Accommodating LLM Training over Decentralized Computational Resources

Binhang Yuan

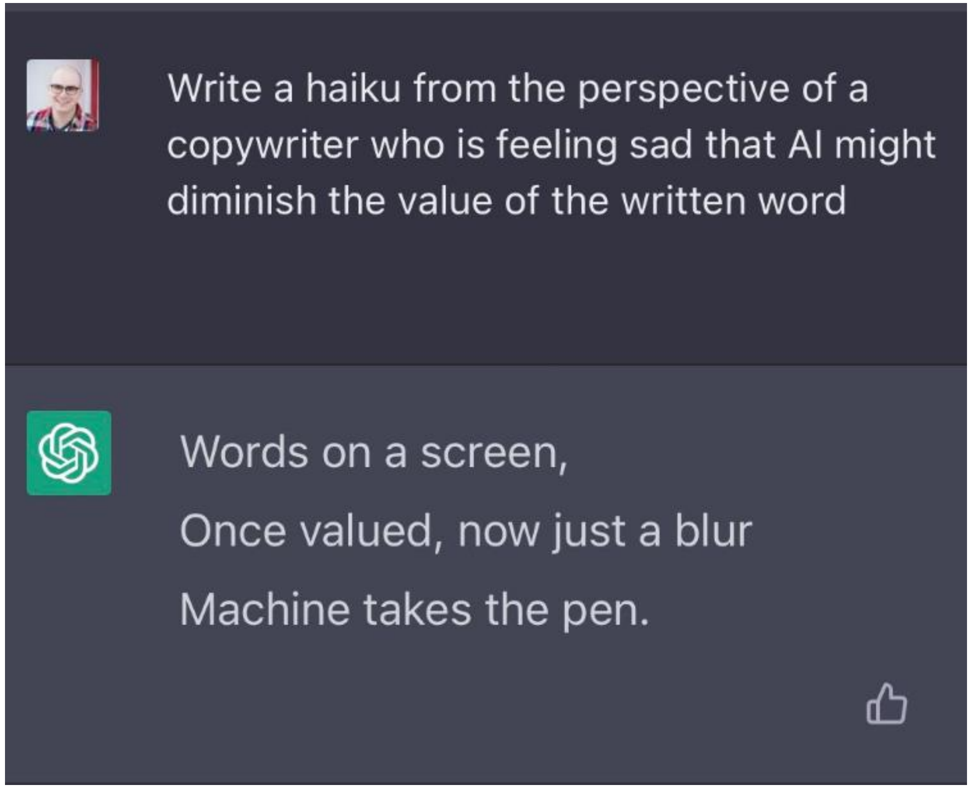
06.11.2024

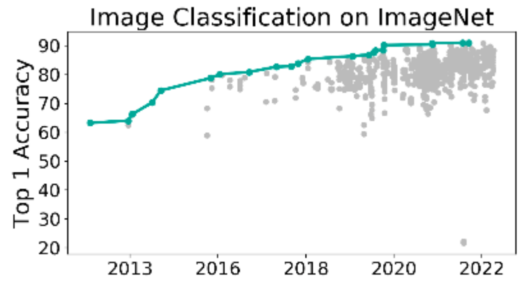
Amazing Progress of ML/AI

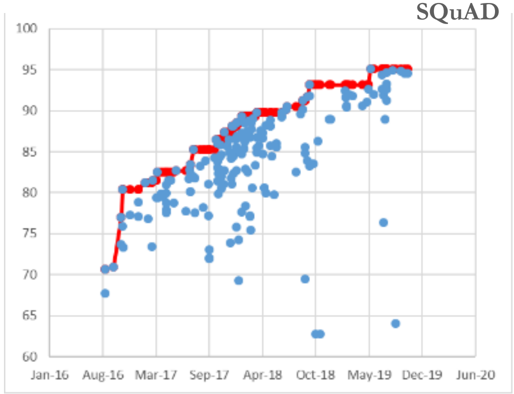
stability.ai runway

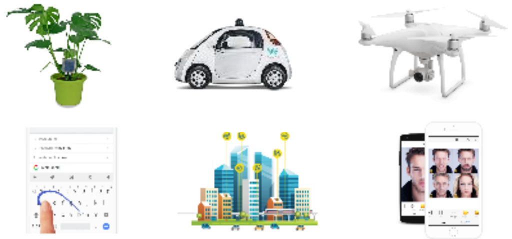
OpenAI

"space robot studying a book in front of Stanford"









The challenge of Today:

(Million $)

Building ML Applications at SOTA scale is expensive!

Further scaling is facing non-linear bottlenecks.

Bottleneck: *Communications & Data Movement*

Distributed training at scale is communication-intensive.

GPT-3

6.7B Parameters

1.20E+22

Floating Point Ops.

32 Machines, 4x A100 GPU each

Each machine send+recv ***4PB*** data

100Gbps = ***93h*** Communication Time

10Gbps = ***930h*** Communication Time

~**200h** Computation Time

GPT-3

175B Parameters

3.14E+23

Floating Point Ops.

196 Machines, 8x A100 GPU each

Each machine send+recv ***12PB*** data

100Gbps = ***279h*** Communication Time

10Gbps = ***2790h*** Communication Time

~**400h** Computation Time

(Future) *10x further scaling requires fast connections between 10x machines. Becoming challenging even for data center.*

NVIDIA DGX SuperPOD:
Up to 256 GPUs

(Today) *Model training today is largely restricted to centralized data centers with fast network connections. Hard to use cheaper alternatives (Non 1st tier clouds, Spot Instances, Volunteer Computes, etc.).*

Optimizing Communications for Distributed and Decentralized Learning.

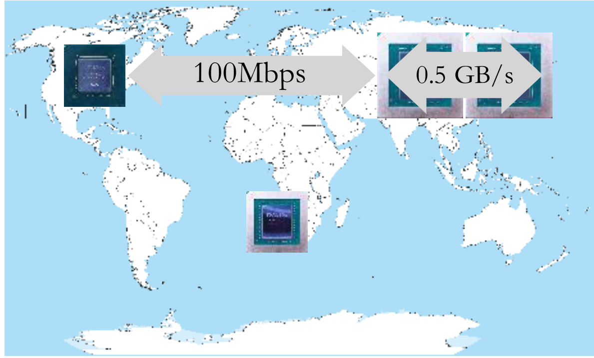
Communication Bottlenecks across Infrastructure

communication becomes slower, open up more choices (and some can be cheaper)

Data Center

(Multi-cloud) Spot Instances

Serverless Environment



Decentralized Network

The more we can optimize communications, the more choices we have when building our infrastructure.

$$\min_x \mathbb{E}_\xi f(\xi, x)$$

$$\min_{x} \mathbb{E}_{\xi} f(\xi, x)$$

Data

- *(ImageNet) 1.3M Images (est. 160+ GB)*
- *(GPT-3) 300 Billion Tokens (est. 2+ TB)*

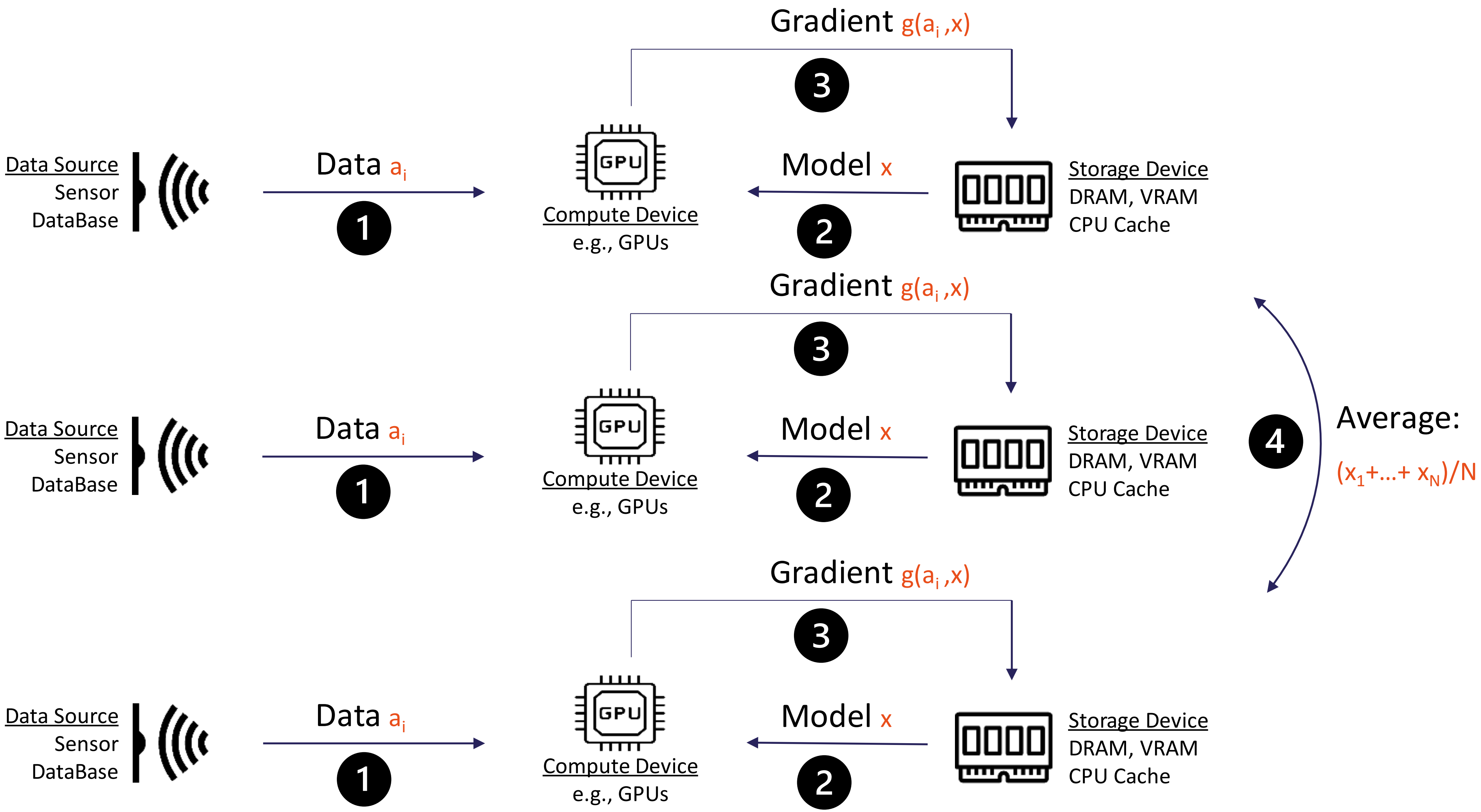
Model

- *(GPT-2) 1.3 Billion Parameters (2.6 GB fp16)*
- *(GPT-3) 175 Billion Parameters (350GB fp16)*

Compute

- *(GPT-2) est. 2.5 GFLOPS/token*
- *(GPT-3) est. 0.4 TFLOPS/token*

Data Parallel SGD



System Optimizations and Relaxed Algorithms

Amazing Systems ⟷ Amazing algorithm

Uber HOROVOD

ByteDance BytePS

NVIDIA Apex

Microsoft DeepSpeed

Decentralized Training

Asynchronous Training

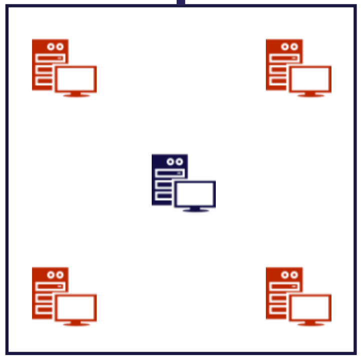
Communication Quantization

Communication Sparsification

Baseline: Centralized, Synchronous, Lossless, SGD

```
x = sync_model()
a = get_data()
g = get_grad(x, a)
x = update(x, g)
```

Synchronous Average



Centralized

Lossless Data Movement

Idea

- Distribute batch gradient calculation to multiple workers;
- Synchronize workers with a central server (or AllReduce).

Mathematical Formulation

$$x_{t+1} = x_t - \gamma \sum_{i=1..n} g_i(x_t; a_i)$$

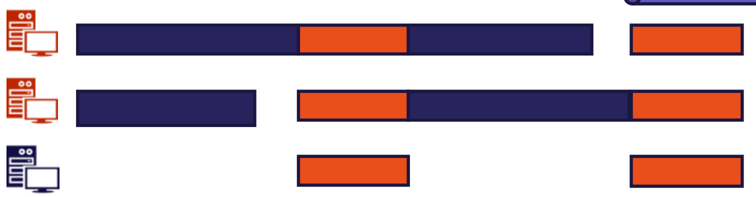
Convergence

$$O(1/\sqrt{nT})$$

Goal 1: Keep This Similar

System Profile

Goal 2: Make this Faster



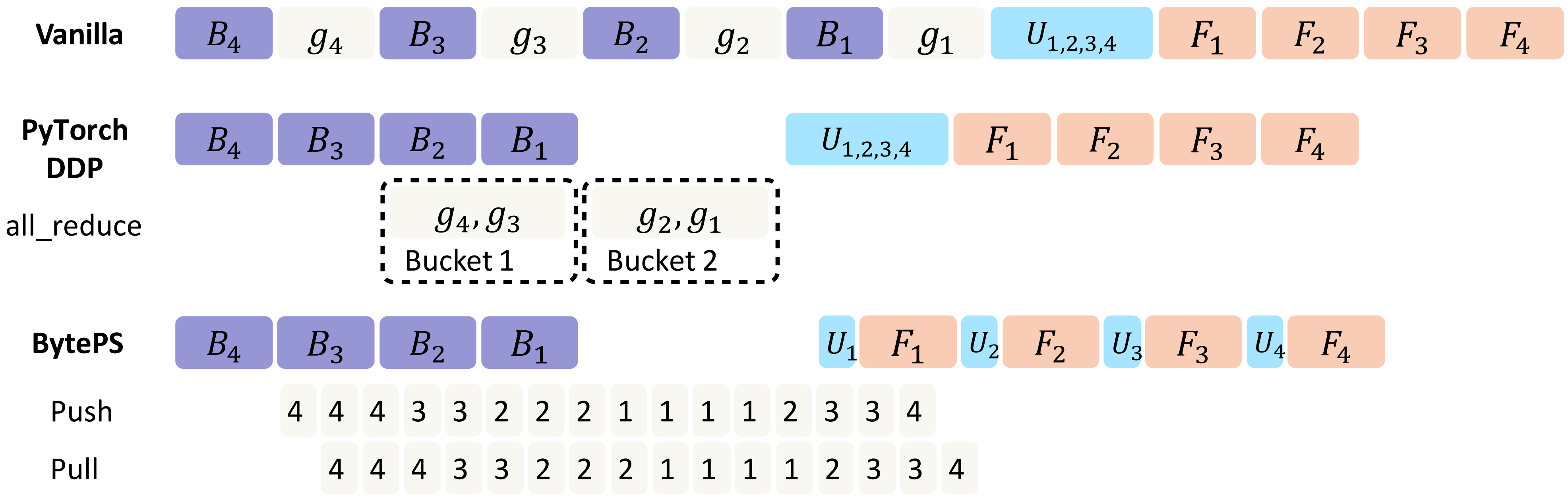
Computation

Communication

System Optimizations

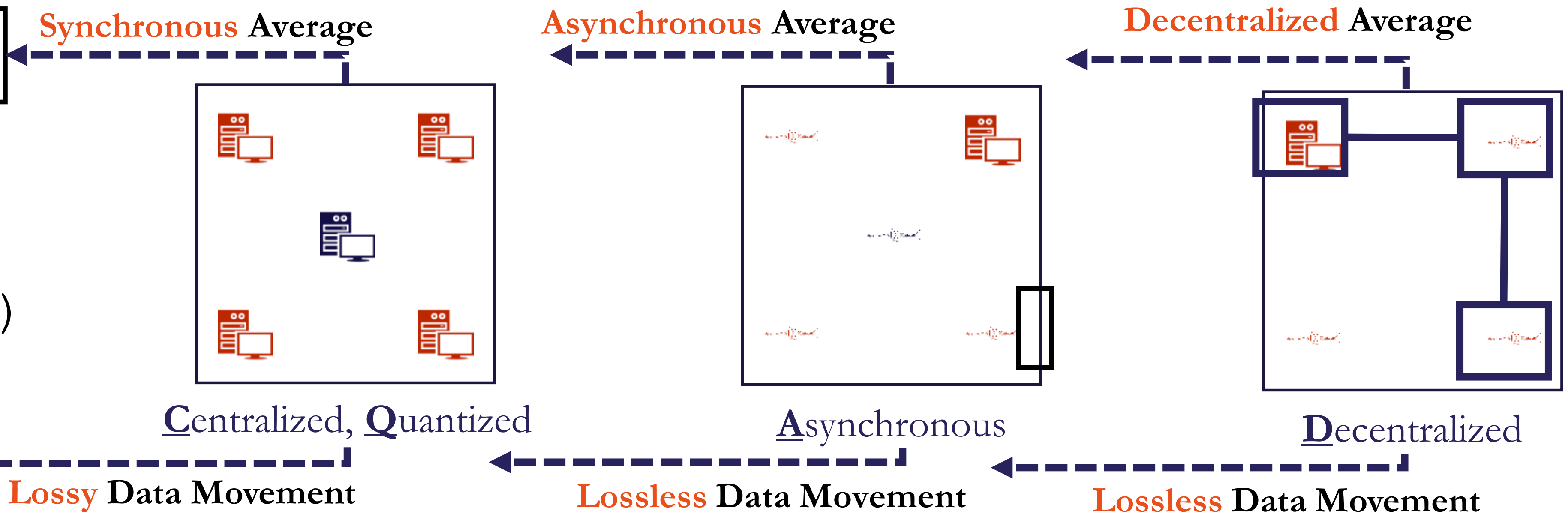
- **Existing Systems:**

Optimize the standard DP-SGD computation:



Relaxed Algorithms

```
x = sync_model()
a = get_data()
g = get_grad(x,a)
x = update(x,g)
```



Mathematical Formulation	$x_{t+1} = x_t - \gamma Q\left(\sum_{i=1..n} Q(g_i(x_t, a_i))\right)$	$x_{t+1} = x_t - \gamma g(x_{t-\tau_t}; a_i)$ staleness caused by async	$x_{t+1,i} = \frac{x_{t,i-1} + x_{t,i} + x_{t,i+1}}{3} - \gamma g(x_{t,i}; a_i)$
Convergence	$O(1/\sqrt{nT} + \boxed{\epsilon/\sqrt{T}})$ Quantization error: ϵ	$O(1/\sqrt{nT} + \boxed{\tau/T})$	$O(1/\sqrt{nT} + \boxed{\rho/T^{1.5}})$ ρ: network topology constant

Attempt 1

Automatic System Optimization for Relaxed Algorithms

Amazing Systems

Uber

HOROVOD

ByteDance

BytePS

NVIDIA

Apex

Microsoft

DeepSpeed

GAP: *Current Amazing Systems Don't Support Recently Developed Amazing Techniques*

OUR GOAL: Distributed Learning with SOTA Communication Optimization Techniques.

Amazing algorithm

Decentralized Training

Asynchronous Training

Communication Quantization

Communication Sparsification

BAGUA: Scaling up Distributed Learning with System Relaxations

Shaoduo Gan*, Jiawei Jiang†, Binhang Yuan, Ce Zhang
ETH Zürich, Switzerland
{firstname.lastname}@inf.ethz.ch

Xiangru Lian*, Rui Wang, Jianbin Chang, Chengjun Liu, Hongmei Shi, Shengzhuo Zhang, Xianghong Li, Tengxu Sun, Sen Yang, Ji Liu
kuaishou technology, China
admin@mail.xrlian.com;ji.liu.uwisc@gmail.com

ABSTRACT

Recent years have witnessed a growing list of systems for distributed data-parallel training. Existing systems largely fit into two paradigms, i.e., parameter server and MPI-style collective operations. On the algorithmic side, researchers have proposed a wide range of techniques to lower the communication via "system relaxations": *quantization*, *decentralization*, and *communication delay*. However, most, if not all, existing systems only rely on standard synchronous and asynchronous stochastic gradient (SG) based optimization, therefore, cannot take advantage of all possible optimizations that the machine learning community has been developing recently. Given this emerging gap between the current landscapes of systems and theory, we build BAGUA, a MPI-style communication library, providing a collection of primitives, that is both flexible and modular to support state-of-the-art system relaxation techniques of distributed training. Powered by this design, BAGUA has a great ability to implement and extend various state-of-the-art distributed learning algorithms. In a production cluster with up to 16 machines (128 GPUs), BAGUA can outperform PyTorch-DDP, Horovod and BytePS in the end-to-end training time by a significant margin (up to 2×) across a diverse range of tasks. Moreover, we conduct a rigorous tradeoff exploration showing that different algorithms and system relaxations achieve the best performance over different network conditions.

PVLDB Reference Format:
Shaoduo Gan, Xiangru Lian, Rui Wang, Jianbin Chang, Chengjun Liu, Hongmei Shi, Shengzhuo Zhang, Xianghong Li, Tengxu Sun, Jiawei Jiang, Binhang Yuan, Sen Yang, Ji Liu, Ce Zhang. BAGUA: Scaling up Distributed Learning with System Relaxations. PVLDB, 15(4): 804 - 813, 2022.
doi:10.14778/3503585.3503590

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/BaguaSys/bagua.

1 INTRODUCTION

The increasing performance of distributed machine learning systems has been one of the main driving forces behind the rapid advancement of machine learning techniques. From AlexNet [35] in 2012 to GPT-3 [11] in 2020, each leap in model quality is enabled by the growth of both the model size and the amount of data one can train a model with, along with a rapid increase in computations [47]. Behind this improvement are two major enabling factors: hardware accelerations (e.g., GPUs and TPUs) and the development of efficient and scalable distributed training algorithms [4, 7, 72, 73, 76]. It is not unfair to say that a scalable distributed training system is the cornerstone of modern deep learning techniques.

In this paper, we scope ourselves and focus on *data parallel training*, one of the most popular distributed training paradigms in which the data set is partitioned across different workers and the model fits into a single device. Not surprisingly, recently years have witnessed a growing list of systems for distributed data parallel training. Existing systems fit into two paradigms, following the seminal work done by Li et al. [38] on *parameter server* and Sergeev et al. [56] on using MPI collective operations such as *Allreduce*. Both paradigms have enabled industrial-scale distributed training systems [47]: Adam (Microsoft) [13], early TensorFlow (Google) [3], Poseidon (Petuum) [77], Angel (Tencent) [32], and BytePS (ByteDance) [33] are based on *parameter server*, while PyTorch-DDP (Facebook) [39], Mariana (Tencent) [82], MALT (NEC Labs) [37], NCCL (NVIDIA) [2], and Horovod (Uber) [56] are based on *MPI-style collective operations*. These systems often involve joint efforts from machine learning, systems, and data management communities, and have been successful in making distributed training easier and more scalable.

On the theory and algorithm side, researchers have also been active in improving the performance of standard *synchronous* and *asynchronous* stochastic gradient (SG) based algorithms. Rightly noticing that a major system bottleneck is communication, researchers have proposed a range of techniques to lower the communication overhead mainly by "*relaxing*" certain aspects of the communication. Examples include (1) *communication compression* (e.g., quantization [4, 7, 73, 76], sparsification [5, 68, 70, 72], and error compensation [67]), (2) *communication decentralization* [34, 40, 42, 43, 64, 66], and (3) *communication delay* (e.g., LocalSGD [21, 44, 61, 69]) and *asynchronization* [43, 52, 60, 80, 81]. These techniques are optimized for *different workloads* and different *network conditions*. These techniques *together* hold promises to significantly decrease the communication overheads, in terms of both bandwidth and latency, or increase the tolerance to the existence of stragglers.

In this paper, we are motivated by one emerging gap between the current landscapes of systems and theory: *Despite the recent advance of distributed learning theory and algorithm on system relaxations, most, if not all, existing systems only rely on standard synchronous and*

Proceedings of the VLDB Endowment, Vol. 15, No. 4 ISSN 2150-8097.
doi:10.14778/3503585.3503590

*Equal contribution.
†Corresponding author.

[VLDB 2022]

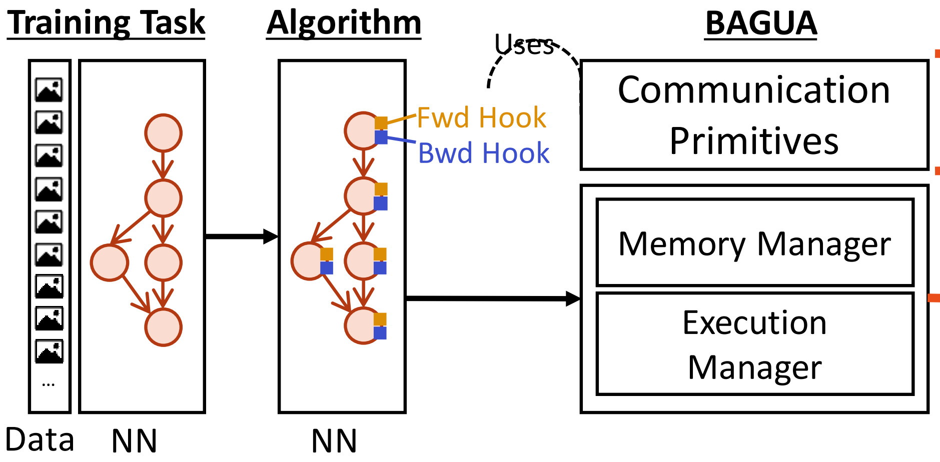
It is not easy to translate *algorithmic flexibility* into *system performance gain*.

Bagua: System Design & Implementation

github.com/BaguaSys/bagua

- *A modular design to accommodate the diversity of different algorithms and communication patterns.*
- *An optimization framework that applies automatically to an algorithm implemented in BAGUA.*

End user: *simply wrap up your training script with BAGUA. Specify the algorithm you want to use*



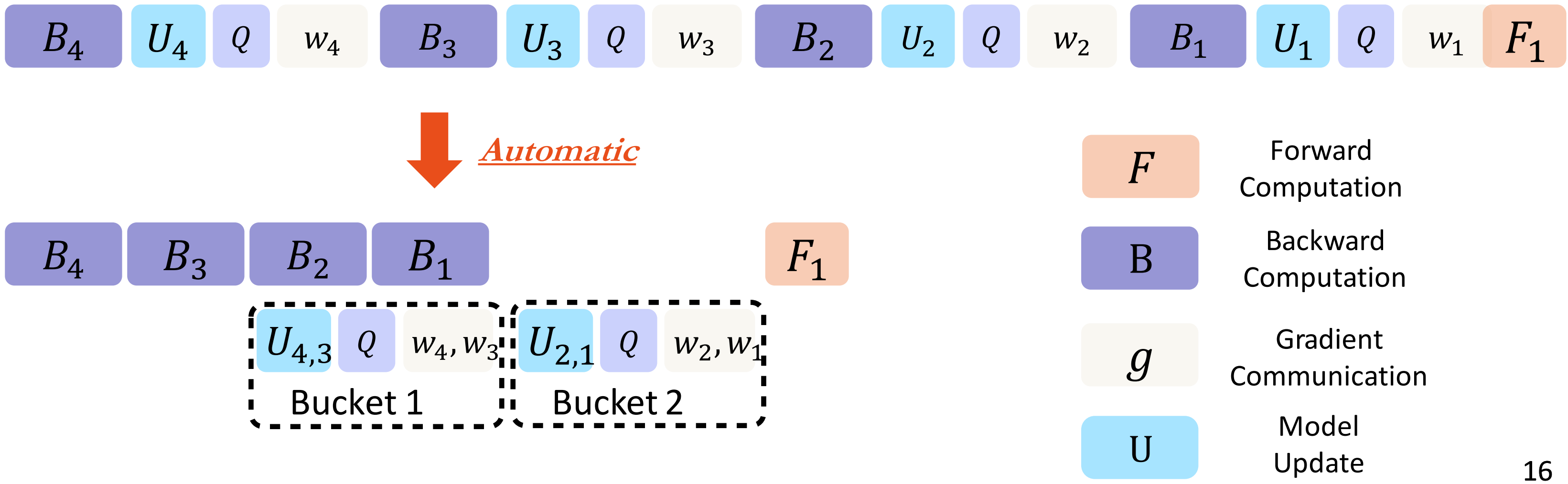
MPI-Style

FCS: Full Prec., Centarlized, Sync
FDS: Full Prec., Decentarlized, Sync
LCS: Low Prec., Centralized, Sync
LDS: Low Prec., Decentarlized, Sync
…

```python
import torch
from bagua import bagua_init, DefaultAlgo

def main():
    args = parse_args()

    # define model and optimizer
    model = MyNet().to(args.device)
    optimizer = torch.optim.SGD(model.parameters(),lr=args.lr)
    # transform to BAGUA wrapper
    model,optimizer = bagua_init(model,optimizer,DefaultAlgo,
      is_intra)

    # train the model over the dataset
    for epoch in range(args.epochs):
        for b_idx,(inputs,targets) in enumerate(train_loader):
            outputs = model(inputs)
            loss = torch.nn.CrossEntropyLoss(outputs,targets)
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()
```

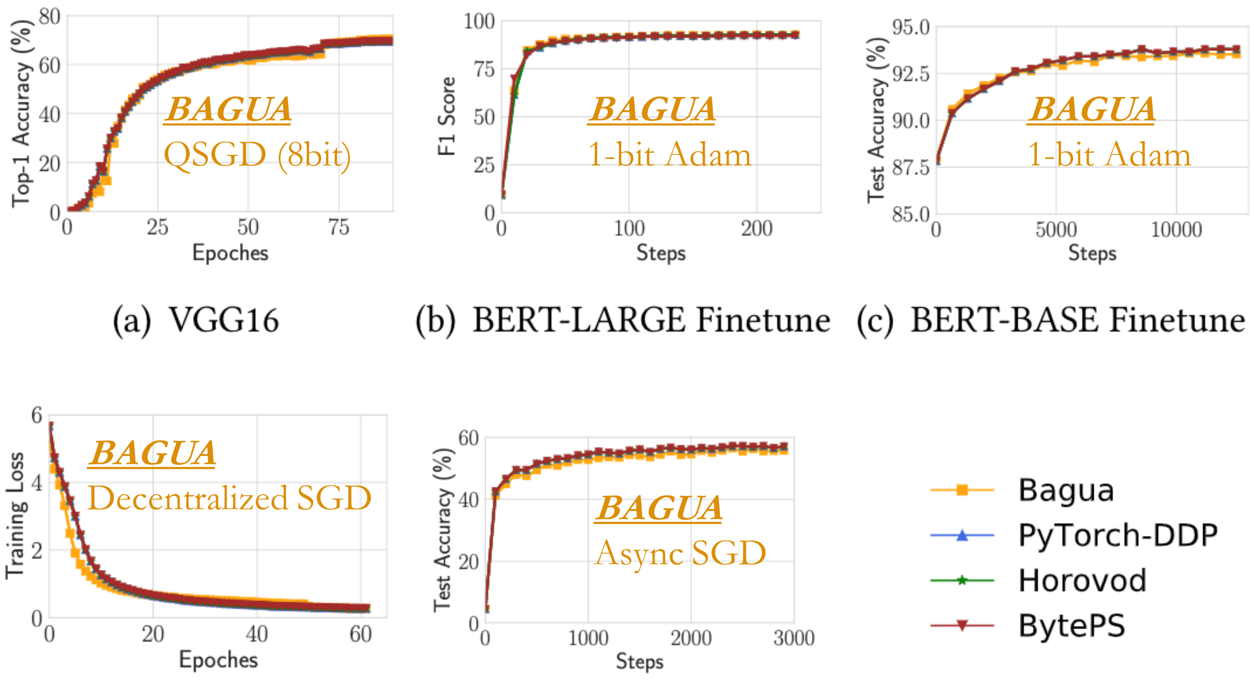
Optimizer: *automatically optimize communication and computations*

E.g., Decentralized, Low Precision Alg.



Bagua Results

Setup: *16 machines, each 8 V100 GPUs. Connected via {10Gbps, 25Gbps, 100Gbps} networks.*



(a) VGG16 (b) BERT-LARGE Finetune (c) BERT-BASE Finetune

(d) Transformer (e) LSTM+AlexNet

Same Convergence with Relaxed Algorithms

Network Conditions	VGG16	BERT-LARGE	BERT-BASE	Transformer	LSTM+AlexNet
100 Gbps	1.1×	1.05×	1.27×	1.2×	1.34×
25 Gbps	1.1×	1.05×	1.27×	1.2×	1.34×
10 Gbps	1.94×	1.95×	1.27×	1.2×	1.34×

Significant speed-up over {Torch-DDP,Horovod 32bits, Horovod 16bits, BytePS}

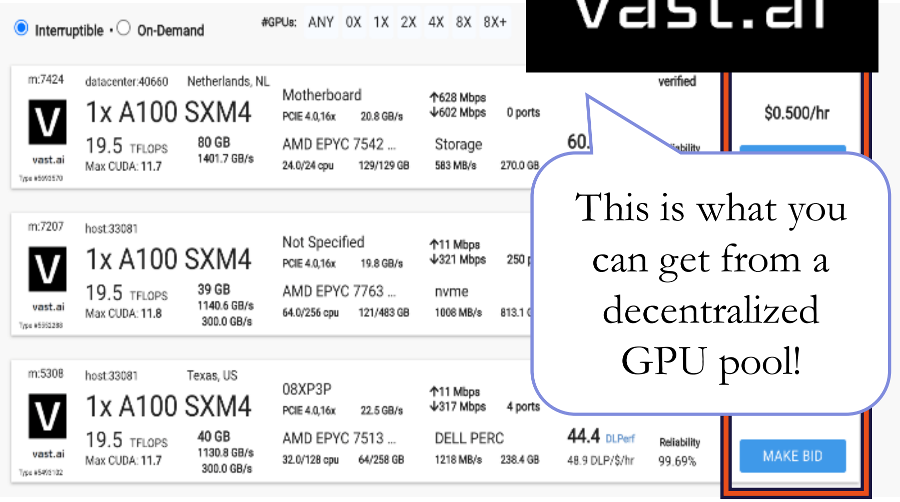
Supporting a diverse set of algorithms can provide significant improvements over existing systems.

From Cloud to Decentralized Compute Resource

Instance Size	vCPUs	Instance Memory (GiB)	GPU – A100	GPU memory	Network Bandwidth (Gbps)	GPUDir… RDM…		…rage (GB)	…Bandwidth (Gbps)	…demand Price/hr
p4d.24xlarge	96	1152	8	320 GB HBM2	400 ENA and EFA	Yes	60… NV…			
p4de.24xlarge (preview)	96	1152	8	640 GB HBM2e	400 ENA and EFA	Yes	60… NV…			

This is $4.09/hour for an A100 GPU.

vast.ai



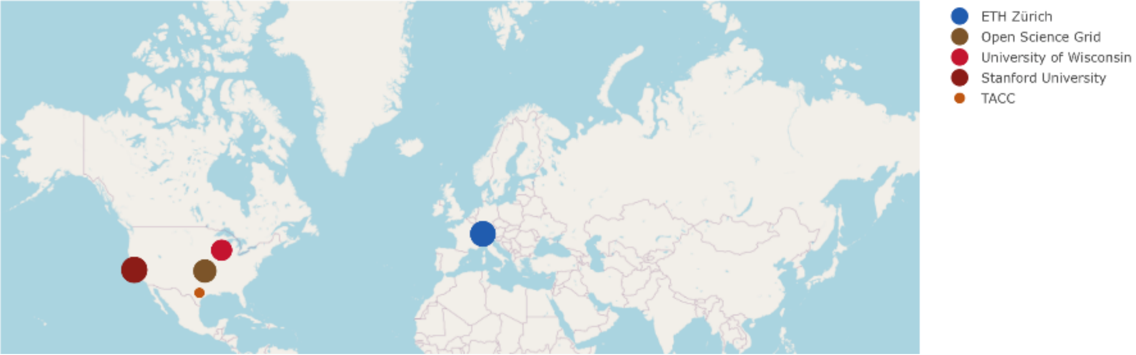
This is what you can get from a decentralized GPU pool!

Available TFlops
71,509 TFlops

Available GPUs
536

Total TFlops
124,428 TFlops

Status Global View



Attempt 2

These algorithmic building blocks need to *be put together!*

CocktailSGD: Fine-tuning Foundation Models over 500Mbps Networks

Jue Wang [*1] **Yucheng Lu** [*2] **Binhang Yuan** [1] **Beidi Chen** [3] **Percy Liang** [4] **Christopher De Sa** [2] **Christopher Re** [4] **Ce Zhang** [1]

Abstract

Distributed training of foundation models, especially large language models (LLMs), is communication-intensive and so has heavily relied on centralized data centers with fast interconnects. *Can we train on slow networks and unlock the potential of decentralized infrastructure for foundation models?* In this paper, we propose COCKTAILSGD, a novel communication-efficient training framework that combines three distinct compression techniques—random sparsification, top-K sparsification, and quantization—to achieve much greater compression than each individual technique alone. We justify the benefit of such a hybrid approach through a theoretical analysis of convergence. Empirically, we show that COCKTAILSGD achieves up to 117× compression in fine-tuning LLMs up to 20 billion parameters without hurting convergence. On a 500Mbps network, COCKTAILSGD only incurs ~ 1.2× slowdown compared with data center networks.

1. Introduction

In recent years, foundation models (Bommasani et al., 2021), including large language models (Brown et al., 2020; Chowdhery et al., 2022; Bommasani et al., 2021; Zhang et al., 2022; Liang et al., 2022; Scao et al., 2022), have enabled rapid advancement for various machine learning tasks, especially in natural language processing (Brants et al., 2007; Austin et al., 2021). Such a significant improvement on quality has been fueled by an ever-increasing amount of data and computes that are required in training these models (Kaplan et al., 2020). Today, training even modest scale models requires a significant amount of compute: For example, fine-tuning GPT-J-6B (6 billion parameters) over merely 10 billion tokens would require 6 petaflops-days: 8 A100 GPUs running at 50% capacity for 5 days!

When training foundation models in a distributed way, *communication* is the key bottleneck in scaling. As an example, fine-tuning GPT-J-6B over 10 billion tokens with a batch size of 262K tokens over 4 machines (each with 2 A100 GPUs) would require 915.5 TB data being communicated during the whole training process! The computation time for such a workload is 114 hours, which means that we need to have at least 20 Gbps connections between these machines to bring the communication overhead to the same scale as the computation time. Not surprisingly, today's infrastructure for training and fine-tuning foundation models are largely *centralized*, with GPUs connected via fast 100Gbps–400Gbps connections (Microsoft, 2020).

Such a heavy reliance on centralized networks increases the cost of infrastructure, and makes it incredibly hard to take advantage of cheaper alternatives, including tier 2 to tier 4 clouds, spot instances and volunteer compute. For example, while volunteering compute projects such as Folding@Home can harvest significant amount of computes for embarrassingly parallelizable workloads (e.g., 2.43exaflops in April 2020 (Larson et al., 2009)), it is challenging to harvest these cycles for foundation model training due to the communication bottleneck. Recently, there has been an exciting collection of work focusing on the decentralized training of neural networks, including those that are algorithmic (Lian et al., 2017; Ryabinin & Gusev, 2020; Diskin et al., 2021; Ryabinin et al., 2021; Yuan et al., 2022; Jue et al.) as well as system efforts such as Training Transformer Together (Borzunov et al., 2022b), and PETALS (Borzunov et al., 2022a). However, despite of these recent efforts, communication is still a significant bottleneck, and one can only compress the communication by at most 10-30× in these recent efforts without hurting convergence. To fully close the gap between centralized infrastructure (100Gbps) and decentralized infrastructure (100Mbps-1Gbps), we need to decrease the communication overhead by at least 100×!

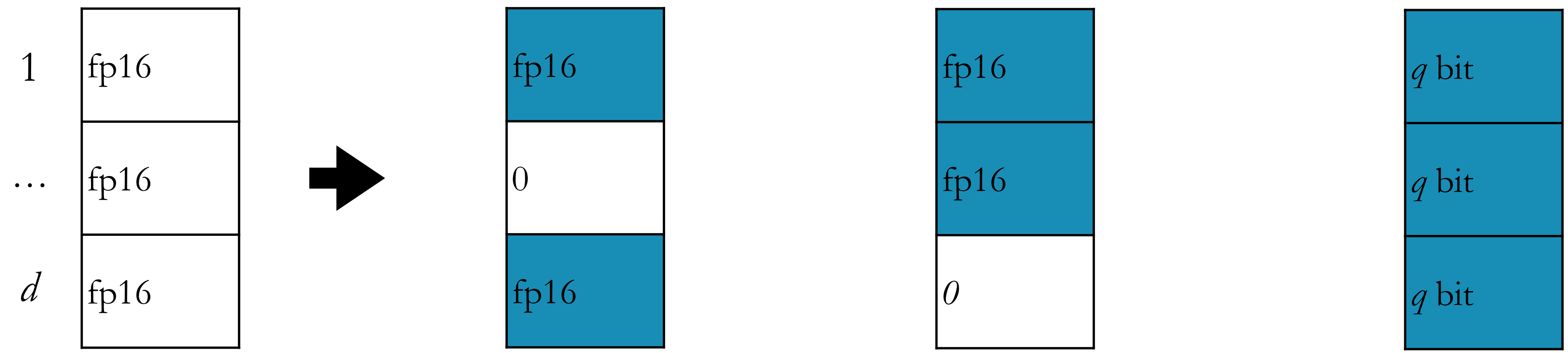
Luckily, there have also been rapid development of communication-efficient optimization algorithms and these efforts provide the foundational building blocks of this paper. Researchers have proposed a wide range of

[*]Equal contribution [1]ETH Zürich, Switzerland [2]Cornell University, USA [3]Carnegie Mellon University, USA [4]Stanford University, USA. Correspondence to: Jue Wang <juewang@inf.ethz.ch>.

Proceedings of the 40[th] International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

[ICML 2023]

Three Methods of Compression



$16d$ bits

top-K

Values

$16K$ bits

Indices

$K\log_2 d$ bits

or d bits

Expensive to compute and to encode Indices

Random Sample

Values

$16pd$ bits

Indices

1 random seed

Might not keep top values as in Top-K

Quantization

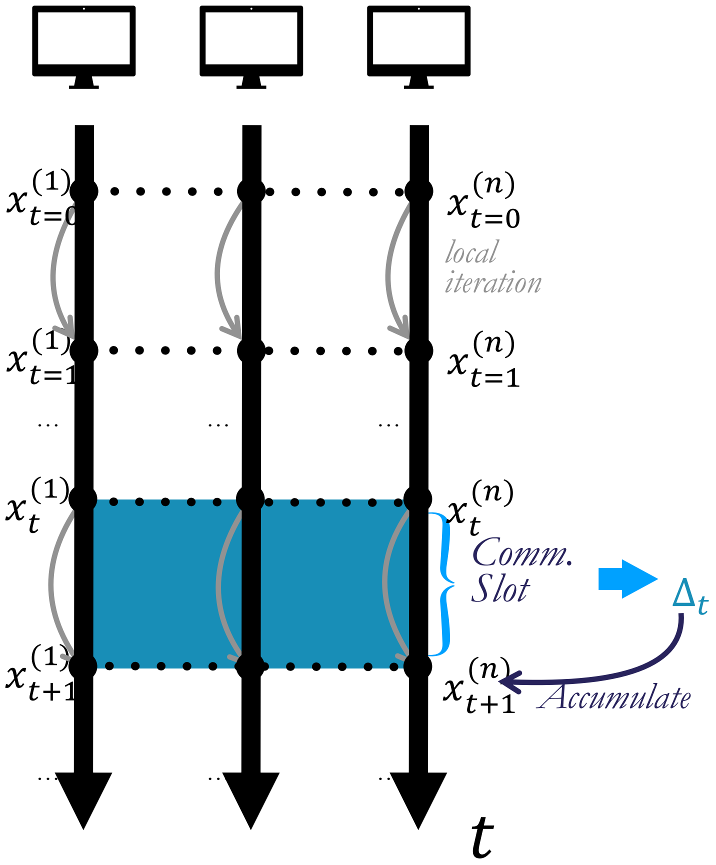
Values

qd bits

Only provide up to 16x compression; hard to go aggressive

It is very hard to reach *100X compression* ratio with a single method.

CocktailSGD: Mixture of Compression Methods



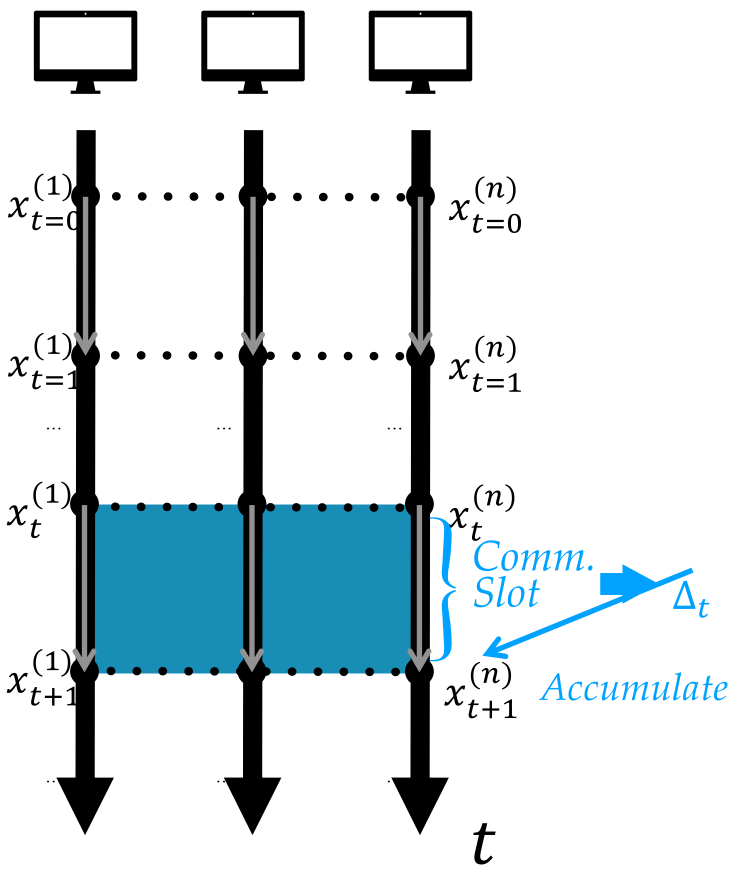
As long as Communication fully fills the Comm. Slot, no slow down caused by communication.

Idea: A Mixture of communication compression techniques.

Looking at Δ_t:

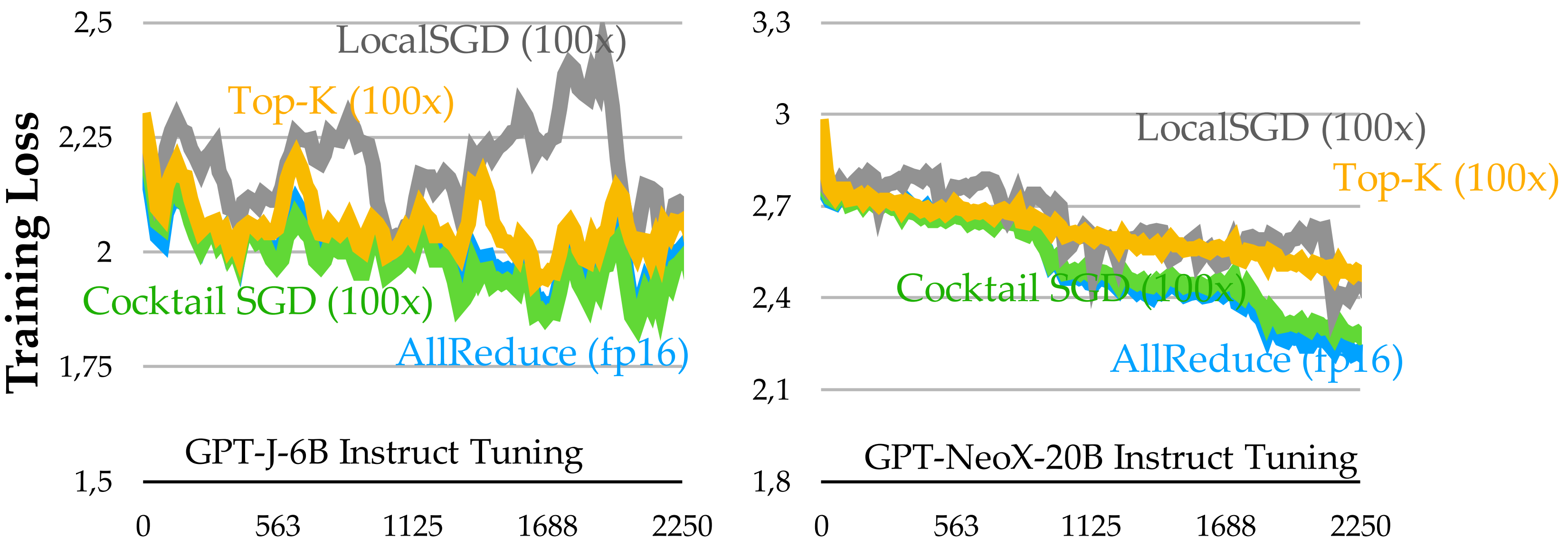
- It has 1-step staleness — // asynchrony
- At t, randomly pick $p\%$ parameters to communicate — // local training: compress $\sim \frac{1}{p\%} \times$
- For selected parameters, let $\delta_t^{(i)}$ be local model updates since last communication:
  - $\tilde{\delta}_t^{(i)} = top\text{–}K\%(\delta_t^{(i)})$ — // topK: compress $\sim \frac{1}{K\%} \times$
  - $\hat{\tilde{\delta}}_t^{(i)} = Quantize(\tilde{\delta}_t^{(i)}, q\,bits)$ — // Quantization: compress $\sim \frac{16}{b} \times$
- Communicate: $\Delta_t = \sum_i \hat{\tilde{\delta}}_t^{(i)}$

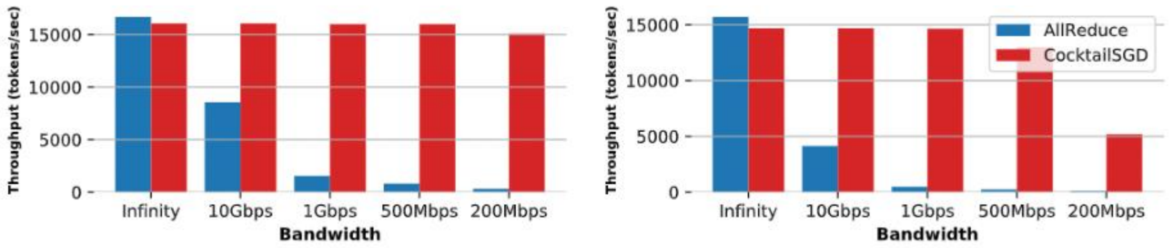
“Cocktail SGD”: Data Parallel over 1Gbps



As long as Communication fully fills the Comm. Slot, no slow down caused by communication.

Different communication compression techniques complement each other and compose well!





(b) GPT-J-6B

(c) GPT-NeoX-20B

Data parallel over ~500 Mbps network!

Large language model training goes *beyond* data parallelism.

Cut 1

Cut 2

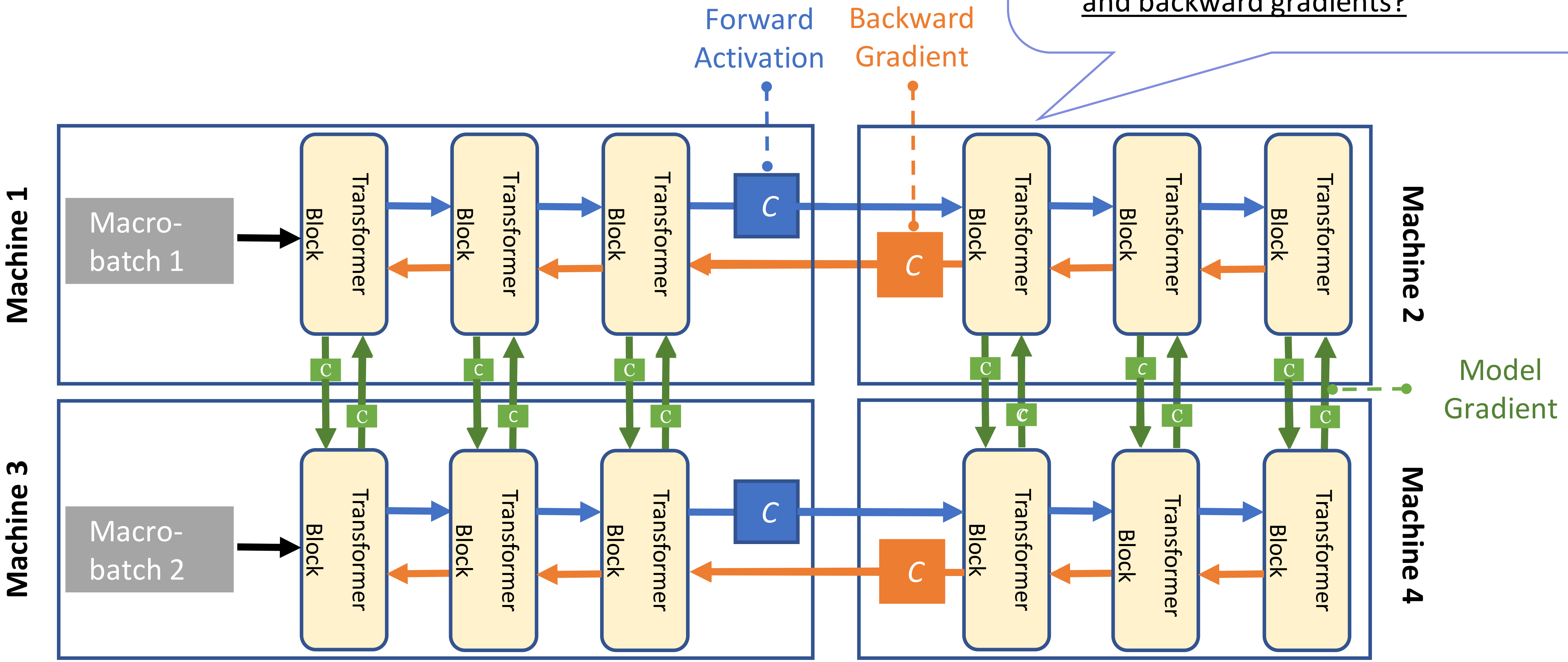
$$\min_{x} \mathbb{E}_{\xi} f(\xi, x) \;\Rightarrow\; \min_{x_f, x_g} \mathbb{E}_{\xi} f(g(\xi, x_g), x_f)$$

Forward Activation

- *(GPT-3) 24MB / 1000tokens*

Pipeline Parallelism

1. How to schedule the communication to accommodate the decentralized connections?
2. How to compress forward activations and backward gradients?



Decentralized Training of Foundation Models

- <u>Decentralized training of FM</u>: the network is 100× slower, but the pre-training throughput is only 1.7~3.5× slower!
- <u>Decentralized fine-tuning of FM</u>: ***AQ-SGD*** communication-efficient pipeline training with activation compression.

Decentralized Training of Foundation Models in Heterogeneous Environments

Binhang Yuan[†*], Yongjun He[†*], Jared Quincy Davis[‡], Tianyi Zhang[‡], Tri Dao[‡], Beidi Chen[‡], Percy Liang[‡], Christopher Re[‡], Ce Zhang[†]

[†]ETH Zürich, Switzerland [‡]Stanford University, USA
{binhang.yuan, yongjun.he, ce.zhang}@inf.ethz.ch
{tz58, jaredq, beidic, trid, pliang, chrismre}@stanford.edu

Abstract

Training foundation models, such as GPT-3 and PaLM, can be extremely expensive, often involving tens of thousands of GPUs running continuously for months. These models are typically trained in specialized clusters featuring fast, homogeneous interconnects and using carefully designed software systems that support both data parallelism and model/pipeline parallelism. Such dedicated clusters can be costly and difficult to obtain. *Can we instead leverage the much greater amount of decentralized, heterogeneous, and lower-bandwidth interconnected compute?* Previous works examining the heterogeneous, decentralized setting focus on relatively small models that can be trained in a purely data parallel manner. State-of-the-art schemes for model parallel foundation model training, such as Megatron, only consider the homogeneous data center setting. In this paper, we present the first study of training large foundation models with model parallelism in a decentralized regime over a heterogeneous network. Our key technical contribution is a scheduling algorithm that allocates different computational "tasklets" in the training of foundation models to a group of decentralized GPU devices connected by a slow heterogeneous network. We provide a formal cost model and further propose an efficient evolutionary algorithm to find the optimal allocation strategy. We conduct extensive experiments that represent different scenarios for learning over geo-distributed devices simulated using real-world network measurements. In the most extreme case, across 8 different cities spanning 3 continents, our approach is 4.8× faster than prior state-of-the-art training systems (Megatron).

Code Availability: `https://github.com/DS3Lab/DT-FM`

1 Introduction

Recent years have witnessed the rapid development of deep learning models, particularly foundation models (FMs) [1] such as GPT-3 [2] and PaLM [3]. Along with these rapid advancements, however, comes computational challenges in training these models: the training of these FMs can be very expensive — a single GPT3-175B training run takes 3.6K Petaflops-days [2]— this amounts to $4M on today's AWS on demand instances, even assuming 50% device utilization (V100 GPUs peak at 125 TeraFLOPS)! Even the smaller scale language models, e.g., GPT3-XL (1.3 billion parameters), on which this paper evaluates, require 64 Tesla V100 GPUs to run for one week, costing $32K on AWS. As a result, speeding up training and decreasing the cost of FMs have been active research areas. Due to their vast number of model parameters, state-of-the-art systems (e.g., Megatron[4], Deepspeed[5], Fairscale[6]) leverage multiple forms of parallelism [4, 7, 8, 9, 10, 11]. However, their design is only tailored to *fast*, *homogeneous* data center networks.

* Equal contribution.

1

[NeurIPS 2022-(a)]

Fine-tuning Language Models over Slow Networks using Activation Compression with Guarantees

Jue Wang[†*], Binhang Yuan[†*], Luka Rimanic[†*], Yongjun He[†], Tri Dao[‡], Beidi Chen[†], Christopher Re[†], Ce Zhang[†]

[†]ETH Zürich, Switzerland [‡]Stanford University, USA
{jue.wang, binhang.yuan, luka.rimanic, yongjun.he, ce.zhang}@inf.ethz.ch
{beidic, trid, chrismre}@stanford.edu

Abstract

Communication compression is a crucial technique for modern distributed learning systems to alleviate their communication bottlenecks over slower networks. Despite recent intensive studies of gradient compression for data parallel-style training, compressing the *activations* for models trained with pipeline parallelism is still an open problem. In this paper, we propose AC-SGD, a novel activation compression algorithm for communication-efficient pipeline parallelism training over slow networks. Different from previous efforts in activation compression, instead of compressing activation values directly, AC-SGD compresses the *changes of the activations*. This allows us to show, to the best of our knowledge for the first time, that one can still achieve $O(1/\sqrt{T})$ convergence rate for non-convex objectives under activation compression, without making assumptions on gradient unbiasedness that do not hold for deep learning models with non-linear activation functions. We then show that AC-SGD can be optimized and implemented efficiently, without additional end-to-end runtime overhead. We evaluated AC-SGD to fine-tune language models with up to 1.5 billion parameters, compressing activations to 2-4 bits. AC-SGD provides up to 4.3× end-to-end speed-up in slower networks, without sacrificing model quality. Moreover, we also show that AC-SGD can be combined with state-of-the-art gradient compression algorithms to enable "end-to-end communication compression": *All communications between machines, including model gradients, forward activations, and backward gradients are compressed into lower precision*. This provides up to 4.9× end-to-end speed-up, without sacrificing model quality.

Code Availability: `https://github.com/DS3Lab/AC-SGD`

1 Introduction

Recent efforts in improving communication efficiency for distributed learning have significantly decreased the dependency of training deep learning models on fast data center networks — the *gradient* can be compressed to lower precision or sparsified [1, 2, 3, 4], which speeds up training over low bandwidth networks, whereas the *communication topology* can be decentralized [5, 6, 7, 8, 9, 10], which speeds up training over high latency networks. Indeed, today's state-of-the-art training systems, such as Pytorch [11, 12], Horovod [13], Bagua [14], and BytePS [15], already support many of these communication-efficient training paradigms.

However, with the rise of large foundation models [16] (e.g., BERT [17], GPT-3 [18], and CLIP[19]), improving communication efficiency via compression becomes more challenging. Current training systems for foundation models such as Megatron [20], Deepspeed [21], and Fairscale [22], allocate different layers of the model onto multiple devices and need to communicate — *in addition to* the gradients on the models — the

* Equal contribution.

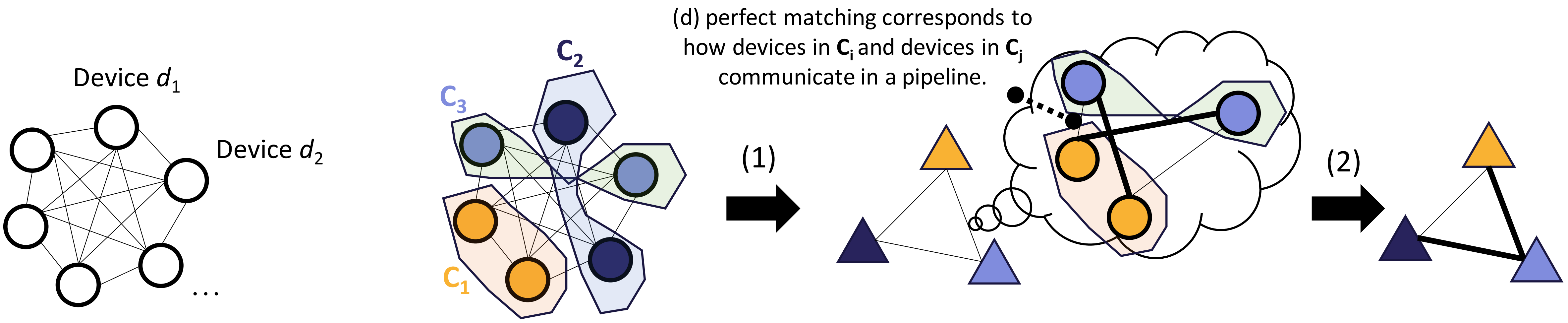
1

[NeurIPS 2022-(b)]

Accommodate Communication in a Decentralized network

A bi-level scheduling algorithm based on an extended balanced graph partition to estimate the communication cost:

- Data parallel communication cost: nodes handling the same stage need to exchange gradients;
- Pipeline parallel communication cost: nodes handling nearby stages for the same micro-batch need to communicate activation in the forward propagation and gradients of the activation in the backward propagation.



(a) Communication Topology Graph **G** over *N* devices

(b) Each partition $\mathbf{C}_i$ deals with one stage, running data parallel within each partition

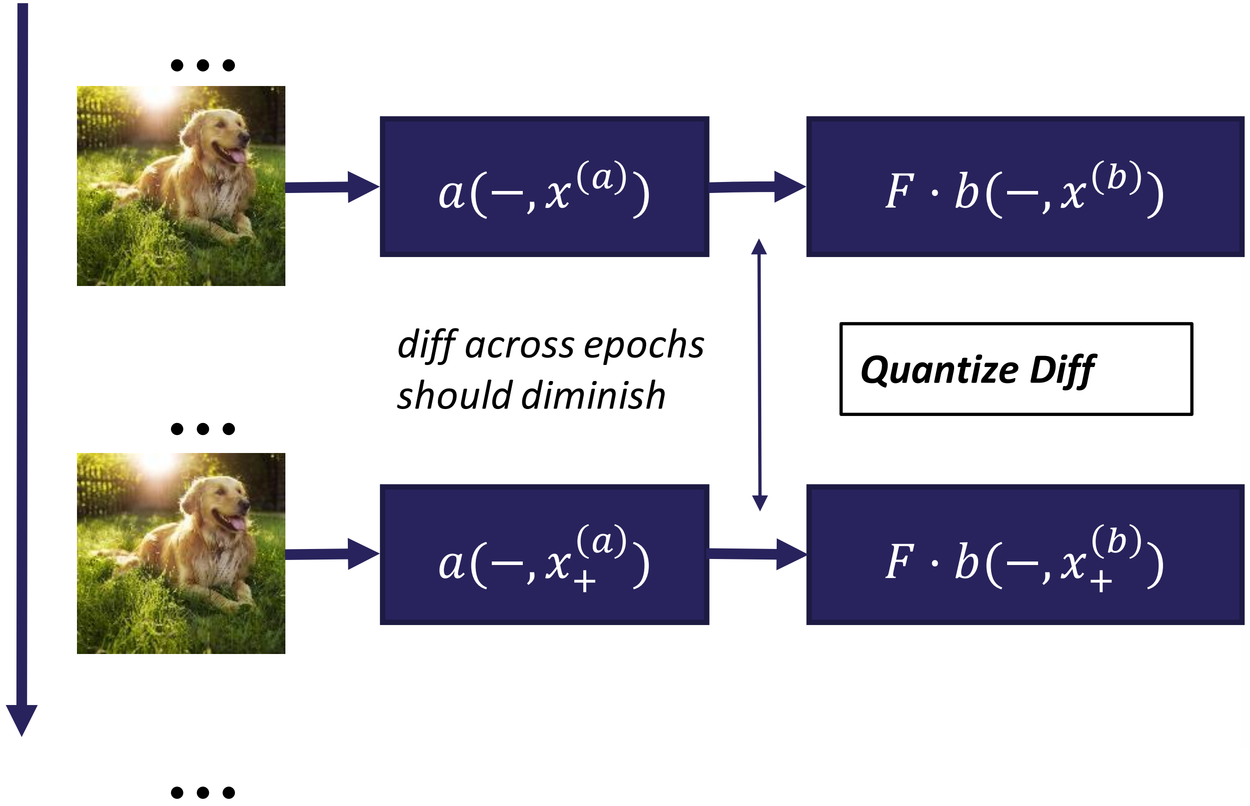
(c) Coarsened graph $\hat{G}$ decoding the cost of pipeline parallel

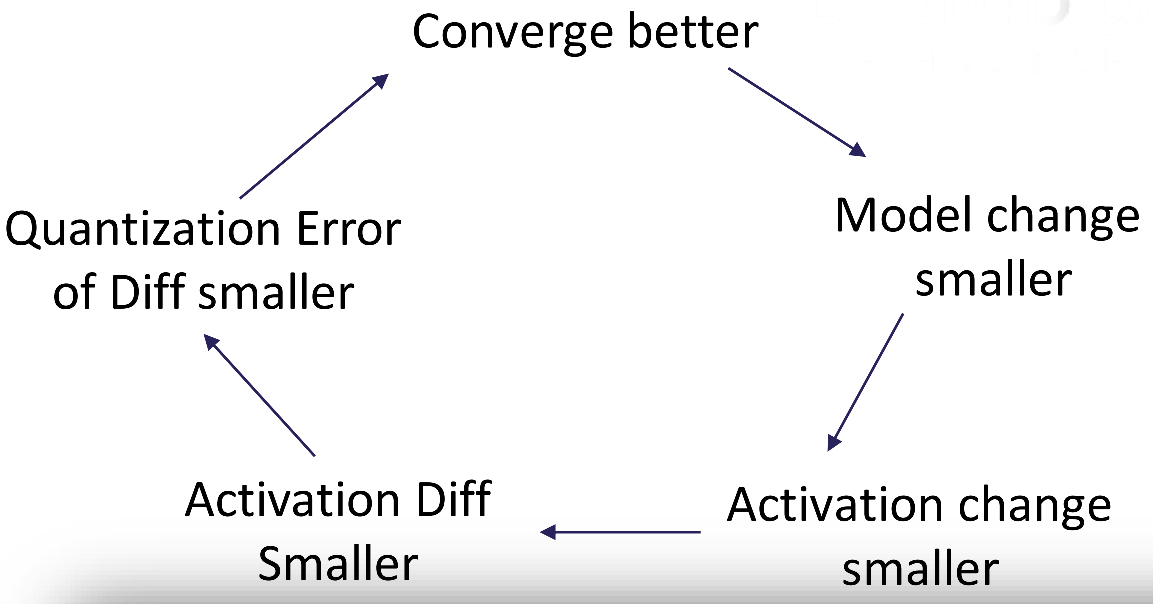
(e) Open-loop-traveling-salesman provides a pipeline structure

AQ-SGD

$$\min_{x \in \mathbb{R}^d} f(x) := \mathbb{E}_{\xi \sim \mathcal{D}} F(b(a(\xi, x^{(a)}), x^{(b)}))$$

Direct quantization only works to some degree.





- **(A1: Lipschitz assumptions)** We assume that ∇f, $\nabla(f \circ b)$ and a are L_f, $L_{f \circ b}$-, and ℓ_a-Lipschitz, respectively, recalling that a function g is L_g-Lipschitz if
$$\|g(x) - g(y)\| \leq L_g \|x - y\|, \quad \forall x, \forall y.$$
Furthermore, we assume that a and $f \circ b$ have gradients bounded by C_a and $C_{f \circ b}$, respectively, i.e. $\|\nabla a(x)\| \leq C_a$, and $\|\nabla (f \circ b)(x)\| \leq C_{f \circ b}$.
- **(A2: SGD assumptions)** We assume that the stochastic gradient g_ξ is unbiased, i.e. $\mathbb{E}_\xi[g_\xi(x)] = \nabla f(x)$, for all x, and with bounded variance, i.e. $\mathbb{E}_\xi \|g_\xi(x) - \nabla f(x)\|^2 \leq \sigma^2$, for all x.

Theorem 3.1. *Suppose that Assumptions A1, A2 hold, and consider an unbiased quantization function $Q(x)$ which satisfies that there exists $c_Q < \sqrt{1/2}$ such that $\mathbb{E}\|x - Q(x)\| \leq c_Q \|x\|$, for all x.[1] Let $\gamma = \frac{1}{3(C + 3L_f)\sqrt{T}}$ be the learning rate, where*

$$C = \frac{4 c_Q \ell_a (1 + C_a) L_{f \circ b} N}{\sqrt{1 - 2c_Q^2}}.$$

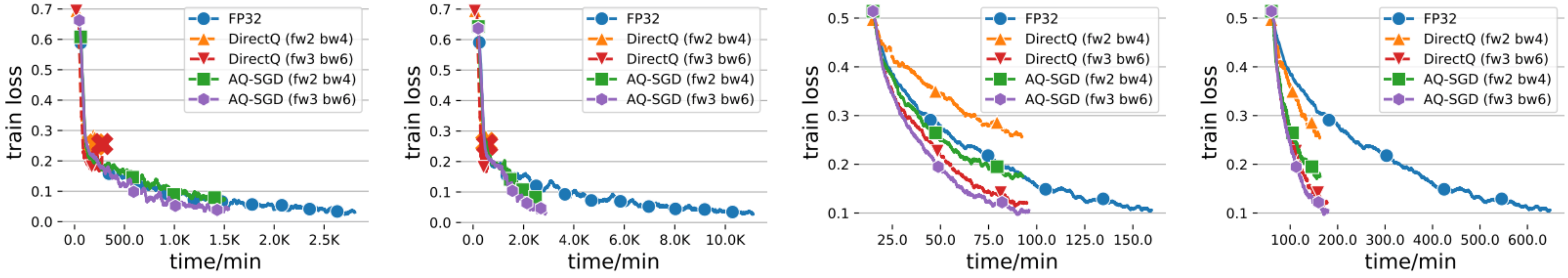
Then after performing T updates one has

$$\frac{1}{T} \sum_{t \in [T]} \mathbb{E}\|\nabla f(x_t)\|^2 \lesssim \frac{(C + L_f)(f(x_1) - f^*)}{\sqrt{T}} + \frac{\sigma^2 + (c_Q C_a C_{f \circ b})^2}{\sqrt{T}}. \tag{3.1}$$

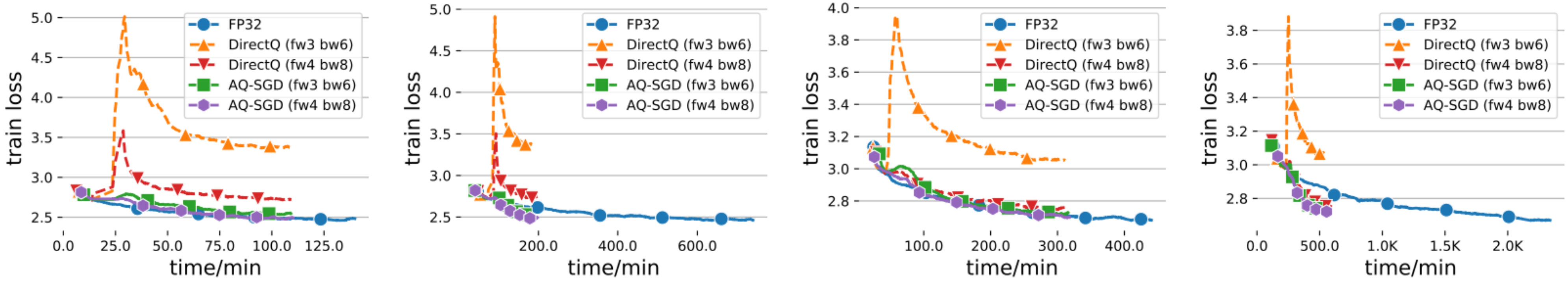
[NeurIPS 2022-(b)]

AQ-SGD Results

- End-to-end training performance over different networks. x represents divergence.



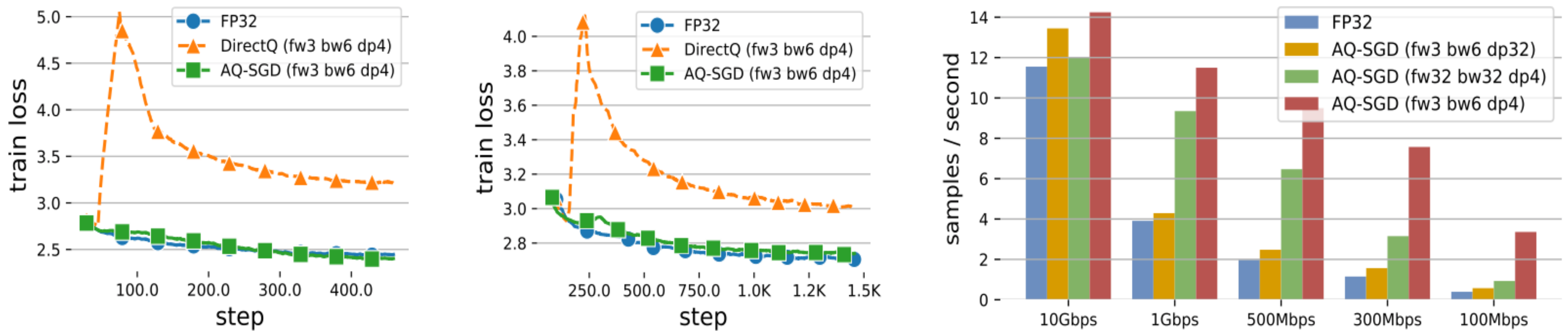
(a) QNLI, 500Mbps (b) QNLI, 100Mbps (c) CoLA, 500Mbps (d) CoLA, 100Mbps



(e) WikiText2, 500Mbps (f) WikiText2, 100Mbps (g) arXiv, 500Mbps (h) arXiv, 100Mbps

AQ-SGD Results

- Convergence and Throughput of AQ-SGD combined with gradient compression.

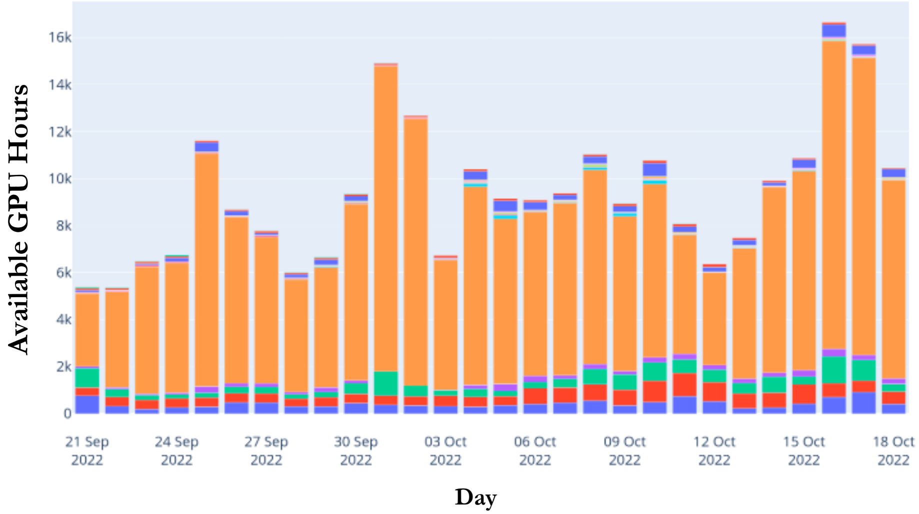


(a) WikiText2, GPT2-1.5B (b) arXiv, GPT2-1.5B (c) Training Throughput

Some Small Steps Towards *Decentralized ML.*

Open Research on the Together Decentralized Cloud

Connecting idle compute across academic institutions.



HELM

11 billion tokens
60K GPU Hours
10 Open Models

BLOOM	176B	July 2022
T0pp	11B	October 2021
GPT-J	6B	July 2021
GPT-NeoX	20B	February 2022
GLM	130B	August 2022
UL2	20B	October 2022
T5	11B	February 2020
OPT	175B	June 2022
OPT	66B	June 2022
YaLM	100B	June 2022

Summary

- ***Communication*** is a key bottleneck of distributed learning, both for centralized data center network and decentralized environments.

- **We can develop *Algorithms* to alleviate communication bottlenecks:**
  - *Data Parallel: {asynchronous, local training, compression, quantization, decentralized topology} & their combinations.*
  - *Model Parallel: Careful error compensation.*

- **Innovation of *Systems* is need to unleash the full potential *Algorithms*:**
  - *Bagua: Automatic optimization framework.*
  - *System Scheduling of communication in decentralized environments.*

Ongoing Research.

Large language model training goes *beyond* data & pipeline parallelism.

Scheduling in a Bigger Scope

- Heterogeneous hardware:
  - A100, A800, A40, 3090, 4090, etc.
- Heterogeneous connections:
  - NVSwitch, NVLink, RDMA, etc.
  - Cross data-center/ @home Network.
- Parallel schemas:
  - Data parallelism;
  - Pipeline parallelism;
  - Tensor model parallelism;
  - Optimizer parallelism (FSDP).

Communication Efficient Parallel LLM Training Algorithms

- Communication compression for different parallel schemas:
  - <u>Data parallelism;</u>
- <u>Pipeline parallelism;</u>
- Tensor model parallelism (?)
- Optimizer parallelism (FSDP) (?)

CocktailSGD: Fine-tuning Foundation Models over 500Mbps Networks

Jue Wang, Yucheng Lu, Binhang Yuan, Beidi Chen, Percy Liang, Christopher De Sa, Christopher Re, Ce Zhang

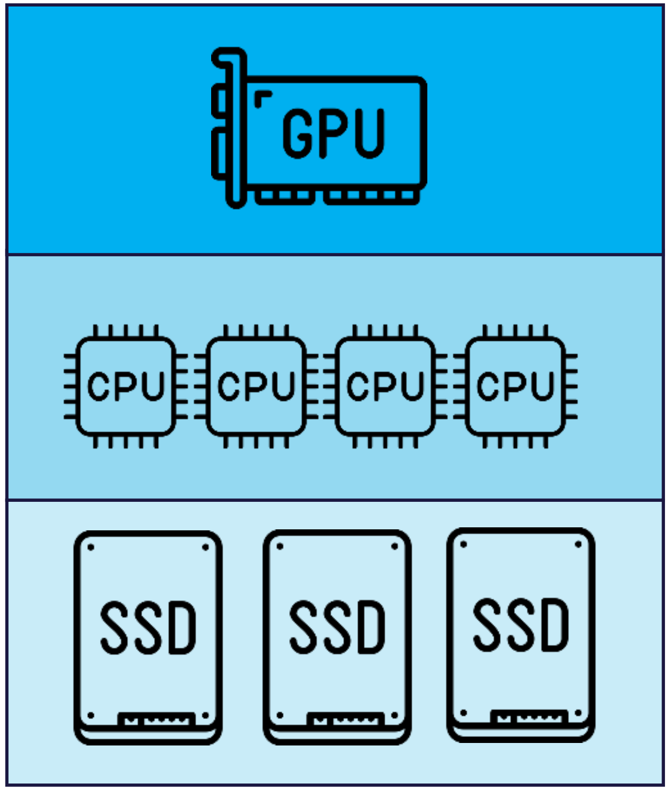
[ICML 2023]

Fine-tuning Language Models over Slow Networks using Activation Compression with Guarantees

Jue Wang†*, Binhang Yuan†*, Luka Rimanic†*, Yongjun He†, Tri Dao‡, Beidi Chen†, Christopher Re‡, Ce Zhang†
†ETH Zürich, Switzerland ‡Stanford University, USA
{jue.wang, binhang.yuan, luka.rimanic, yongjun.he, ce.zhang}@inf.ethz.ch
{beidic, trid, chrismre}@stanford.edu

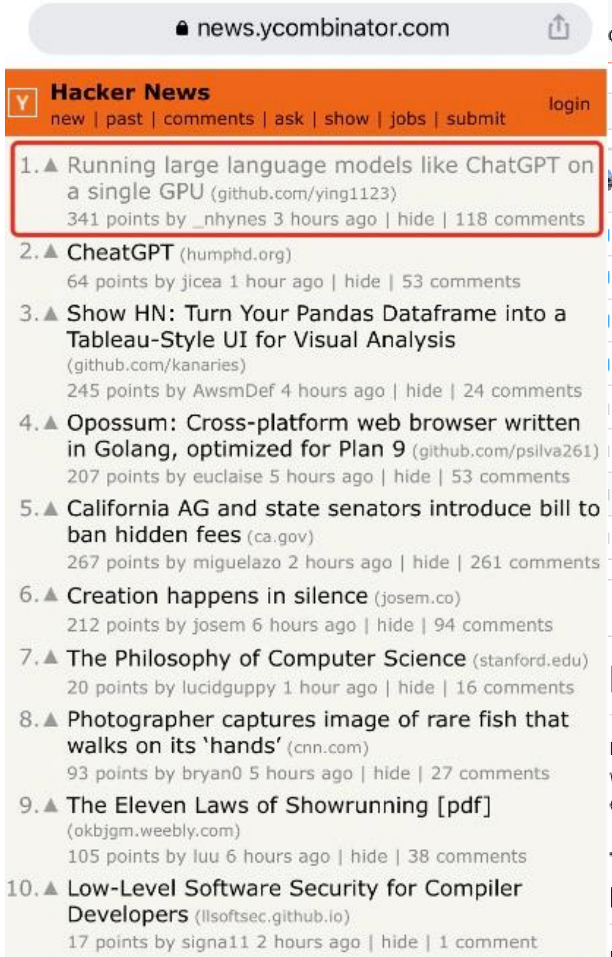
[NeurIPS 2022]

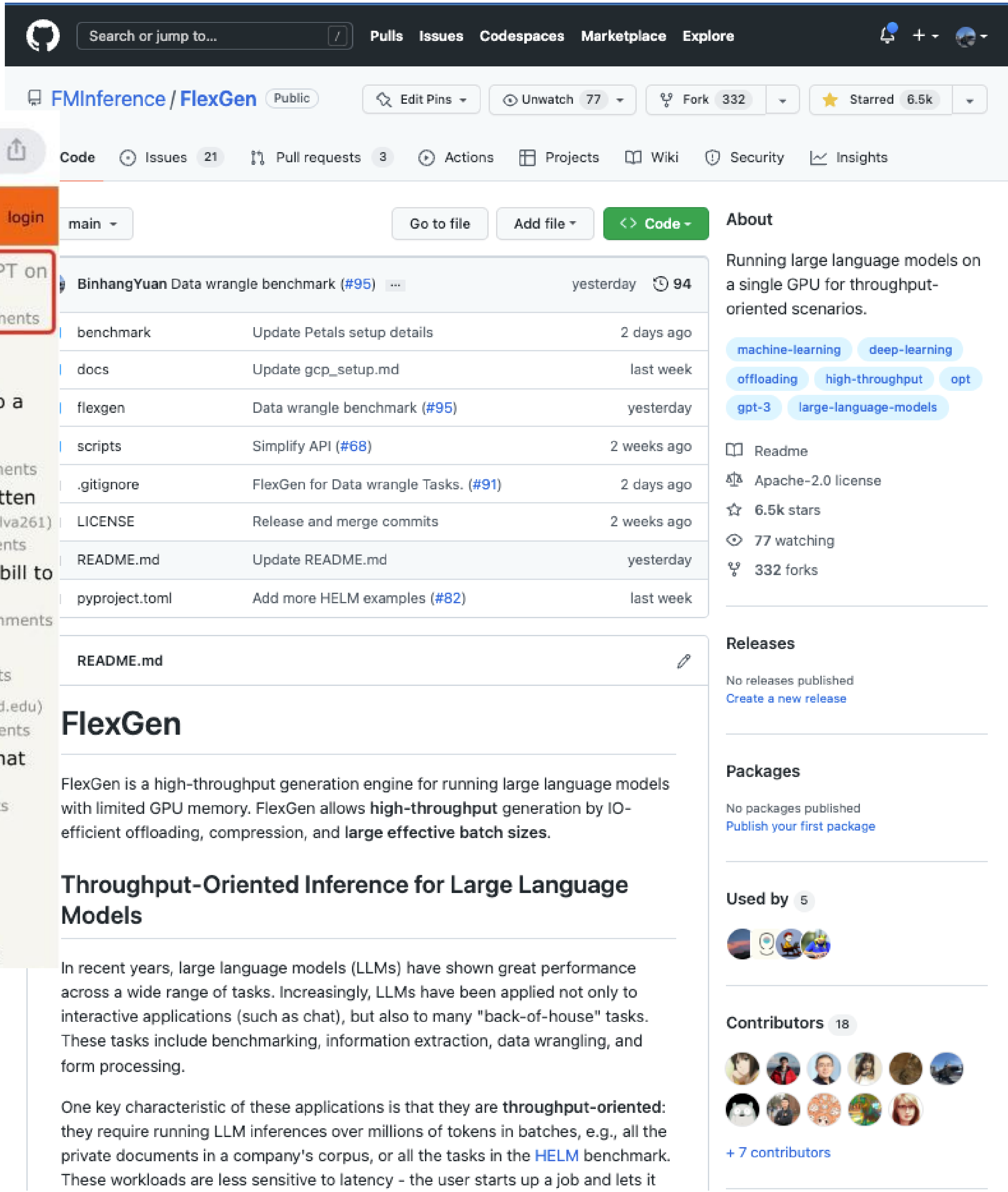
Enhancing High-Throughput LLM Inference through Off-loading Systems.



FlexGen

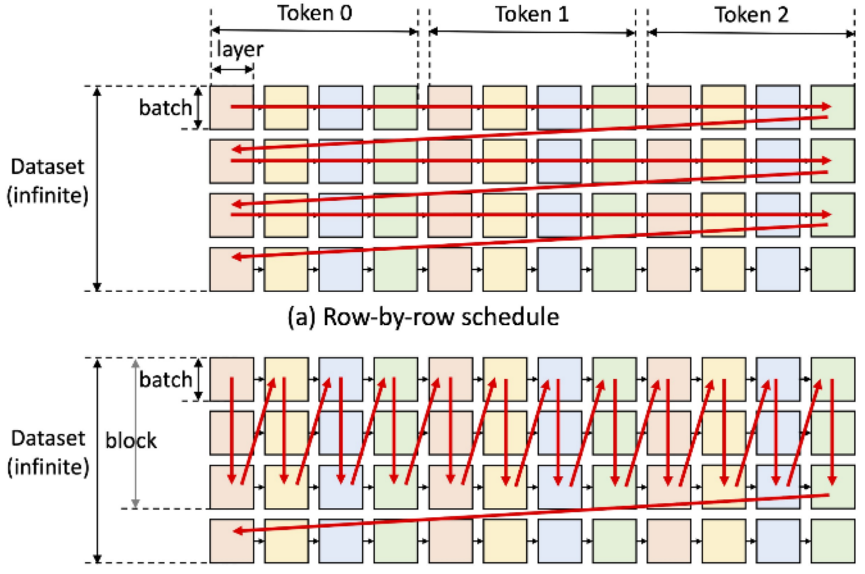
- *OPT-175B Scale Inference on a single GPU:*
  - 6.5K on Github;
  - Top discussion on Hacker News;
  - High throughput scenario: *1 token/s*.





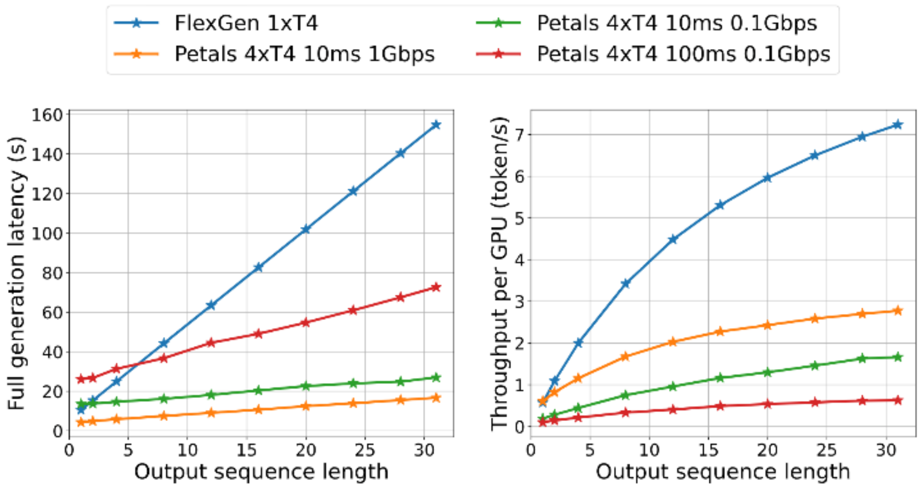
FlexGen

High-Throughput Generative Inference of Large Language Models with a Single GPU



(a) Row-by-row schedule

(b) Zig-zag block schedule



FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU

Ying Sheng[1] Lianmin Zheng[2] Binhang Yuan[3] Zhuohan Li[2] Max Ryabinin[4,5] Beidi Chen[6,7] Percy Liang[1] Christopher Ré[1] Ion Stoica[2] Ce Zhang[3]

Abstract

The high computational and memory requirements of large language model (LLM) inference make it feasible only with multiple high-end accelerators. Motivated by the emerging demand for latency-insensitive tasks with batched processing, this paper initiates the study of high-throughput LLM inference using limited resources, such as a single commodity GPU. We present FlexGen, a high-throughput generation engine for running LLMs with limited GPU memory. FlexGen can be flexibly configured under various hardware resource constraints by aggregating memory and computation from the GPU, CPU, and disk. By solving a linear programming problem, it searches for efficient patterns to store and access tensors. FlexGen further compresses the weights and the attention cache to 4 bits with negligible accuracy loss. These techniques enable FlexGen to have a larger space of batch size choices and thus significantly increase maximum throughput. As a result, when running OPT-175B on a single 16GB GPU, FlexGen achieves significantly higher throughput compared to state-of-the-art offloading systems, reaching a generation throughput of 1 token/s for the first time with an effective batch size of 144. On the HELM benchmark, FlexGen can benchmark a 30B model with a 16GB GPU on 7 representative sub-scenarios in 21 hours. The code is available at https://github.com/FMInference/FlexGen.

Figure 1. The total latency for a block and throughput trade-offs of three offloading-based systems for OPT-175B (left) and OPT-30B (right) on a single NVIDIA T4 (16 GB) GPU with 208 GB CPU DRAM and 1.5TB SSD. FlexGen achieves a new Pareto-optimal frontier with 100× higher maximum throughput for OPT-175B. Other systems cannot further increase throughput due to out-of-memory issues. "(c)" denotes compression.

1. Introduction

In recent years, large language models (LLMs) have demonstrated strong performance across a wide range of tasks (Brown et al., 2020; Bommasani et al., 2021; Zhang et al., 2022; Chowdhery et al., 2022). Along with these unprecedented capabilities, generative LLM inference comes with unique challenges. These models can have billions, if not trillions of parameters (Chowdhery et al., 2022; Fedus et al., 2022), which leads to extremely high computational and memory requirements to run. For example, GPT-175B requires 325GB of GPU memory simply to load its model weights. Fitting this model onto GPUs would require at least five A100 (80GB) GPUs and complex parallelism strategies (Pope et al., 2022; Aminabadi et al., 2022). Thus, lowering LLM inference resource requirements has recently attracted intense interest.

In this paper, we focus on a setting that we call *throughput-oriented generative inference*. In addition to interactive use cases such as chatbots, LLMs are also applied to many "back-of-house" tasks such as benchmarking (Liang et al., 2022), information extraction (Narayan et al., 2018), data wrangling (Narayan et al., 2022), and form processing (Chen et al., 2021). One key characteristic of these tasks is that they often require running LLM inference in batches over a large number of tokens (e.g., all the documents in a company's

[1]Stanford University [2]UC Berkeley [3]ETH Zurich [4]Yandex [5]HSE University [6]Meta [7]Carnegie Mellon University. Correspondence to: Ying Sheng <ying1123@stanford.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

[ICML 2023 Oral]

FlexGen 2.0

- Efficient support of multiple weak GPUs;
- Easy support for general transformer models;
- Integrate multiple relaxed computations (sparsified, quantized, etc.).

Summary

- ***Communication*** **is a key bottleneck of distributed learning, both for centralized data center network and decentralized environments.**

- **We can develop *Algorithms* to alleviate communication bottlenecks:**
  - *Data Parallel: {asynchronous, local training, compression, quantization, decentralized topology} & their combinations.*
  - *Model Parallel: Careful error compensation.*

- **Innovation of *Systems* is need to unleash the full potential *Algorithms*:**
  - *Bagua: Automatic optimization framework.*
  - *System Scheduling of communication in decentralized environments.*

Personal page:
https://binhangyuan.github.io/site/

Thank you!