# ALADDIN: Asymmetric Centralized Training for Distributed Deep Learning

CIKM 2021
1-5 NOVEMBER


Yunyong Ko[1], Kibong Choi[1], Hyunseung Jei[2], Dongwon Lee[3], and Sang-Wook Kim[1]

Hanyang University, Republic of Korea[1]

SK Telecom, Republic of Korea[2]

The Pennsylvania State University, PA, USA[3]

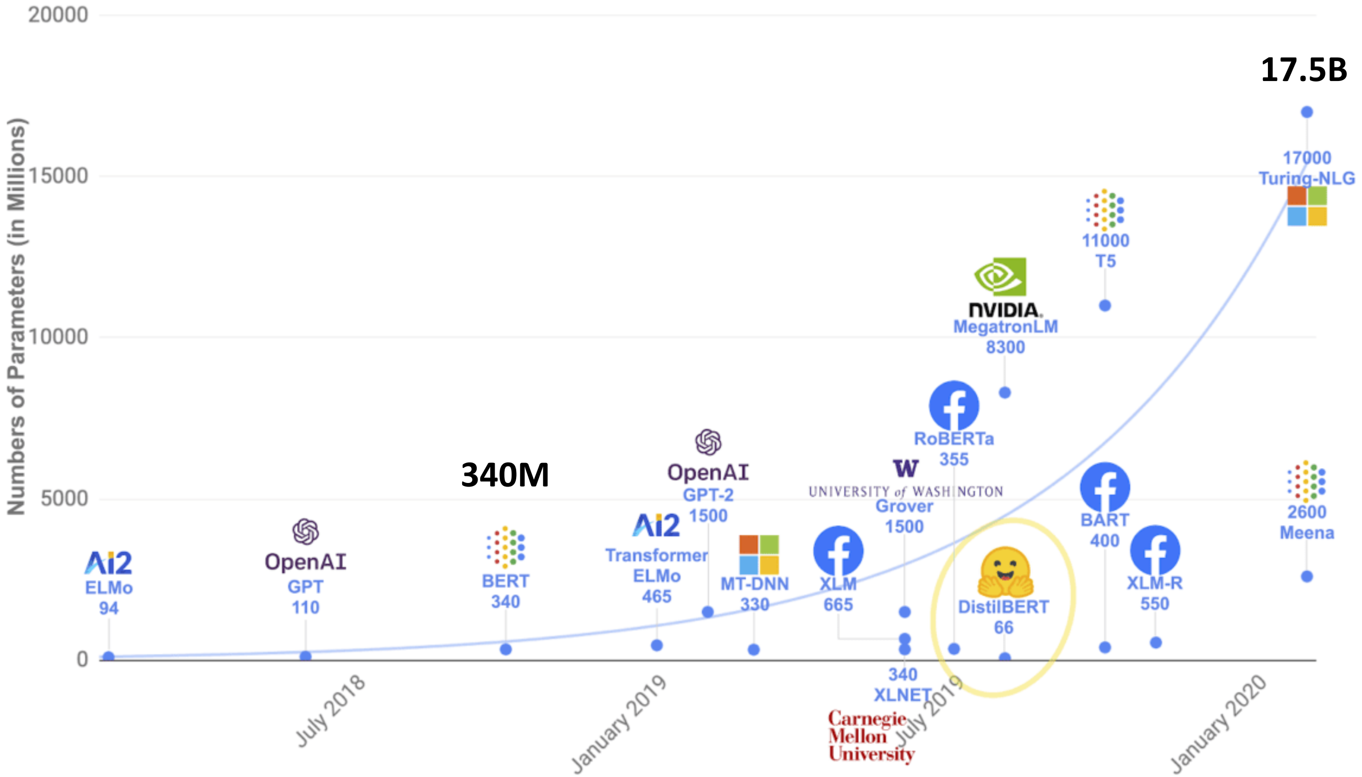
# Table of Contents

- **Background**
  - Distributed deep learning
  - Centralized and decentralized training
- **Proposed algorithm: ALADDIN**
  - Motivation and key idea
  - Algorithm details
  - Convergence analysis
- **Experiments**
- **Conclusions**

# Challenge of Deep Learning

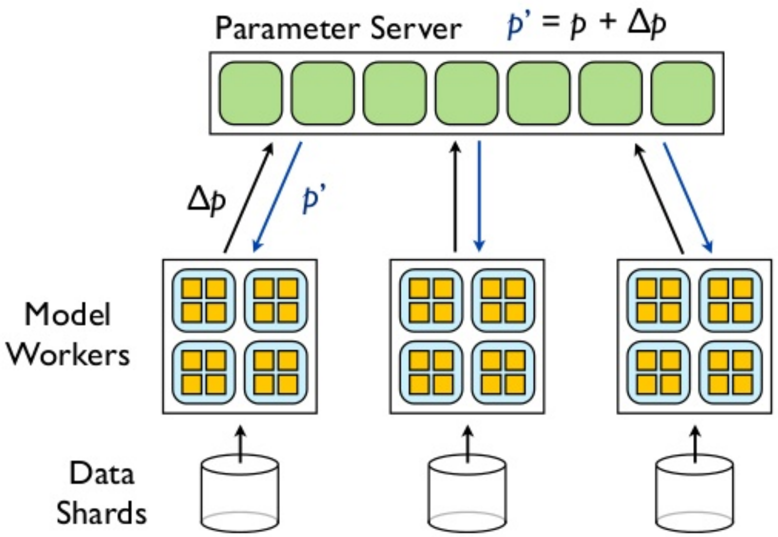
## ☐ Training of DNNs *requires massive time*

- The increasing sizes of DNN models and training datasets
  1) Increasing computation overhead (forward, backward, and update)
  2) Increasing the number of training iterations

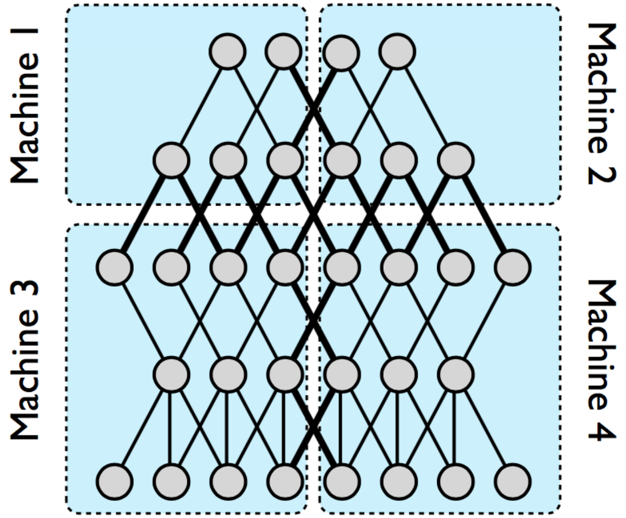



# Distributed Deep Learning

- ***Data parallelism (our focus)***
  - Splitting and distributing ***training data*** into multiple workers
- **Model parallelism**
  - Splitting and distributing ***a model*** into multiple workers




**<Data parallelism>**

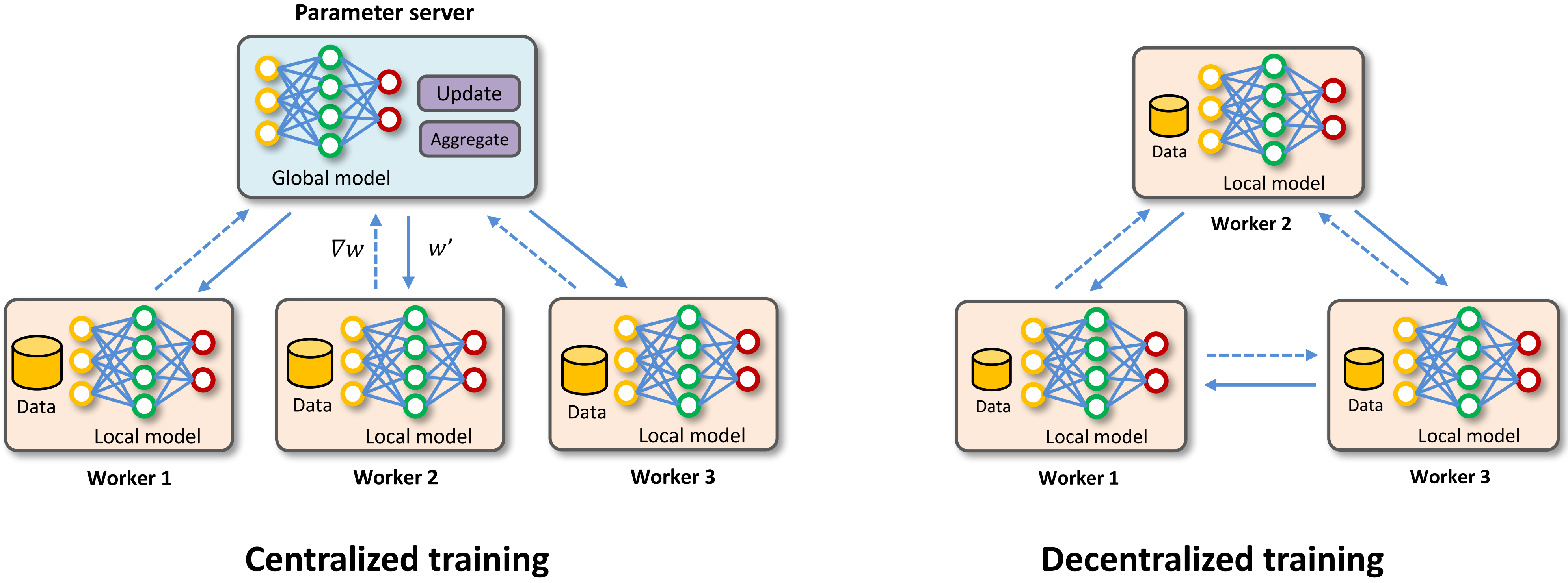



**<Model parallelism>**

# Data Parallelism Approach

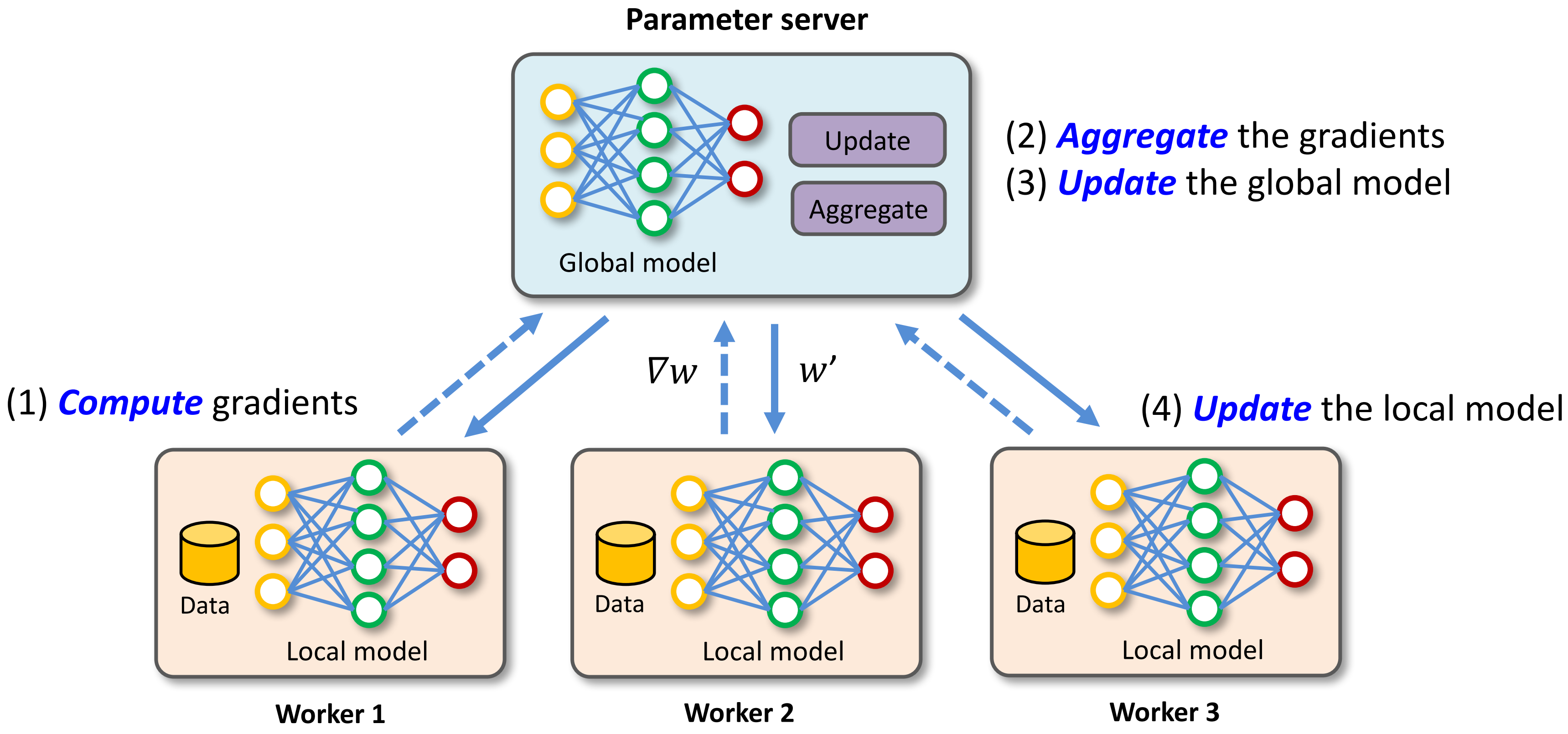
## □ Splitting and distributing ***training data*** into multiple workers

- Each worker trains a model based on its local data ***in parallel***
- Then, the training results are aggregated via communication



# Centralized Training

- ***The parameter server (PS)***, **managing the global model, aggregates the training results from workers,**



# Deficiency of Centralized Training

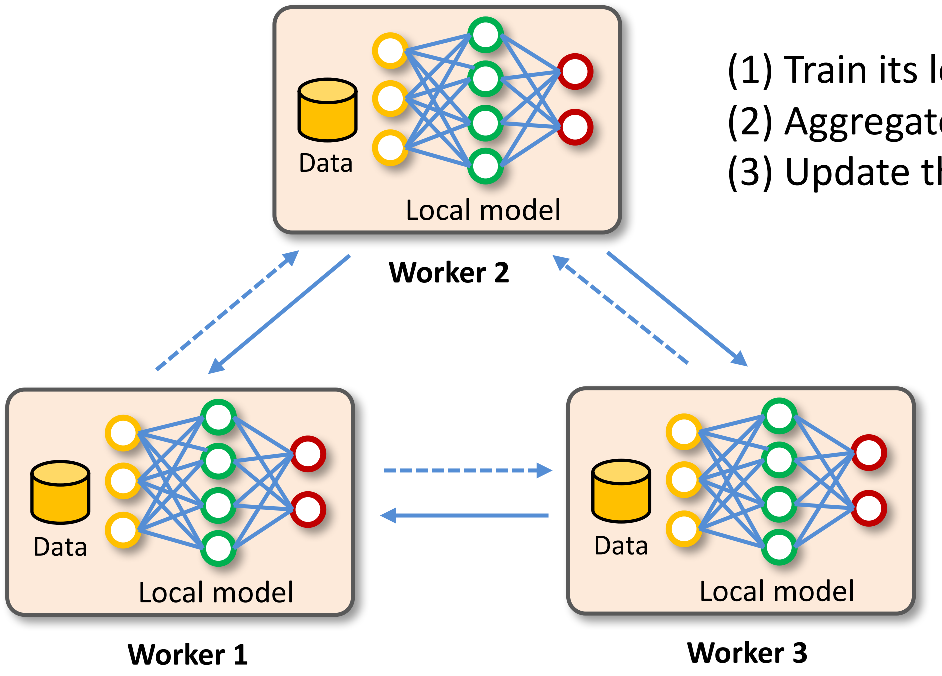
## ☐ *Large* communication overhead

- The PS can be a bottleneck of the whole training
- More than 60% of the entire training (# of workers >= 4)


**<Communication overhead w.r.t the number of workers>**
(VGG-16 training)

# Decentralized Training

- **The training results of workers are aggregated via *peer-to-peer communication***
  - To avoid the problem of *PS being bottleneck*

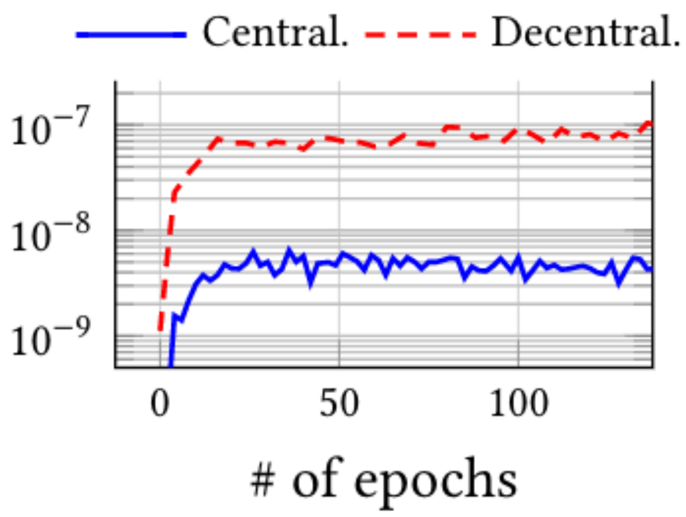



(1) Train its local model
(2) Aggregate the results via ***p2p communication***
(3) Update the local model

# Deficiency of Decentralized Training

## ☐ *High* parameter variance

- To sufficiently aggregate workers' results via p2p communication is difficult




**<Parameter variance as the training progress>**

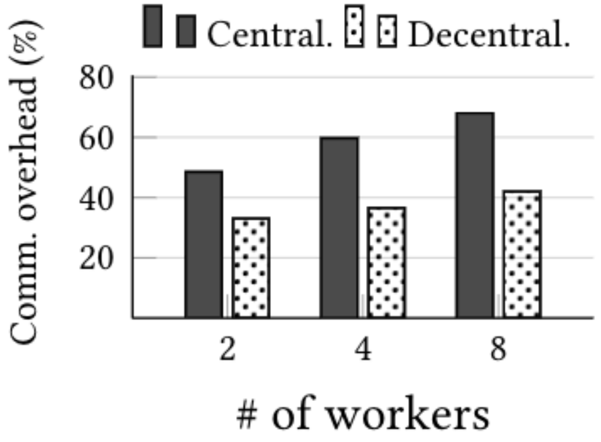
# Summary of Centralized and Decentralized Training

## ☐ Centralized training

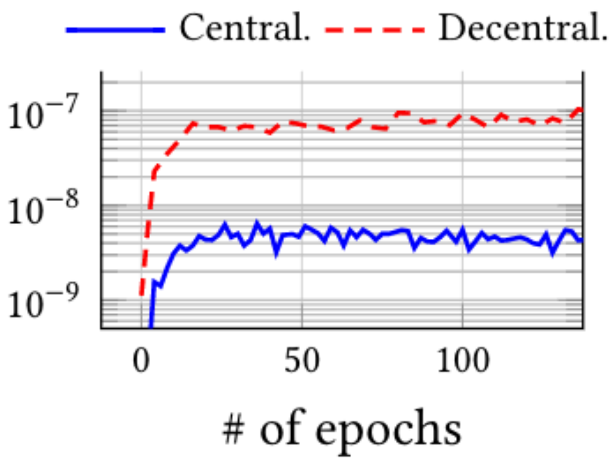
- Large communication overhead
- Low parameter variance among workers

## ☐ Decentralized training

- Small communication overhead
- High parameter variance among workers



(a) Communication overhead

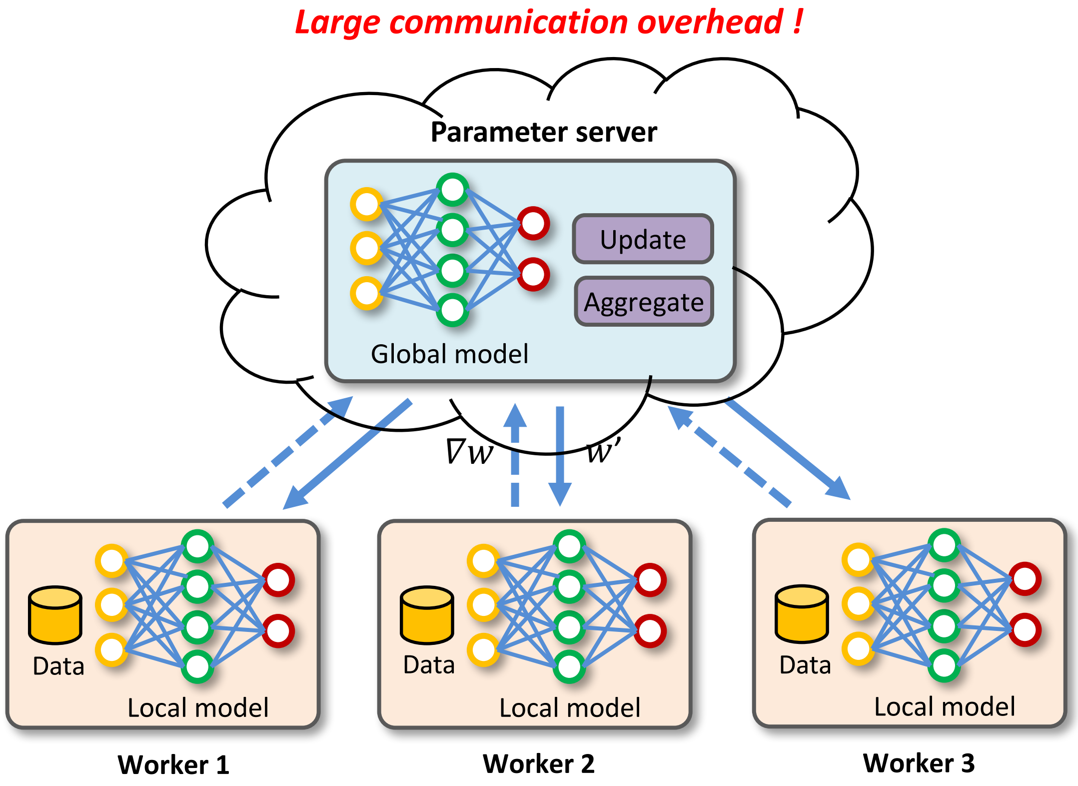


(b) Parameter variance

# Our Research Direction

## ☐ To improve the performance of centralized training

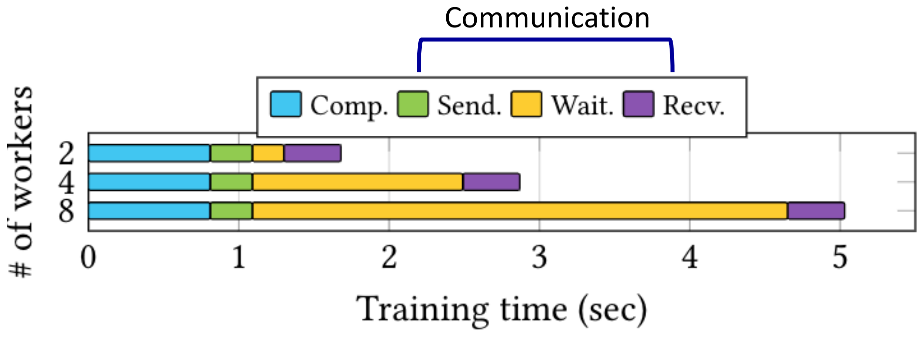
- By addressing the problem of large communication overhead



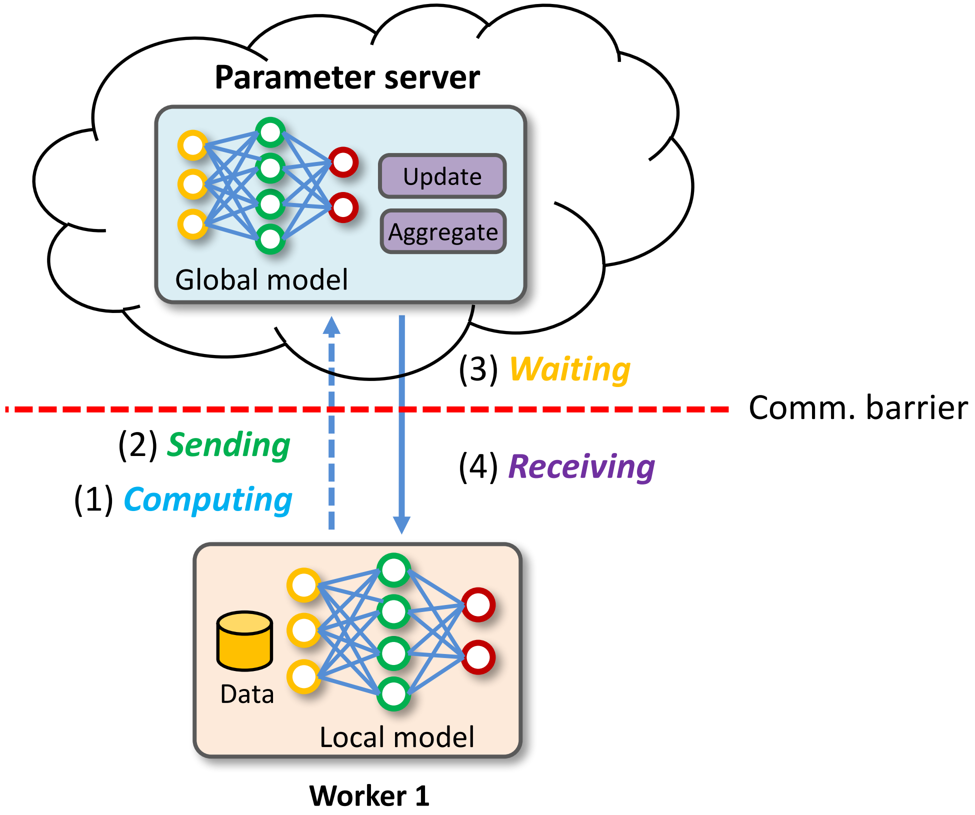
# Motivation: Symmetric Communication

## The cause of performance degradation in centralized training

- Each worker ***symmetrically*** waits for the updated model from PS
- The PS bottleneck increases the ***waiting*** at communication barrier




**<Breakdown of training time>**
(VGG-16 training)



# Key Idea: Asymmetric Training

## ☐ Goal

- To reduce the idle time of workers, *symmetrically* waiting for PS

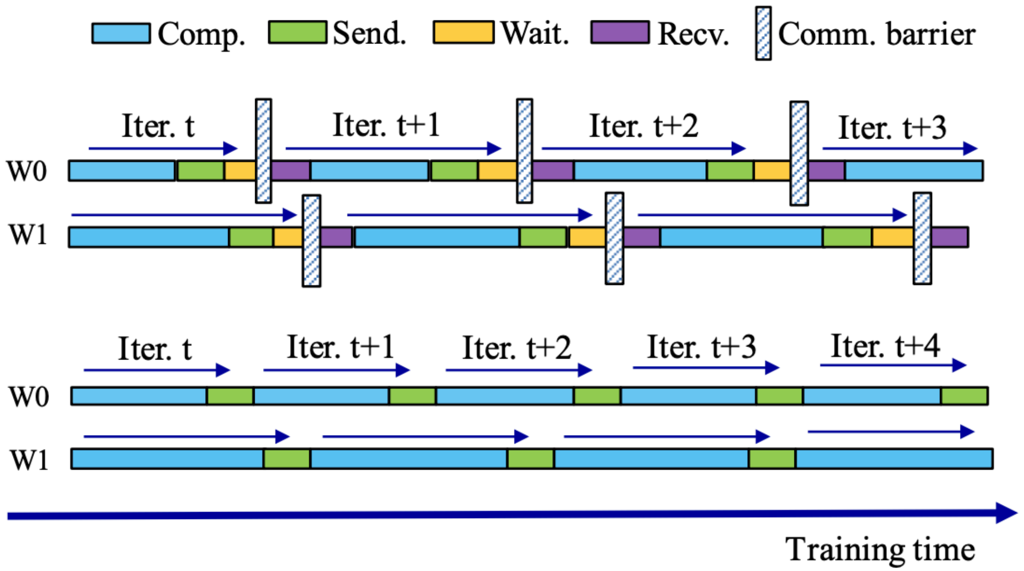
## ☐ Asymmetric training between PS and workers

- A worker sends gradients to PS, and then ***immediately proceeds*** to the next step

Symmetric training

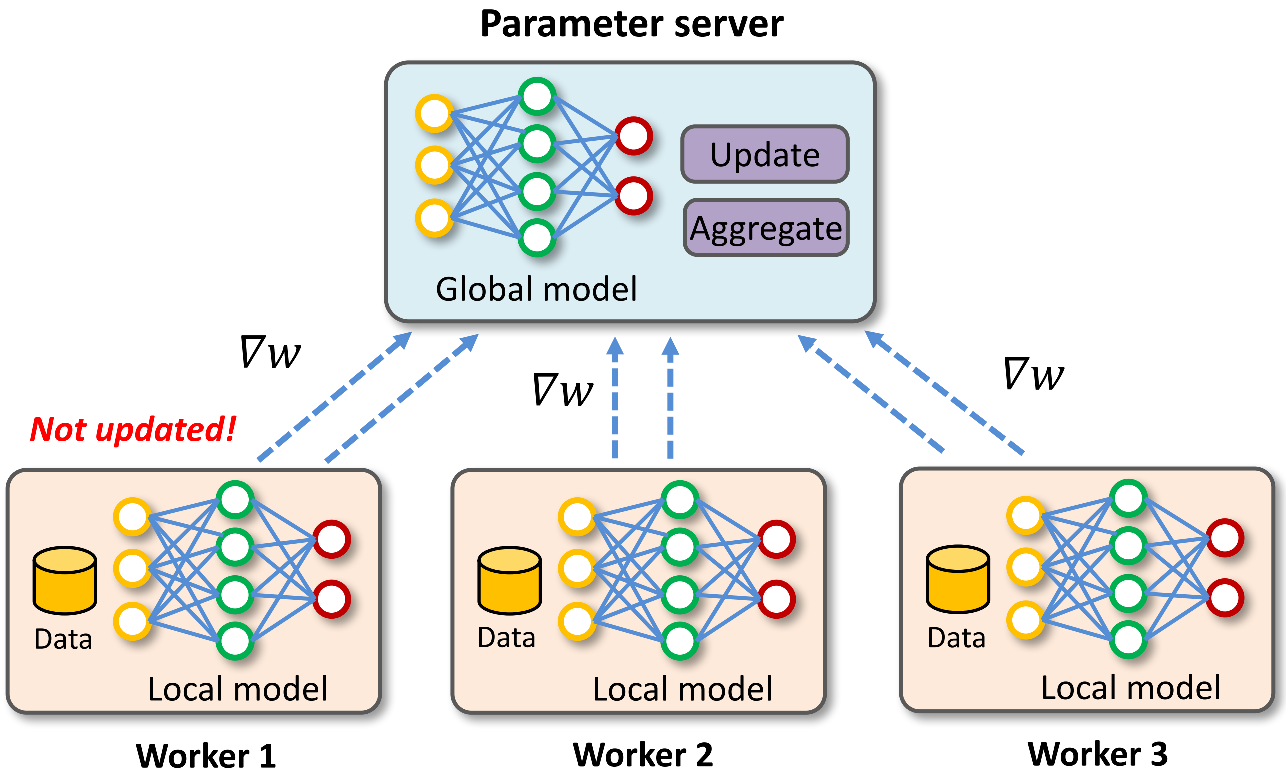
**VS**

***Asymmetric training***



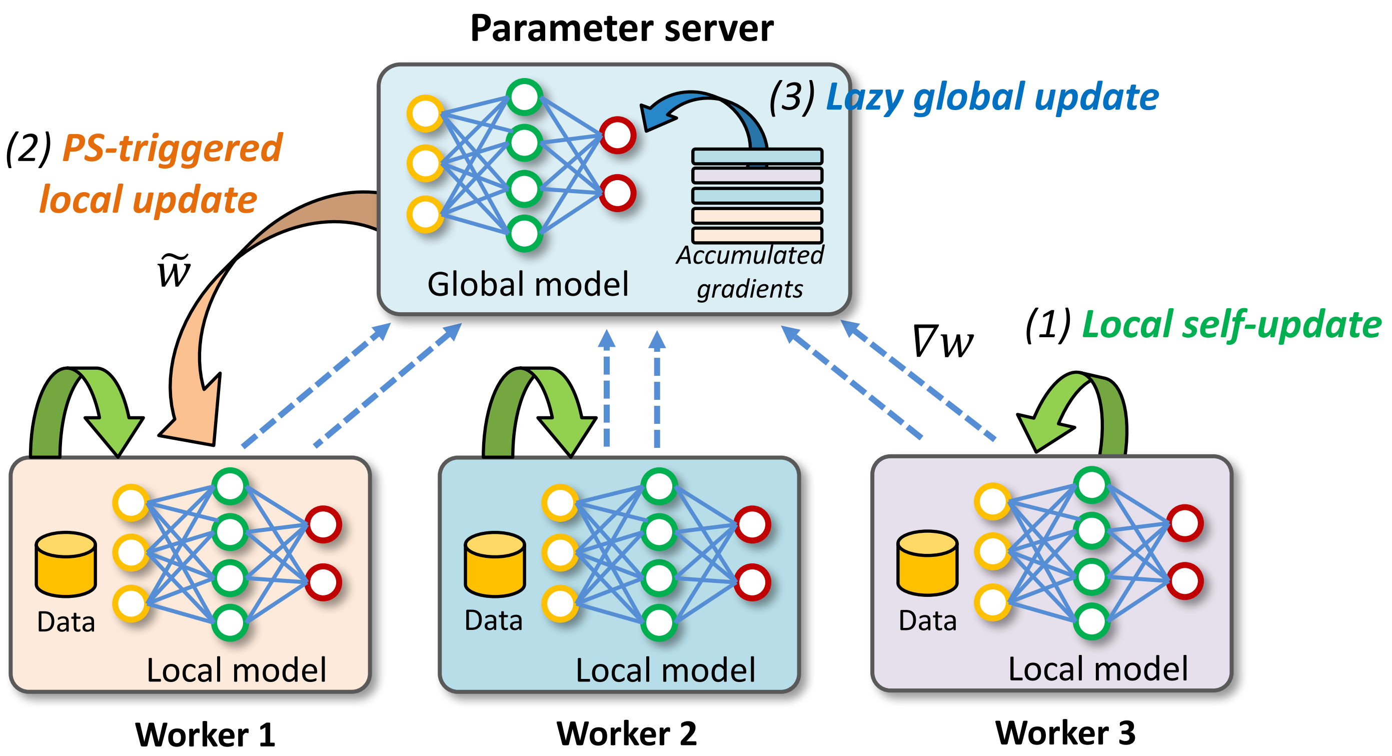
# Limitation of Naïve Asymmetric Training

- **The local model of each worker is *not updated***
  - Each worker *never receive the global model*
  - Degrading the quality of gradients computed at each worker



# Three Update Strategies of ALADDIN

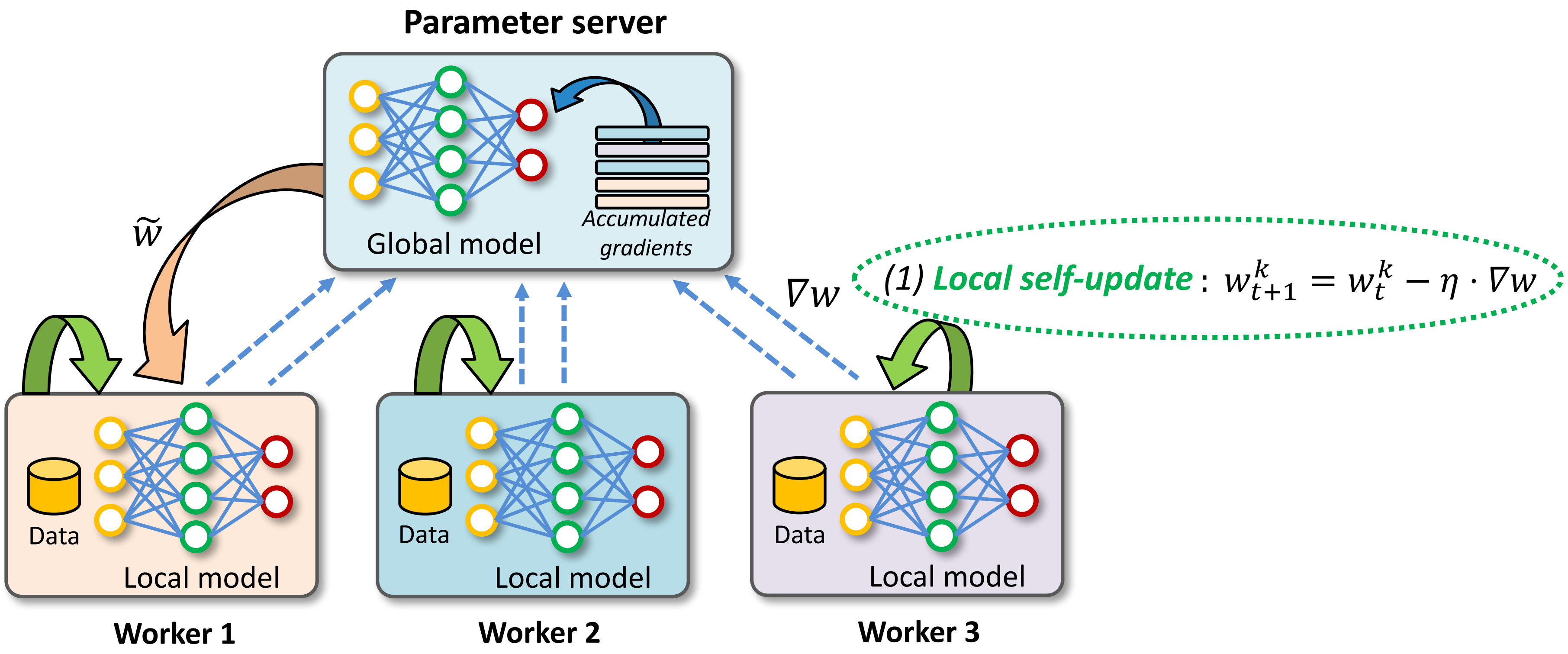
- To speed up the model convergence
  - (1) ***Local self-update***, (2) ***PS-triggered local update,*** and (3) ***Lazy global update***



# Strategy 1: Local Self-Update

## ☐ Each worker applies the computed gradients to its local model

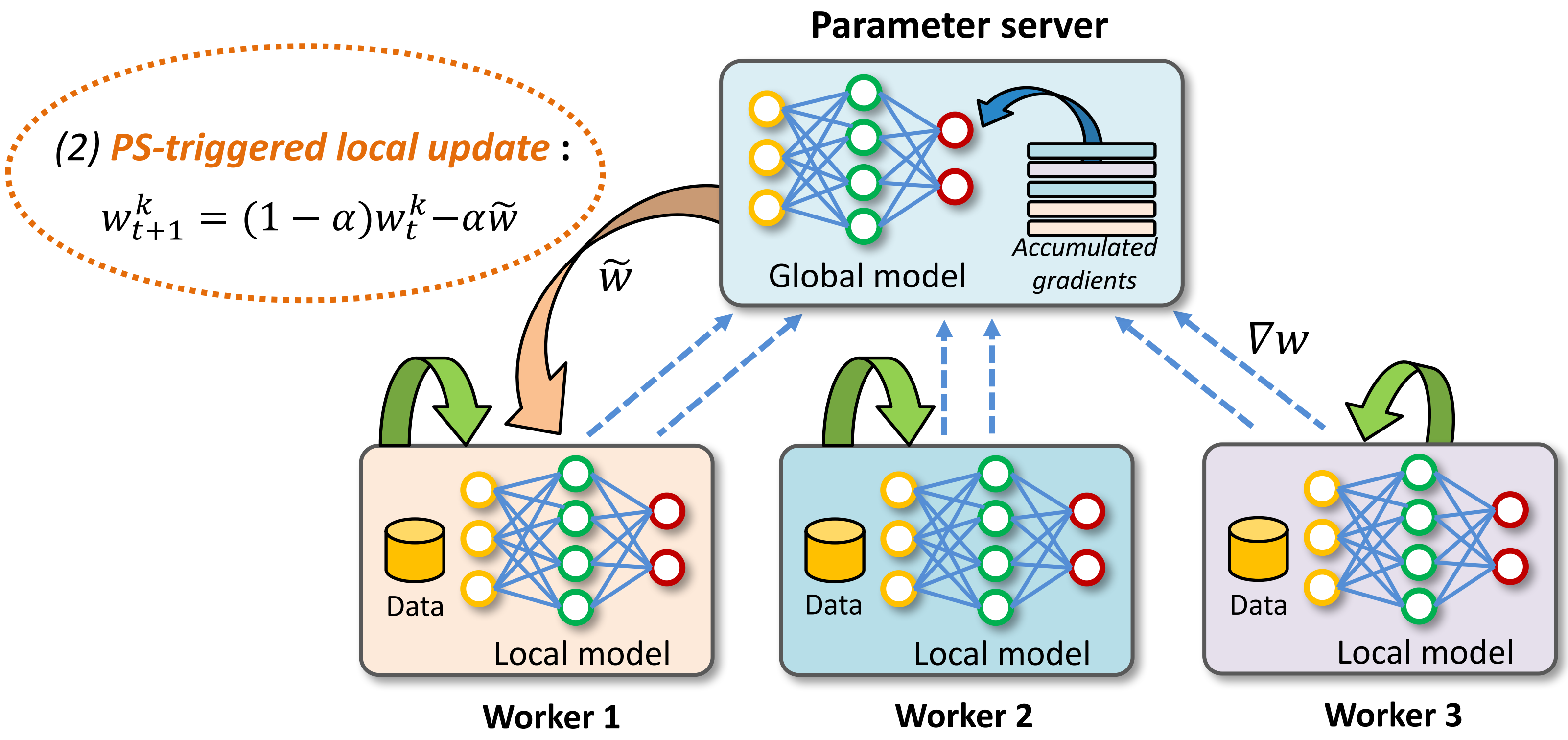
- Improving the quality of gradients computed at each worker



# Strategy 2: PS-Triggered Local Update

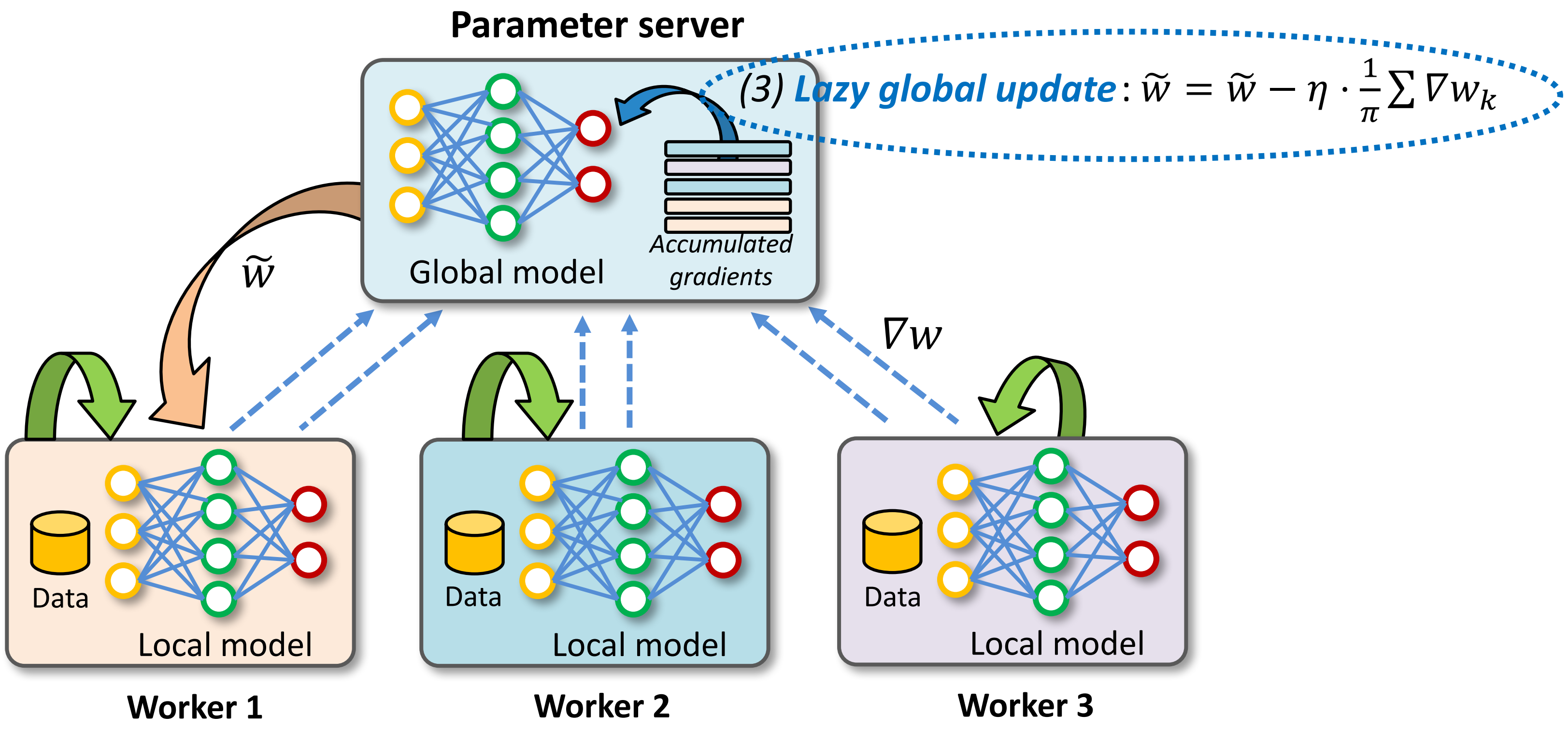
## ☐ PS sends the up-to-date global model to each worker periodically

- The local models of workers are mixed indirectly, reducing parameter variance



# Strategy 3: Lazy Global Update

- **PS aggregates the gradients, and updates the global model *lazily***
  - Reducing the number of operations without any loss (e.g., *apply_gradients()*)



# Algorithms: ALADDIN

## ☐ The entire training process

**Algorithm 1** Training of worker $i$ in ALADDIN

**Require:** Initialize $x_0^i$ for worker $i \in \{1, 2, \ldots, n\}$, dataset $\xi^i$, batch size $B$, learning rate $\eta$, weight factor $\alpha$

**for** $t = 0, 1, \ldots$ **do**
- $g_t^i \leftarrow \frac{1}{B}\sum_{j=1}^{B} \nabla F(x^i, \xi_{t,j}^i)$
- $x_{t+1}^i \leftarrow x_t^i - \eta \cdot g_t^i$ *(Local self-update)*
- Send $g_t^i$ to PS
- **if** $\tilde{x}$ **from PS**
  - $x^i \leftarrow (1-\alpha)x^i + \alpha\tilde{x}$ *(PS-triggered local update)*
- **end if**

**end for**

**Algorithm 2** Training of PS in ALADDIN

**Require:** Global model $\tilde{x}_0$, learning rate $\eta$, update period $\pi$, period count $c_i$ for workers

$\tilde{c} \leftarrow 0, c_i \leftarrow 0$

**for** $t = 0, 1, \ldots$ **do**
- Receive $g^i$ from worker $i$
- $\tilde{c} \leftarrow \tilde{c} + 1, c_i \leftarrow c_i + 1$
- **if** $c_i < \pi$
  - $\tilde{g} \leftarrow \tilde{g} + g^i$ *(Gradient accumulation)*
- **else**
  - $\tilde{x} \leftarrow \tilde{x} - \eta \cdot \tilde{g},\ \tilde{g} \leftarrow \frac{1}{\tilde{c}} \cdot \tilde{g}$ *(Lazy global update)*
  - Send $\tilde{x}$ to worker $i$ *(PS-triggered local update)*
  - $\tilde{g} \leftarrow \emptyset, \tilde{c} \leftarrow 0, c_i \leftarrow 0$
- **end if**

**end for**

# Convergence Analysis for ALADDIN

- **Requirements for the convergence on non-convex optimization**
  - The mixing matrix of a distributed algorithm has to satisfies the two conditions:
    - ***Doubly-stochastic*** and ***spectral gap*** conditions

- **By lemmas 1 and 2, we show that the mixing matrix of ALADDIN satisfies the two conditions**

**Lemma 1** *Let $x_k^i$ and $\tilde{x}_k$ be the local model of worker i and the global model at iteration k in ALADDIN, respectively. Then, the averaged model of all workers is equivalent to the global model.*

$$\tilde{x}_k = \frac{1}{n}\sum_{i=1}^{n} x_k^i \tag{5}$$

**Lemma 2** *Let $W_k$ be an $n \times n$ mixing matrix and $\alpha$ be the weight of the global model in the PS-triggered local update. Then, there exists $W_k$ causing the same consequence as the training of ALADDIN.*

$$W_k = \begin{bmatrix} 1-\frac{\alpha}{n}(n-1) & \frac{\alpha}{n} & \cdots & \frac{\alpha}{n} \\ \frac{\alpha}{n} & 1-\frac{\alpha}{n}(n-1) & \cdots & \frac{\alpha}{n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\alpha}{n} & \frac{\alpha}{n} & \cdots & 1-\frac{\alpha}{n}(n-1) \end{bmatrix} \tag{6}$$

# Convergence Analysis for ALADDIN

## ☐ Theorem 1 (Convergence of ALADDIN)

- The convergence rate of ALADDIN is consistent with that of single-worker training

## ☐ Corollary 1 (Linear speedup)

- The convergence is accelerated linearly as the number of workers increases

**Theorem 1** (Convergence of ALADDIN) *Let $L$, $\sigma^2$, $\zeta$, $\pi$, and $\tilde{x}^*$ be the Lipschitz constant, variance bound for gradients, magnitude of second largest eigenvalue, update period, and optimal model, respectively. Then, the convergence rate of ALADDIN is as follows.*

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\|\nabla f(\tilde{x}_k)\|^2 \leq \frac{2[f(\tilde{x}_0) - f(\tilde{x}_*)]}{\eta K} + \frac{\eta L \sigma^2}{n} + \eta^2 L^2 \sigma^2 \left(\frac{1+\zeta^2}{1-\zeta^2}\pi - 1\right) \quad (7)$$

**Corollary 1** With a proper learning rate $\eta = \frac{1}{L}\sqrt{\frac{n}{K}}$,

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\|\nabla f(\tilde{x}_k)\|^2 \leq \frac{2L[f(\tilde{x}_0) - f(\tilde{x}_*)] + \sigma^2}{\boxed{\sqrt{nK}}} + \cancel{\frac{n\sigma^2}{K}\left(\frac{1+\zeta^2}{1-\zeta^2}\pi - 1\right)} \quad (8)$$

$$\Rightarrow O\left(\frac{1}{\sqrt{nK}}\right)$$

# Experimental Setup

## ☐ Models

- ResNet-50 (23M parameters), VGG-16 (128M parameters)

## ☐ Dataset

- CIFAR-10 (50K train images, 10K test images)
- ImageNet-1K (1.2M train images, 50K test images)

## ☐ Competing algorithms

- Centralized: ASP (NeuIPS’11), EASGD (NeuIPS’15)
- Decentralized: AR-SGD, SGP (ICML’19)

## ☐ The cluster with four machines

- 2 * NVIDIA 2080Ti GPU (14.90 TFLOPS, 12GB memory)
- Intel i-7 CPU with 64 GB memory
- 10Gbps Ethernet

# Questions to Answer

## ☐ Q1: Model accuracy

- Does ALADDIN provide the higher accuracy than existing algorithms?

## ☐ Q2: Convergence rate

- Does ALADDIN provide the higher convergence rate than existing algorithms?

## ☐ Q3: Scalability (speedup w.r.t # of numbers)

- Does ALADDIN provide the better scalability than existing algorithms?

## ☐ Q4: Robustness to heterogeneous environments

- Does ALADDIN is more robust to heterogeneous environments than existing algorithms?

# Q1: Model Accuracy

## ☐ Goal

- To compare the *final accuracy* and *training time*

## ☐ Results

| | ResNet-50 (CIFAR-10) | | | | VGG-16 (CIFAR-10) | | | |
|---|---|---|---|---|---|---|---|---|
| | Test Acc. | Train time | Test Acc. (1 hrs.) | Train time (90%) | Test Acc. | Train time | Test Acc. (2 hrs.) | Train time (90%) |
| AR-SGD | 0.9353 | 2.37 hrs. | 0.8992 | 0.98 hrs. | **0.9238** | 14.46 hrs. | 0.7765 | 6.40 hrs. |
| ASP | 0.9315 | 2.24 hrs. | 0.8810 | 1.16 hrs. | 0.9171 | 10.96 hrs. | 0.4973 | 8.22 hrs. |
| EASGD | 0.9112 | 2.21 hrs. | 0.8431 | 1.36 hrs. | 0.9124 | 9.04 hrs. | 0.7002 | 6.18 hrs. |
| SGP | 0.9340 | 2.11 hrs. | 0.8839 | 1.15 hrs. | 0.9228 | 6.66 hrs. | 0.8247 | 3.94 hrs. |
| **ALADDIN** | **0.9378** | **1.81** hrs. | **0.9071** | **0.92** hrs. | 0.9226 | **4.97** hrs. | **0.8687** | **2.87** hrs. |

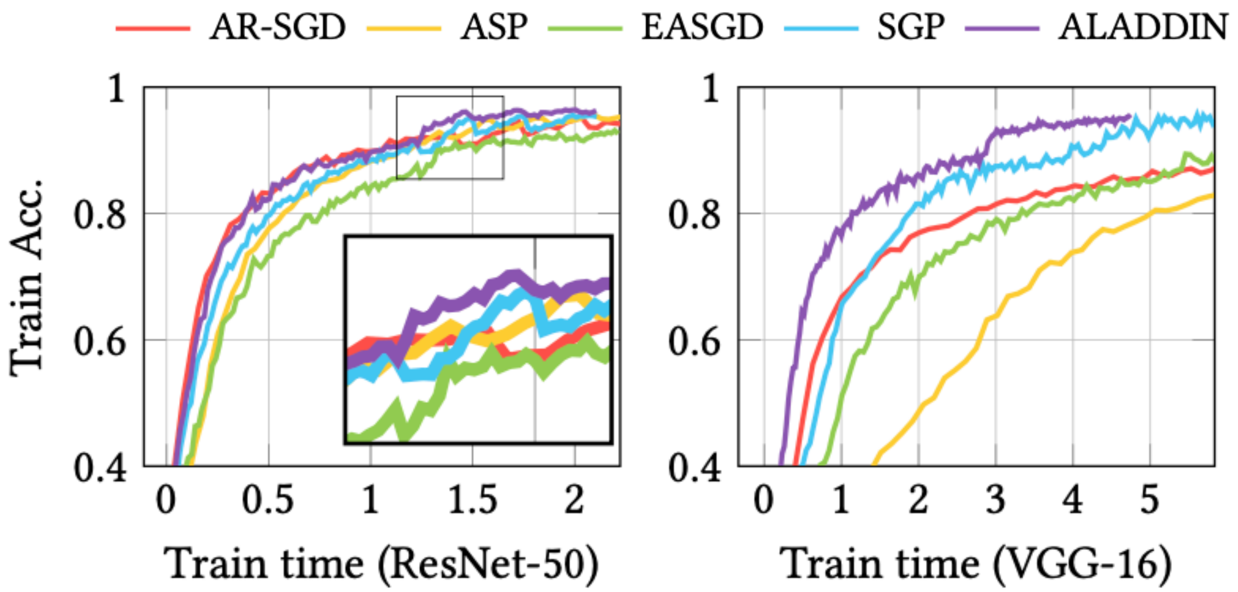
- The models trained by ALADDIN converge to ***high accuracies comparable to those of AR-SGD*** (the ground truth)
- ALADDIN finishes the training in ***shortest time*** for all cases

# Q2: Convergence Rate

## ☐ Goal

- To compare the *convergence rate* with respect to training time

## ☐ Results



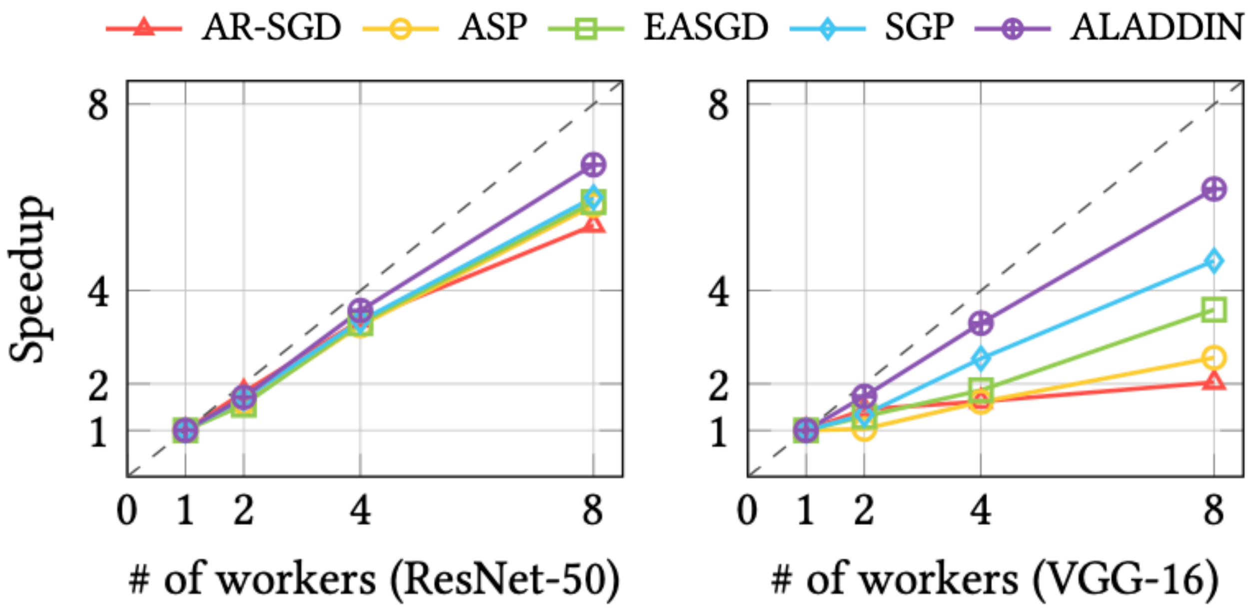

- ***ALADDIN outperforms all competing algorithms*** in the time-wise convergence rate
  - ☐ Achieving ***highest accuracies*** within the given time

# Q3: Scalability

## ☐ Goal

- To compare *scalability* with the increasing number of workers

## ☐ Results



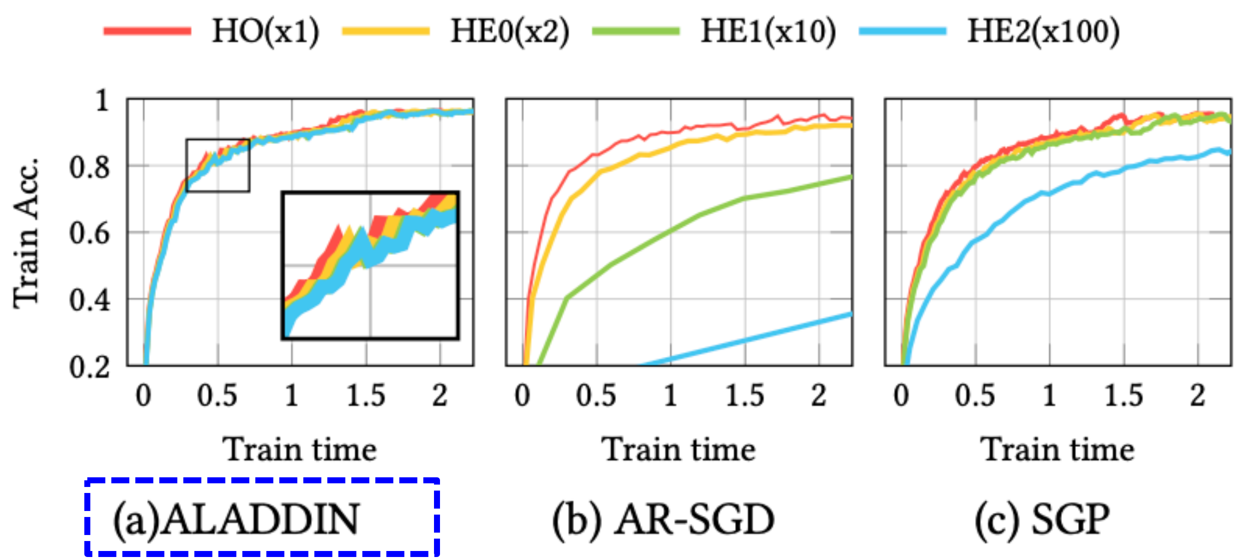

- ***ALADDIN provides the best speed-ups (almost linear)*** in both models and datasets

# Q4: Robustness to heterogeneous clusters

## ☐ Goal

- To compare the *robustness* to heterogeneous environments (x2, x10, x100)

## ☐ Results




| | ALADDIN | | AR-SGD | | SGP | |
|---|---|---|---|---|---|---|
| | ResNet | VGG | ResNet | VGG | ResNet | VGG |
| HO (×1) | **6.70** | **6.18** | 5.40 | 2.03 | 6.00 | 4.64 |
| HE0 (×2) | **6.30** | **5.74** | 3.22 | 1.46 | 5.21 | 4.15 |
| HE1 (×10) | **5.95** | **5.49** | 0.71 | 0.62 | 4.83 | 3.81 |
| HE2 (×100) | **5.87** | **5.41** | 0.08 | 0.07 | 1.93 | 1.89 |

- ***ALADDIN is most robust*** to all heterogeneous clusters
  - ☐ In terms of both *convergence rate* and *speedup*

# Conclusions

- **We identified the deficiencies of centralized and decentralized training**
  - Large communication overhead and increased parameter variance problems
- **We proposed a novel asymmetric training algorithm, ALADDIN**
  - Successfully addressing both problems at the same time
- **We provided the theoretical analysis for the convergence of ALADDIN**
- **Through comprehensive experiments, we showed that**
  - ALADDIN finishes the training within ***the shortest time***, while achieving high accuracies comparable to those of a synchronous algorithm (AR-SGD)
  - ALADDIN shows almost ***linear speed-up*** as the number of workers increases
  - ALADDIN is ***most robust to heterogeneous environments***

# Thank You !

Email: koyunyong@hanyang.ac.kr