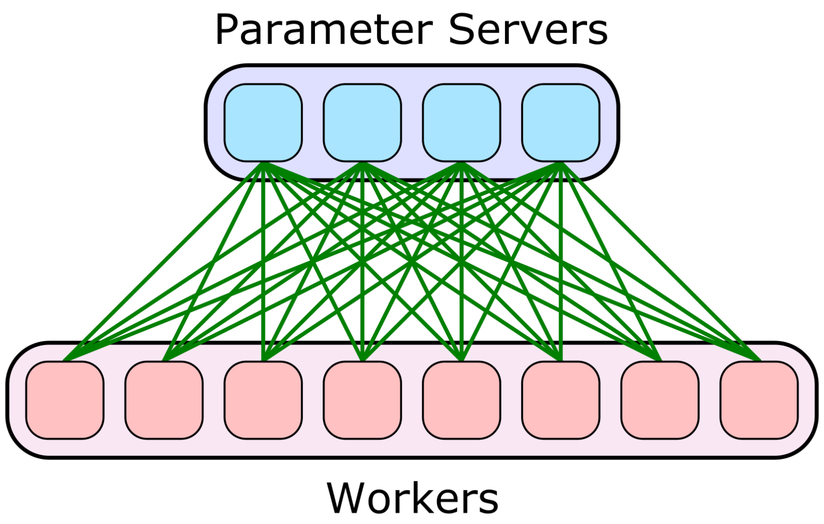
GOAT: Gradient Scheduling with Collaborative In-Network Aggregation for Distributed Training

Jin Fang

Gongming Zhao, Hongli Xu, Zhuolong Yu, Bingchen Shen, Liguang Xie

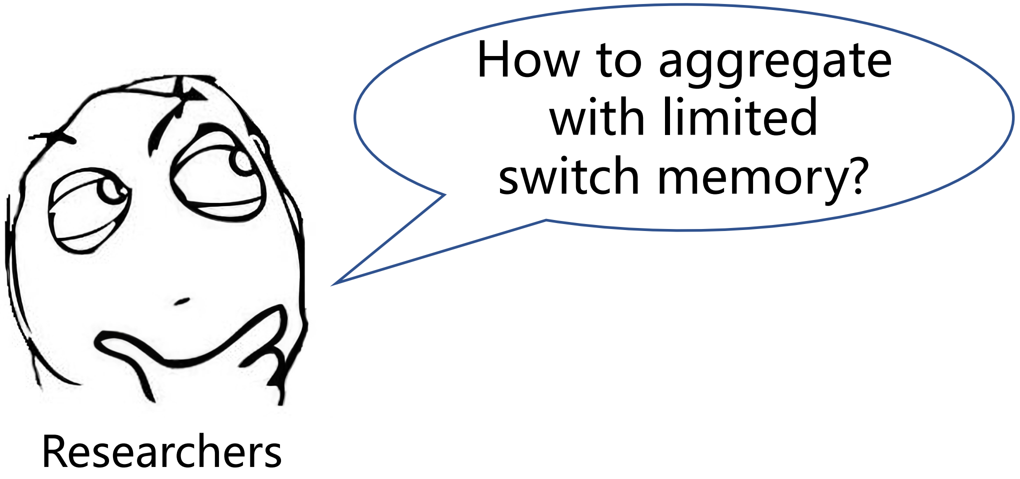
In-network Aggregation for DT

- With the increasing complexity of machine learning (ML) applications, the scale of ML tasks grows explosively
- **Distributed training** is proposed to meet the needs of training large-scale ML tasks
- Communication overhead has become the main bottleneck
- **In-network Aggregation**: utilize programmable switches to aggregate gradients within the network



Problem: Switch Memory Limitation

- Switch memory is used to buffer the intermedia aggregation value
- Current programmable switch has limited on-chip memory
  - Intel Tofino 1: 22MB
  - Intel Tofino 2: 64 MB
- Size of popular DNN models usually exceeds the size of switch memory
  - ResNet-50: 98MB
  - VGG-16: 528MB



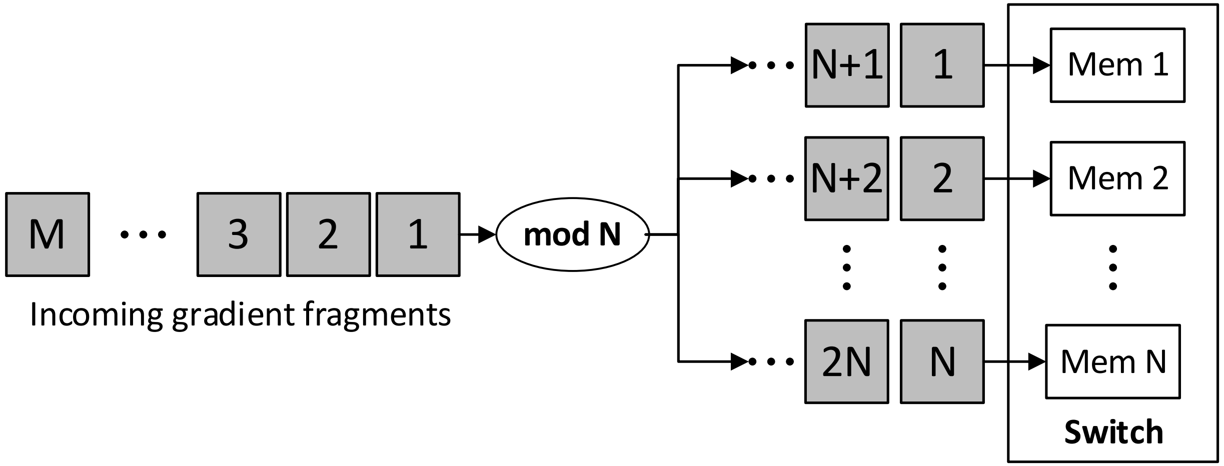
Existing Solution

Increase Memory Size

- ➢ Directly increasing the on-chip memory size **High cost!**
- ➢ TEA (SIGCOMM 20): utilizing external server memory to extend **Additional latency!**

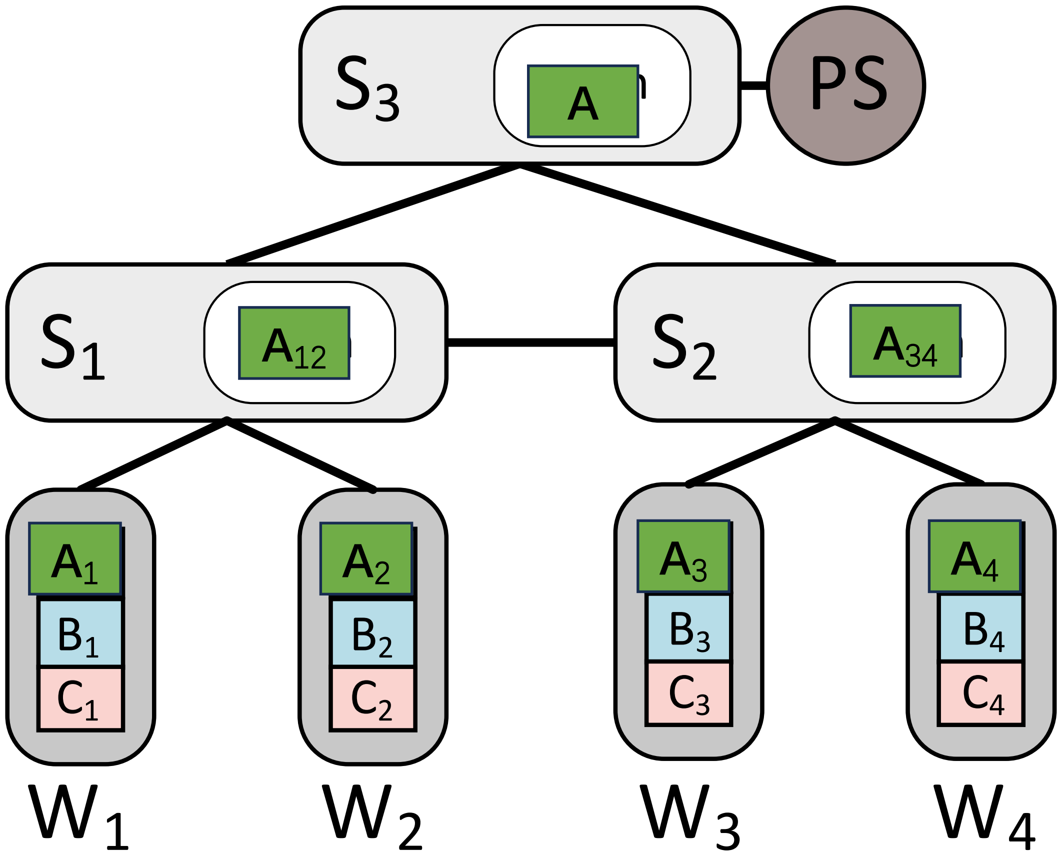
Memory Sharing Scheme

- ➢ ATP (NSDI 21): manage and **reuse** the switch on-chip memory



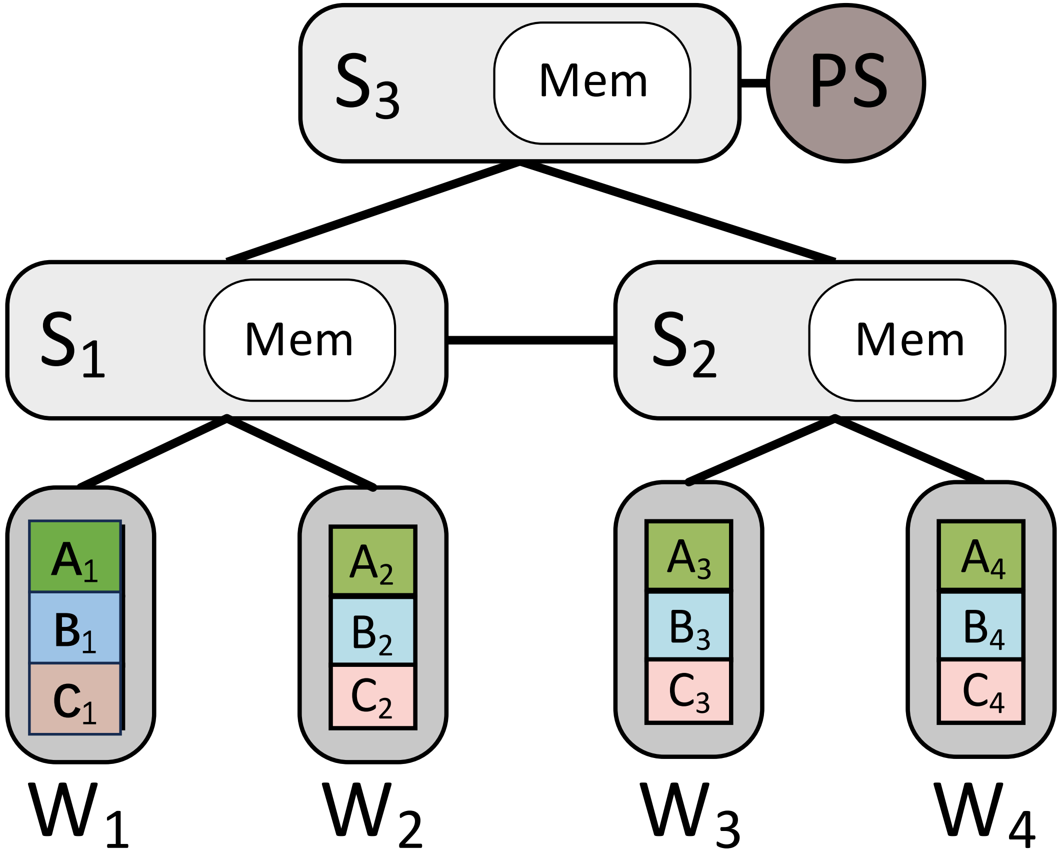
A Motivating Example

- Memory sharing scheme requires gradient fragments arriving at switches **simultaneously**



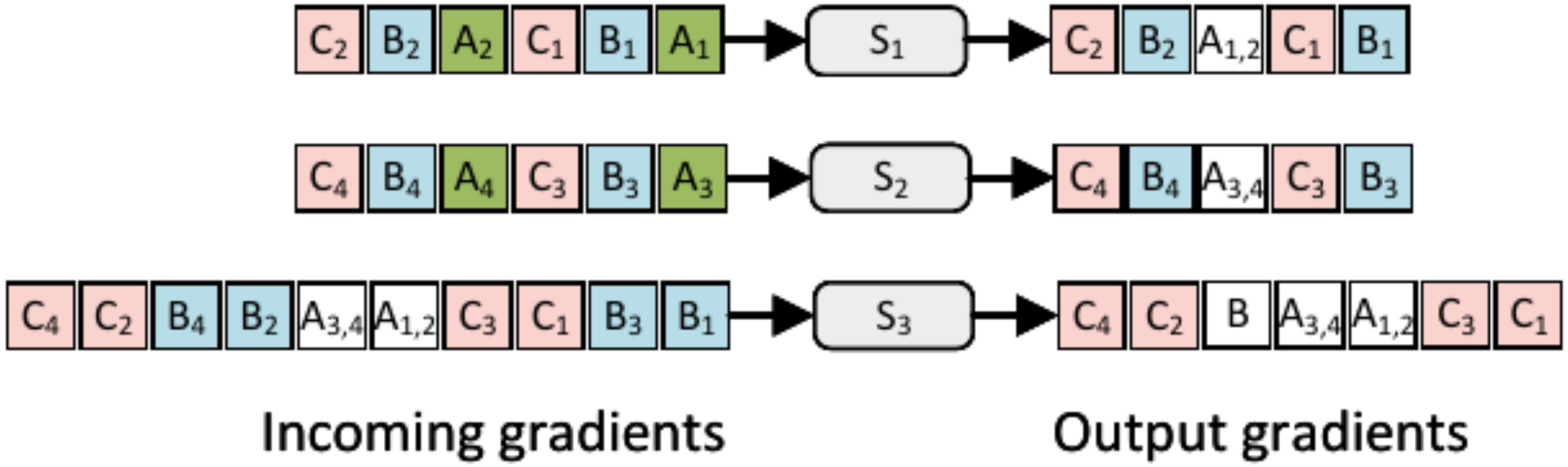
A Motivating Example

➢ **Asynchronously arriving gradient** fragments will increase the aggregation overhead of the PS



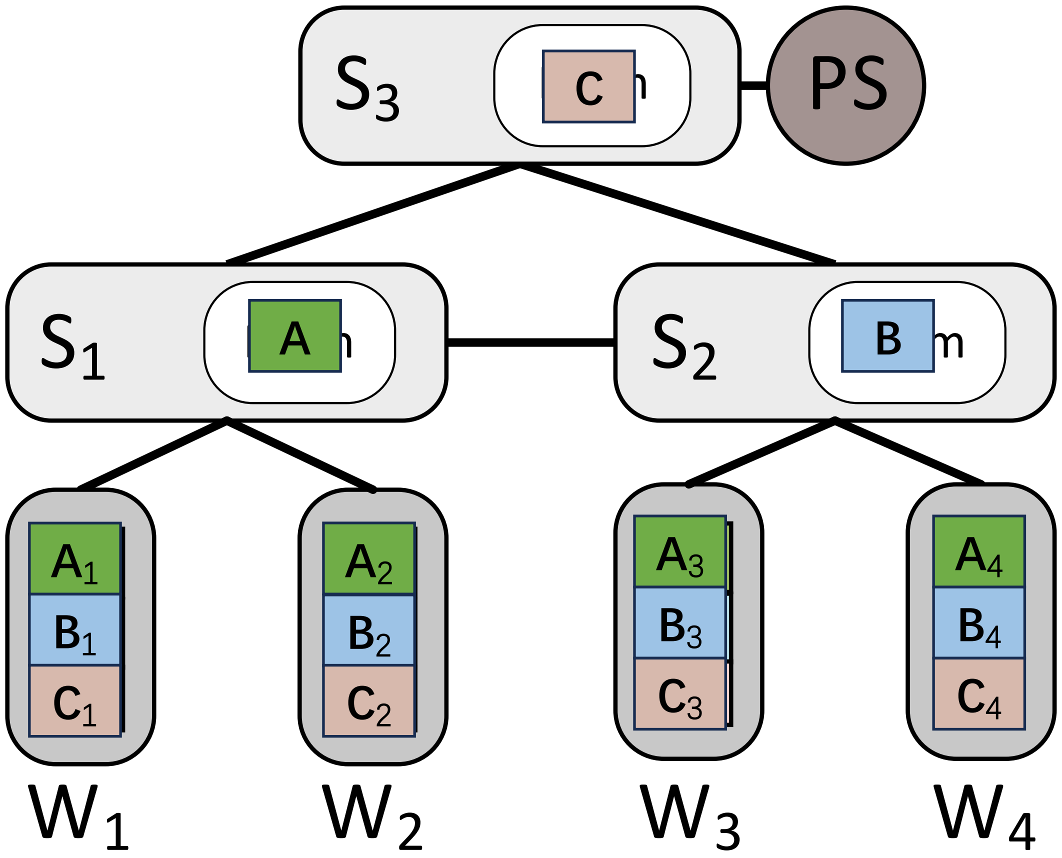
A Motivating Example

➢ The aggregation overhead of the PS is **7**



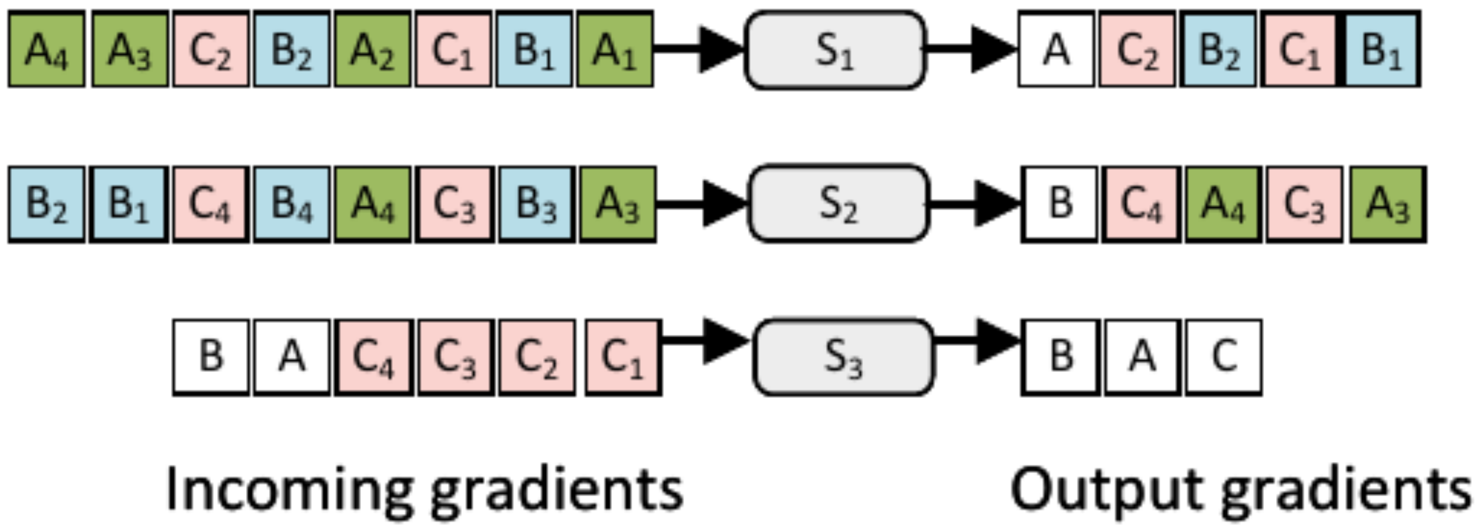
A Motivating Example

➢ Each switch buffers sub-model gradient to **collaborative** perform in-network aggregation



A Motivating Example

- The aggregation overhead of the PS is **3 (optimal)**
- Incur **additional scheduling cost?**



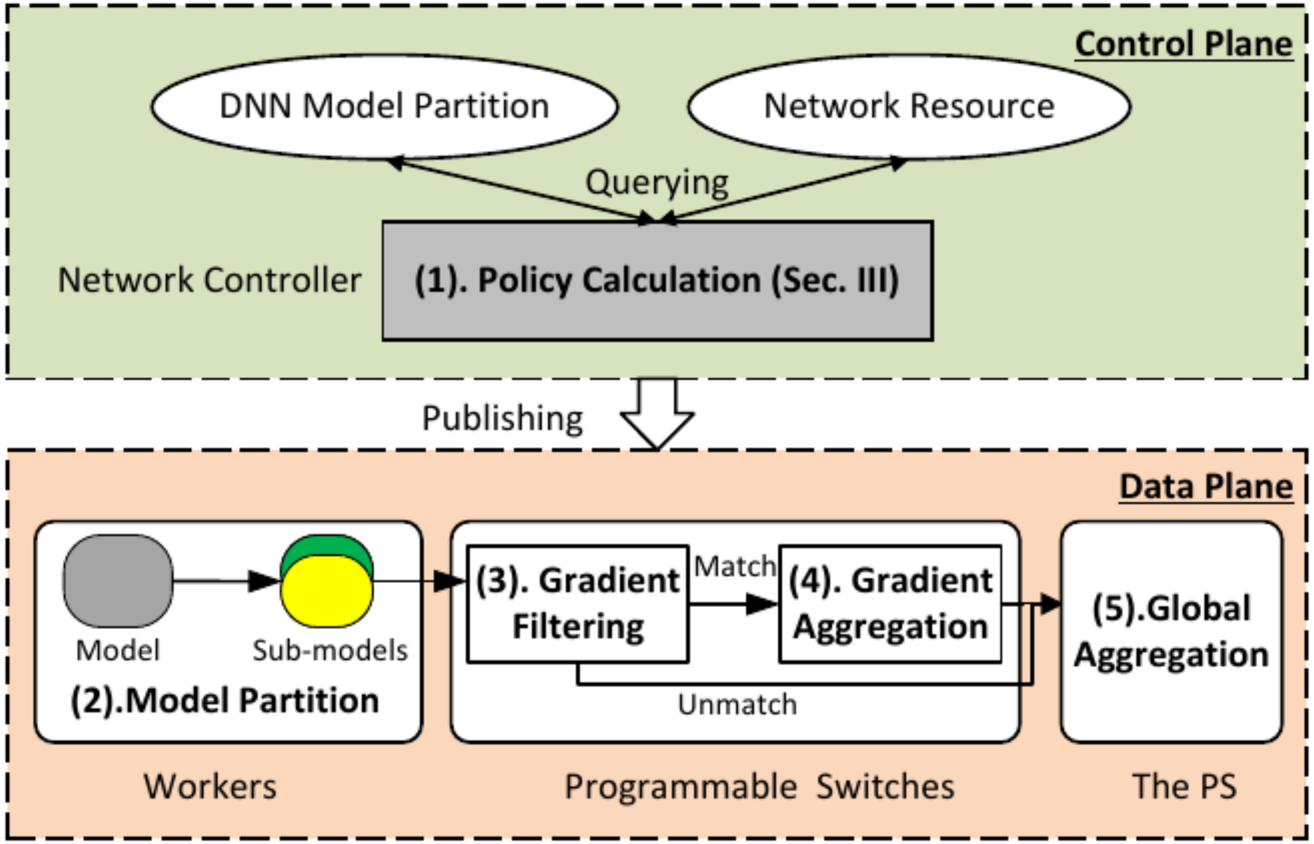
GOAT Overview

Control plane

- **Where** to buffer sub-model gradients?
- In **which** node to aggregate gradients?

Data plane

- Model partition
- Gradient filtering
- Gradient aggregation
- Global aggregation



Problem Formulation

Parameter Server Architecture

- Parameter server: α
- Worker set: $W = \{w_1, w_2, \dots, w_{|W|}\}$

DNN Model Training

- Gradient set of sub-model: $G = \{g_1, g_2, \dots, g_{|G|}\}$

Programmable Network

- Programmable switch set: $S = \{s_1, s_2, \dots, s_{|S|}\}$

Problem Formulation

- Objective: minimize the communication overhead
  - **Non-aggregated gradients** sent from workers to aggregation nodes
  - **Aggregated gradients** sent from switches to the PS

- Sub-model aggregation constraint
- Aggregation node constraint
- Assignment constraint
- Switch memory constraint

$$\min \sum_{g\in G}\Big(\sum_{w\in W}\sum_{s\in S\cup\{\alpha\}} y_{w,g}^{s}\cdot D_w(s)+\sum_{s\in S} x_g^s\cdot D_s(\alpha)\Big)\cdot b(g)$$

$$S.t.\begin{cases}\sum\limits_{s\in S\cup\{\alpha\}} x_g^s \geq 1, & \forall g\in G\\ \sum\limits_{s\in S\cup\{\alpha\}} y_{w,g}^s = 1, & \forall w\in W, g\in G\\ y_{w,g}^s \leq x_g^s, & \forall w\in W, g\in G, s\in S\cup\{\alpha\}\\ \sum\limits_{g\in G} x_g^s\cdot b(g)\leq B(s), & \forall s\in S\\ x_g^s\in\{0,1\}, & \forall g\in G, s\in S\cup\{\alpha\}\\ y_{w,g}^s\in\{0,1\}, & \forall w\in W, g\in G, s\in S\cup\{\alpha\}\end{cases} \tag{1}$$

Algorithm Design

- Convert the problem into an equivalent maximization problem
  - So we only need to consider **the total distance from switches to the PS**

$$\min \sum_{g\in G}\Big(\sum_{w\in W}\sum_{s\in S\cup\{\alpha\}} y^s_{w,g}\cdot D_w(s) + \sum_{s\in S} x^s_g \cdot D_s(\alpha)\Big)\cdot b(g)$$

$$S.t. \begin{cases} \sum\limits_{s\in S\cup\{\alpha\}} x^s_g \geq 1, & \forall g\in G \\ \sum\limits_{s\in S\cup\{\alpha\}} y^s_{w,g} = 1, & \forall w\in W, g\in G \\ y^s_{w,g} \leq x^s_g, & \forall w\in W, g\in G, s\in S\cup\{\alpha\} \\ \sum\limits_{g\in G} x^s_g \cdot b(g) \leq B(s), & \forall s\in S \\ x^s_g \in \{0,1\}, & \forall g\in G, s\in S\cup\{\alpha\} \\ y^s_{w,g} \in \{0,1\}, & \forall w\in W, g\in G, s\in S\cup\{\alpha\} \end{cases} \tag{1}$$

$$\max \sum_{g\in G}\sum_{s\in S}\Big(\sum_{w\in W} y^s_{w,g} - x^s_g\Big)\cdot D_s(\alpha)\cdot b(g)$$

$$S.t. \begin{cases} \sum\limits_{s\in S\cup\{\alpha\}} x^s_g \geq 1, & \forall g\in G \\ \sum\limits_{s\in S\cup\{\alpha\}} y^s_{w,g} = 1, & \forall w\in W, g\in G \\ y^s_{w,g} \leq x^s_g, & \forall w\in W, g\in G, s\in S\cup\{\alpha\} \\ \sum\limits_{g\in G} x^s_g \cdot b(g) \leq B(s), & \forall s\in S \\ x^s_g \in \{0,1\}, & \forall g\in G, s\in S\cup\{\alpha\} \\ y^s_{w,g} \in \{0,1\}, & \forall w\in W, g\in G, s\in S\cup\{\alpha\} \end{cases} \tag{4}$$

Algorithm Design

- Solve the converted problem with a **knapsack-based randomized rounding algorithm**
  - **Relax the converted LP** and obtain the optimal solution
  - **Assign switches** for sub-model gradients with knapsacks
  - **Determine aggregation nodes** for workers' sub-model gradients according to switch assignment

Algorithm 1 KRGS: Knapsack-based Randomized Rounding for Gradient Scheduling

Step 1: Solving the Relaxed Problem
Construct a LP by replacing with $x_g^s, y_{w,g}^s \in [0,1]$.
Obtain the optimal solution $\{\widetilde{x}_g^s, \widetilde{y}_{w,g}^s\}$.
Step 2: Assigning Switches for Sub-Model Gradients
for each sub-model gradient $g \in G$ **do**
  Let $\boxed{k(g) = \left\lceil \sum_{s \in S} \widetilde{x}_g^s \right\rceil}$.
  Put x_g^s $(\forall s \in S)$ into $k(g)$ knapsacks with min-max sum.
  for each knapsack a **do**
    Let $\mathbb{A}$ denote the variables in knapsack a.
    Calculate $S_a = \sum_{\widetilde{x}_g^s \in \mathbb{A}} \widetilde{x}_g^s$.
    Choose s for $\widetilde{x}_g^s \in \mathbb{A}$ with probability $\frac{\widetilde{x}_g^s}{S_a}$.
    Set $\widehat{x}_g^s = 1$ for chosen aggregation node s.
  Let $S(g) = \{s \in S | \widehat{x}_g^s = 1\}$ denote the set of switches responsible for aggregating sub-model gradient g.
Step 3: Determining Aggregation Nodes for Workers' Sub-Model Gradients
for each worker $w \in W$ **do**
  for each gradient $g \in G$ **do**
    Set the probabilities of selecting switch $s \in S(g)$ and the PS to $p_n(s) = \frac{\widetilde{y}_{w,g}^s}{\widetilde{x}_g^s}$ and $p_n(\alpha) = 1 - \sum_{s \in S(g)} p_n(s)$, respectively.
    Select an aggregation node $s \in S \cup \{\alpha\}$ with the probability of $p_n(s)$.

Evaluation

Testbed

- How fast can GOAT **accelerate** the distributed training tasks?
- How much can GOAT **reduce** the aggregation overhead?

Simulation

- Can GOAT handle the **large-scale** distributed task?
- Can GOAT handle the network **dynamic**?

Evaluation: Setup

Topology

- 9 servers
- 3 Wedge100BF-32x programmable switches
- All connected with 100Gbps links

Workload

- 2 DNN models: **ResNet-18(44MB)** and **ResNet-50 (98MB)**
- Dataset: Cifar-100

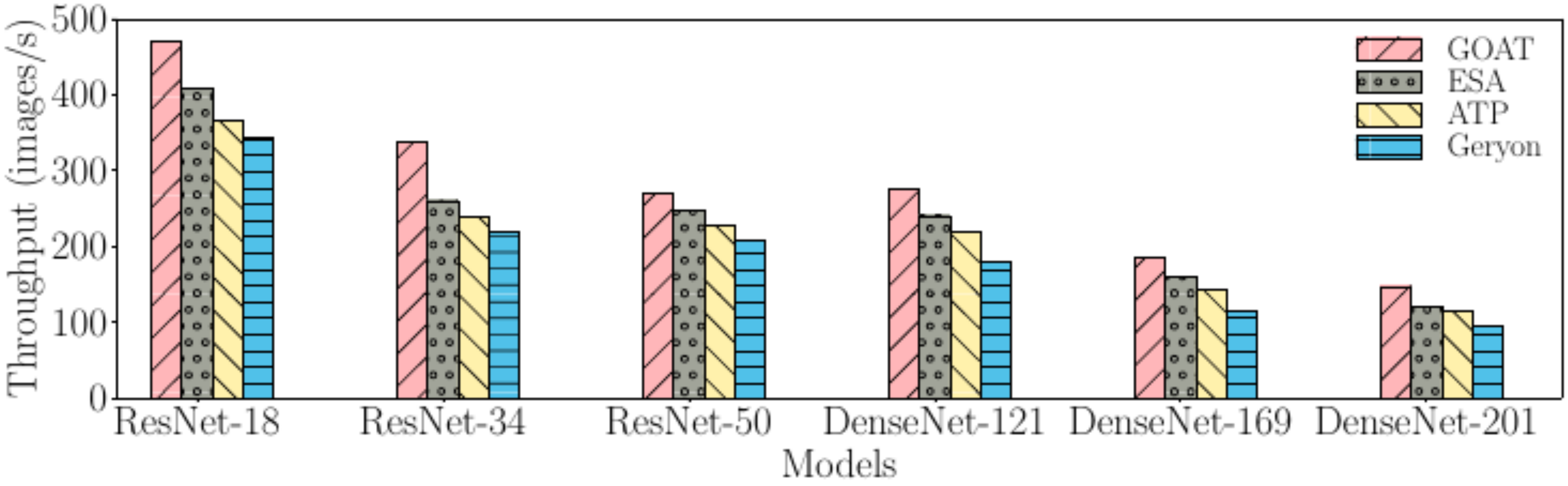
Evaluation: Setup

Benchmark

- **Geryon** (INFOCOM 20): design a communication scheduling scheme **without in-network aggregation**
- **ATP** (NSDI 21): perform in-network aggregation in the first encountered aggregation node **with available memory capacity**
- **ESA**: design a **priority-based memory preemption mechanism** for in-network aggregation

Evaluation: Throughput

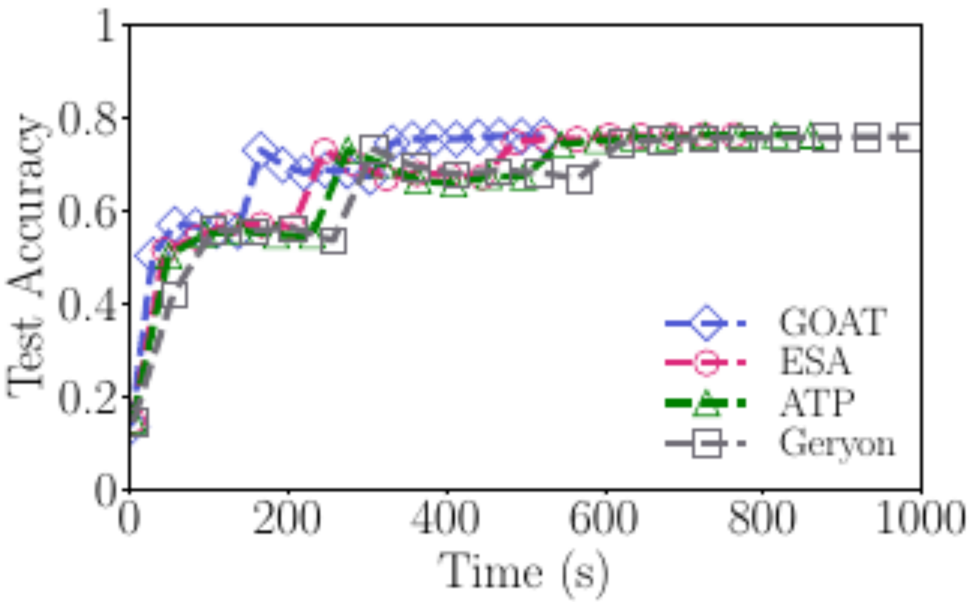
➢ Each distributed training task contains 8 workers



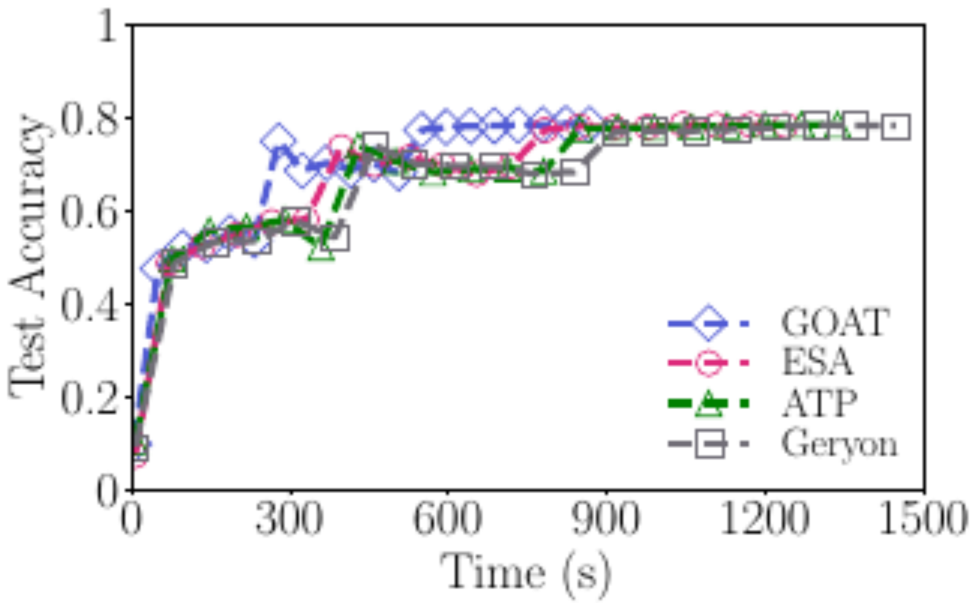
➢ GOAT increases up to **53.3%** in training throughput

Evaluation: Accuracy

➢ Record the test accuracy of training tasks in each epoch



(a) ResNet-18

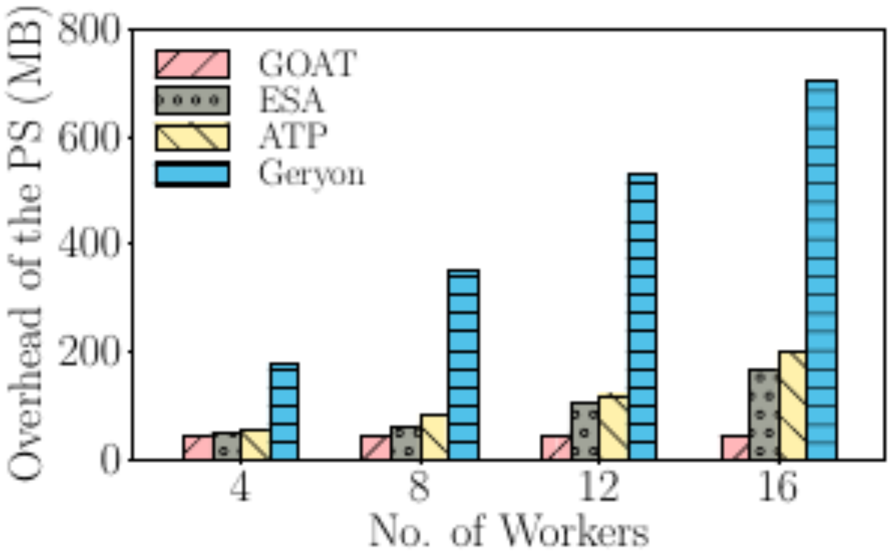


(b) ResNet-50

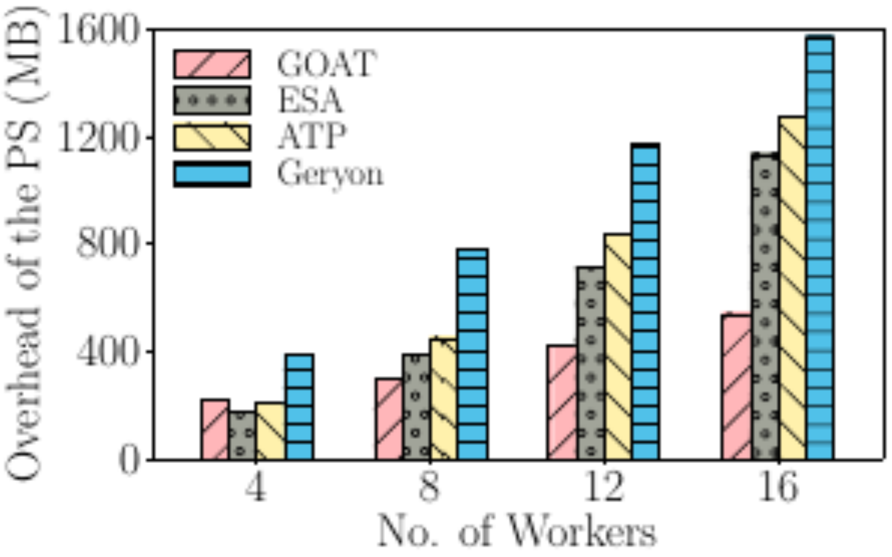
➢ GOAT speeds up distributed training by **1.77x**

Evaluation: Overhead

- Vary the number of workers from 4 to 16



(a) ResNet-18

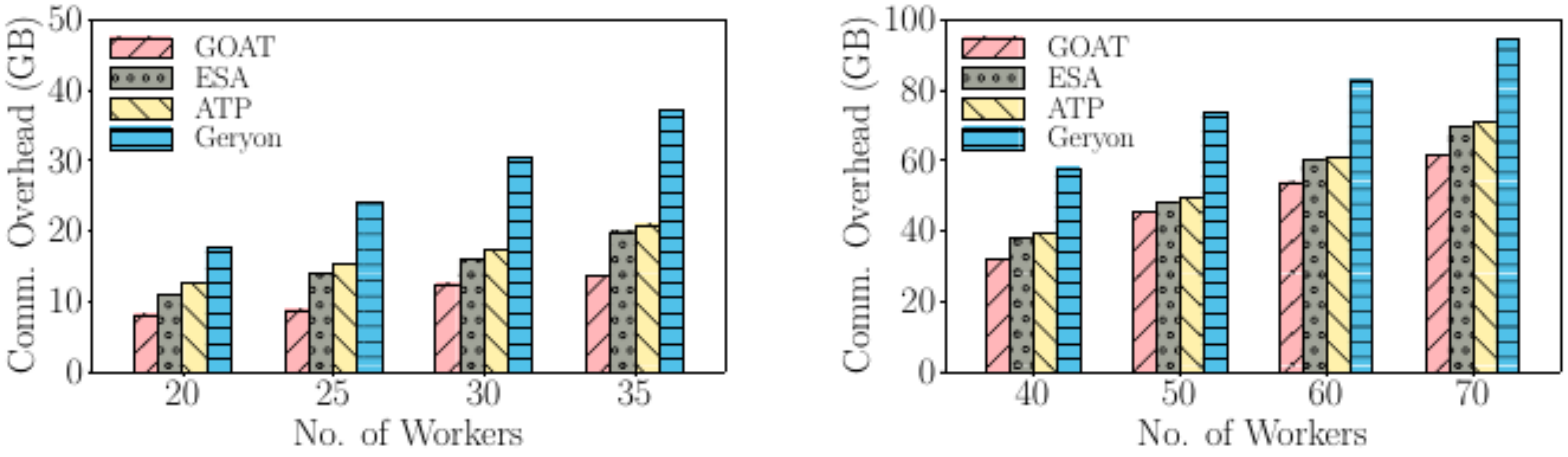


(b) ResNet-50

- GOAT reduces aggregation overhead of the PS by **93.8%**

Evaluation: Scalability

➢ Evaluate GOAT with two practical topologies and more workers



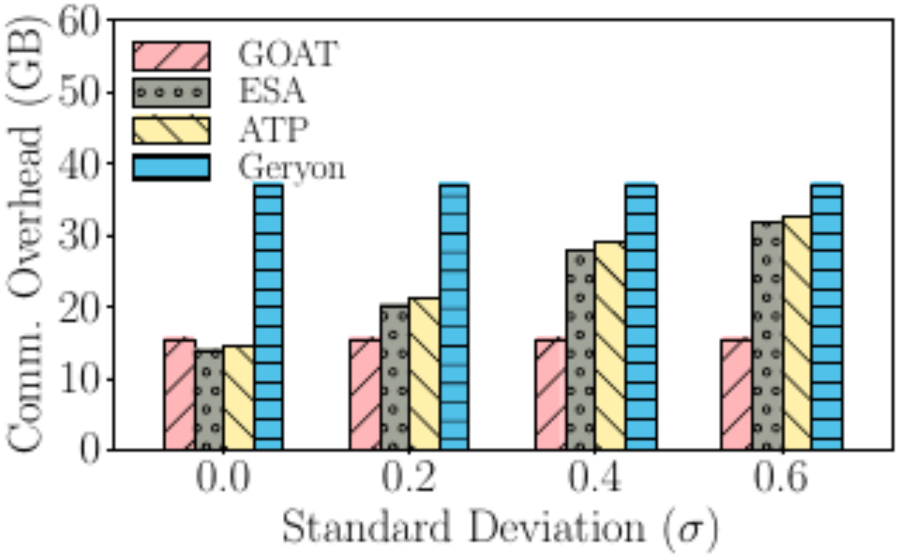
(a) Leaf-Spine

(b) Fat-Tree

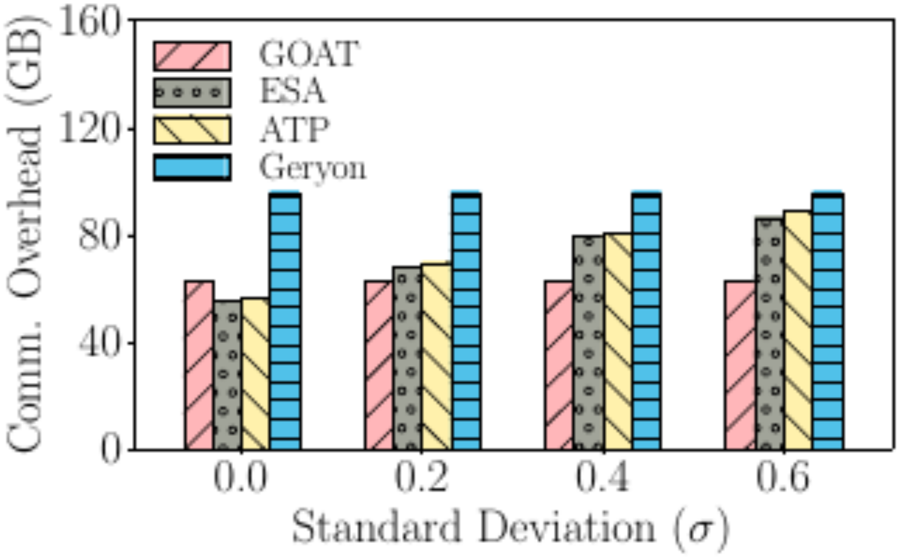
➢ GOAT reduces communication overhead by **63.1%**

Evaluation: Network Dynamic

➢ Vary the sending rate of workers to simulate network dynamic



(a) Leaf-Spine



(b) Fat-Tree

➢ GOAT always achieves the **least** communication overhead

Summary

Goal

- Minimize the communication overhead of distributed training tasks with collaborative in-network aggregation.

Challenges

- Sub-model gradient buffering
- Aggregation node selection
- Switch memory limitation

Solution

- Knapsack-based randomized rounding algorithm with a constant approximation ratio

Thank you!

IEEE/ACM IWQoS 2023

Committee, Reviewers, Volunteers

My Advisors and Collaborators!

Jin Fang
fangjin98@mail.ustc.edu.cn
www.fangjin.site