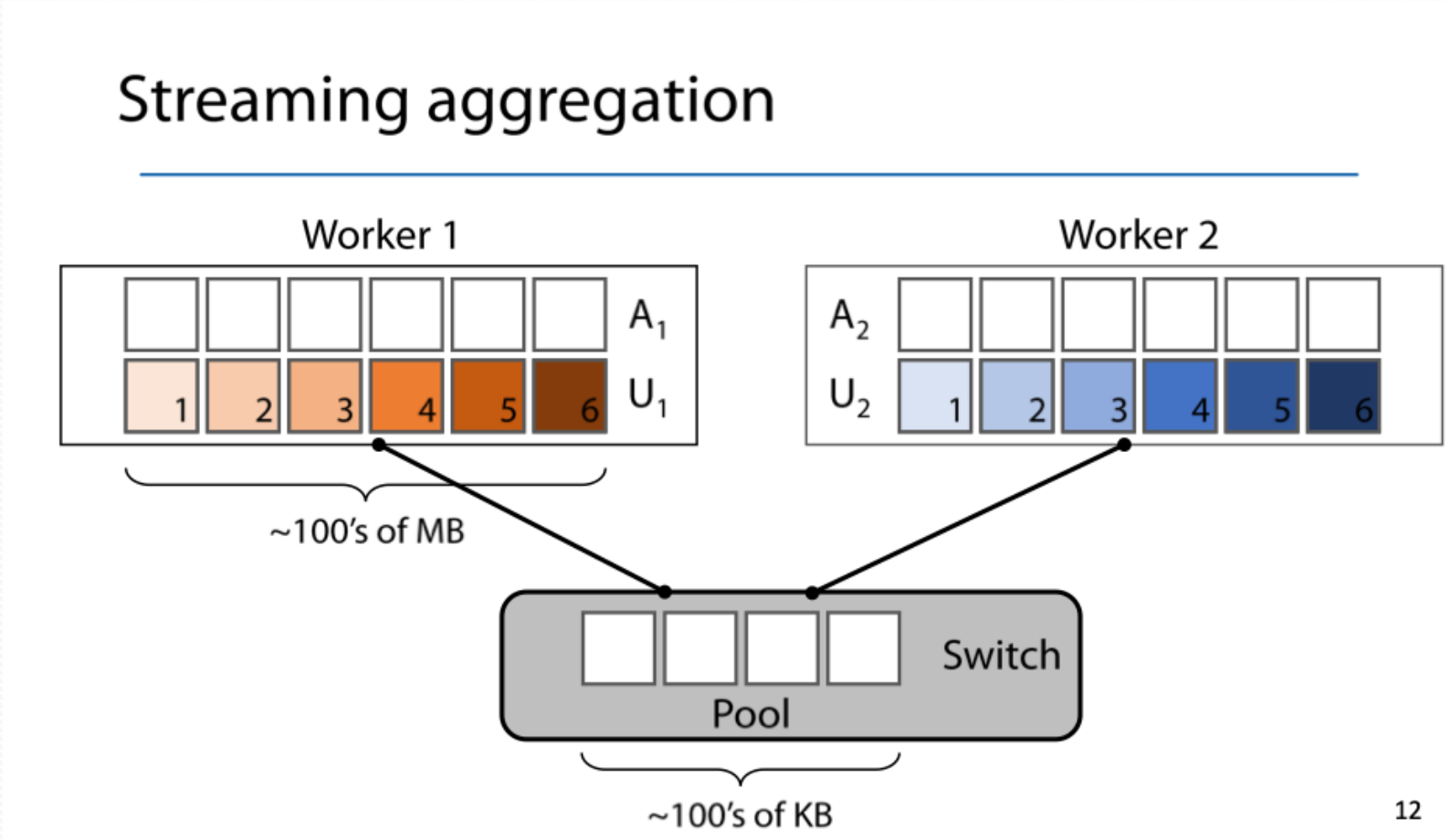
6.5 Tbps programmable data plane

Design

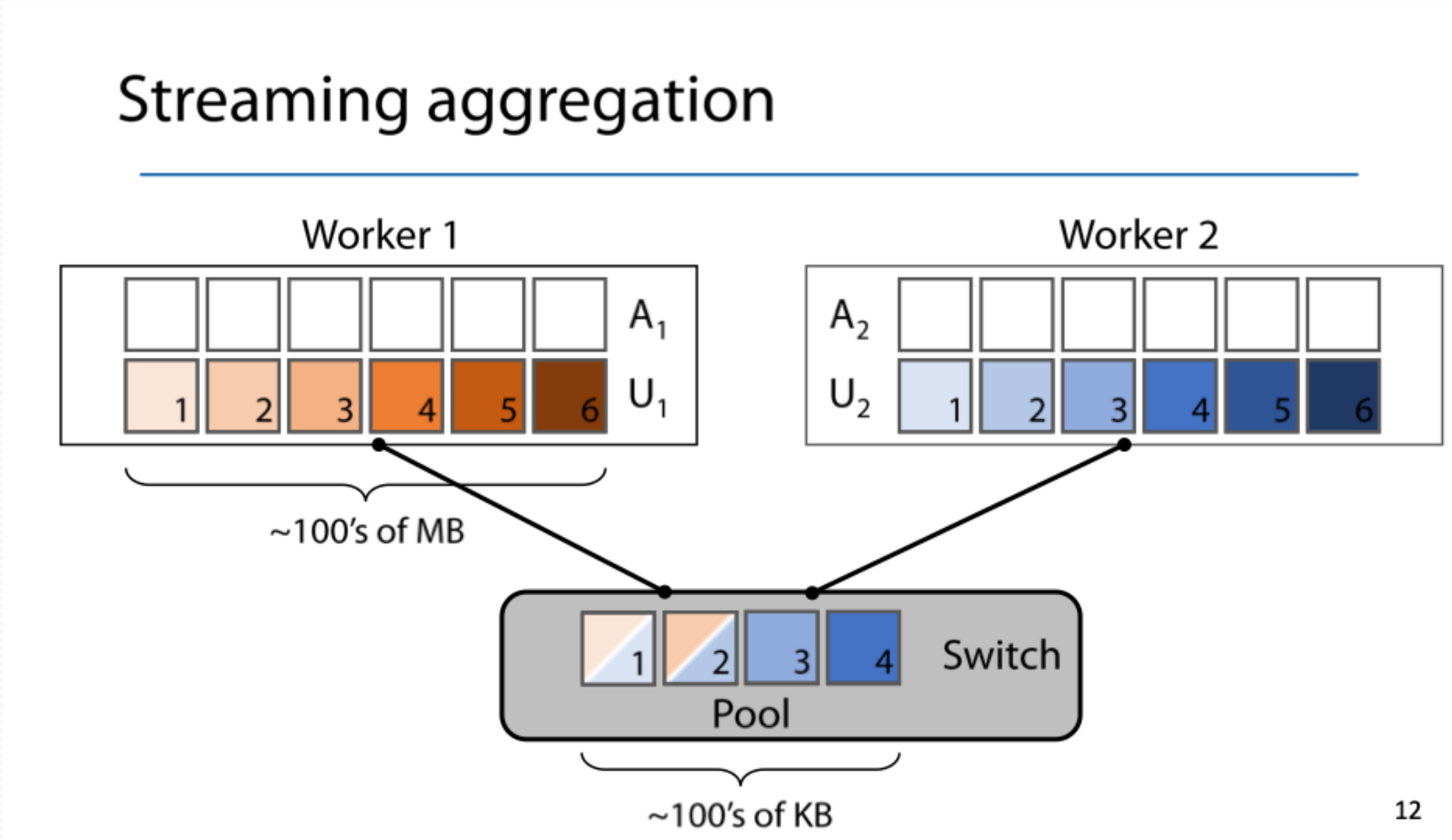
- Combined switch-host architecture
- Pool-based streaming aggregation
- Quantized integer operations
- Failure-recovery protocol
- In-switch RDMA implementation

11

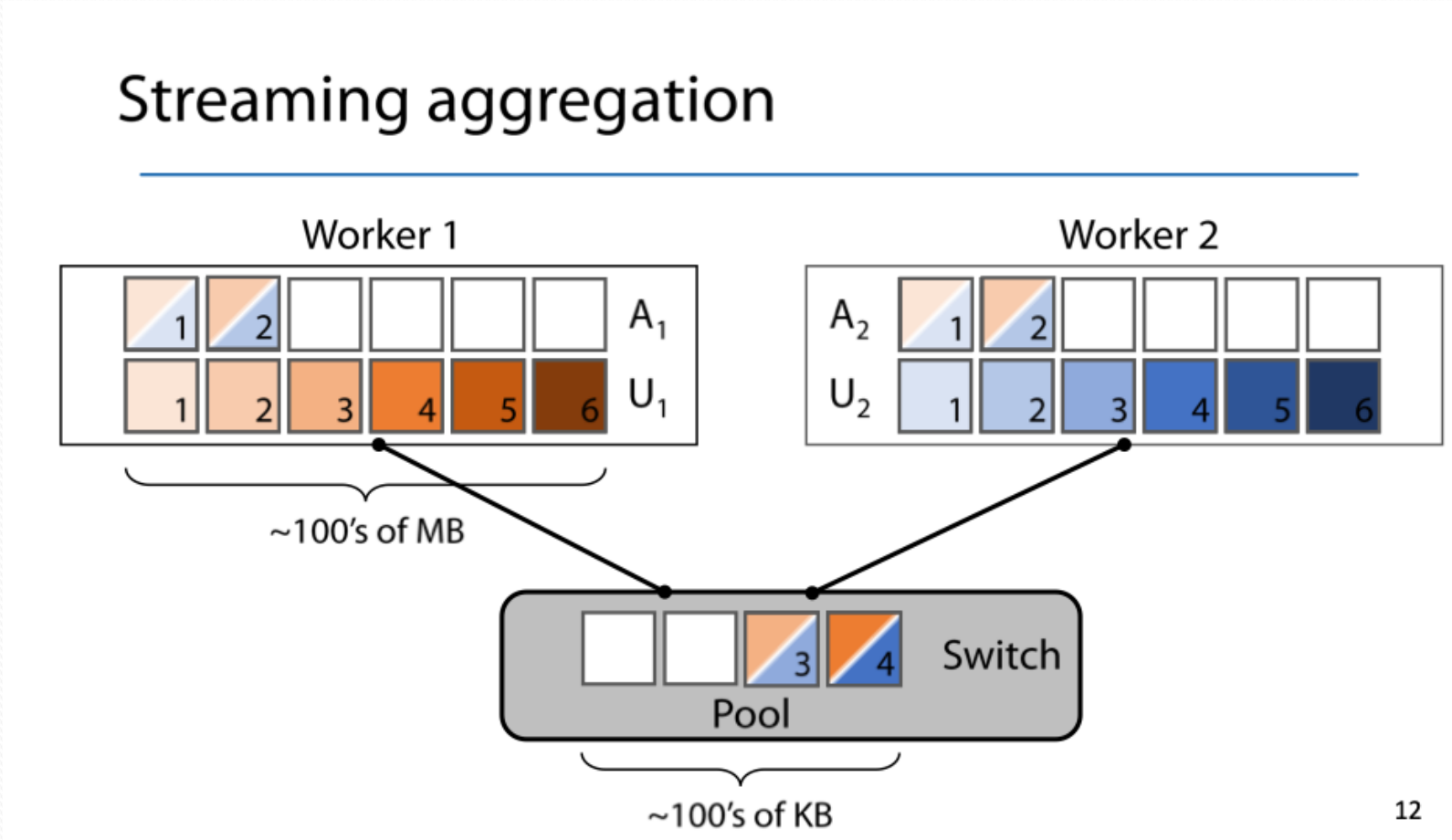
Limited Computation & Resources



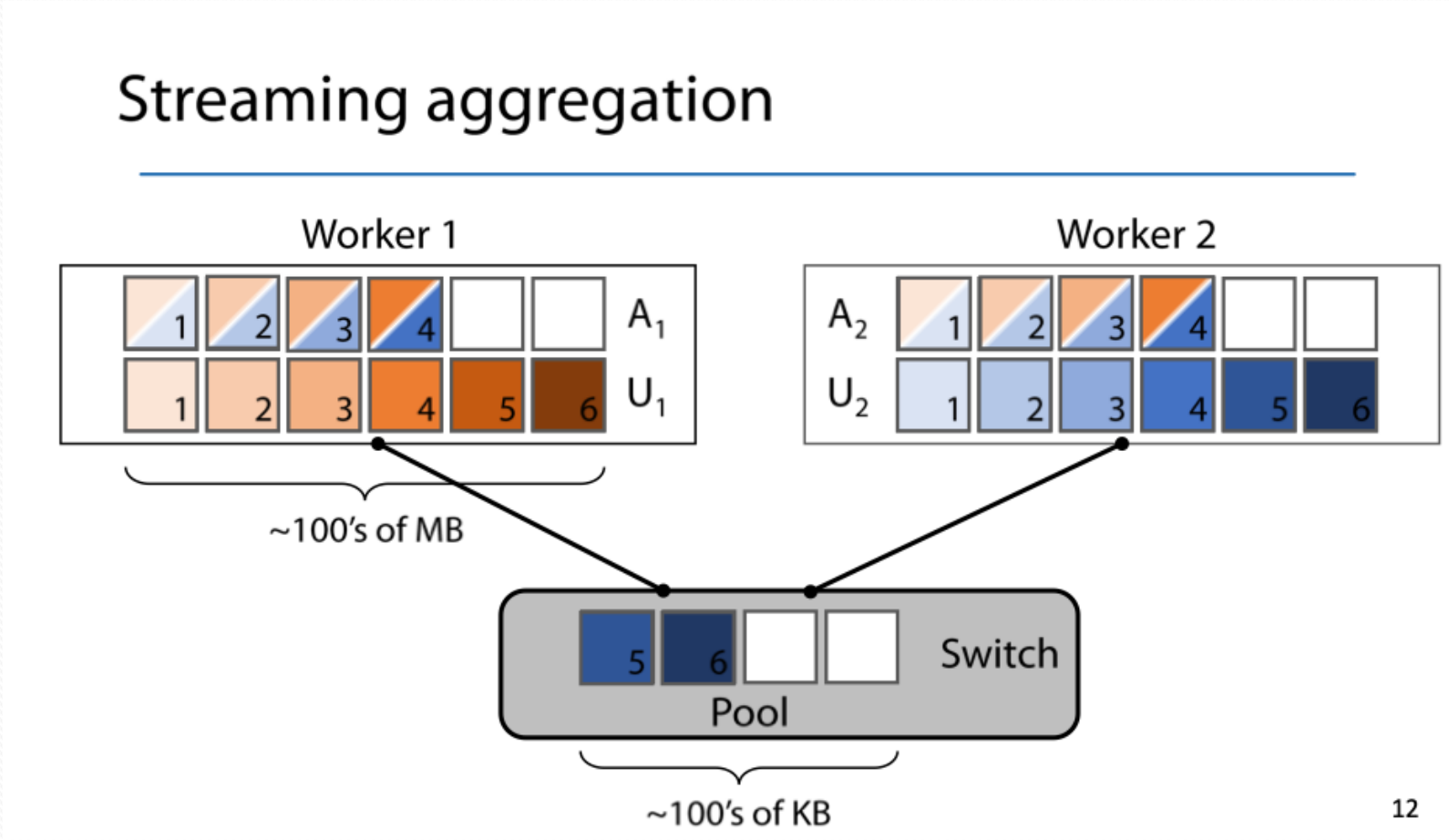
Limited Computation & Resources



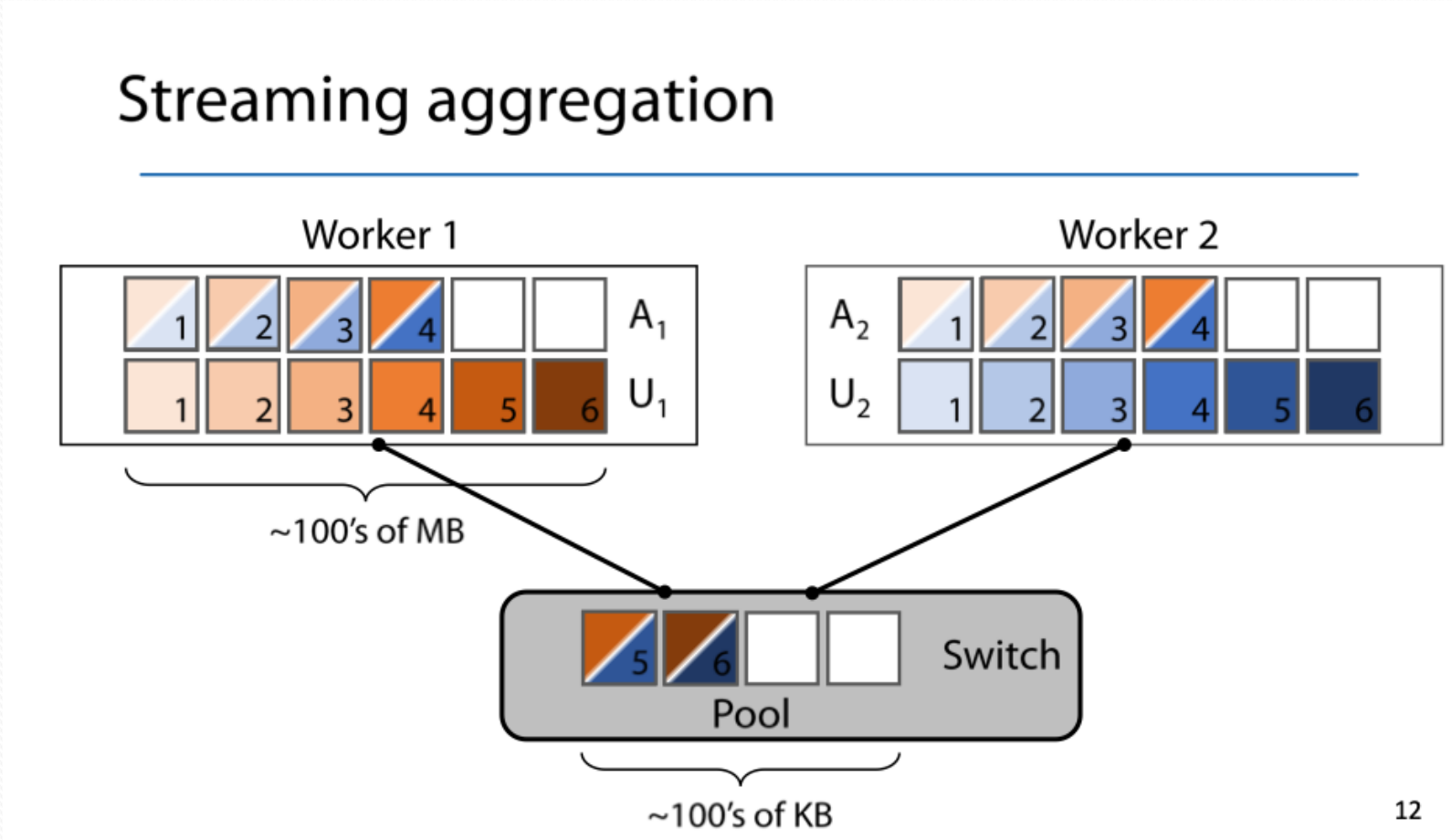
Limited Computation & Resources



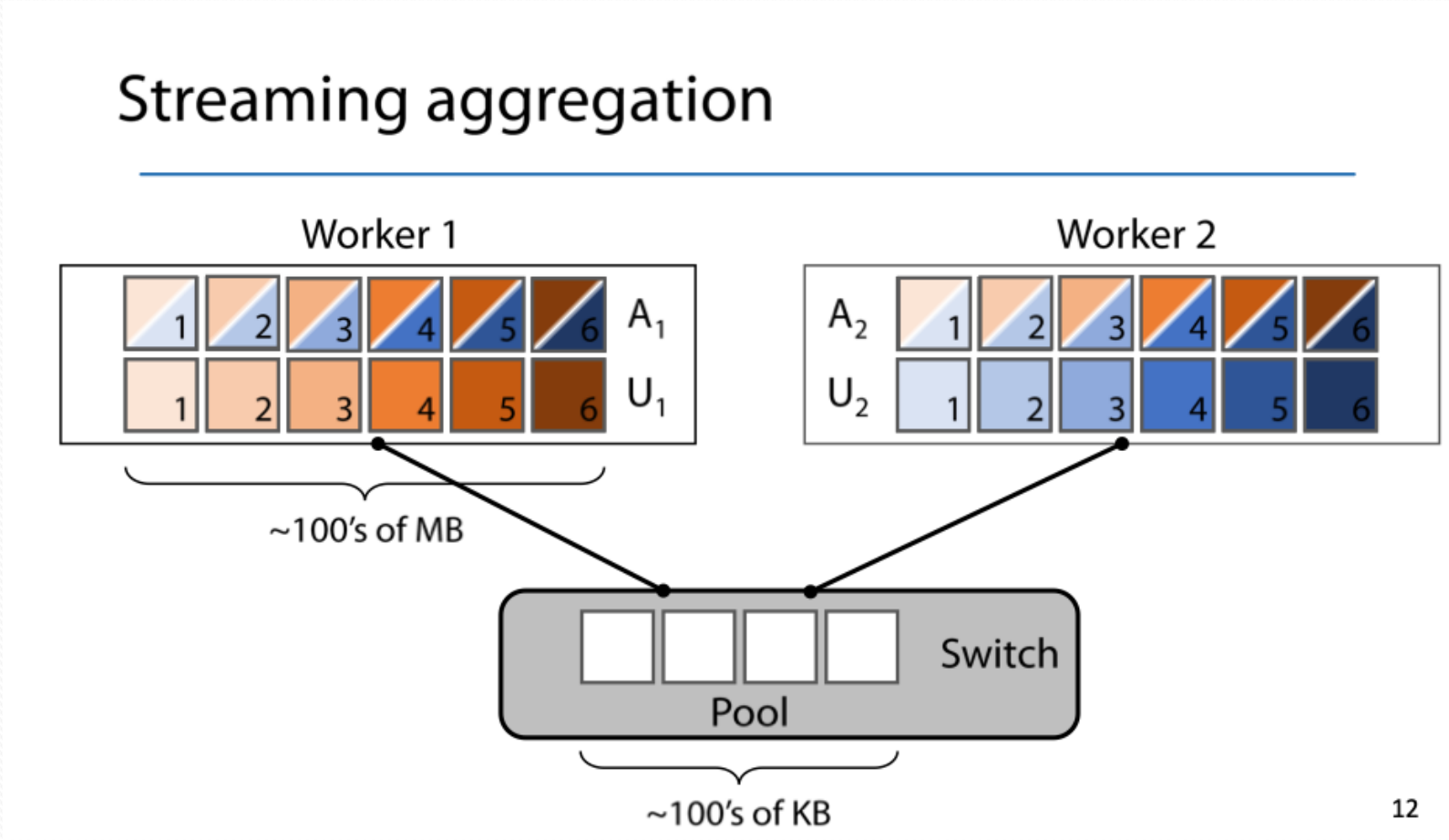
Limited Computation & Resources



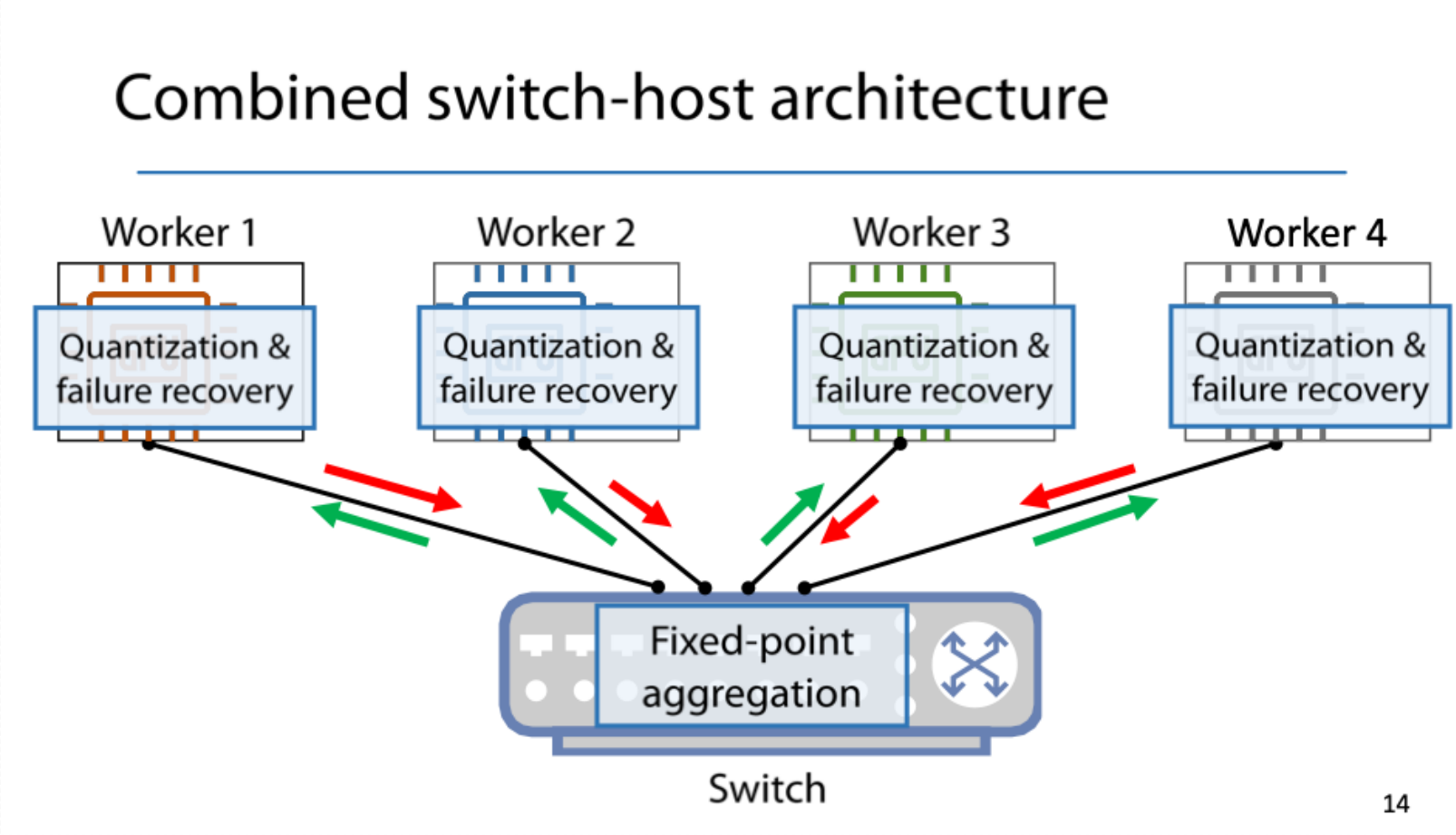
Limited Computation & Resources



Limited Computation & Resources

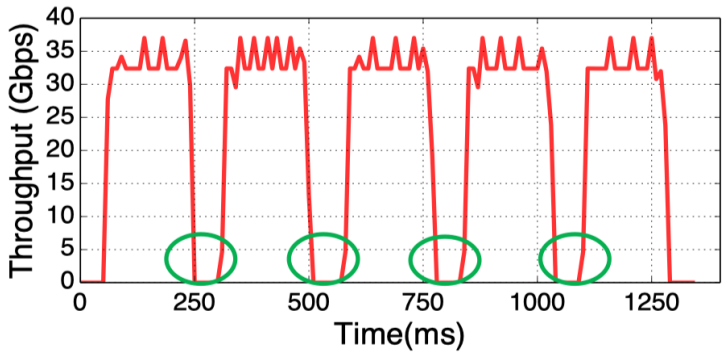


No Floating Number Support



Limitations of Switch ML

- Inefficiently use the switch resources
- Does not consider multi-rack setting



BERT-Large Training Times on GPUs

Time	System	Number of Nodes	Number of V100 GPUs
47 min	DGX SuperPOD	92 x DGX-2H	1,472
67 min	DGX SuperPOD	64 x DGX-2H	1,024

Key Goal

Speed up multiple DT jobs in a cluster while maximizing the benefits from in-network multi-switch aggregation

Outline

- Background
- **ATP**
- Implementation and Evaluation
- Review

Design Goal of ATP

- **Multi-tenant**
- Multi-rack
- Additional challenges
  - Reliability
  - Congestion control
  - Improve floating point computation

Multi-tenant

- Why SwitchML fails to support Multi-tenant?
- SwitchML is designed with:
  - Single training job
  - Static aggregation assignment
  - Dedicated switch memory

Multi-tenant

- However, multiple jobs run simultaneously in real-world environment
- Each job requires:
  - Switch Memory
  - Aggregation slots

Limited Switch Memory

- **Static memory allocation is insufficient!**
- **Example:** Switch memory = 100 aggregators
  - Job A needs 80
  - Job B needs 80

Static Allocation

- Job A → 50 slots
- Job B → 50 slots
- Problem:
  - Poor Performance
  - Low Utilization
  - Unfairness

Shared Without Isolation

- Jobs overwrite each other
- Problem:
  - Incorrect aggregation
  - Job performance interference

ATP's Core Idea

- **Best-Effort In-Network Aggregation**
  - Switch aggregates when resources are available, otherwise, fall back to end host
  - *If switch memory available*
    → aggregate in network
    Else
    → forward to PS
- **Need a PS server**

ATP Packet Format

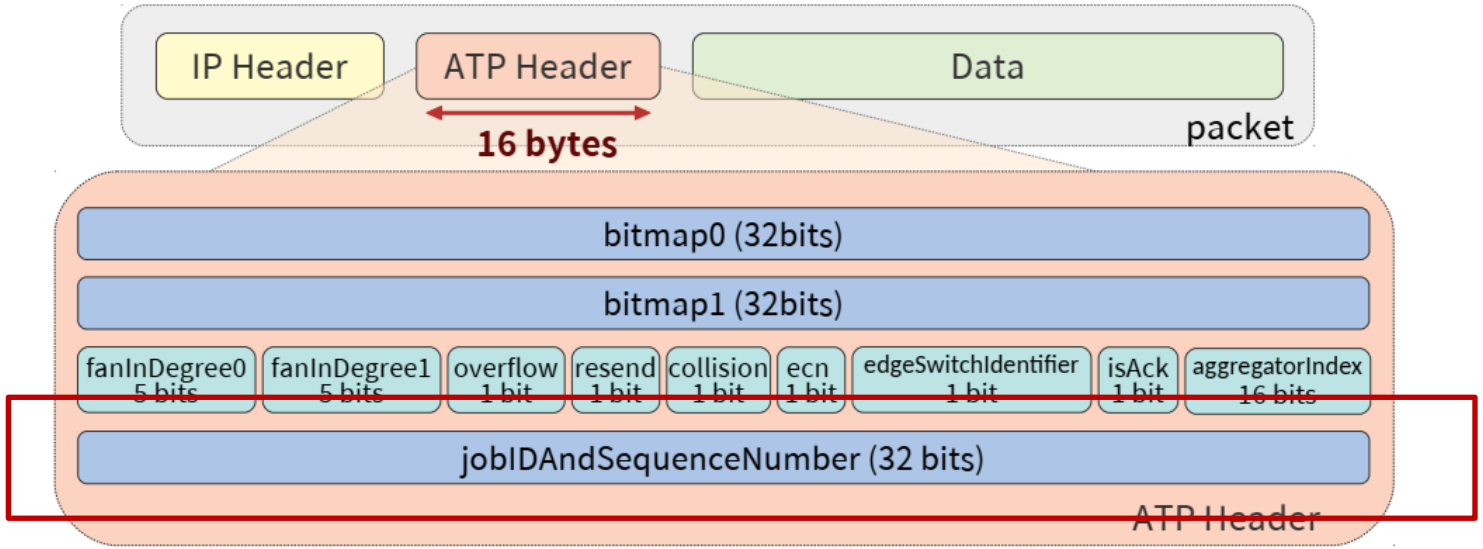


Figure 5: ATP packet format.

ATP Packet Format

- ATP uses **gradient fragment packets**
- Packet Identification
  - jobID
  - sequenceNumber
- Used to identify:
  - Which job
  - Which gradient fragment

ATP Packet Format

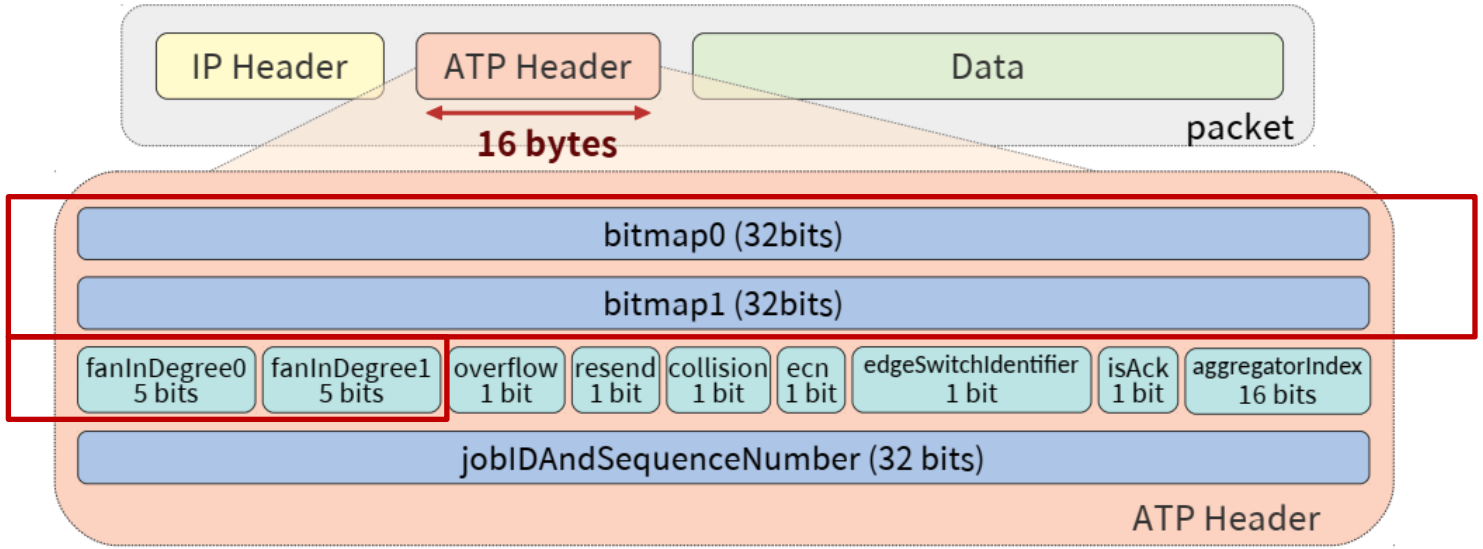


Figure 5: ATP packet format.

ATP Packet Format

- Aggregation Metadata
  - bitmap -> The position of the worker
  - fanInDegree -> The number of workers
- **Track aggregation progress**

ATP Packet Format

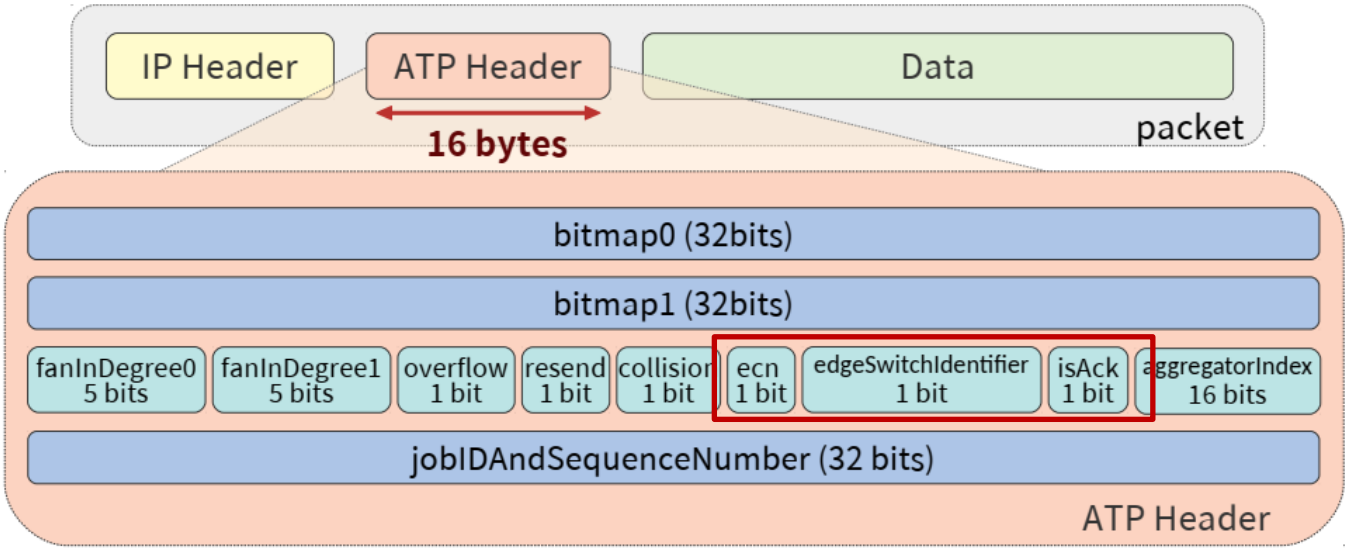


Figure 5: ATP packet format.

ATP Packet Format

- Aggregator Selection
  - Aggregator Index
- **Deterministically derived from other fields**
- Aggregator Index = hash(jobID, seq) % number Of Aggregators
- Consistent across different hosts

ATP Packet Format

- Control Flags
  - collision
  - resend
  - ECN
  - isAck
- Function
  - multi-tenant isolation
  - reliability
  - congestion control

ATP Switch Memory Layout

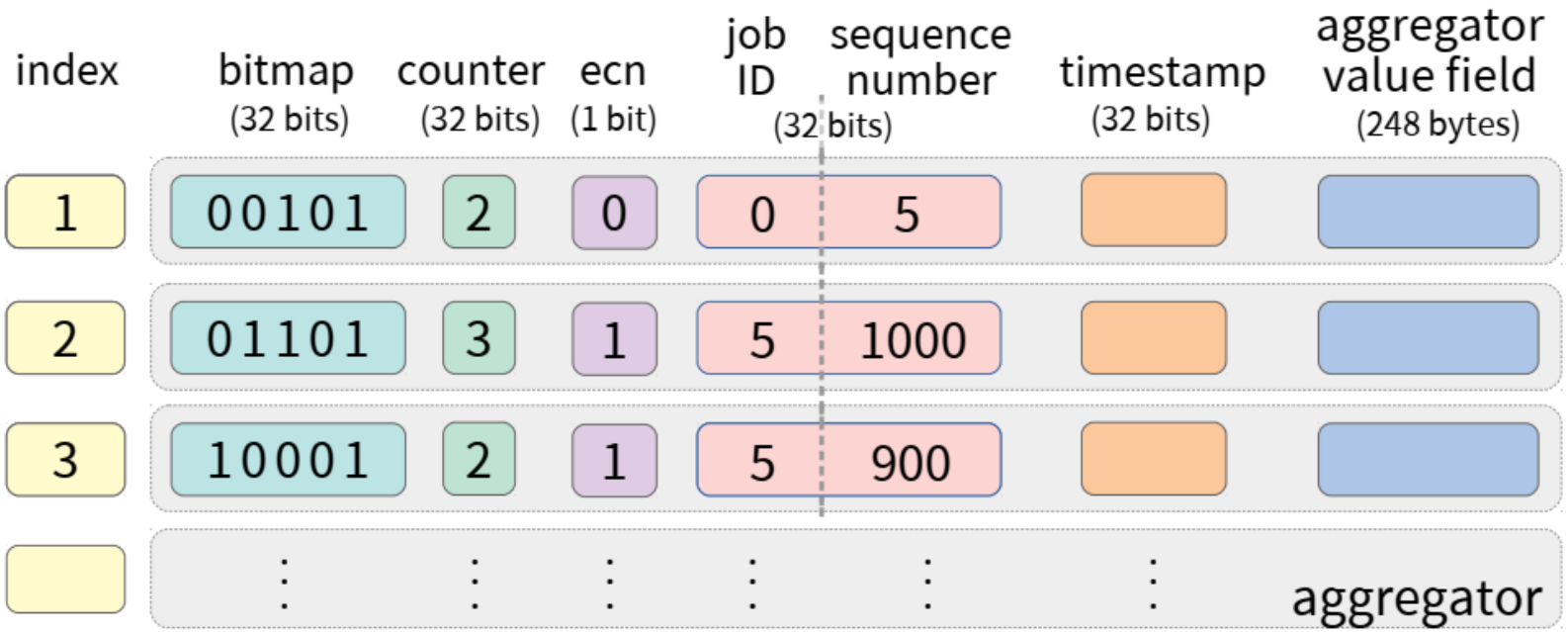


Figure 6: ATP Switch memory layout.

Switch Logic (Simplified Version)

- When a packet arrives
  - check bitmap to determine if the packet has already been aggregated
  - *value += gradients*
  - *counter ++*
- *When aggregation completes:*
  - *counter == fanInDegree*

Endhost Logic

- Push gradient fragment packets toward the PS and receive updated parameters back
- The PS accepts gradient fragment packets, as well as **partially or fully aggregated packets**
  - PS collects aggregated gradients as an array of <bitmap, value> indexed by sequence number.
  - The bitmap tracks which workers' gradient fragments have been aggregated in the value field.

Collision Detection

- If the aggregator has already been used by other jobs (**hash collision**)
- Forward the packet to PS
- Set collision field to 1 to avoid repetitive check next rounds

Why Multi-tenant?

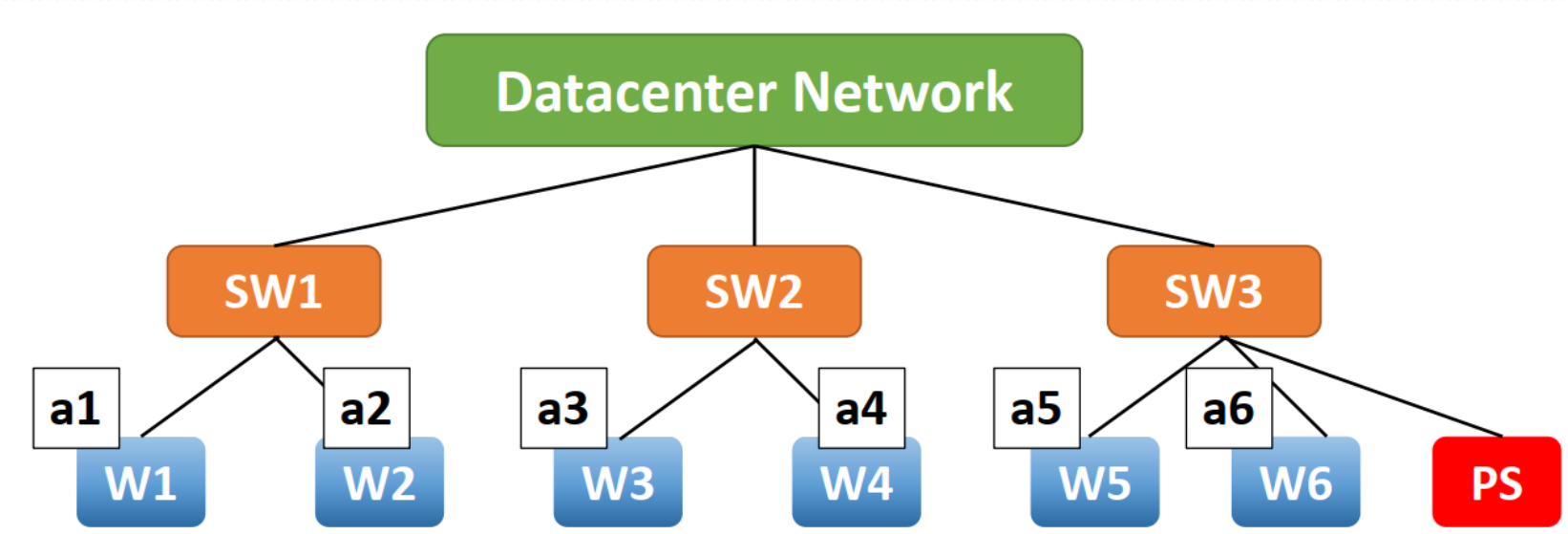
- ATP does not reserve aggregator per job
  - aggregatorIndex = hash(jobID, seq) % N
- Best effort Aggregation
  - If aggregator overwritten, packet lost happen, ATP fallback to PS aggregation
  - Ensure correctness when shared by multiple jobs
- Collision Detection

Design Goal of ATP

- Multi-tenant
- **Multi-rack**
- Additional challenges
  - Reliability
  - Congestion control
  - Improve floating point computation

Inter-Rack Aggregation

- Aggregation at every layer of network topology
  - Nondeterministic routing, i.e., ECMP
- Support two-level aggregation at ToR switches
  - Workers and PS(es) are located in different racks



Design Goal of ATP

- Multi-tenant
- Multi-rack
- **Additional challenges**
  - Reliability
  - Congestion control
  - Improve floating point computation

Additional Challenges

- Rethink reliability
  - Recovery from packet loss
  - Ensure exact once aggregation
  - Memory leak: aggregators are reserved forever, but not used
- Rethink congestion control
  - N flows merged into one flow communication
  - Drop congestion signal, i.e., ECN
- Improve the floating point computation
  - Convert gradients to 32-bit integer at workers by a scaling factor
  - Aggregation overflow at switch

Outline

- Background
- ATP
- **Implementation and Evaluation**
- Review

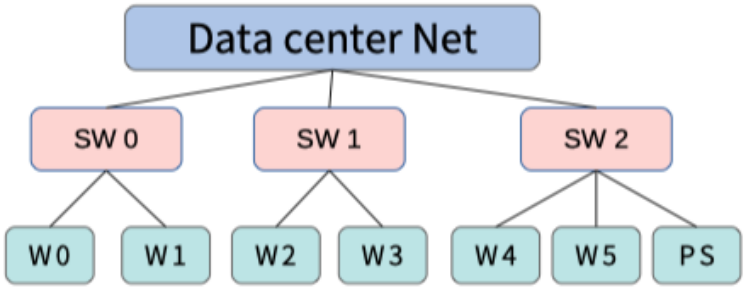
Implementation

- Replace the networking stack of BytePS on the end host
- Use P4 to implement the in-network aggregation service on a Barefoot Tofino switch

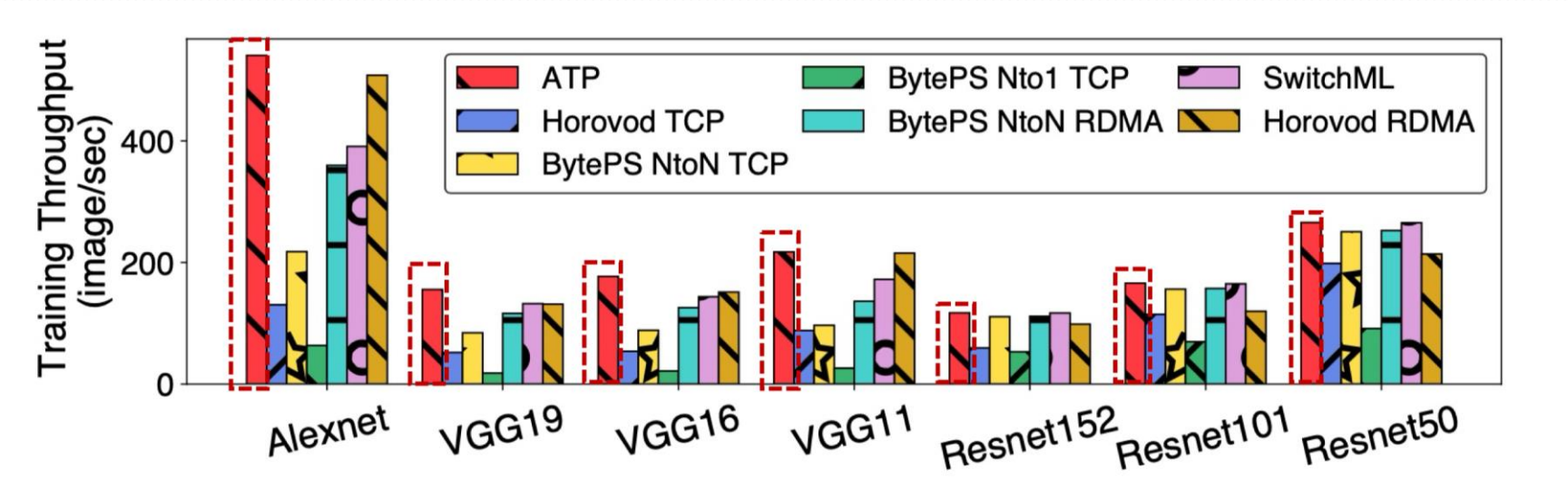
https://github.com/in-ATP/ATP

Evaluation

- Setup: 9 servers, each with one GPU, one 100G NIC
- Baseline: (BytePS + TCP, BytePS+ RDMA) x (N to 1, N to N), SwitchML, Horovod+RDMA, Horovod+TCP
- Metrics: Training Throughput, Time-to-Accuracy
- Workloads: AlexNet, VGG11, VGG16, VGG19, ResNet50, ResNet101, and ResNet152

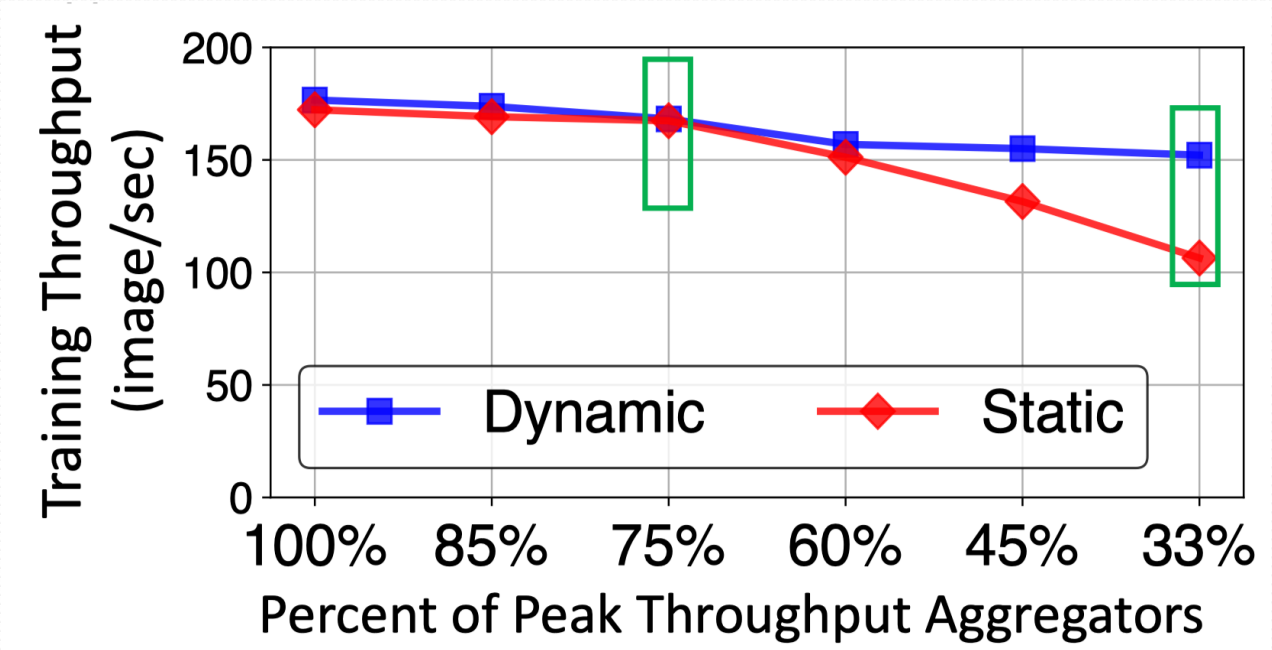


Single Job Performance



- ATP is comparable to, and outperforms the state-of-the-art approaches.
- ATP gets larger performance gains on network-intensive workloads (VGG) than the computation-intensive workloads (ResNet).

Multiple Jobs: Dynamic (ATP) vs Static



3 VGG16 Jobs

- Static approach evenly distributes aggregators to jobs
- PTA: the number of the aggregators to make each job to achieve the peak aggregation throughput

Multiple Jobs: Dynamic (ATP) vs Static

- When switch memory is sufficient, ATP's dynamic ≈ static
- When switch memory is insufficient, ATP's dynamic > static

Outline

- Background
- ATP
- Implementation and Evaluation
- **Review**

Review

- Programmable switches can do **more than just packet switching** nowadays!
- ATP
  - Multi-tenant
  - Multi-rack
- In-network gradient aggregation has been widely deployed to accelerate LLM training with NVIDIA's InfiniBand switch