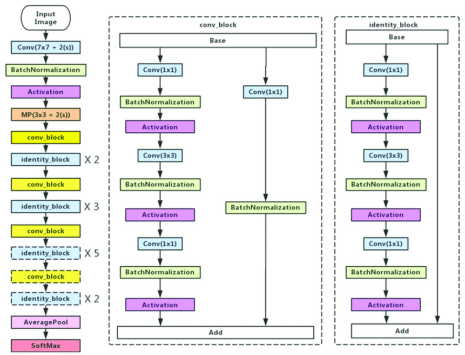
A Unified Architecture for Accelerating Distributed DNN Training in Heterogeneous GPU/CPU Clusters

Yimin Jiang[*†], *Yibo Zhu*[†], *Chang Lan*[‡], *Bairen Yi*[†], *Yong Cui*[*], *Chuanxiong Guo*[†]

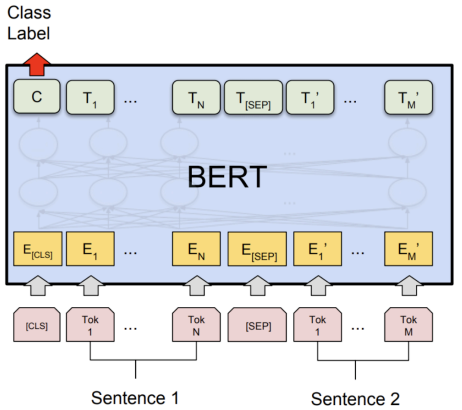
[*]Tsinghua University, [†]ByteDance, [‡]Google

Deep Neural Network (DNN)

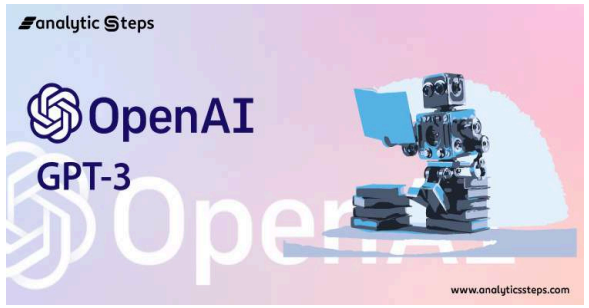
DNN Models



ResNet-50
(26M params)



BERT-Large
(387M params)



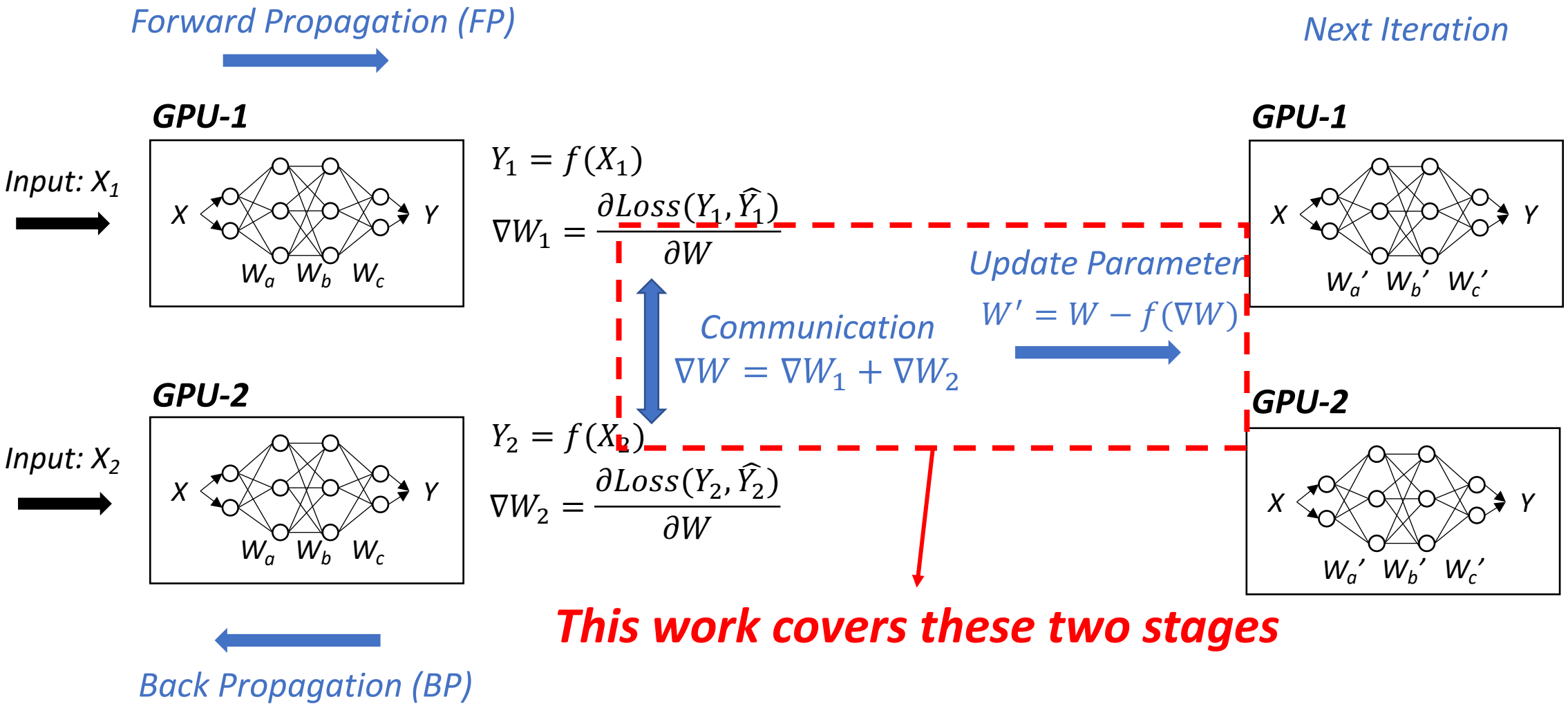
GPT-3
(175B params)

DNN Training

DNN model	Device	Computation time (FP + BP)	Estimated #iters. to converge	Estimated training time
ResNet-50	1x Tesla V100	32ms + 64ms (batch size 32)	3.6M	96 hours
BERT-Large	1x Tesla V100	339ms + 508ms (batch size 35)	8M	78.4 days

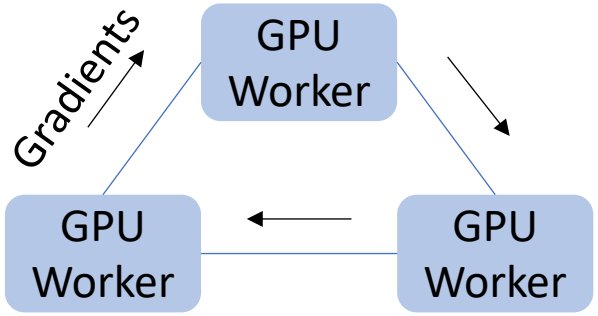
Need distributed training to scale out!

Data-parallel DNN Training



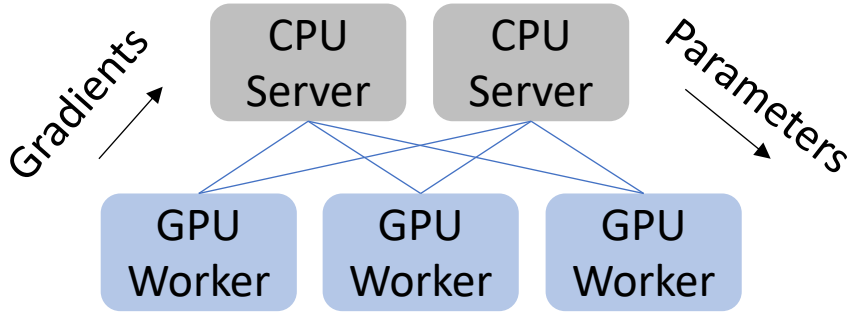
All-reduce and PS

- Architectures based on data parallelism: All-reduce and PS



All-reduce

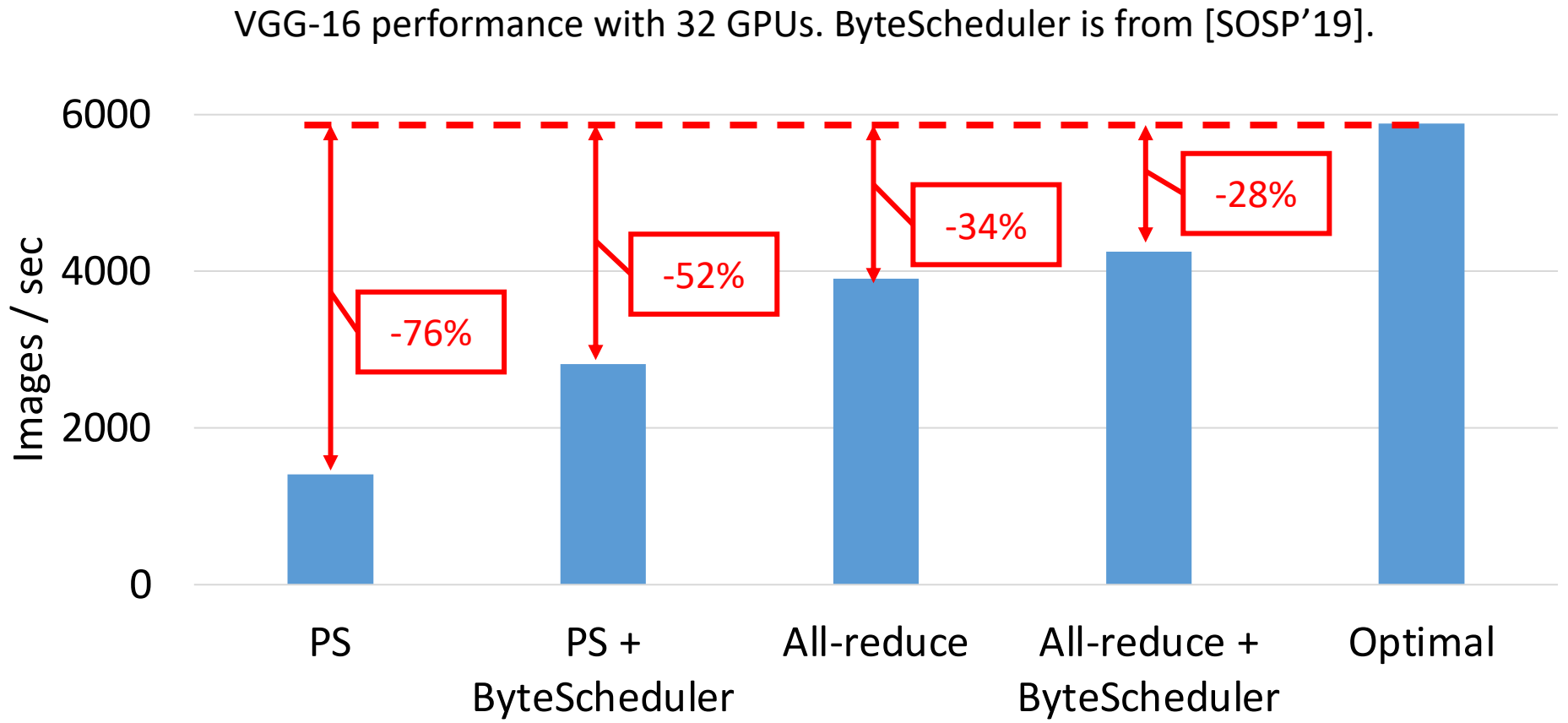
- All workers are homogeneous
- Use collective communication to exchange the gradients



Parameter Server (PS)

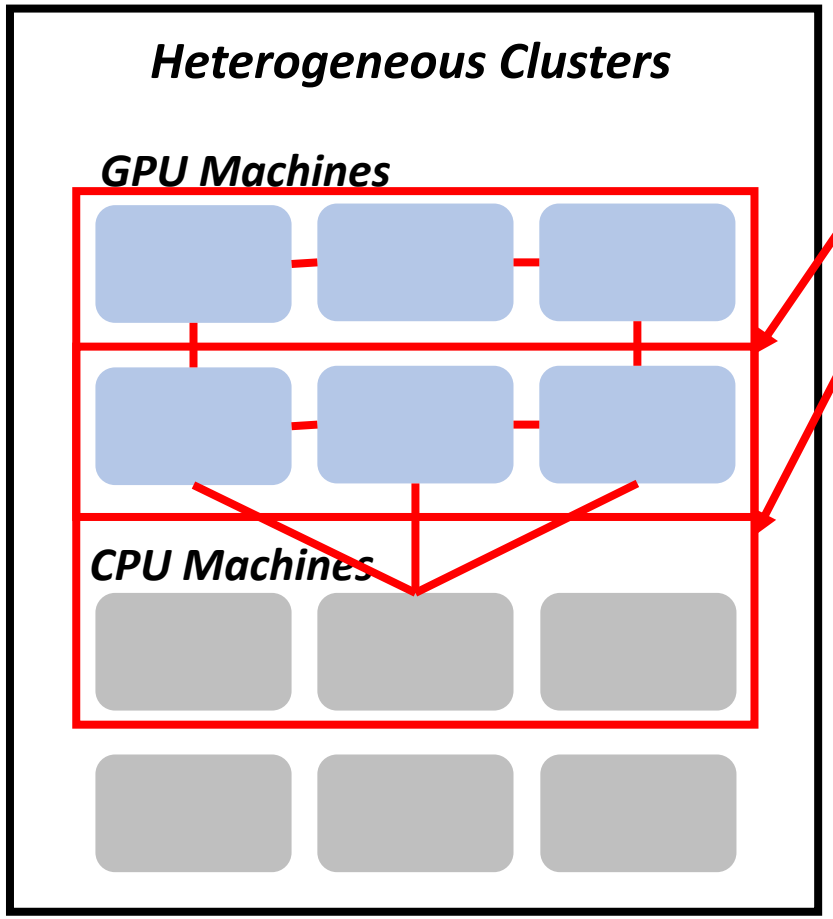
- Heterogeneous bipartite graph
- GPU workers + CPU servers
- Push gradients + pull parameters

Existing Solutions Are Insufficient



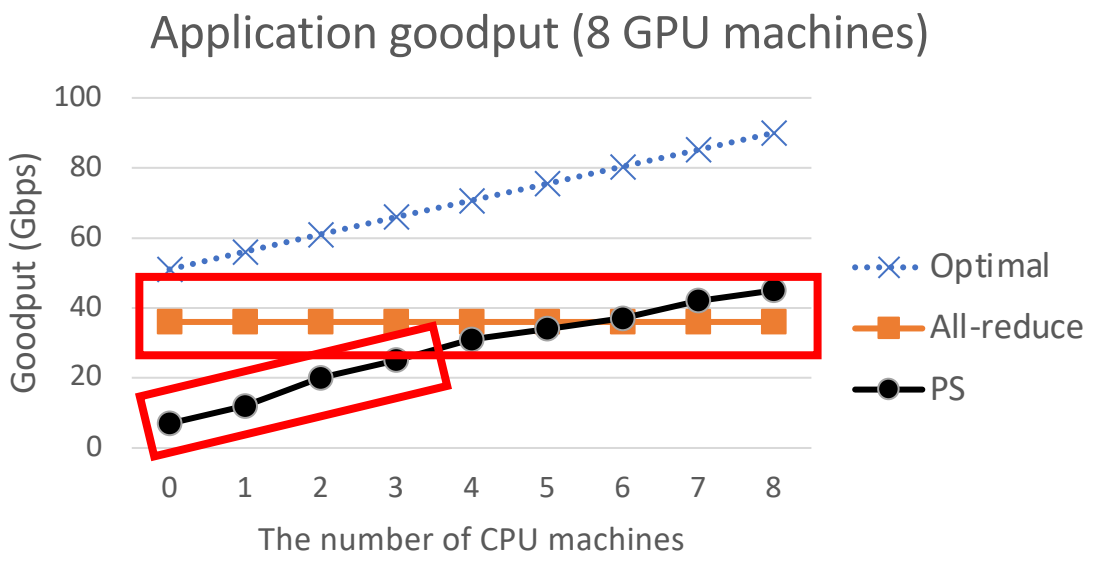
What are the problems of existing solutions?

P1: Sub-optimal Inter-machine Communication



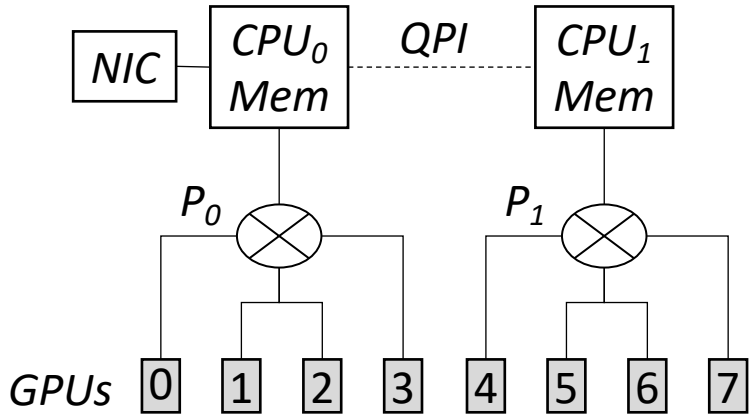
If use all-reduce → cannot leverage CPU machines

If use PS → may create traffic hotspot when CPU not enough



Existing solutions fail to address the characteristics of heterogeneous clusters

P2: Sub-optimal Intra-machine Communication

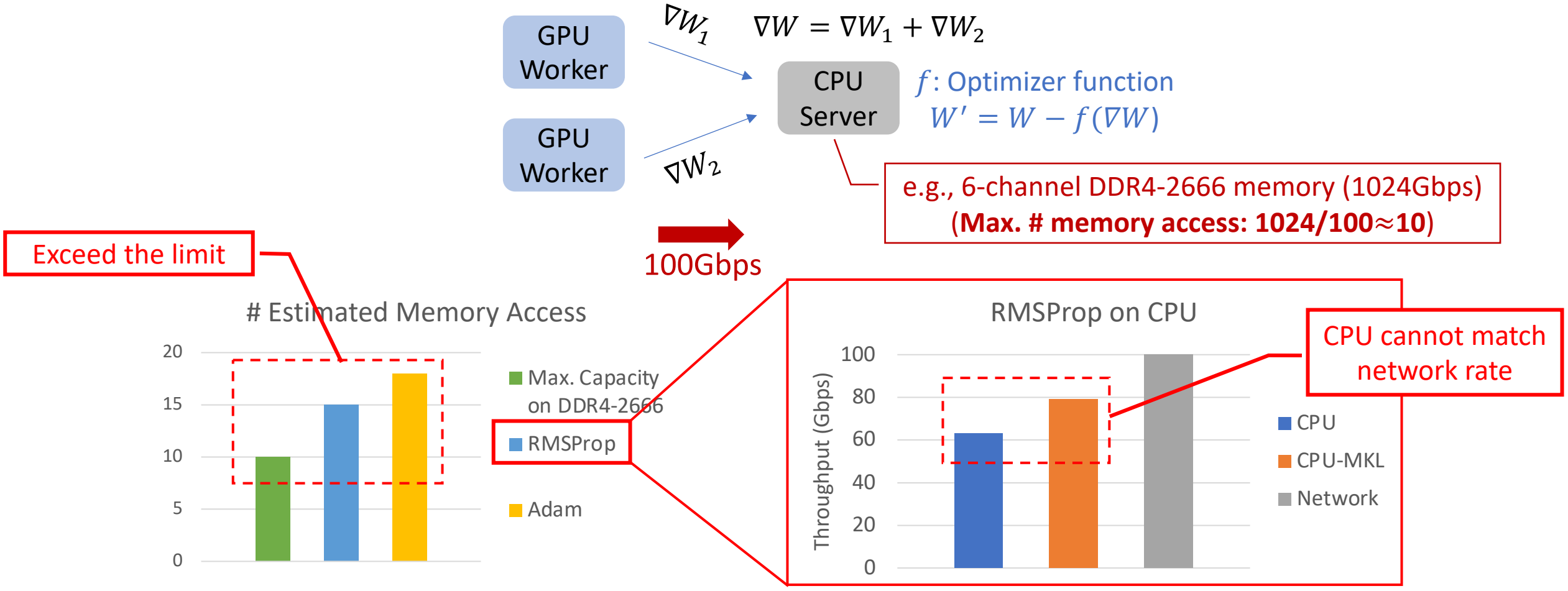


		Bandwidth
NIC	ConnectX-3 EN	10/40 Gbps
	ConnectX-4 EN	25/40/50 Gbps
	ConnectX-5 EN	**100 Gbps**
PCIe 3.0 x16		**128 Gbps**

- The NIC bandwidth is close to PCIe's bandwidth
- Existing solutions cause PCIe contention → NIC not saturated

Need to consider the intra-machine topology carefully

P3: The CPU Bottleneck



How to address the CPU bottleneck?

Our solution: BytePS

Problem 1 Sub-optimal inter-machine communication	→	**An optimal inter-machine communication strategy** that is generic and unifies all-reduce and PS
Problem 2 Sub-optimal intra-machine communication	→	**Intra-machine optimization** that accelerates communication inside GPU machines with diverse topology
Problem 3 The CPU bottleneck	→	**Summation Service** that aggregates gradients on CPUs and moves parameter update to GPUs

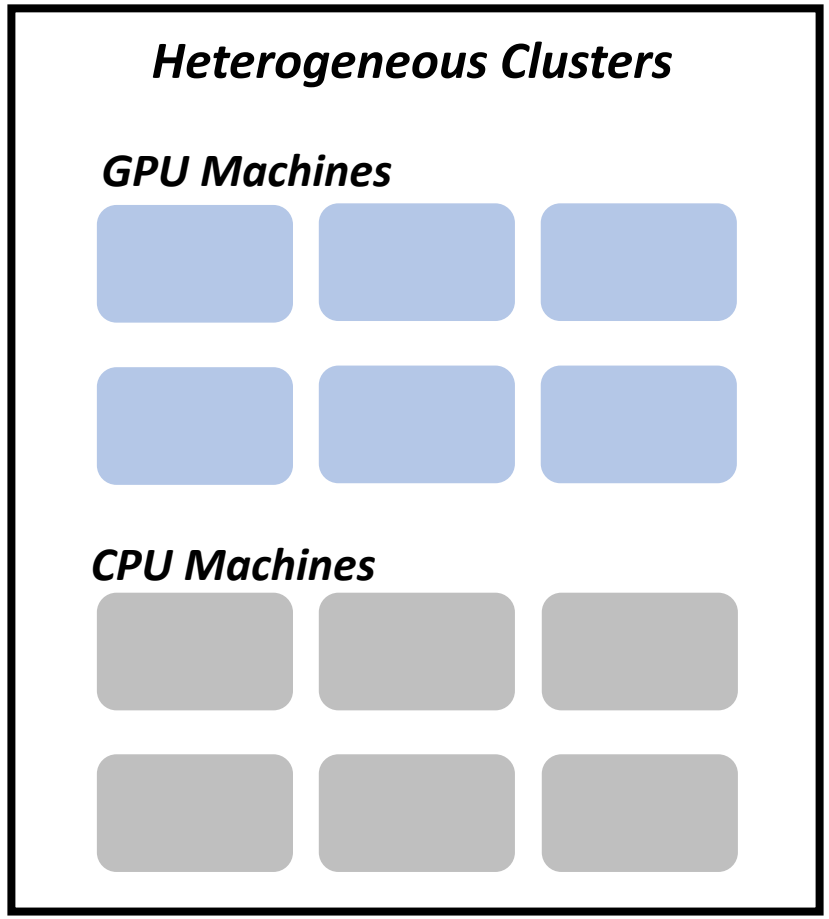
Outline

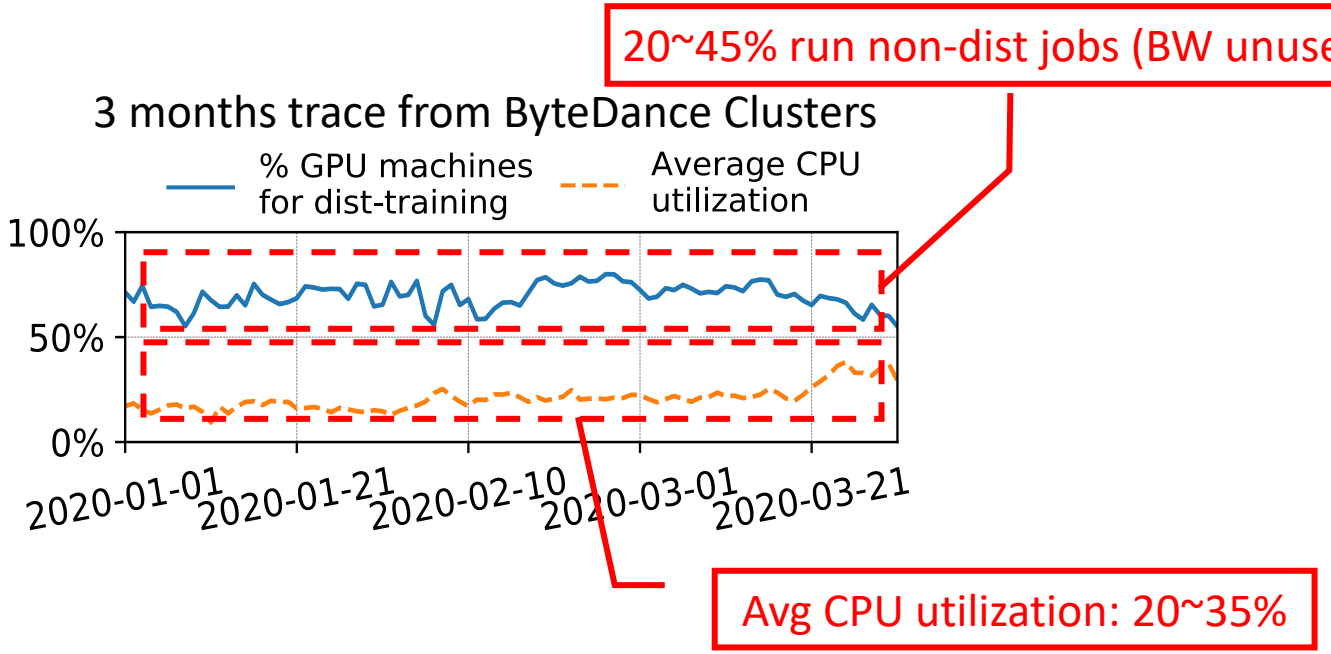
1. Background and Motivation

2. Design and Implementation

3. Evaluation

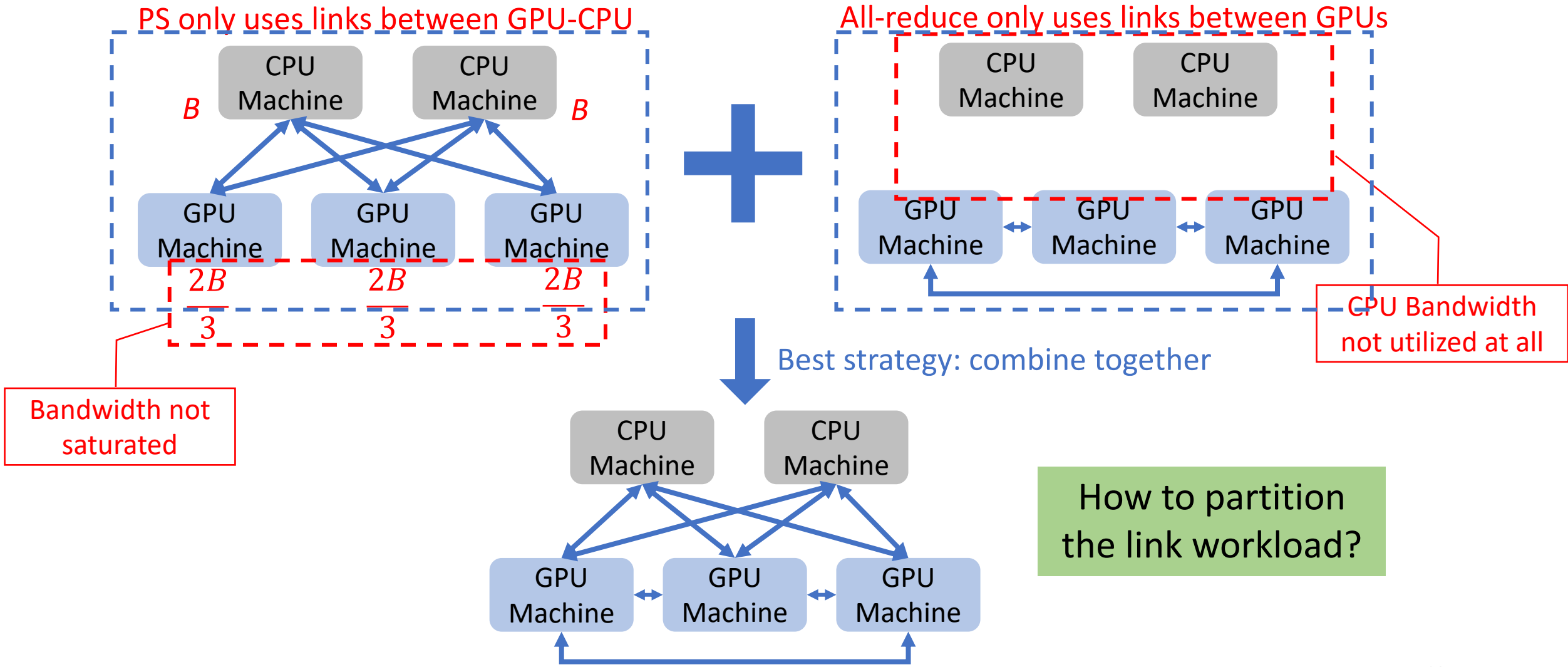
Opportunity and Design Goal





- **Opportunity:** There are spare CPUs and bandwidth in heterogeneous clusters
- **Design goal:** leverage any spare resources

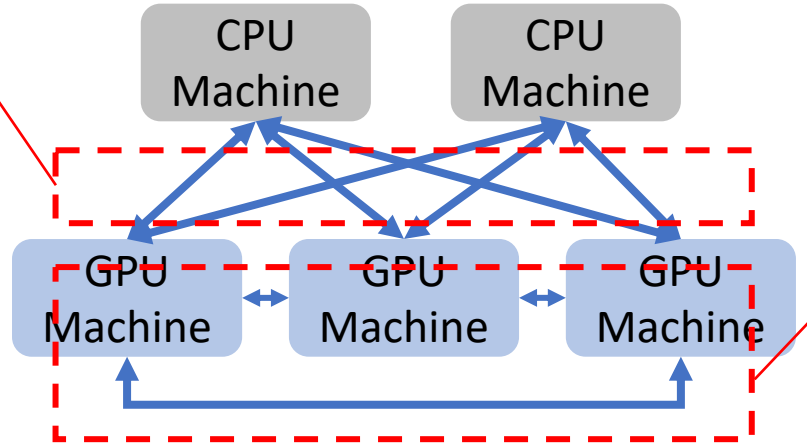
1. Inter-machine Communication



Best Partition Strategy

x: % traffic between GPU-CPU

y: % traffic between GPU-GPU



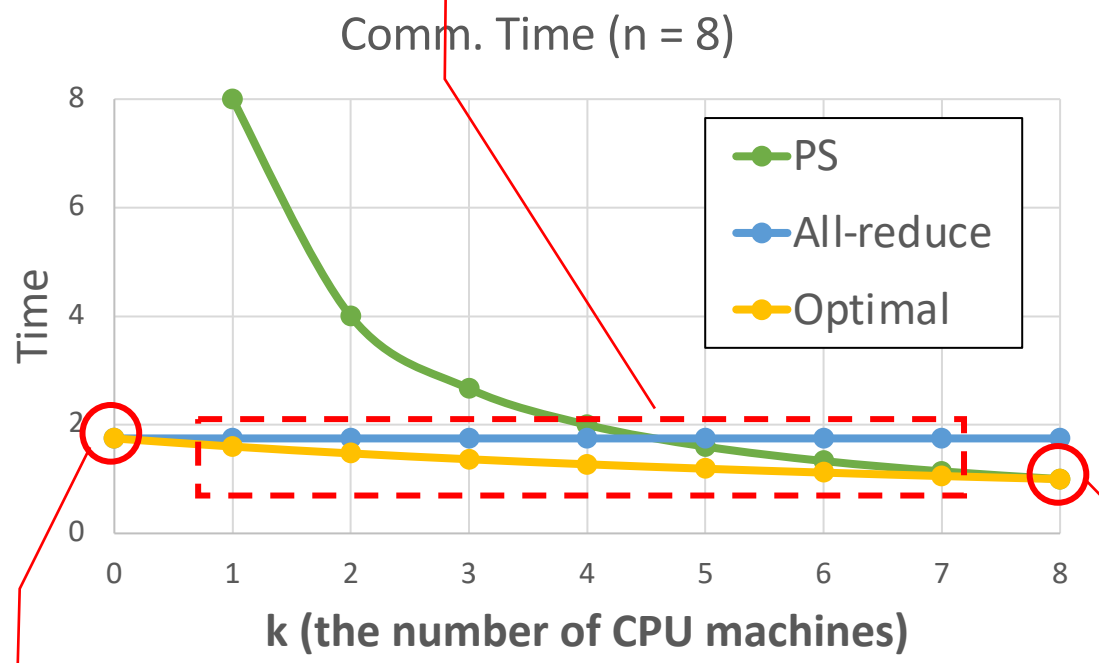
Optimal partition strategy:

$$x = \frac{2k(n-1)}{n^2 + kn - 2k} \qquad y = \frac{n(n-k)}{n^2 + kn - 2k}$$

n: # of GPU machines

k: # of CPU machines

$0<k<n$: better than all-reduce and PS

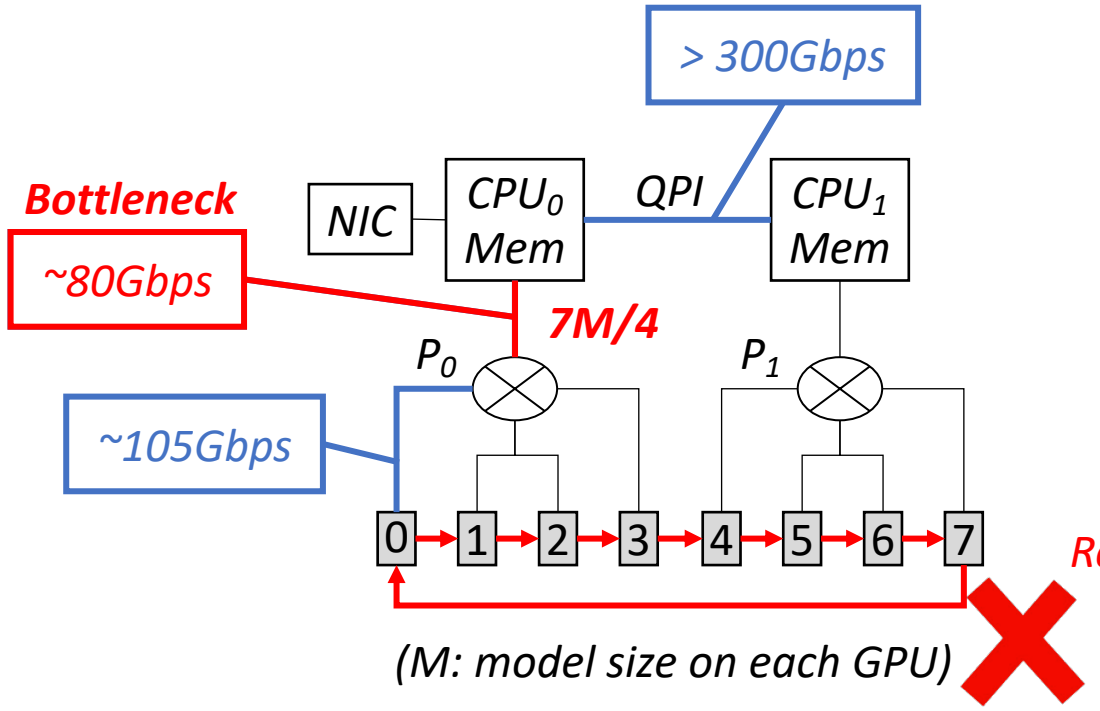


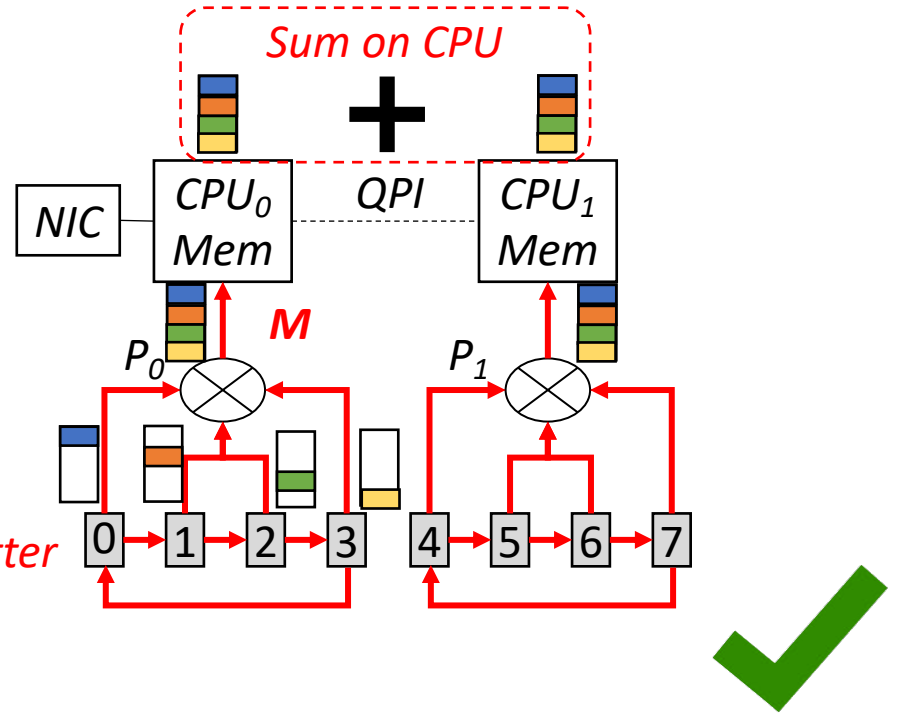
$k = 0$: equal to all-reduce

$k = n$: equal to PS

This strategy unifies PS and all-reduce, and is optimal with any # of CPU machines

2. Intra-machine Communication





Existing solutions (e.g., MPI, NCCL)

Bottleneck link: M*2(8-1)/8 = 7M/4 traffic

Our solution: CPU-assisted Aggregation

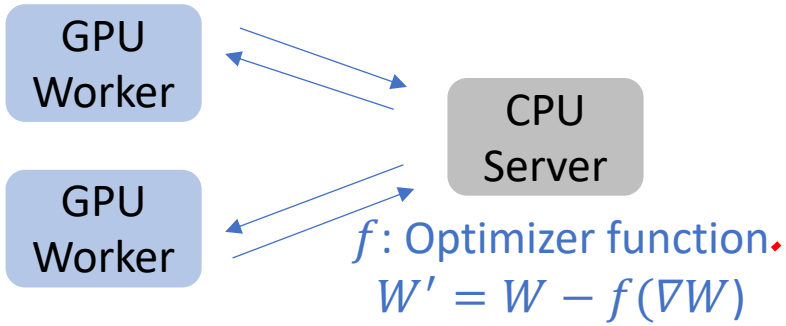
Bottleneck link: M/4*4 = M traffic

- CPU-assisted aggregation **outperforms MPI/NCCL by 24%** in theory
- **Principle**: avoid direct copy between GPUs under different PCIe switches

More Details in the Paper

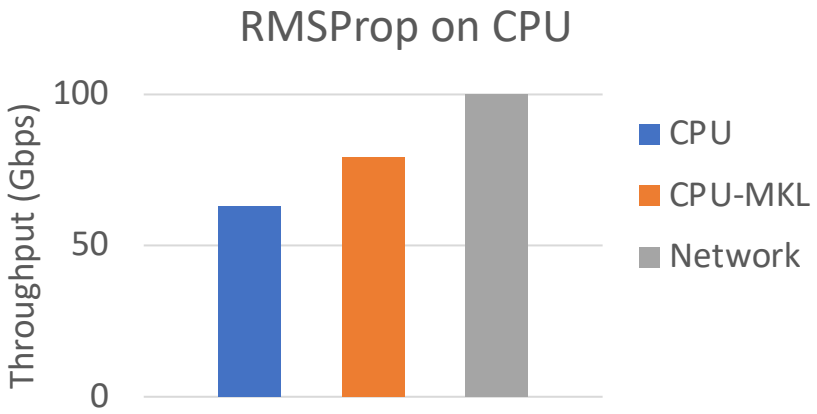
- Solution for NVLink-based machines
- Design principles for different topology
- Optimality analysis
- Discussion about GPU Direct RDMA

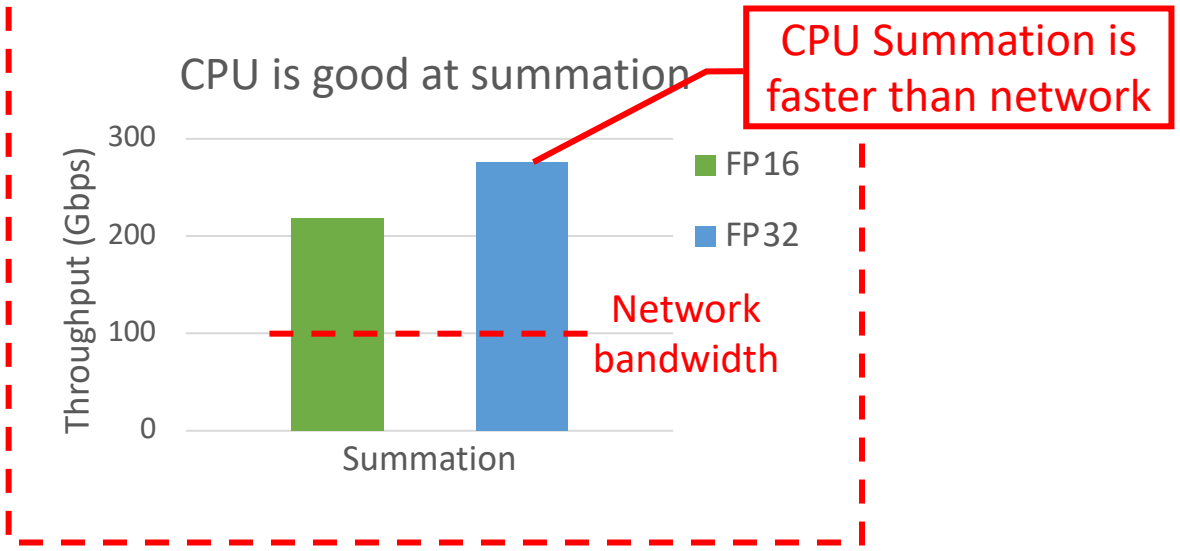
3. Address the CPU Bottleneck



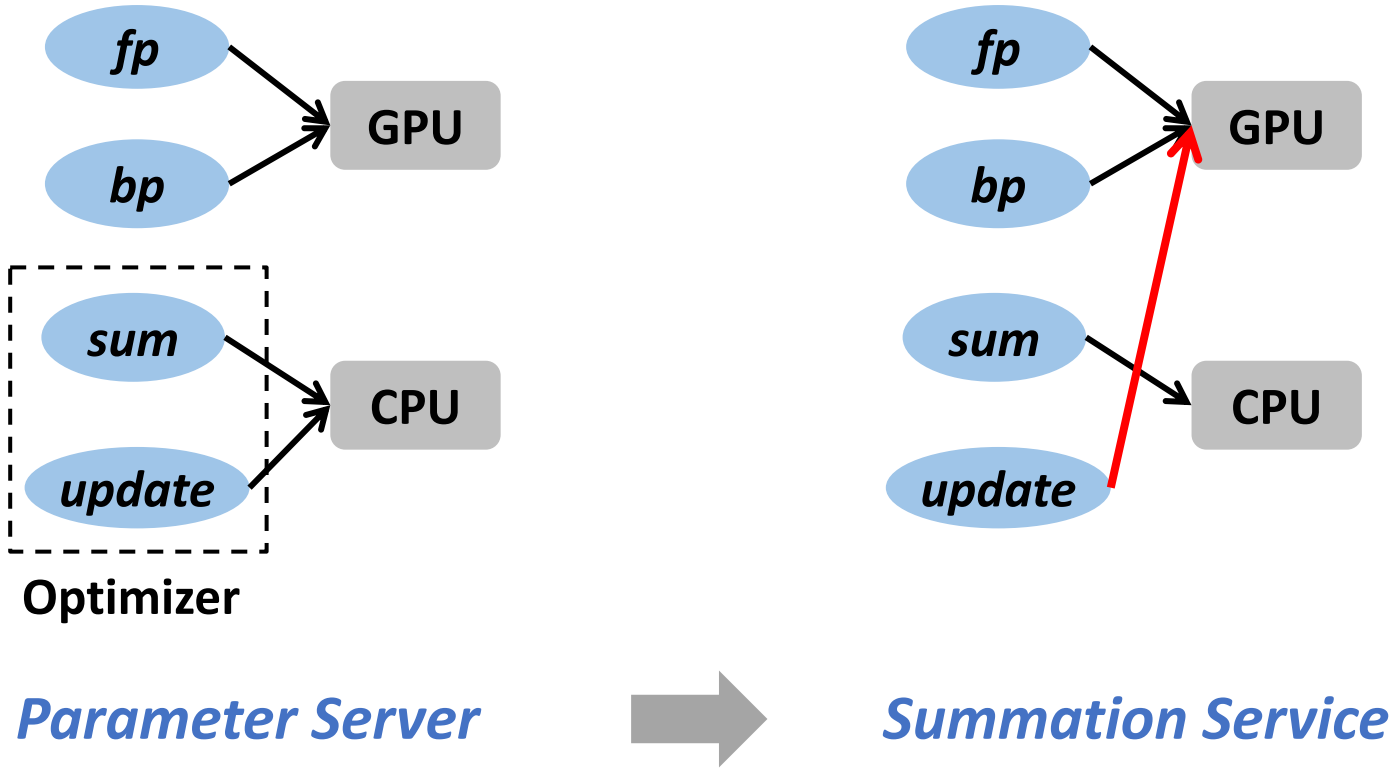
CPU-friendly | Heavy for CPU

Gradient Summation + *Parameter Update*



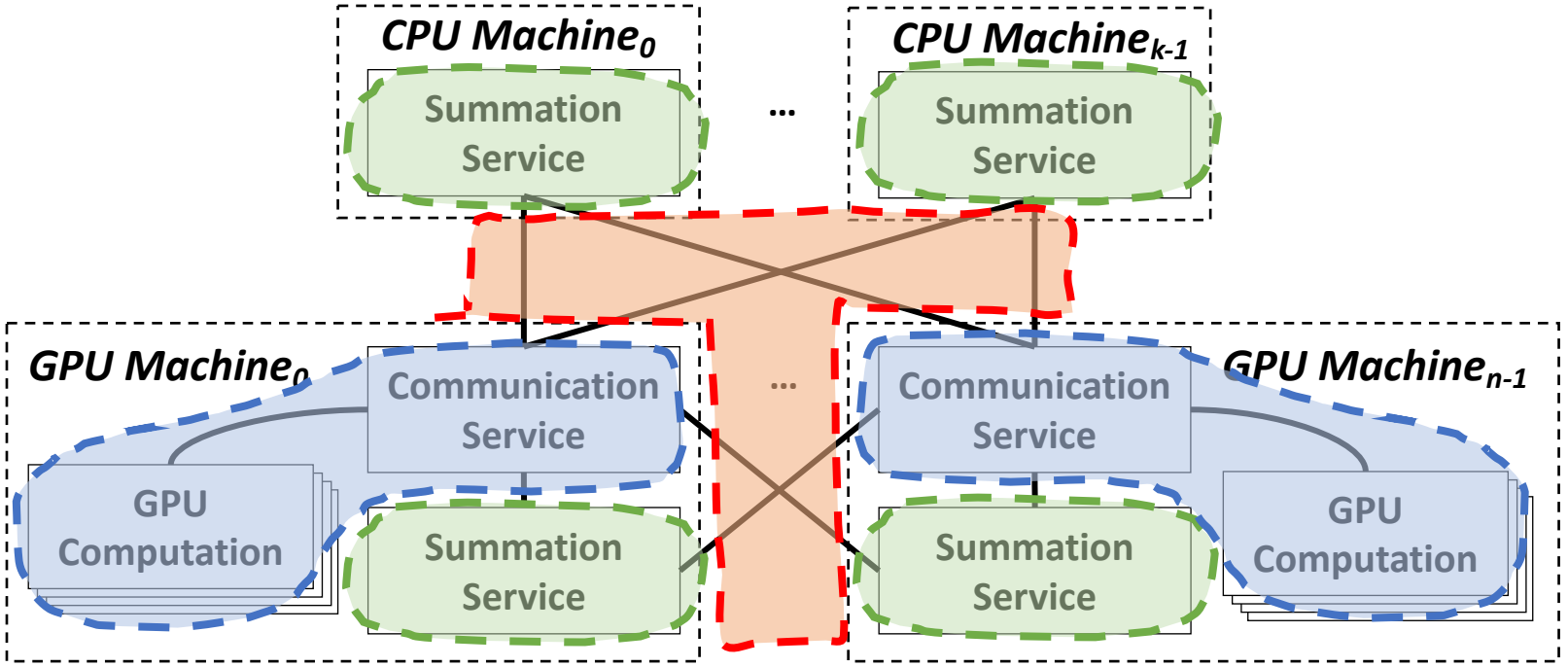


Summation Service



Summation Service can address the CPU bottleneck efficiently

System Architecture



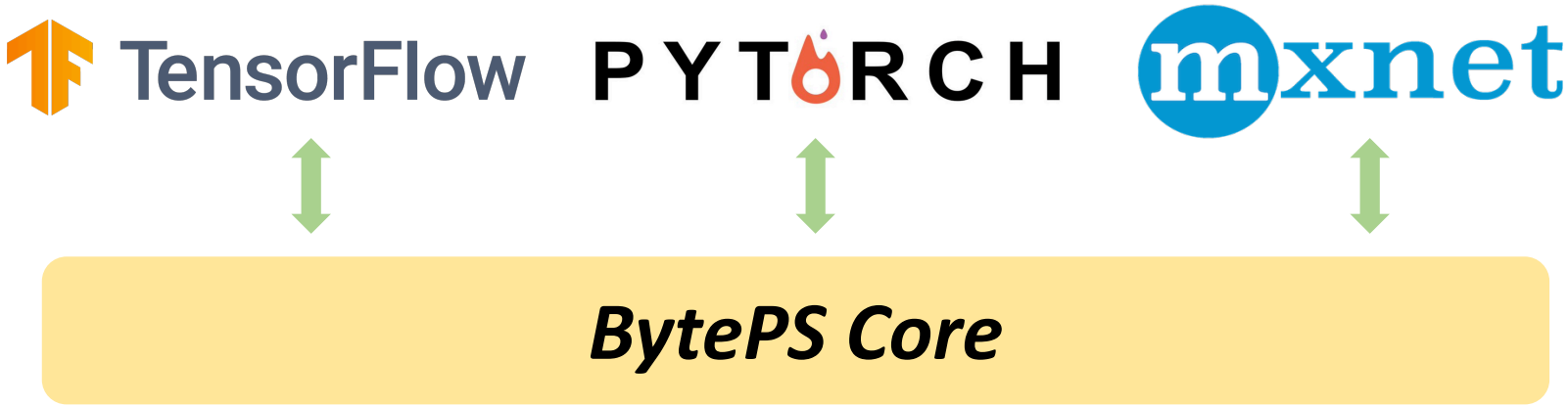
Intra-machine optimization

Optimal Inter-machine strategy

Address the CPU bottleneck

Usage and Deployment

BytePS supports TensorFlow, PyTorch and MXNet



Easy to use: compatible to most widely used APIs

- Horovod-alike APIs
- Native APIs for PyTorch and TensorFlow

BytePS has been deployed in ByteDance for CV/NLP tasks

Outline

1. Background and Motivation

2. Design and Implementation

3. Evaluation

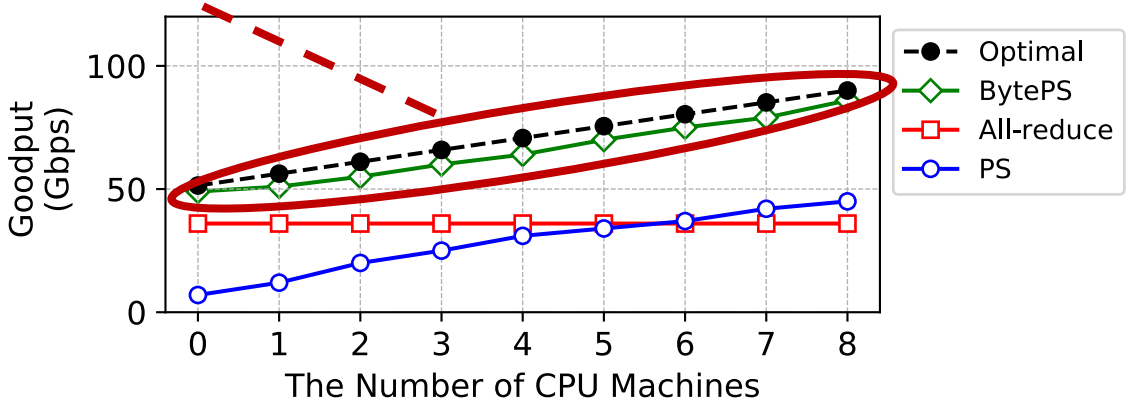
Evaluation Setup

Item	Content
CV models	ResNet-50 (TF), VGG-16 (MXNet), UGATIT-GAN (PyTorch)
NLP models	Transformer (TF), BERT-Large (MXNet), GPT-2 (PyTorch)
Hardware	Each machine with 8x V100 GPUs (32GB) and a 100GbE NIC
Network	RoCEv2, full-bisection bandwidth
Baseline	Horovod, PyTorch DDP, Native-PS of TF and MXNet

- All experiments are performed on production clusters
- All chosen models are representative of production workloads

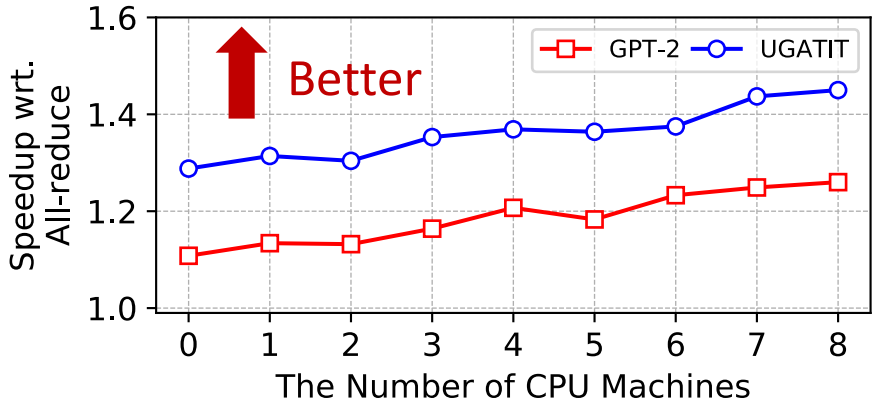
Inter-machine Communication

BytePS achieves near-optimal communication performance



[Traffic microbenchmark] on 8x 1-GPU machines with different # CPU machines

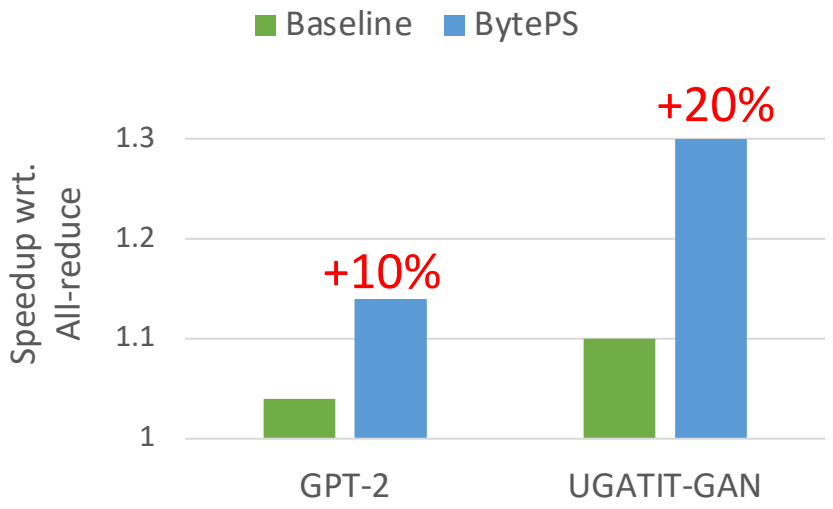
With more CPU machines, BytePS achieves higher E2E performance



[End-to-end] on 8x 8-GPU machines with different # CPU machines. Models: GPT-2 and UGATIT-GAN

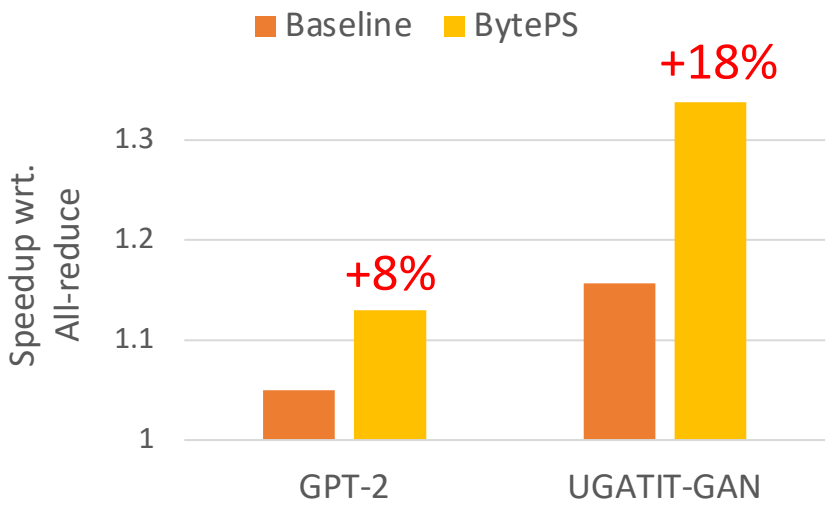
Intra-machine Optimization

For PCIe-only topology, up to 20% gain



[PCIe-only] 8x 8-GPUs

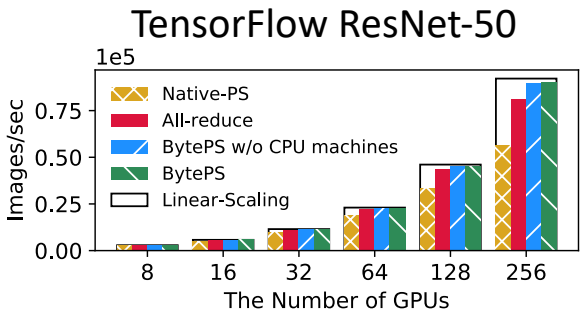
For NVLink-based topology, up to 18% gain

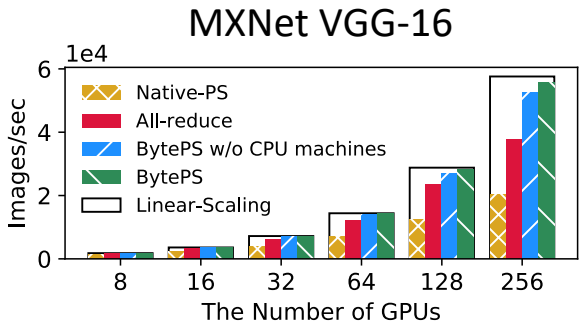


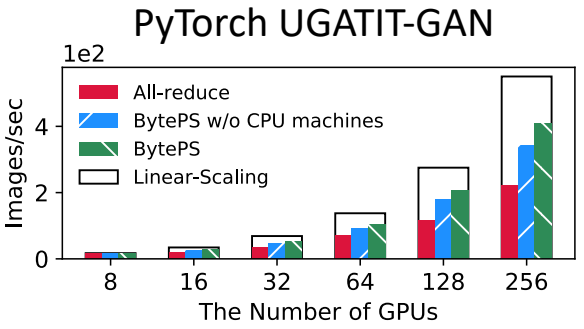
[NVLink-based] 8x 8-GPUs

End-to-end Scalability (up to 256 GPUs)

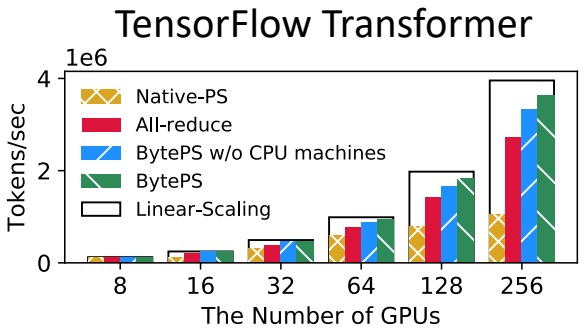
CV Models

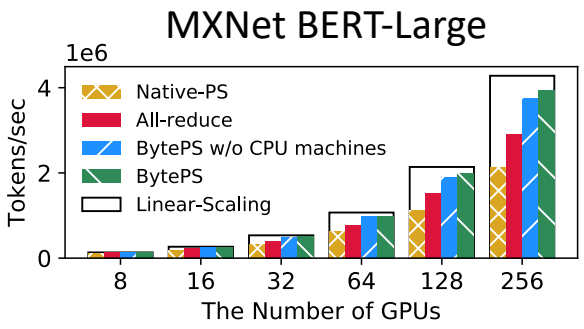


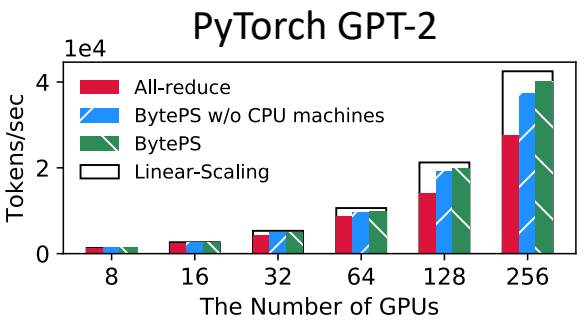




NLP Models

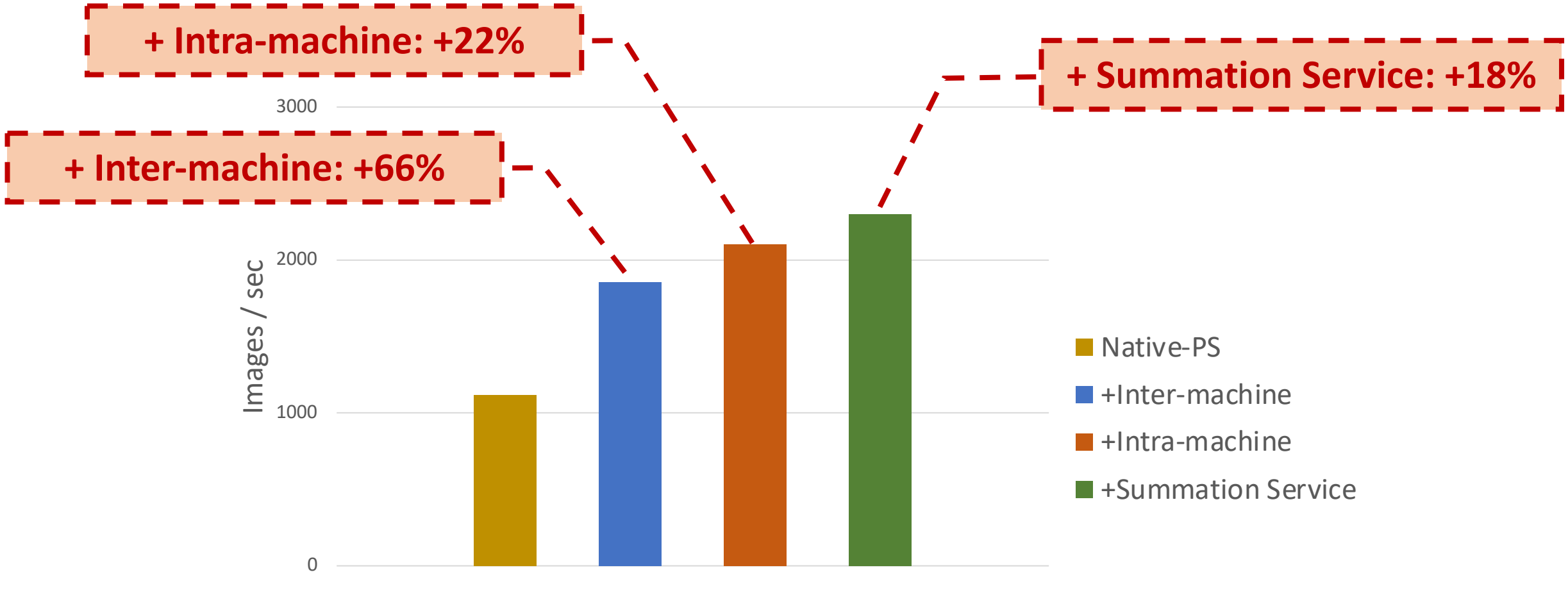






BytePS outperforms All-reduce and PS by up to 84% and 245%, respectively

Breakdown of Performance Gains



VGG-16 (batch size / GPU = 32), 4x 8-GPUs machines, 2x CPU machines

Related Work

- Communication Acceleration
  - Gradient compression: 1-bit, top-k, Adacomp [AAAI'18]
  - Scheduling: ByteScheduler [SOSP'19], P3/TicTac [SysML'19]

  } Complementary to BytePS

  - Pipelined parallelism: PipeDream [SOSP'19] → BytePS can benefit its data-parallel stage
  - Hierarchical all-reduce: BlueConnect [SysML'19] → Still does not leverage heterogeneous resources

- New hardware or architecture for DNN training
  - New AI chips: TPU, Habana → BytePS design is generic and not GPU-specific
  - In-network all-reduce: Infiniband switch ASIC (SHArP)
  - In-network PS: P4 switch [HotNets'17]
  - Rack-scale dedicated servers: PHub [SoCC'18]

  } Require special re-design of hardware or architecture

Conclusion

BytePS: A unified system for distributed DNN training acceleration

- Optimal inter-machine communication
- Topology-aware intra-machine optimizations
- Address the CPU bottleneck with Summation Service

Deployed in ByteDance for CV and NLP training tasks

Open-sourced at https://github.com/bytedance/byteps

- Support TensorFlow, PyTorch, MXNet

Thank you

Yimin Jiang

jymthu@gmail.com