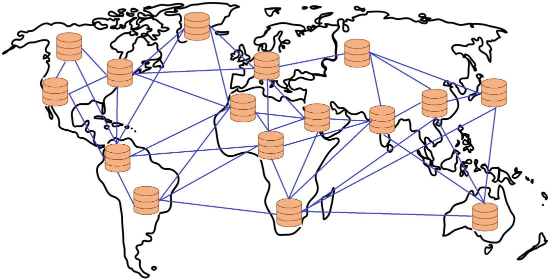
WeightGrad: Geo-Distributed Data Analysis Using Quantization for Faster Convergence and Better Accuracy

Syeda Nahida Akter

Muhammad Abdullah Adnan

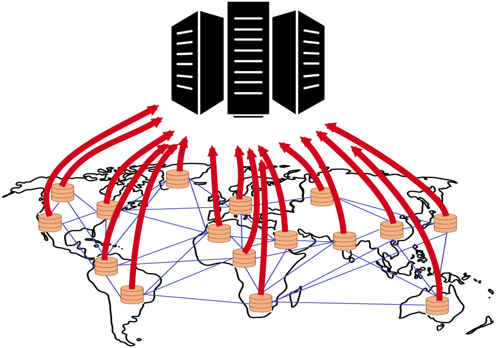
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka, Bangladesh

Problem Overview

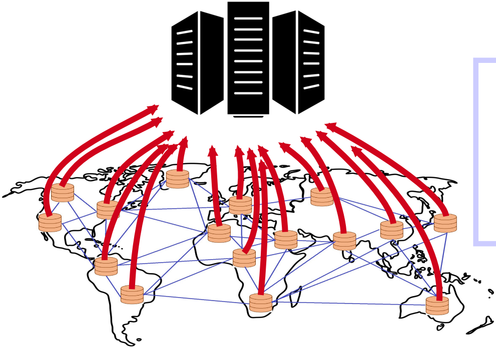


- Large scale cloud organizations are establishing data centers and "edge" clusters globally to provide their users low latency access to their services.
- Microsoft and Google have tens of data centers, with the latter also operating 1500 edges worldwide.

Problem Overview



Problem Overview

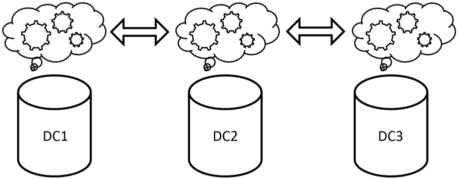


Problem

- Powerful machines.
- Huge memory.
- Large amount of time.

Problem Overview

Clearly, **centralization** is not a feasible solution which motivates the need to **distribute the DNN** system across multiple data centers.



Motivation

- There are many large-scale distributed ML systems, among them, the **parameter server architecture** provides a performance advantage over other systems for many ML applications and has been widely adopted in many ML systems.

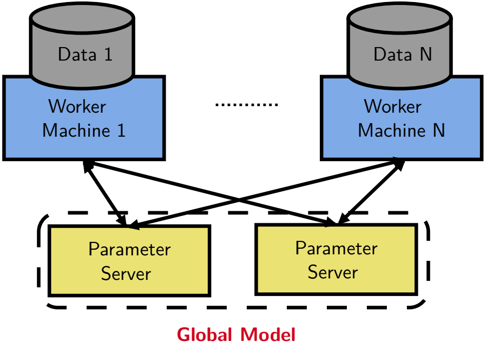


Figure: Basic PS architecture

Motivation

- Deployment of PS architecture on WANs needs to synchronize updates among the workers which is a high cost operation that can significantly slow down the worker machines.

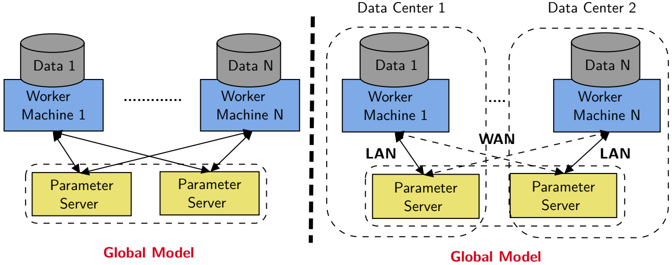


Figure: Deploy PS on WANs

Motivation

- Introduction of quantization has provided significant speedup in convergence in deep learning training.

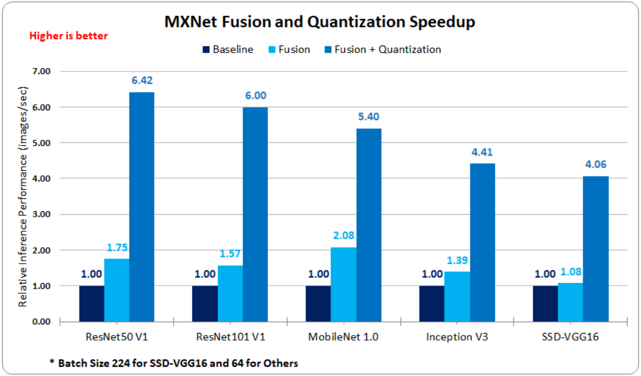


Figure: MXNet speedup graph [Chen et al. (2015)]

Motivation

- But quantizing parameters for faster communication between worker and parameter server also causes loss in precision thus some loss in accuracy. Here is an accuracy table from [Hou et al. (2019)].

Table 3: Top-1 and top-5 accuracies (%) on ImageNet.

weight	gradient	$N=2$		$N=4$		$N=8$	
		top-1	top-5	top-1	top-5	top-1	top-5
FP	FP	55.08	78.33	55.45	78.57	55.40	78.69
LAQ4	FP	53.79	77.21	54.22	77.53	54.73	78.12
	SQ3 (no clipping)	52.48	75.97	52.87	76.40	53.18	76.62
	SQ3 (clip, $c=3$)	54.13	77.27	54.23	77.55	54.34	78.07

Problem Definition

For distributed setup, we define the problem as

Problem Definition

For distributed setup, we define the problem as

- How to efficiently utilize limited WAN b/w

Problem Definition

For distributed setup, we define the problem as

- How to efficiently utilize limited WAN b/w
- How to ensure faster convergence without loss of accuracy

Distributed ML Systems

- As a distributed ML system, the state-of-the-art method is : **"Gaia: Geo-Distributed MachineLearning Approaching LAN Speeds.", Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R.Ganger, Phillip B. Gibbons, and Onur Mutlu. 2017.** [Hsieh et al. (2017)]

Distributed ML Systems

- As a distributed ML system, the state-of-the-art method is : **"Gaia: Geo-Distributed MachineLearning Approaching LAN Speeds.", Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R.Ganger, Phillip B. Gibbons, and Onur Mutlu. 2017.** [Hsieh et al. (2017)]
- The goal of Gaia is to develop a geo-distributed ML system that
  - **minimizes communication over WANs**, so that the system is not bottle-necked by the scarce WAN bandwidth and
  - is **general** enough to be applicable to a wide variety of ML algorithms, without requiring any changes to the algorithms themselves.

Distributed ML Systems

- As a distributed ML system, the state-of-the-art method is : **"Gaia: Geo-Distributed MachineLearning Approaching LAN Speeds.", Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R.Ganger, Phillip B. Gibbons, and Onur Mutlu. 2017.** [Hsieh et al. (2017)]
- The goal of Gaia is to develop a geo-distributed ML system that
  - **minimizes communication over WANs**, so that the system is not bottle-necked by the scarce WAN bandwidth and
  - is **general** enough to be applicable to a wide variety of ML algorithms, without requiring any changes to the algorithms themselves.
- To address the key challenges in designing a general and effective ML system, Gaia proposes a new model called **Approximate Synchronous Parallel (ASP)**.

Basic Significance Filter

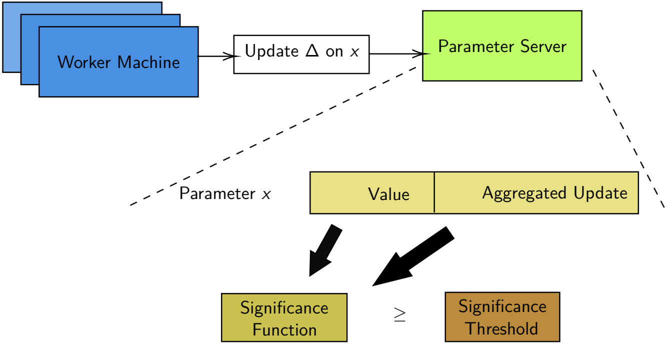


Figure: The Significance Filter [Hsieh et al. (2017)]

Gaia System

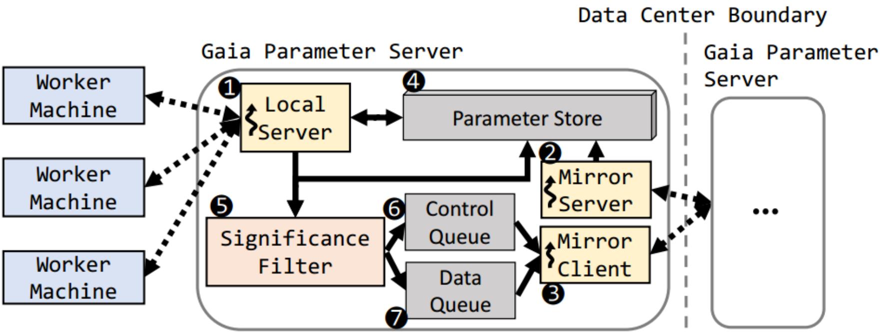


Figure: Key Component of Gaia [Hsieh et al. (2017)]

Problems of Gaia

- Exchanges parameters (weights and biases) with **full-precision**.

Problems of Gaia

- Exchanges parameters (weights and biases) with **full-precision**.
- Reducing communication using threshold value creates **inconsistency** among the data centers.

Problems of Gaia

- Exchanges parameters (weights and biases) with **full-precision**.
- Reducing communication using threshold value creates **inconsistency** among the data centers.
- Choosing **asynchronous structure** for communication over WAN can deteriorate the situation resulting in **divergence** of the model.

Gradient Sparsification

Key Ideas [Strom (2015)]

- Only send gradient larger than a predefined constant (Threshold quantization)
- Thus reduce the volume of data sent

Gradient Sparsification

Key Ideas [Strom (2015)]

- Only send gradient larger than a predefined constant (Threshold quantization)
- Thus reduce the volume of data sent

Disadvantages

- Maintaining the best possible threshold needs large amounts of experiments and largely varies from model to model, from dataset to dataset

Gradient Sparsification

Key Ideas [Dryden et al. (2016)]

- Own adaptive algorithm which uses a fixed proportion π and sends both positive and negative gradient updates maintaining that proportion

Gradient Sparsification

Key Ideas [Dryden et al. (2016)]

- Own adaptive algorithm which uses a fixed proportion π and sends both positive and negative gradient updates maintaining that proportion

Disadvantages

- Poor performance on deep learning models for using full precision parameters

Gradient Sparsification

Key Ideas [Lin et al. (2017)]

- Compressing gradients to solve bandwidth problem

Gradient Sparsification

Key Ideas [Lin et al. (2017)]

- Compressing gradients to solve bandwidth problem

Disadvantages

- Poor performance on deep learning models
- Loss in accuracy due to compression

Parameters and Gradient Quantization

Key Ideas [Seide et al. (2014)]

- 1-bit quantization which allows to significantly reduce data-exchange bandwidth for data-parallel SGD

Parameters and Gradient Quantization

Key Ideas [Seide et al. (2014)]

- 1-bit quantization which allows to significantly reduce data-exchange bandwidth for data-parallel SGD

Disadvantages

- Floating point scalar per column is required, cannot yield speed benefit on CNN
- "Cold Start" method requires first 24h of data processed without parallelism or quantization
- Significant reduction in accuracy if correct conditions are not maintained

Parameters and Gradient Quantization

- Paper: **"TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning"** [Wen et al. (2017)].

Key Ideas

- Quantize gradients to ternary levels

Parameters and Gradient Quantization

- Paper: **"TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning"** [Wen et al. (2017)].

Key Ideas

- Quantize gradients to ternary levels

Disadvantages

- Poor performance in distributed setup
- Loss in accuracy due to quantization

Parameters and Gradient Quantization

- Paper: **"Analysis of Quantized Models"** [Hou et al. (2019)].

Key Ideas

- Both weight and gradient quantization

Parameters and Gradient Quantization

- Paper: **“Analysis of Quantized Models”** [Hou et al. (2019)].

Key Ideas

- Both weight and gradient quantization

Disadvantages

- Poor performance in distributed setup
- Sacrificed upto 1.22% loss in top-1 accuracy for ImageNet dataset due to quantization
- Does not provide any synchronization model across multiple data centers

Efficient and faster communication over LAN and WAN

Efficient and faster communication over LAN and WAN

- For distributed setup, **scarce of WAN bandwidth** can significantly slowdown the **convergence** of the Deep neural networks.

Efficient and faster communication over LAN and WAN

- For distributed setup, **scarce of WAN bandwidth** can significantly slowdown the **convergence** of the Deep neural networks.
- DNN models hold **billions of parameters** and exchanging them in full precision (**hundreds to thousands mega bytes of data**) among the data centers in each epoch can dismiss the benefit of distributed setup.

Maintaining accuracy and convergence

- Though **quantization** has perceived **significant speedup**, quantizing gradients
  - slows convergence by a factor related to the **gradient quantization resolution and dimension** which contributes to fall in accuracy.

Goal

From Challenge 1

- We need a model which not only **reduces communication time and bandwidth usage efficiently**, but also provides **a proper architecture to ensure convergence**.

Goal

From Challenge 1

- We need a model which not only **reduces communication time and bandwidth usage efficiently**, but also provides **a proper architecture to ensure convergence**.

From Challenge 2

- We need to design a structure that utilizes the **speedup** achieved from the quantization and guarantees **same accuracy as the full precision** model.

Methodology

We propose **WeightGrad** that

Methodology

We propose **WeightGrad** that

- adapts both **weight and gradient quantization** to provide best speedup possible on WAN

Methodology

We propose **WeightGrad** that

- adapts both **weight and gradient quantization** to provide best speedup possible on WAN
- proposes a **synchronous structure** to prevent the loss in accuracy due to quantization

WeightGrad System

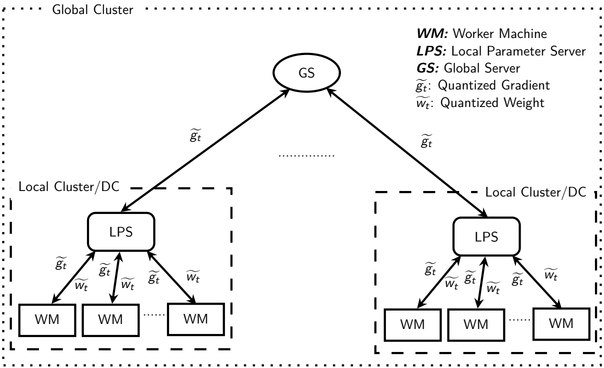


Figure: WeightGrad Tree Structure

WeightGrad System

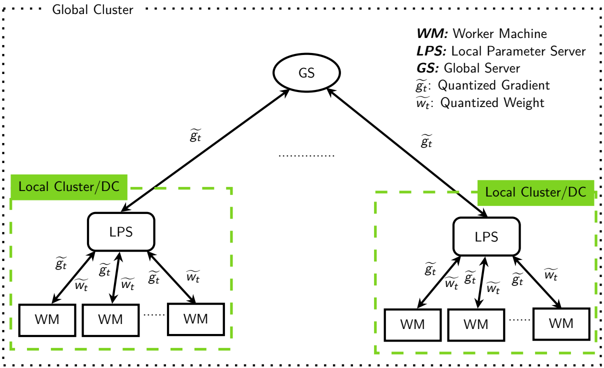


Figure: WeightGrad Tree Structure

WeightGrad System

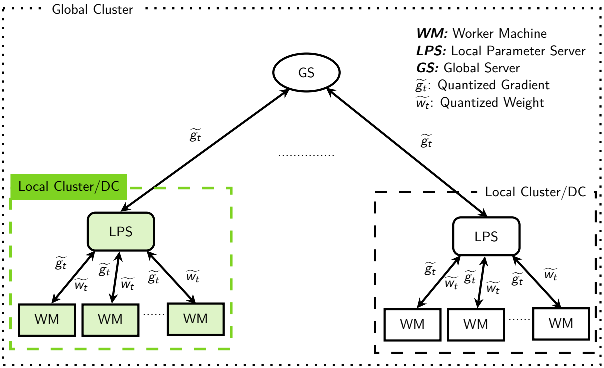


Figure: WeightGrad Tree Structure

WeightGrad System

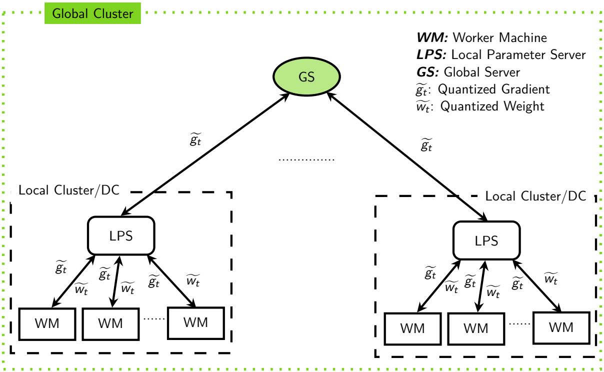


Figure: WeightGrad Tree Structure

WeightGrad System

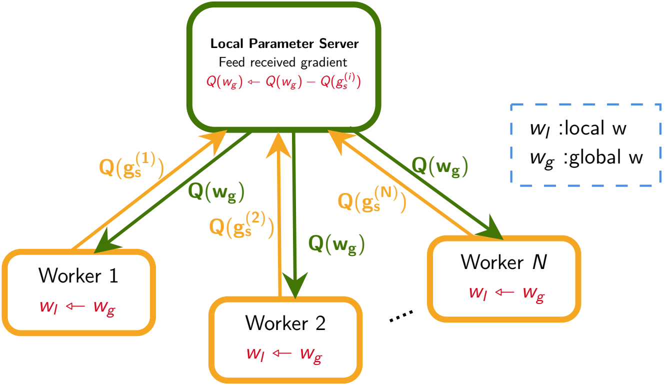


Figure: WeightGrad: Local Cluster

WeightGrad System

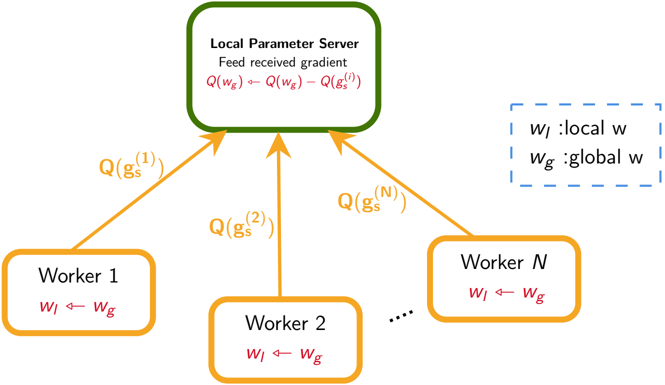


Figure: WeightGrad: Local Cluster

WeightGrad System

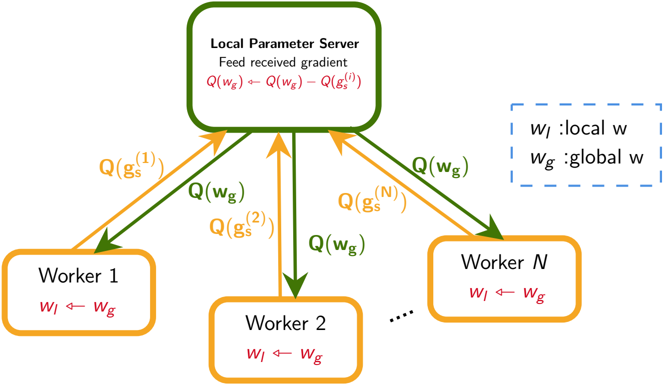


Figure: WeightGrad: Local Cluster

LPS with Gradient Synchronizer

LPS with Gradient Synchronizer

Dynamic Threshold

- Gradient Synchronizer maintains a dynamic threshold for distributing the gradient updates.

LPS with Gradient Synchronizer

Dynamic Threshold

- Gradient Synchronizer maintains a dynamic threshold for distributing the gradient updates.

Fixed Interval

- To synchronize the communication process, Gradient Synchronizer maintains a fixed interval T, within which it receives aggregated gradient values from the GS.
- If an LPS does not get update from the GS within T, it stops sending updates to the WMs until gradient updates are received from the GS.

LPS with Gradient Synchronizer

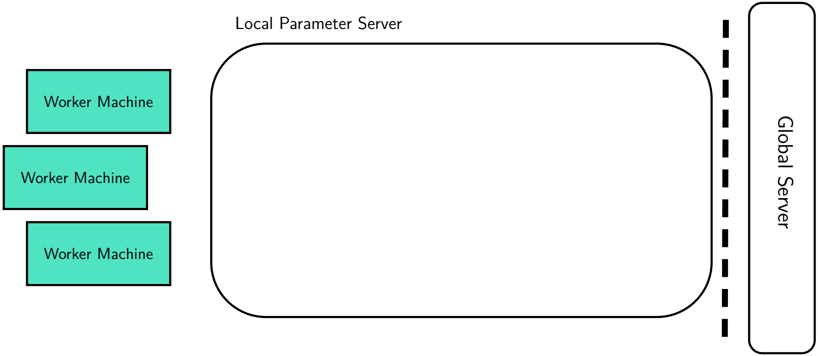


Figure: LPS Structure

LPS with Gradient Synchronizer

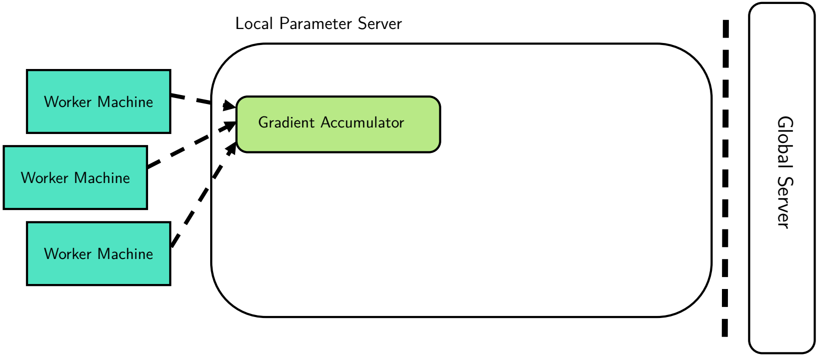


Figure: LPS Structure

LPS with Gradient Synchronizer

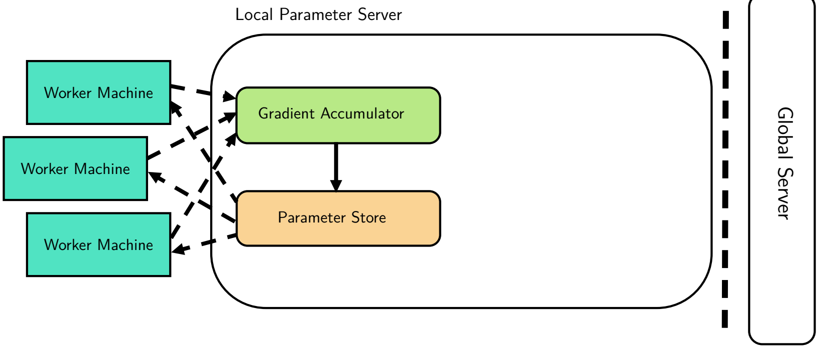


Figure: LPS Structure

LPS with Gradient Synchronizer

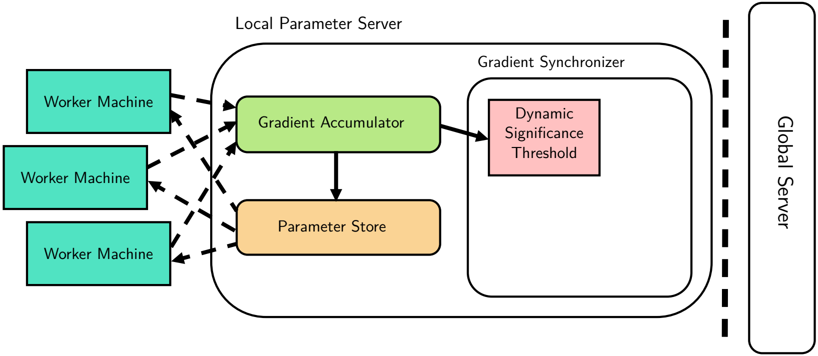


Figure: LPS Structure

LPS with Gradient Synchronizer

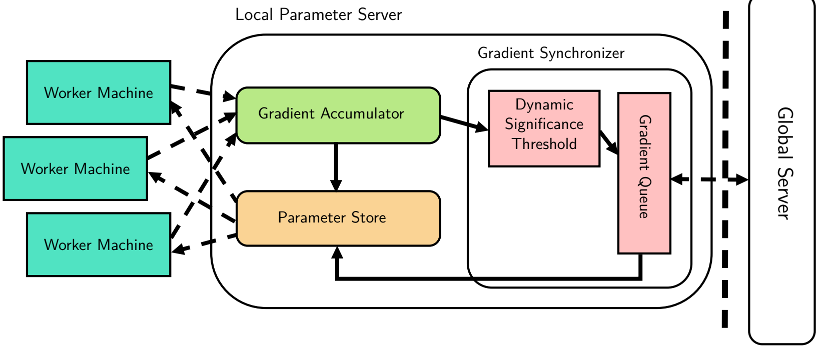


Figure: LPS Structure

LPS with Gradient Synchronizer

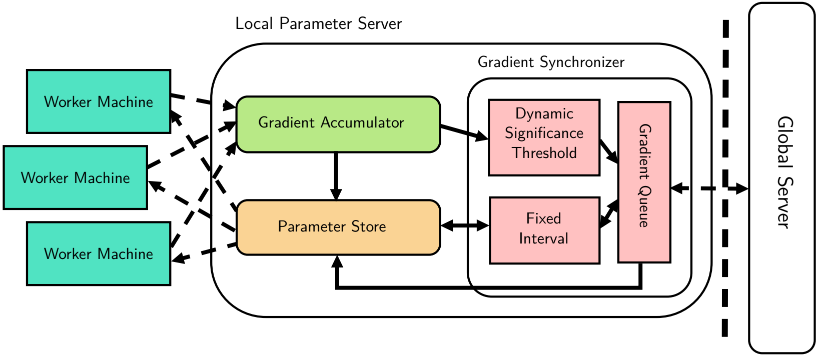


Figure: LPS Structure

Two Level Structure

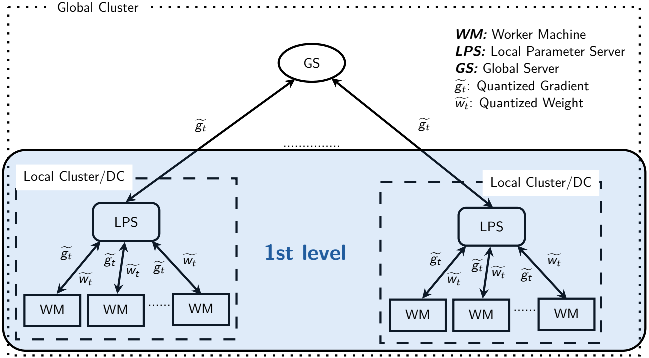


Figure: WeightGrad Tree Structure

Two Level Structure

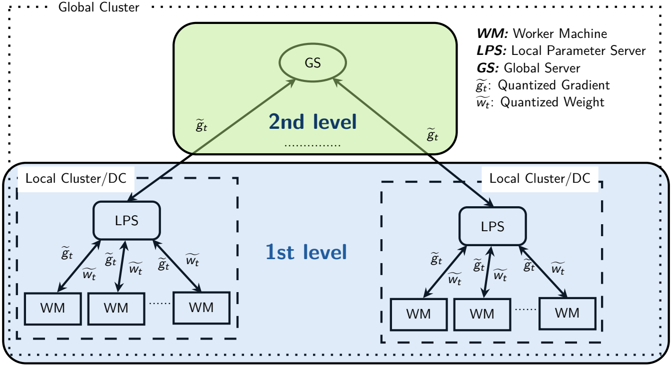


Figure: WeightGrad Tree Structure

Basic Algorithm

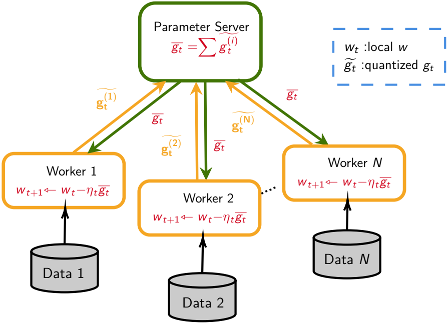


Figure: Traditional Worker-Parameter Server

- Send quantized gradients in each epoch to PS
- Workers receive average quantized gradient

WeightGrad

Worker Machine

- $\widetilde{g_s^{(i)}} = \mathit{ComputeTernarizedGradient}\left(z_s^{(i)}, W_s^{(i)}, {}_s^{(i)}\right)$
- $\mathit{SendToLPS}\left(\widetilde{g_s^{(i)}}\right)$
- $W_{s+1}^{(i)} = \mathit{ReceiveParameters}()$

Local Parameter Server

- $\overline{g_s} = \frac{1}{N}\sum_{i=1}^{N}\widetilde{g_s^{(i)}}$ N: Number of Workers
- $g_u = \mathit{AggregateGradient}\left(\overline{g_s}\right)$
- **if** $\left(\frac{g_u}{g_s} > G_{thresh}\right)$ **then** $\mathit{SendGradientToGS}(g_u)$
- $\mathit{PullGradWithinTandUpdateParameter}\left(\overline{g_u}\right)$

Global Server

- $\overline{g_u} = \frac{1}{N}\sum_{i=1}^{N} g_u^{(i)}$

  N: Number of LPS
- $\mathit{SendToAllLPS}\left(\overline{g_u}\right)$

WeightGrad Tuning

Worker Machine

- $\widetilde{g_s^{(i)}} = ComputeTernarizedGradient\left(z_s^{(i)}, W_s^{(i)}, ,_s^{(i)}\right)$
- $SendToLPS\left(\widetilde{g_s^{(i)}}\right)$
- $W_{s+1}^{(i)} = ReceiveParameters()$

Local Parameter Server

N: Number of Workers

- **for** $i \in N$
  $UpdateParameters\left(\widetilde{g_s^{(i)}}\right)$
  $g_u = AggregateGradient\,(\overline{g_s})$
- $g_{max} = MaxGradient\,(\widetilde{g_s})$
- **if** $\left(\frac{g_u}{g_{max}} > G_{thresh}\right)$ **then** $SendGradientToGS(g_u)$
- $PullGradWithinTandUpdateParameter\,(\overline{g_u})$
- **if** $(\overline{g_u} > g_{max})$ **then** $UpdateParameters\,(\overline{g_u} - g_{max})$

Global Server

- $\overline{g_u} = \frac{1}{N}\sum_{i=1}^{N} g_u^{(i)}$
  N: Number of LPS
- $SendToAllLPS\,(\overline{g_u})$

Quantized Network Accuracy

To reduce the fall of convergence rate due to gradient quantization

- We quantize weights and gradients to **ternary levels** {-1, 0, 1} proposed in TernGrad [Wen et al. (2017)].

Quantized Network Accuracy

To reduce the fall of convergence rate due to gradient quantization

- We quantize weights and gradients to **ternary levels** {-1, 0, 1} proposed in TernGrad [Wen et al. (2017)].
- We use a **layer-wise quantization** scheme to reduce the performance degradation.

Quantized Network Accuracy

To reduce the fall of convergence rate due to gradient quantization

- We quantize weights and gradients to **ternary levels** {-1, 0, 1} proposed in TernGrad [Wen et al. (2017)].
- We use a **layer-wise quantization** scheme to reduce the performance degradation.
- A **smaller learning rate** is maintained to preserve the good performance achieved from quantization [Wu et al. (2018)].

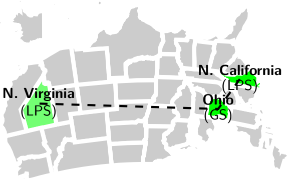
Quantized Network Accuracy

To reduce the fall of convergence rate due to gradient quantization

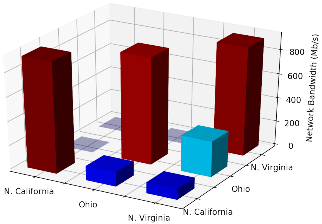
- We quantize weights and gradients to **ternary levels** {-1, 0, 1} proposed in TernGrad [Wen et al. (2017)].
- We use a **layer-wise quantization** scheme to reduce the performance degradation.
- A **smaller learning rate** is maintained to preserve the good performance achieved from quantization [Wu et al. (2018)].

We observe that **gradient clipping with momentum correction** and **layer-wise ternarizing** gives the best convergence rate among the state-of-the-art systems.

Amazon EC-2



(a)



(b)

Figure: (a) Deployment Regions in AWS, (b) Measured network bandwidth between Amazon EC2 sites

Amazon EC-2

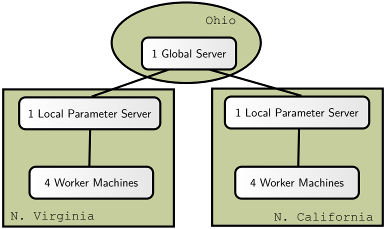
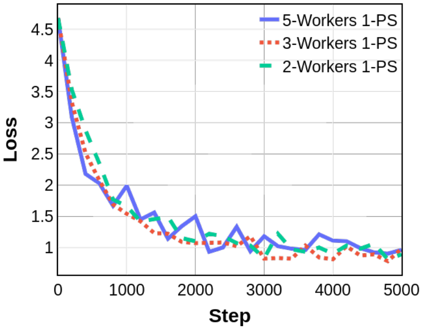


Figure: Instance Hierarchy

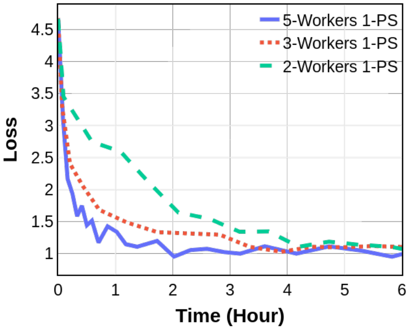
Instances	Instance Type	RAM	vCPU	GPU	B/W
11	g3s.xlarge, 64-bit Ubuntu Server 16.04 LTS	30.5 GiB	4	NVIDIA Tesla M60 GPU	10 Gbps

Table: Instance Configuration

Convergence Analysis



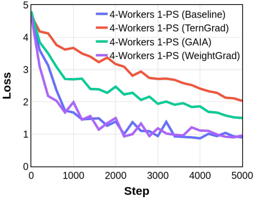
(a) Loss vs Step



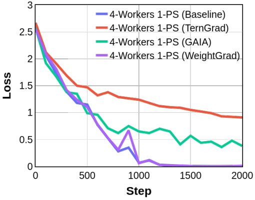
(b) Loss vs Time (hour)

Figure: Convergence of WeightGrad for CIFAR10 dataset

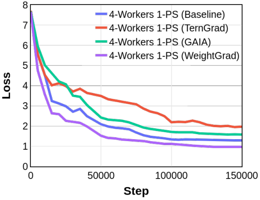
Training Loss Analysis



(a) Training Loss for CifarNet



(b) Training Loss for VGGNet



(c) Training Loss for ImageNet

Figure: (a) Training loss for CifarNet model on CIFAR-10 dataset, (b) Training loss for VGGNet model on CIFAR-10 dataset, (c) Training loss for AlexNet on ImageNet dataset

SpeedUp Analysis

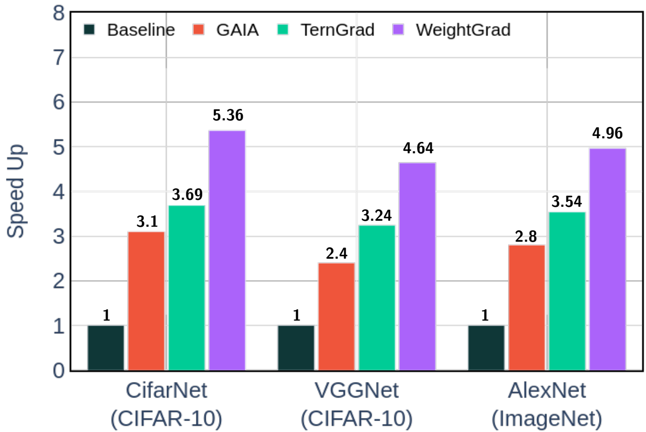


Figure: Training Speed Comparison

SpeedUp Analysis

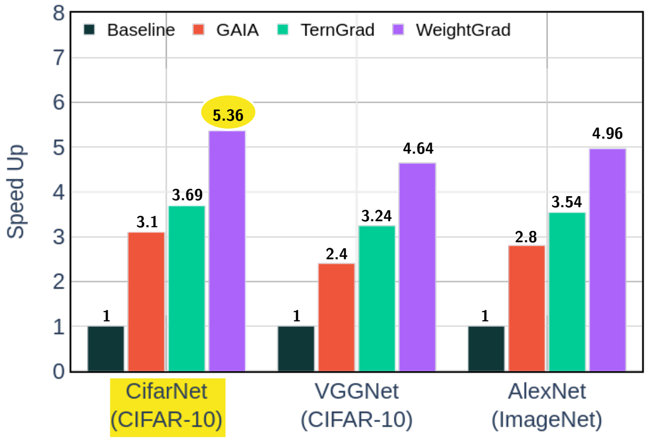


Figure: Training Speed Comparison

SpeedUp Analysis

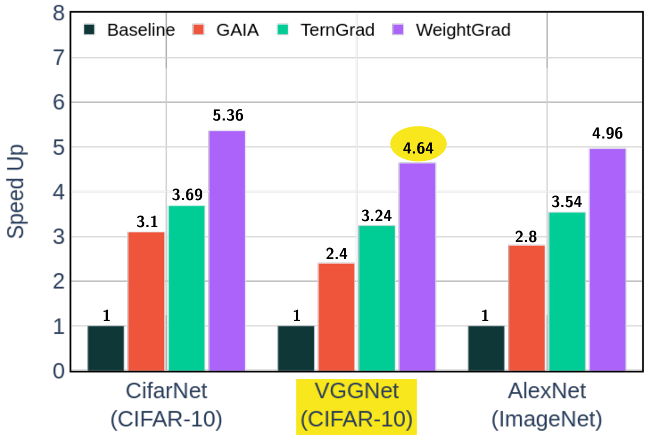


Figure: Training Speed Comparison

SpeedUp Analysis

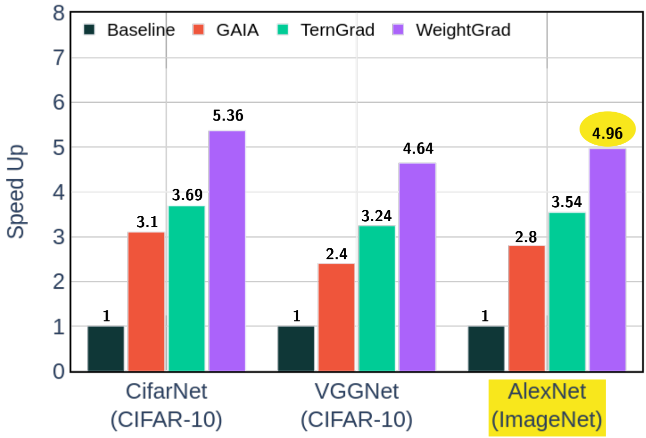


Figure: Training Speed Comparison

Accuracy Comparison: CIFAR-10

Model	SGD	Base LR	Total mini-batch size	Steps	Gradients	Workers	Accuracy
CifarNet	GD	0.1	128	50k	Baseline	4	84.56%
					Gaia	4	83.48%(-1.08%)
					TernGrad	4	82.41%(-2.15%)
					WeightGrad	**4**	**84.56%(-0.00%)**
	GD	0.1	512	50k	Baseline	8	83.19%
					Gaia	8	83.04%(-0.13%)
					TernGrad	8	81.40%(-1.79%)
					WeightGrad	**8**	**83.21%(+0.03%)**
VGG-Net	GD	0.1	512	50k	Baseline	8	88.14%
					Gaia	8	87.19%(-0.95%)
					TernGrad	8	86.3%(-1.84%)
					WeightGrad	**8**	**88.13%(-0.01%)**

Table: Result of WeightGrad on CIFAR-10 dataset

Accuracy Comparison: ImageNet

Model	Steps	Training Method	Top-1 Accuracy	Top-5 Accuracy
AlexNet	185k	Baseline	58.17%	80.19%
		Gaia	58.02%(-0.15%)	80.20%(+0.01%)
		TernGrad	57.32%(-0.85%)	80.18%(-0.01%)
		Deep Gradient Compression	58.20%(+0.03%)	80.20%(+0.01%)
		WeightGrad	**59.28%(+1.06%)**	**80.25%(+0.06)**

Table: Comparison of training methods on ImageNet data

Paper also includes

- Convergence proof for quantized weight and quantized gradient architecture

Paper also includes

- Convergence proof for quantized weight and quantized gradient architecture
- Dynamic significance threshold calculation

Paper also includes

- Convergence proof for quantized weight and quantized gradient architecture
- Dynamic significance threshold calculation
- Analyzing challenges and maximum worker machine for a single LPS

Conclusions

Conclusions

- **Problem**: How to perform DNN on geo-distributed data?

Conclusions

- **Problem**: How to perform DNN on geo-distributed data?
  - Limited WAN bandwidth and slower convergence rate
  - DNN models contain billions of parameters

Conclusions

- **Problem**: How to perform DNN on geo-distributed data?
  - Limited WAN bandwidth and slower convergence rate
  - DNN models contain billions of parameters
- We introduce **WeightGrad**,
  - **A loss-aware weight-quantized deep learning system with quantized gradient**

Conclusions

- **Problem**: How to perform DNN on geo-distributed data?
  - Limited WAN bandwidth and slower convergence rate
  - DNN models contain billions of parameters
- We introduce **WeightGrad**,
  - **A loss-aware weight-quantized deep learning system with quantized gradient**
  - Maintains a **tight synchronization over LAN** to simulate each data center as a centralized system and a **loose synchronization over WAN** to reduce communication cost.

Conclusions

- **Problem**: How to perform DNN on geo-distributed data?
  - Limited WAN bandwidth and slower convergence rate
  - DNN models contain billions of parameters
- We introduce **WeightGrad**,
  - **A loss-aware weight-quantized deep learning system with quantized gradient**
  - Maintains a **tight synchronization over LAN** to simulate each data center as a centralized system and a **loose synchronization over WAN** to reduce communication cost.
- **Results:** WeightGrad achieves,
  - **2.4-5.36× speedup** over state-of-the-art distributed systems

Conclusions

- **Problem**: How to perform DNN on geo-distributed data?
  - Limited WAN bandwidth and slower convergence rate
  - DNN models contain billions of parameters
- We introduce **WeightGrad**,
  - **A loss-aware weight-quantized deep learning system with quantized gradient**
  - Maintains a **tight synchronization over LAN** to simulate each data center as a centralized system and a **loose synchronization over WAN** to reduce communication cost.
- **Results:** WeightGrad achieves,
  - **2.4-5.36**$\times$ **speedup** over state-of-the-art distributed systems
  - **0.03-1.06% accuracy gain** over Baseline

Future Works

- Integrating XNOR-Net [Rastegari et al. (2016)] with WeightGrad back-propagation phase in parameter server.
  - XNOR-Net gives **58**$\times$ faster convolutional operations and **32**$\times$ memory savings.

THANK YOU!

Syeda Nahida Akter and Muhammad Abdullah Adnan, "WeightGrad: Geo-Distributed Data Analysis Using Quantization for Faster Convergence and Better Accuracy," Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD 2020), San Diego, CA, USA, August 23-27, 2020.