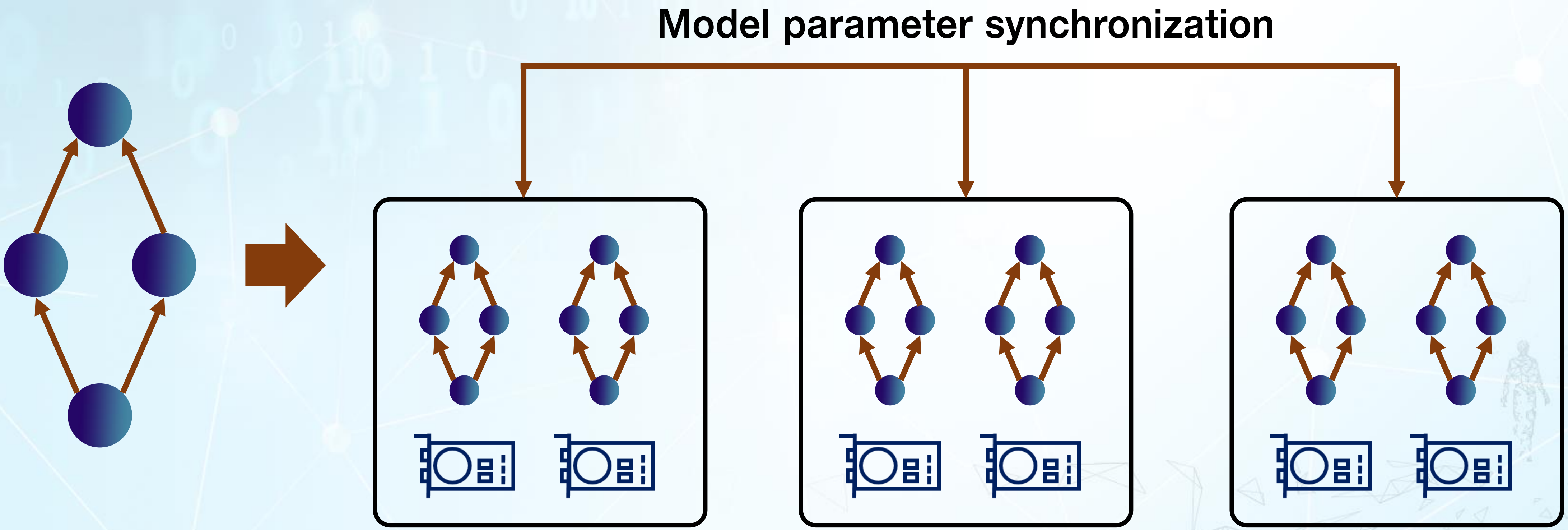
빅데이터와 머신러닝 소프트웨어

분산 머신러닝/딥러닝 개요

Distributed ML Training

- Speed up ML training convergence by parallelizing it across multiple devices (e.g., GPUs)



Distributed ML Training

Weak scaling

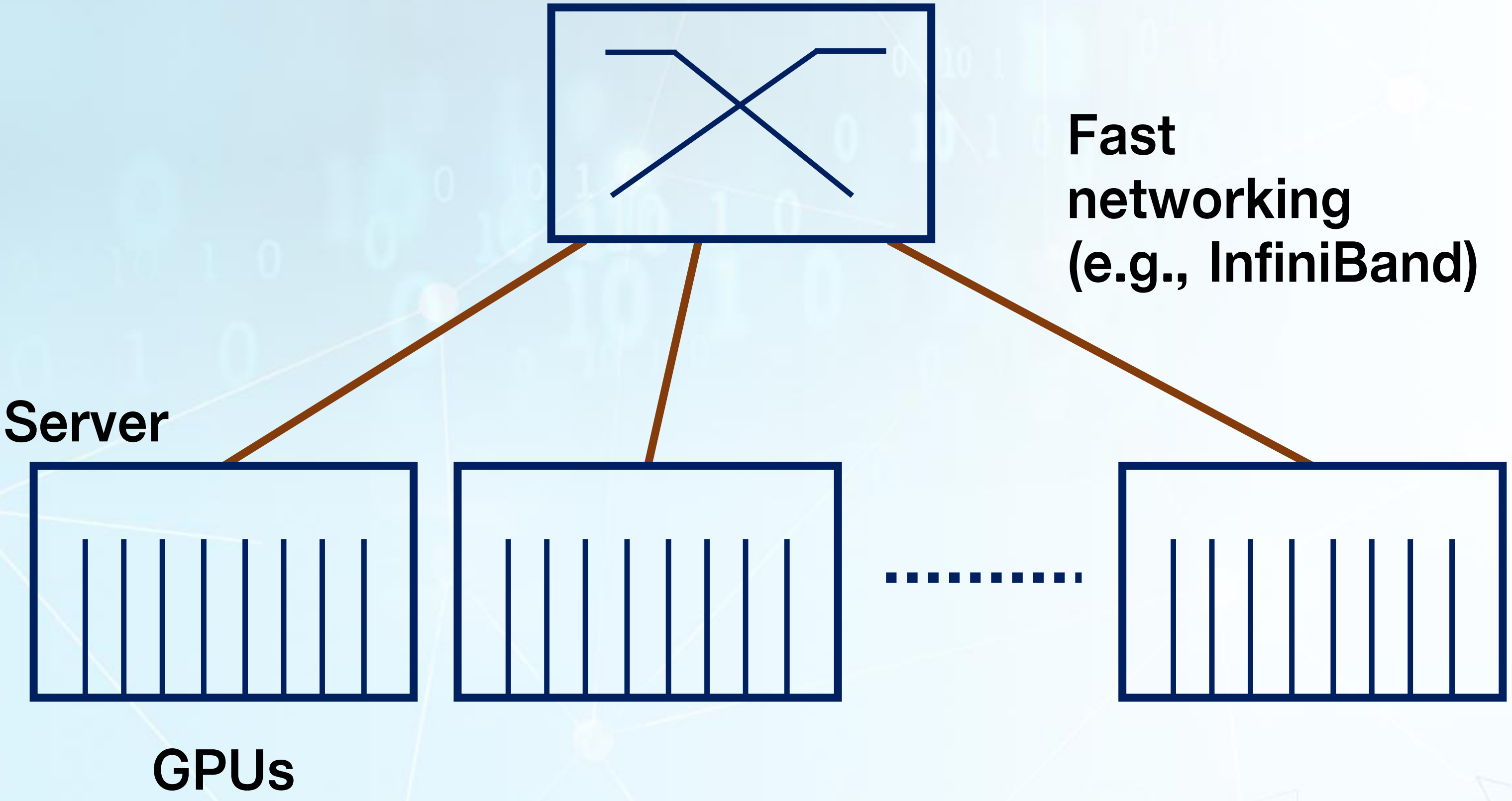
- Fix the batch size per device (e.g., GPU)
- As we add more devices, the total batch size increases
- To mitigate communication overheads, weak scaling is typically used, but it requires hyperparameter tuning

Strong scaling

- Fix the total batch size for all devices
- As we add more devices, the batch size per device decreases

MEMO

Cluster for Distributed ML Training



MEMO

Cluster for Distributed ML Training

Distributed ML job example

- Facebook used 256 NVIDIA P100 GPUs to train ImageNet in 1 hour (CVPR 2017)
- Fast.ai trained ImageNet in 18 minutes using 16 AWS P3 instances, each with 8 NVIDIA V100 GPUs (http://www.fast.ai/2018/08/10/fastai-diu-imagenet/, August 10, 2018)

MEMO

Cluster for Distributed ML Training

Cluster for big data processing

- Lots of machines, each of which has tens of CPU cores
- Lots of storage: HDDs for big data
- Medium-speed networking: 10Gbps Ethernet

MEMO

Distributed training issues

- **Parallelism**
  - data parallelism, model parallelism, hybrid parallelism
- **Model parameter synchronization**
  - synchronous, asynchronous, bounded synchronous
- **Training architecture**
  - parameter server architecture, Allreduce/Allgather architecture

MEMO

Parallelism

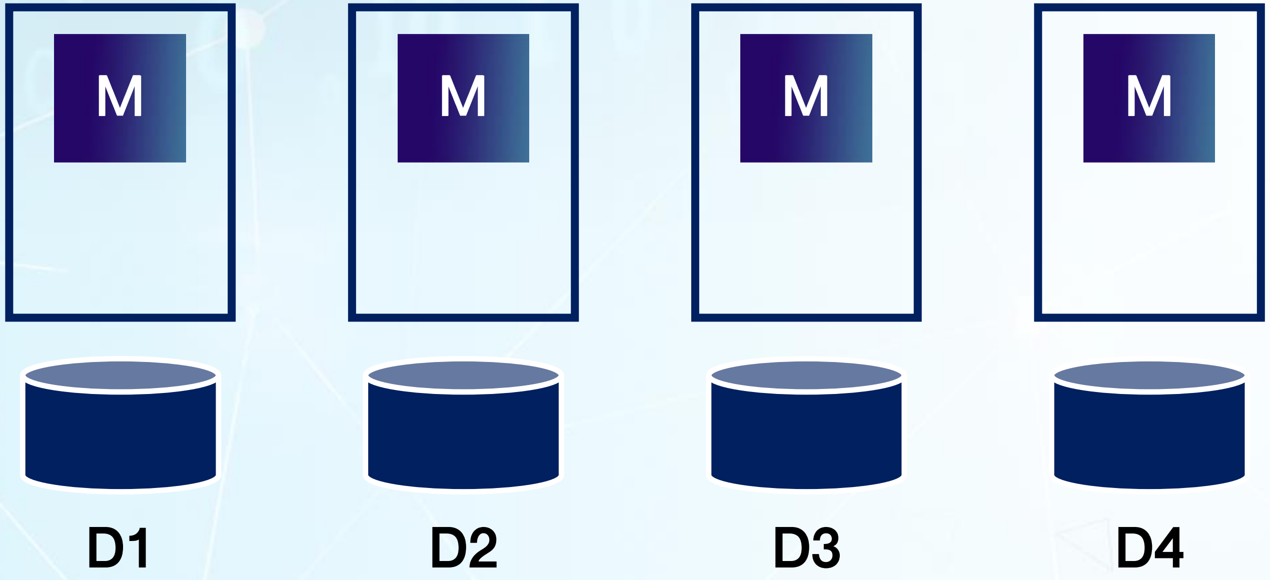
- Data Parallelism
- Model Parallelism
- Hybrid Parallelism

MEMO

Parallelism

Data Parallelism

- D = D1 U D2 U ⋯ U Dn
- Each worker i processes M with Di.

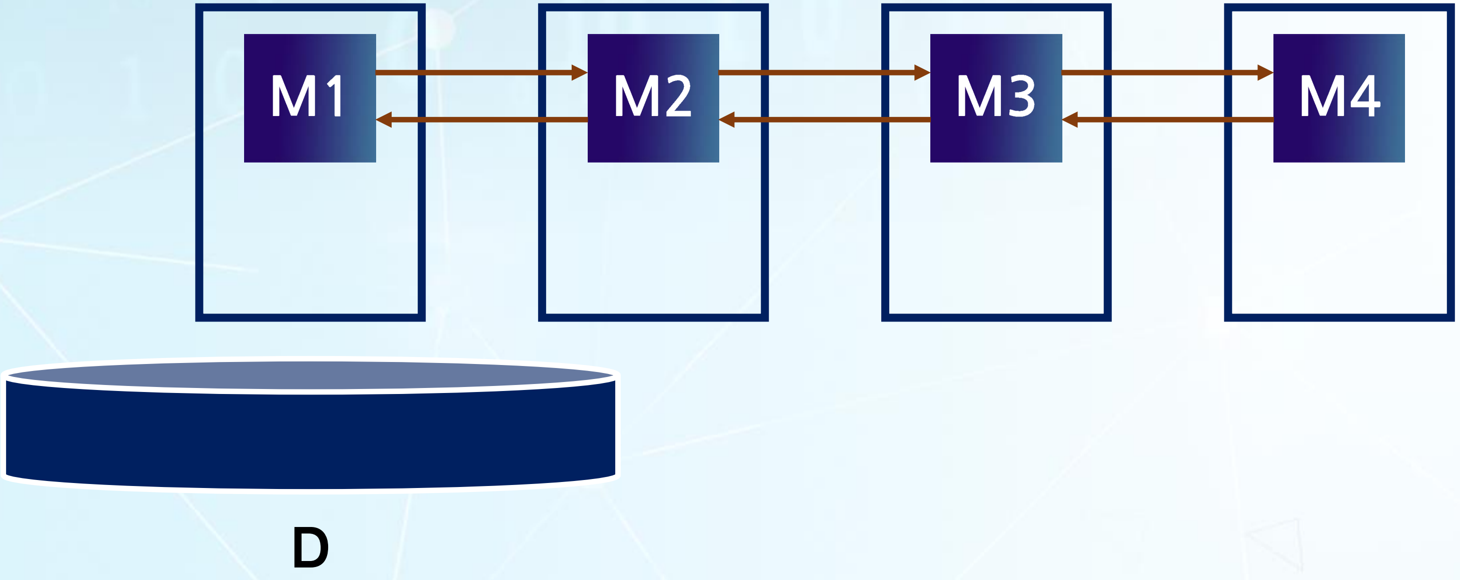


MEMO

Parallelism

Model Parallelism

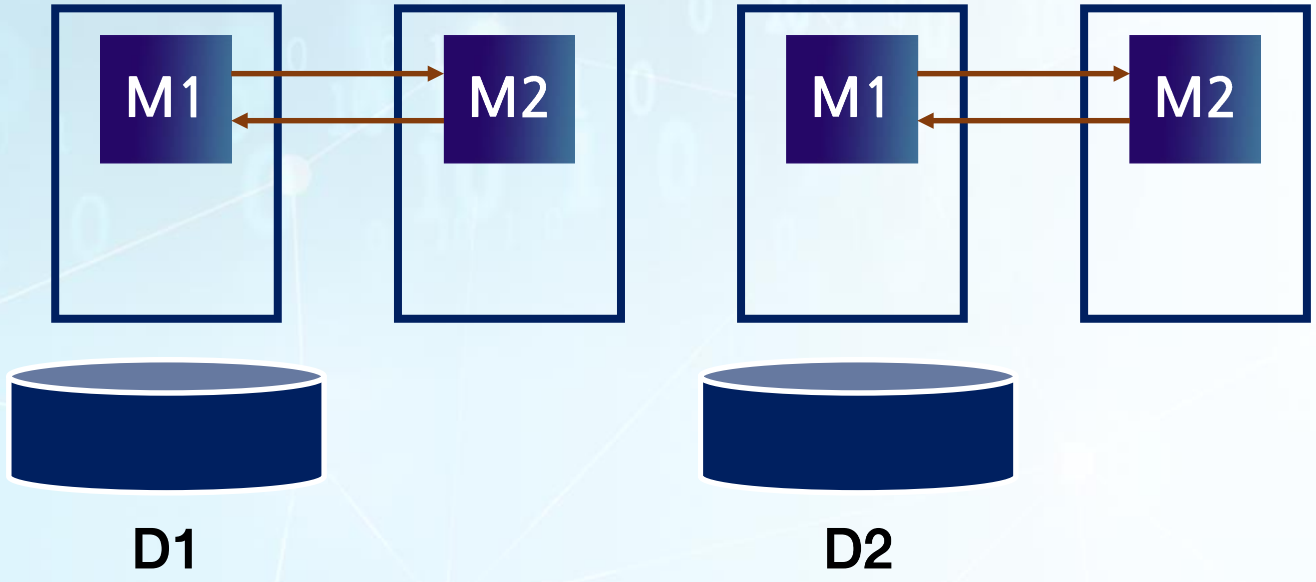
- M = M1 U M2 U ⋯ U Mn
- Each worker i processes Mi with D.



MEMO

Parallelism

Hybrid Parallelism



MEMO

Synchrony of Training

- Synchronous training
- Asynchronous training
- Bounded-synchronous training

MEMO

Synchrony of Training

- Synchronous training
  - Each worker computes gradients with the up-to-date model parameters
  - Gradients from all workers are aggregated
  - Model parameters are updated with the aggregated gradients

- There is a barrier to update model parameters

MEMO

Synchrony of Training

- Asynchronous training
  - Each worker computes gradients with the current model parameters
  - Each worker updates model parameters with the gradients computed independently
- Each worker executes as fast as it can, but it uses a model that does not reflect gradients aggregated from all workers
- Statistical efficiency vs. training throughput tradeoffs

MEMO

Synchrony of Training

Bounded synchronous training

- Limit the difference between the version of the global model and the version of the model stored at each worker

MEMO

Summary

- Distributed ML training
- Distributed ML training issues: parallelism, synchrony

MEMO

빅데이터와 머신러닝 소프트웨어

분산 머신러닝/딥러닝 아키텍쳐

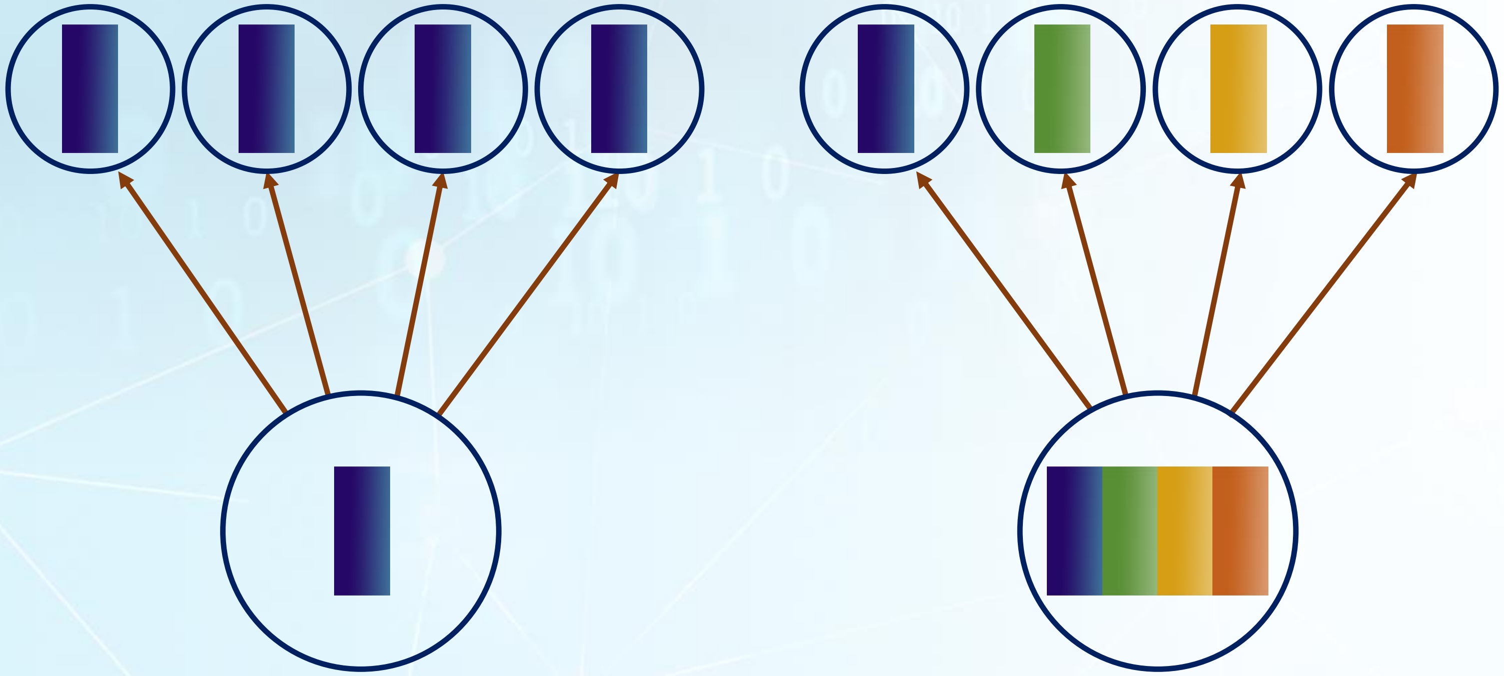
Distributed ML Training Architecture

- Allreduce/Allgather architecture

- Parameter server architecture

MEMO

Collective Communication

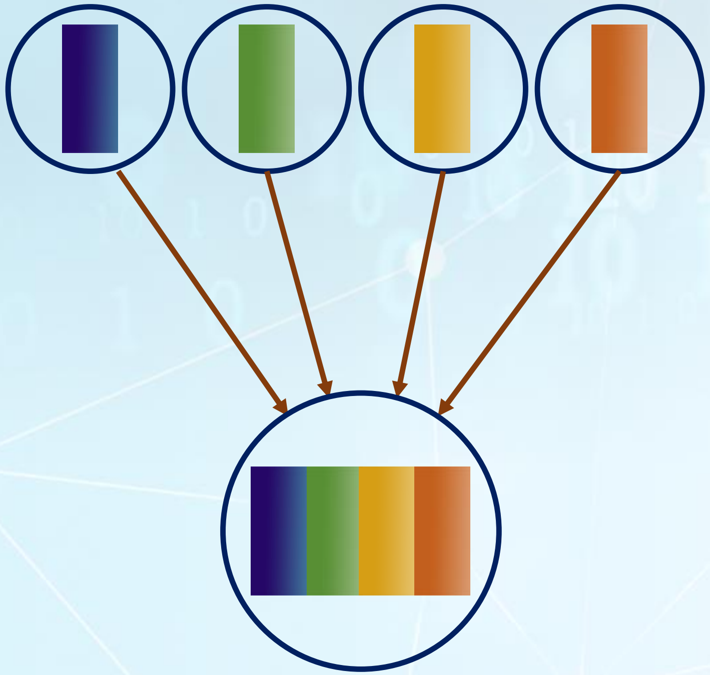


Broadcast

Scatter

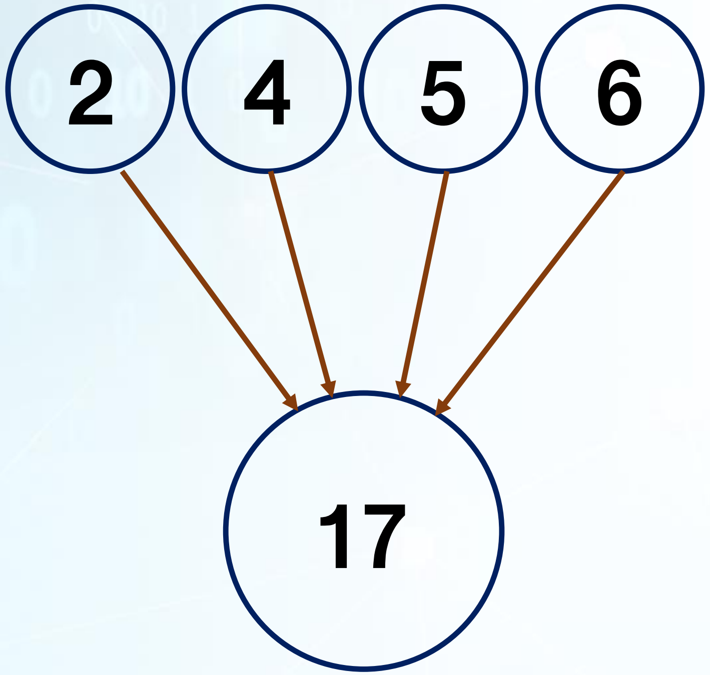
MEMO

Collective Communication



Gather

Allgather – Gather + Broadcast



Reduce

Allreduce – Reduce + Broadcast

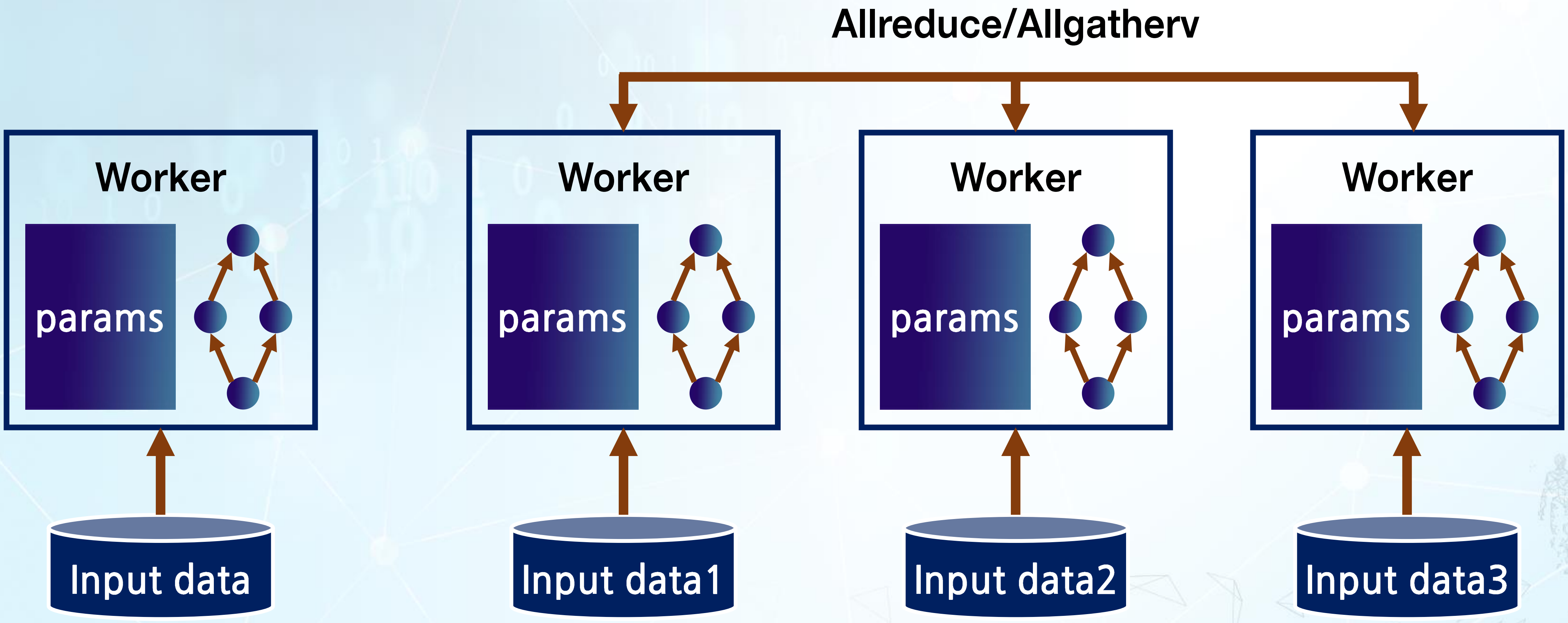
MEMO

Allreduce/Allgather synchronous training

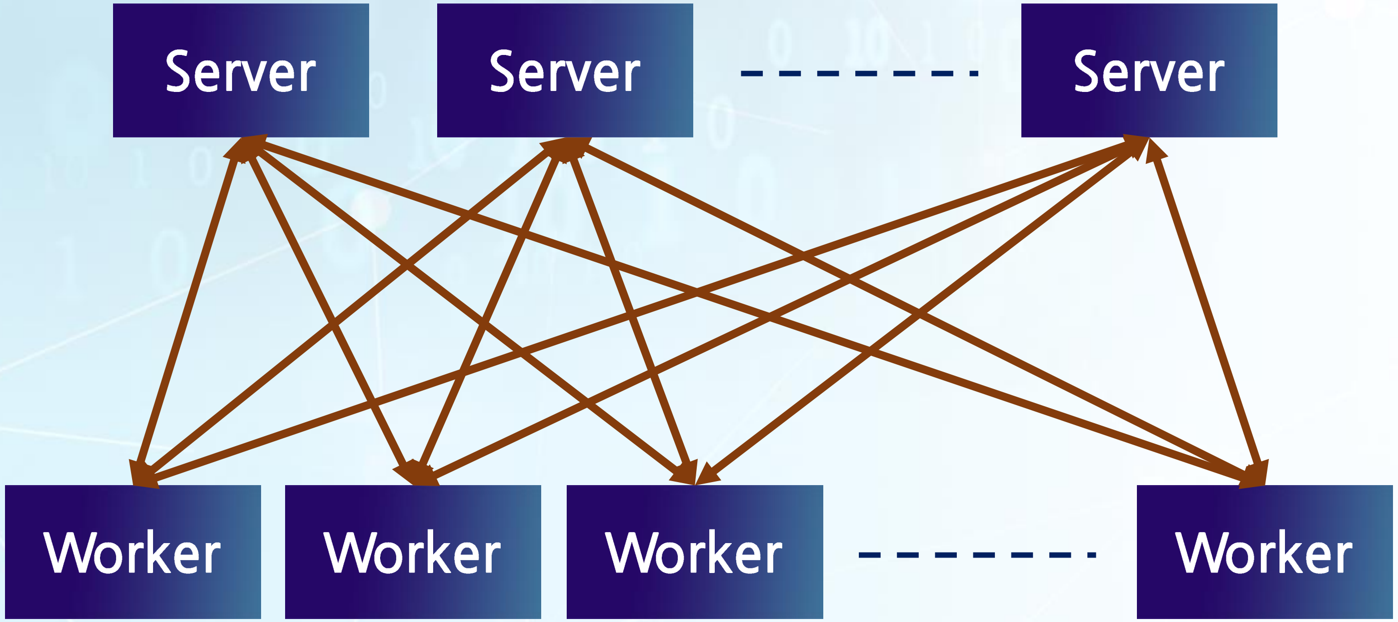
- Step 1
  - ► Workers compute gradients with training data
- Step 2
  - ► Workers run Allreduce (or Allgather) to aggregate them and apply the sums to update the model parameters
- The above steps iterate until training converges

MEMO

Allreduce/Allgather synchronous training



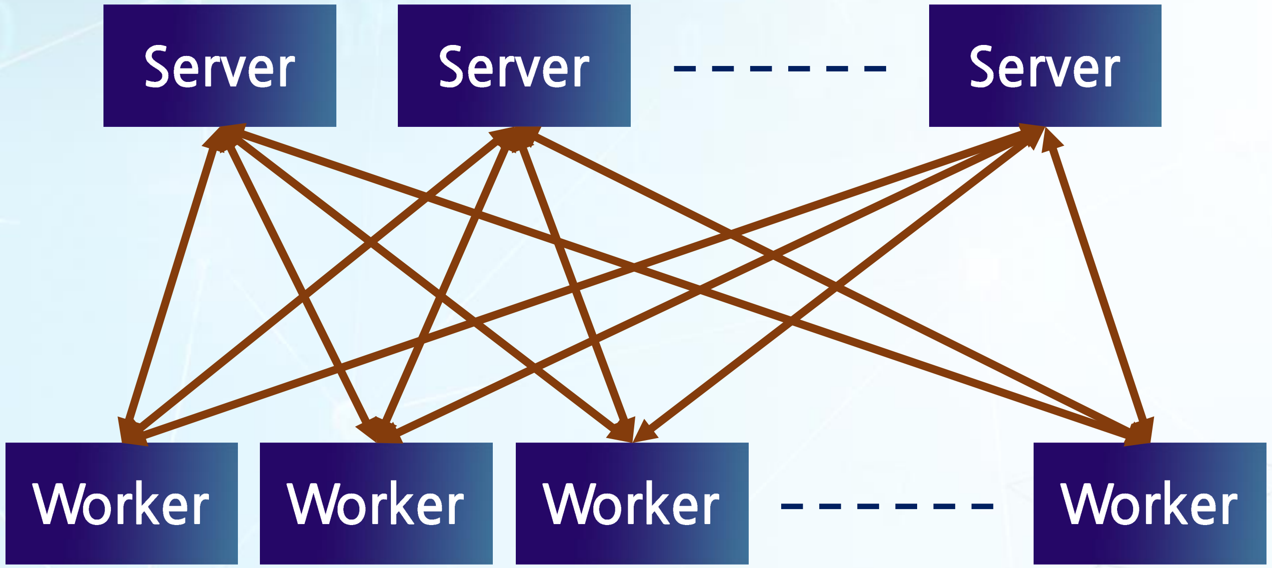
Parameter Server Architecture



MEMO

Parameter Server Architecture

- **Server**
  - Maintains a partition of the globally shared parameters
  - Performs global aggregation steps

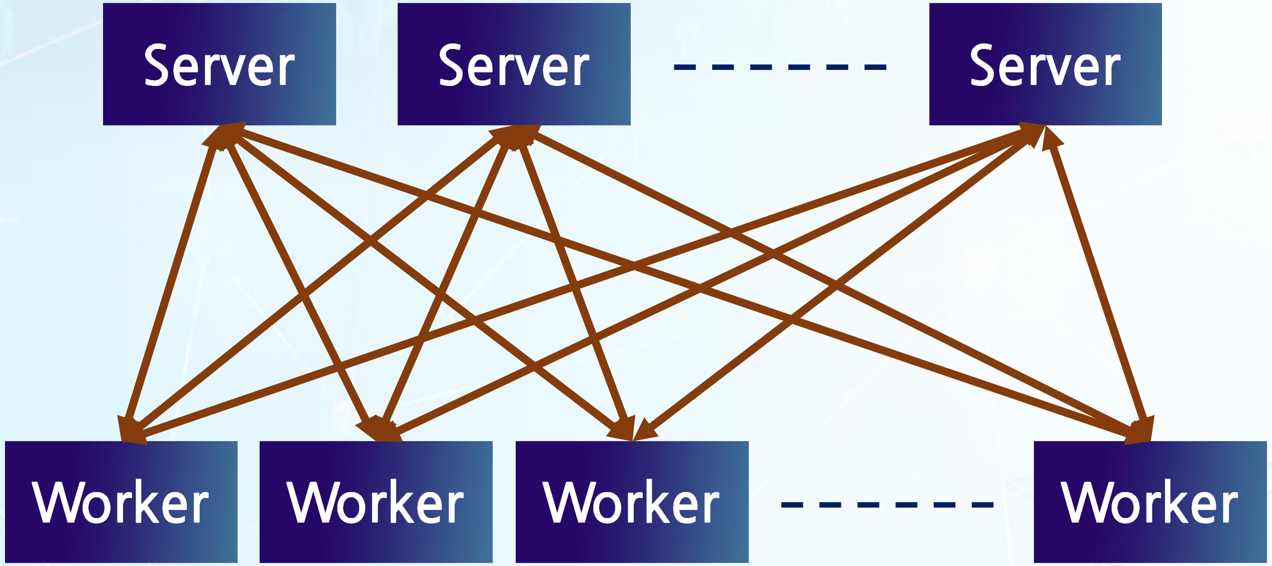


MEMO

Parameter Server Architecture

Worker

- Performs computation with (a portion of) training data communicates with servers
- Updating and retrieving the shared parameters



MEMO

Parameter Server Architecture

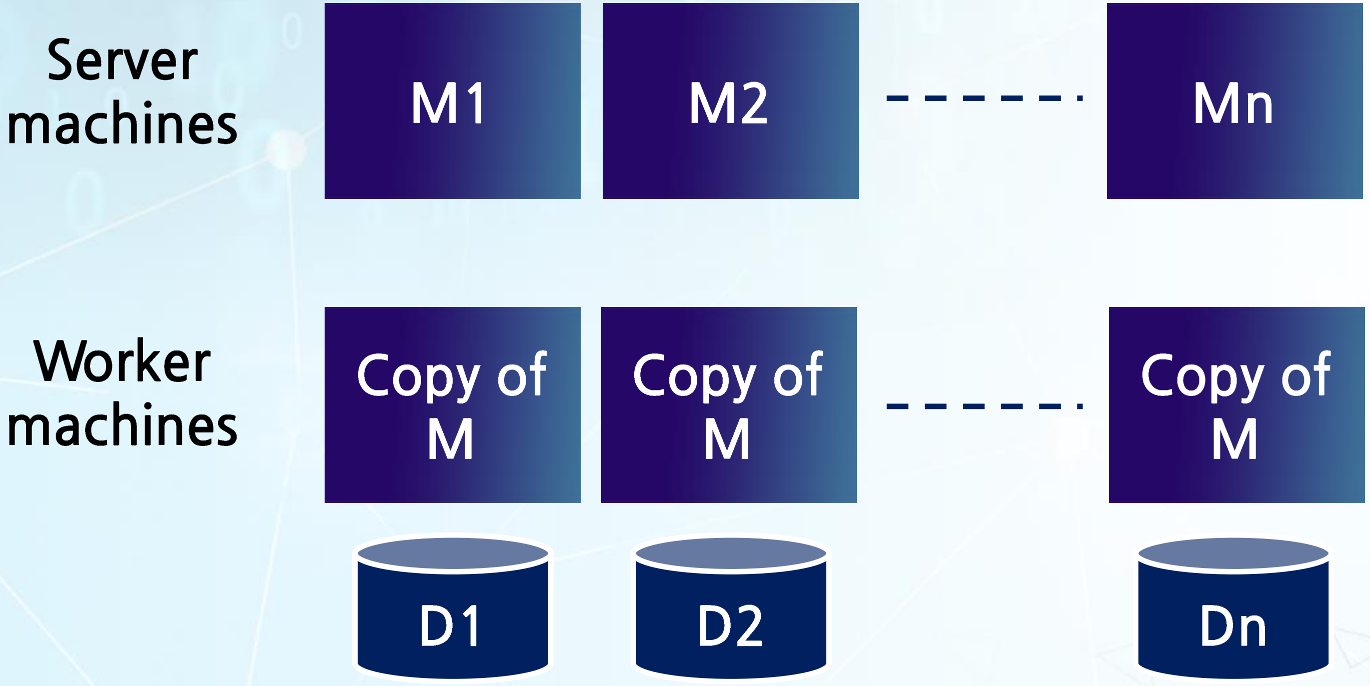
Synchronous Training

- Step 1
  - Workers compute gradients with training data and push them to servers
- Step 2
  - Each Server receives gradients from Workers, aggregates them, and applies the sums to update the model parameters
- Step 3
  - Workers pull the new model parameters
- The above steps iterate until training converges

MEMO

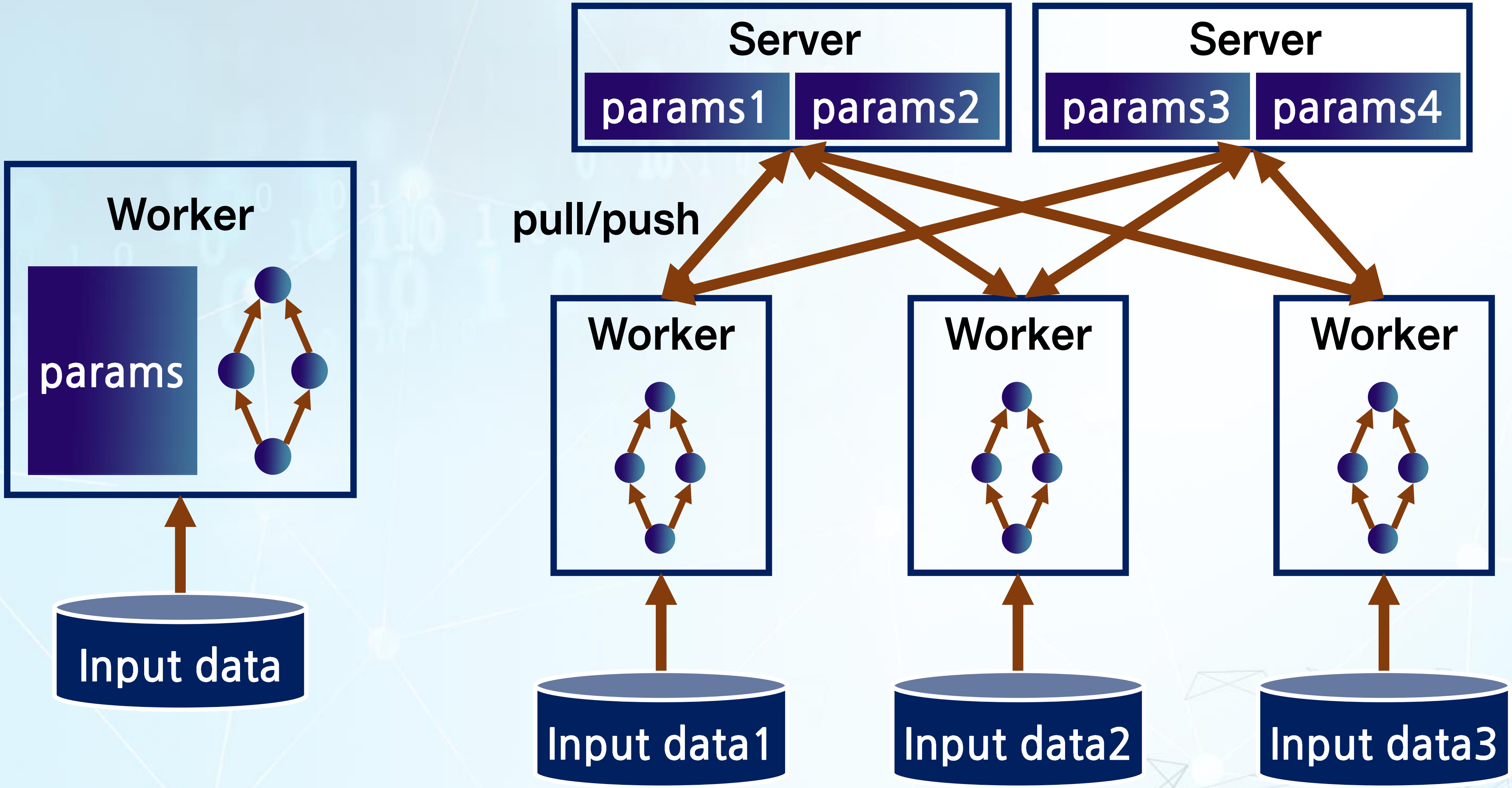
Parameter Server Architecture

- Data parallel training



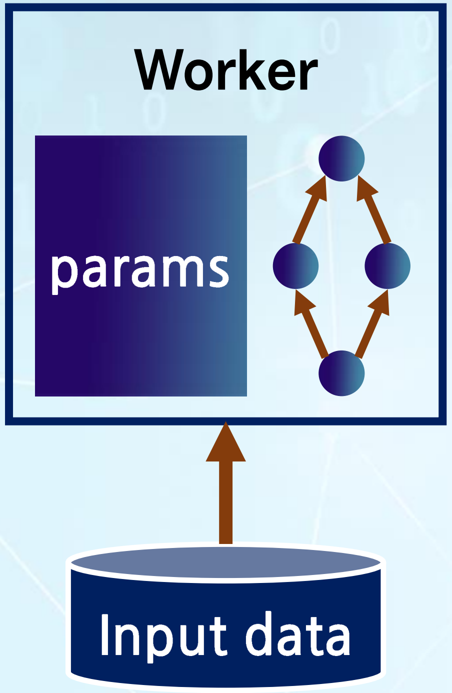
MEMO

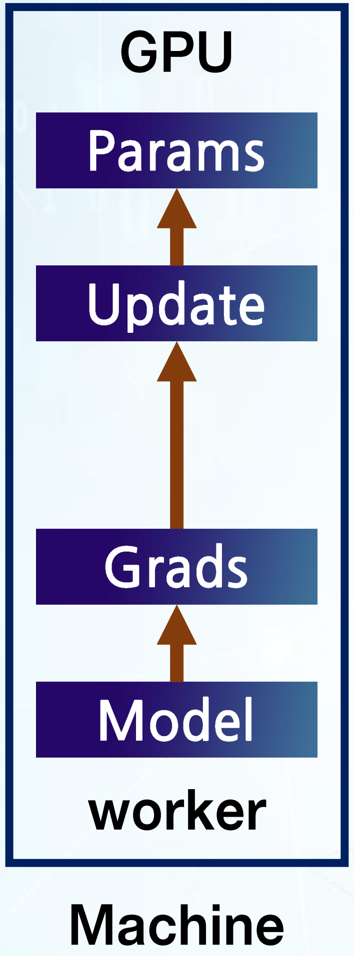
Parameter Server Architecture



TensorFlow Graph Transformation

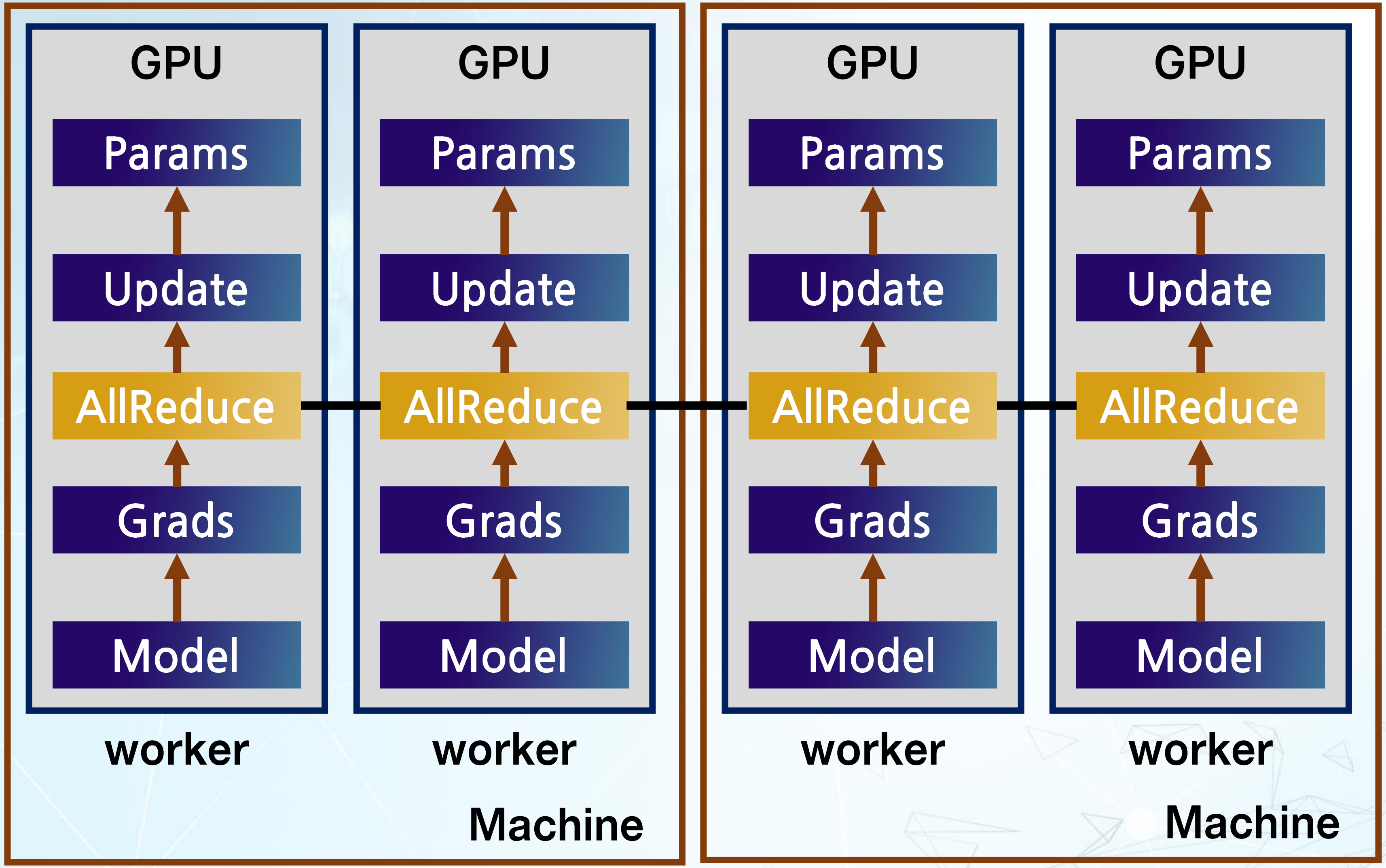
- Single-GPU graph



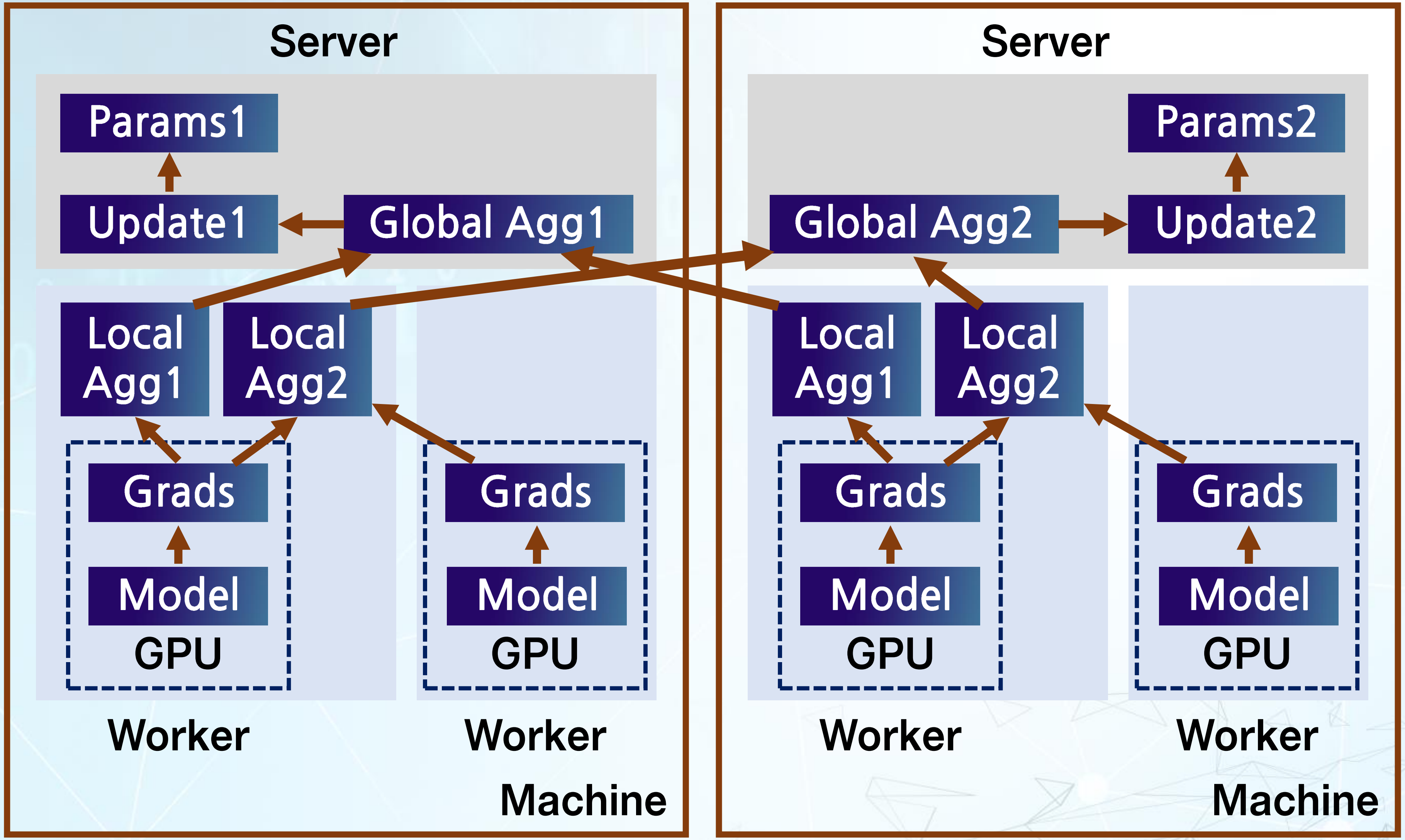


MEMO

TensorFlow Graph Transformation: Allreduce Architecture



TensorFlow Graph Transformation: Parameter Server Architecture



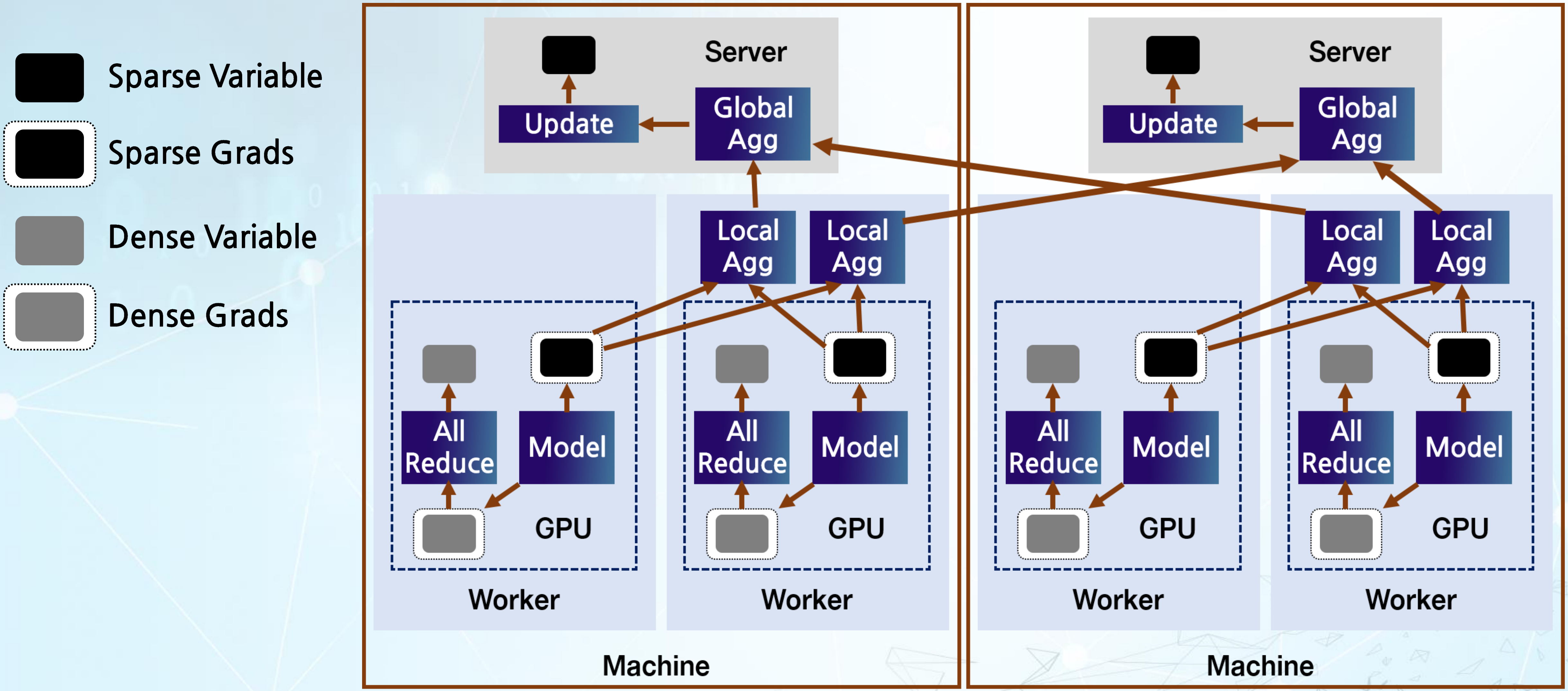
TensorFlow Graph Transformation

- **Dense model (e.g., Resnet50)**
  - dense parameters → Allreduce architecture

- **Sparse model (e.g., Language Model, Neural Machine Translation)**
  - sparse parameters + dense parameters
    → Parameter server architecture

MEMO

TensorFlow Graph Transformation: Parallax Hybrid Architecture



Summary

- Distributed ML training architecture
- Allreduce architecture
- Parameter server architecture
- TensorFlow graph transformation

MEMO