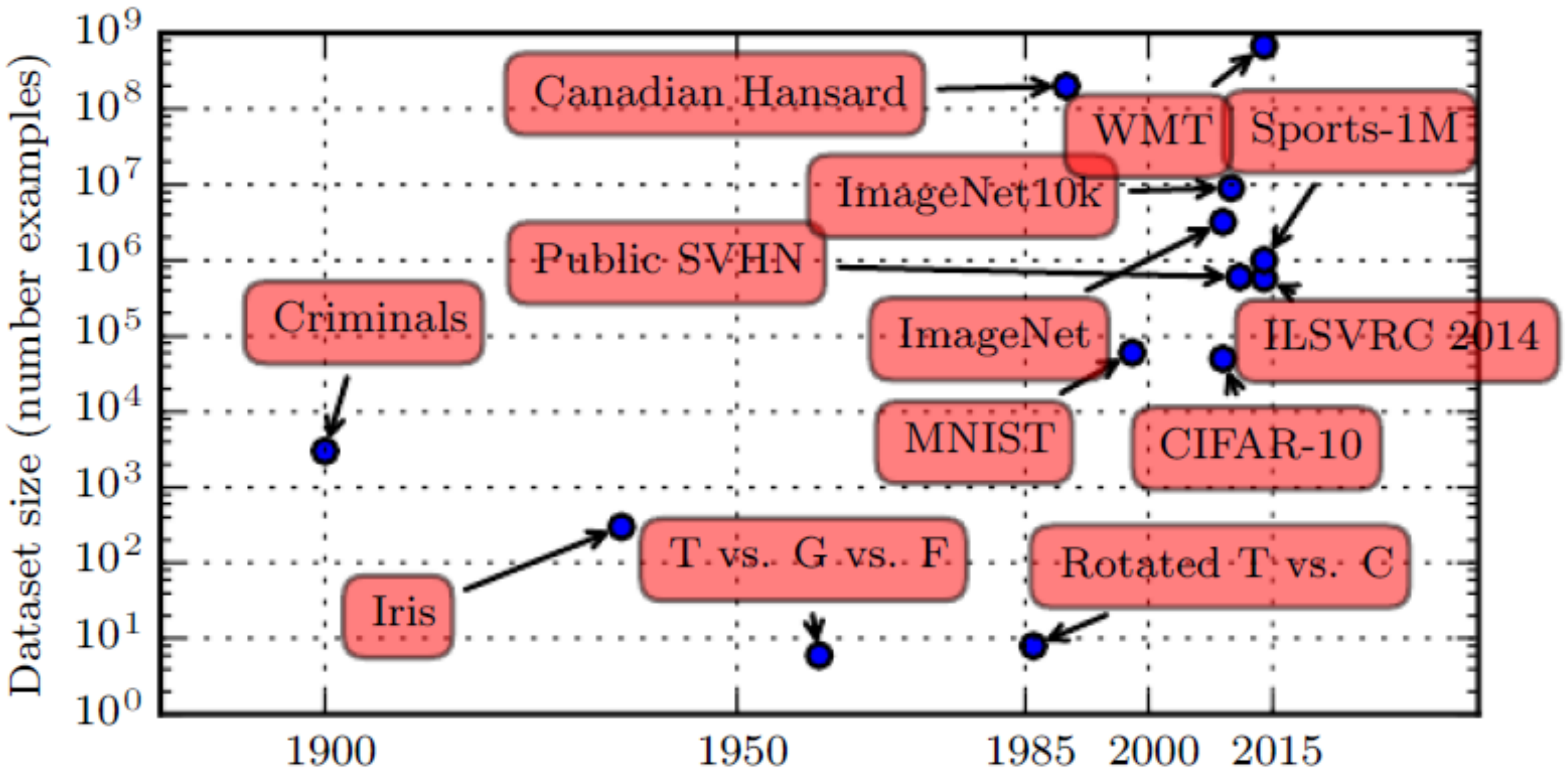
# Lecture 13: Networking and Deep Learning

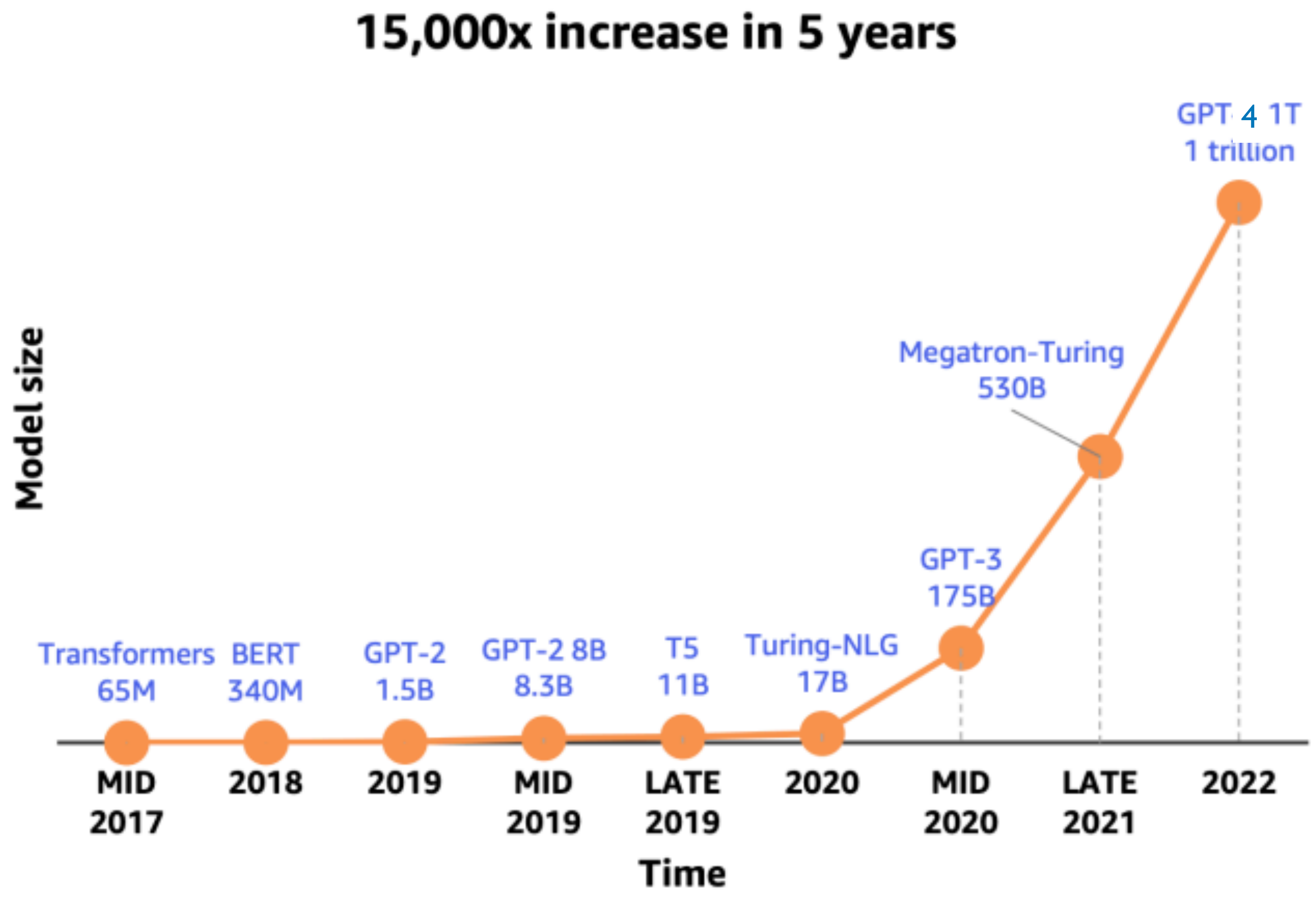
CS 234 / NetSys 210: Advanced Computer Networks

Sangeetha Abdu Jyothi

Parts of this lecture were adapted from talks on Parameter Server, Horovod, and TicTac

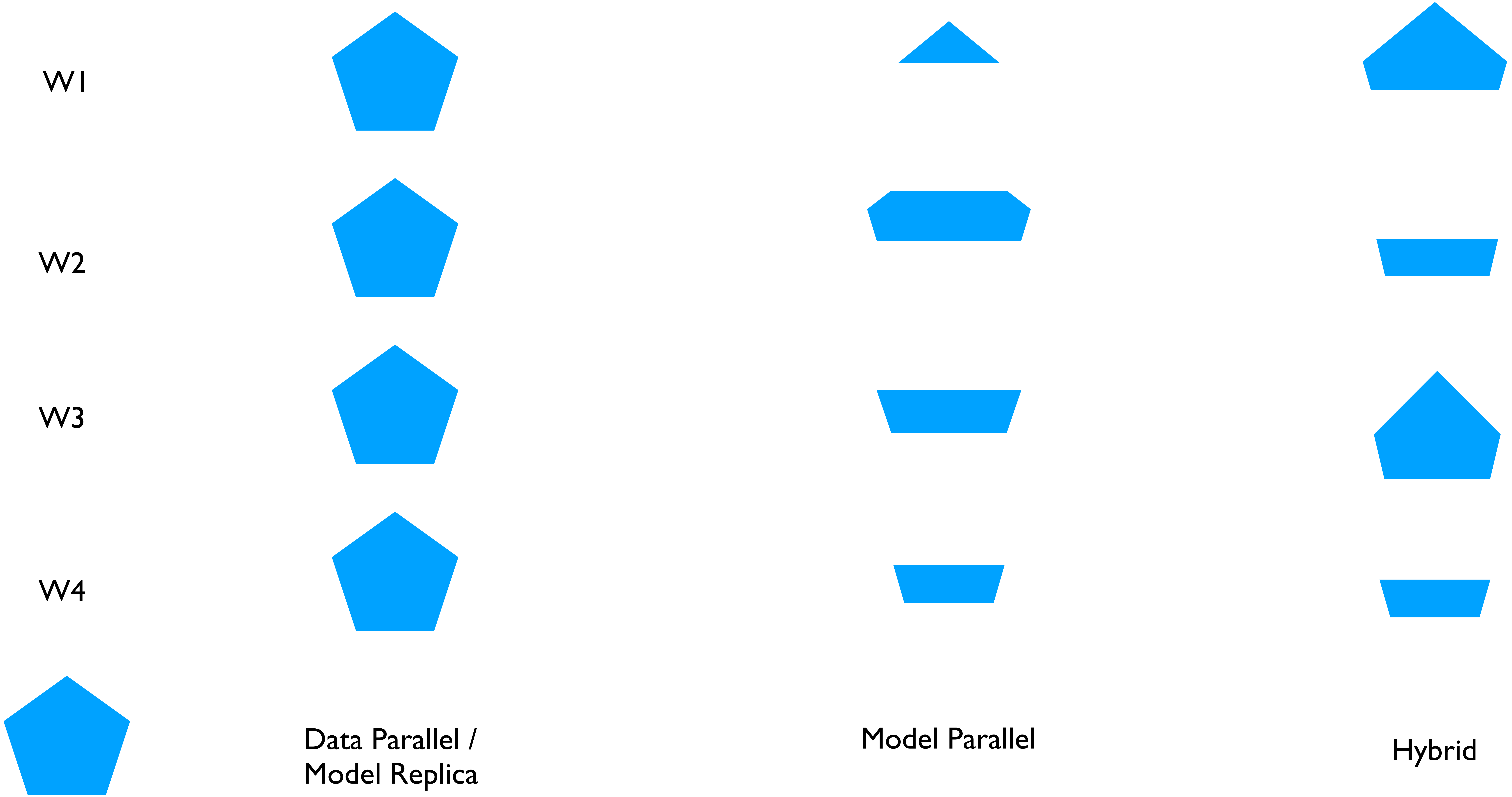
# Rapid Growth



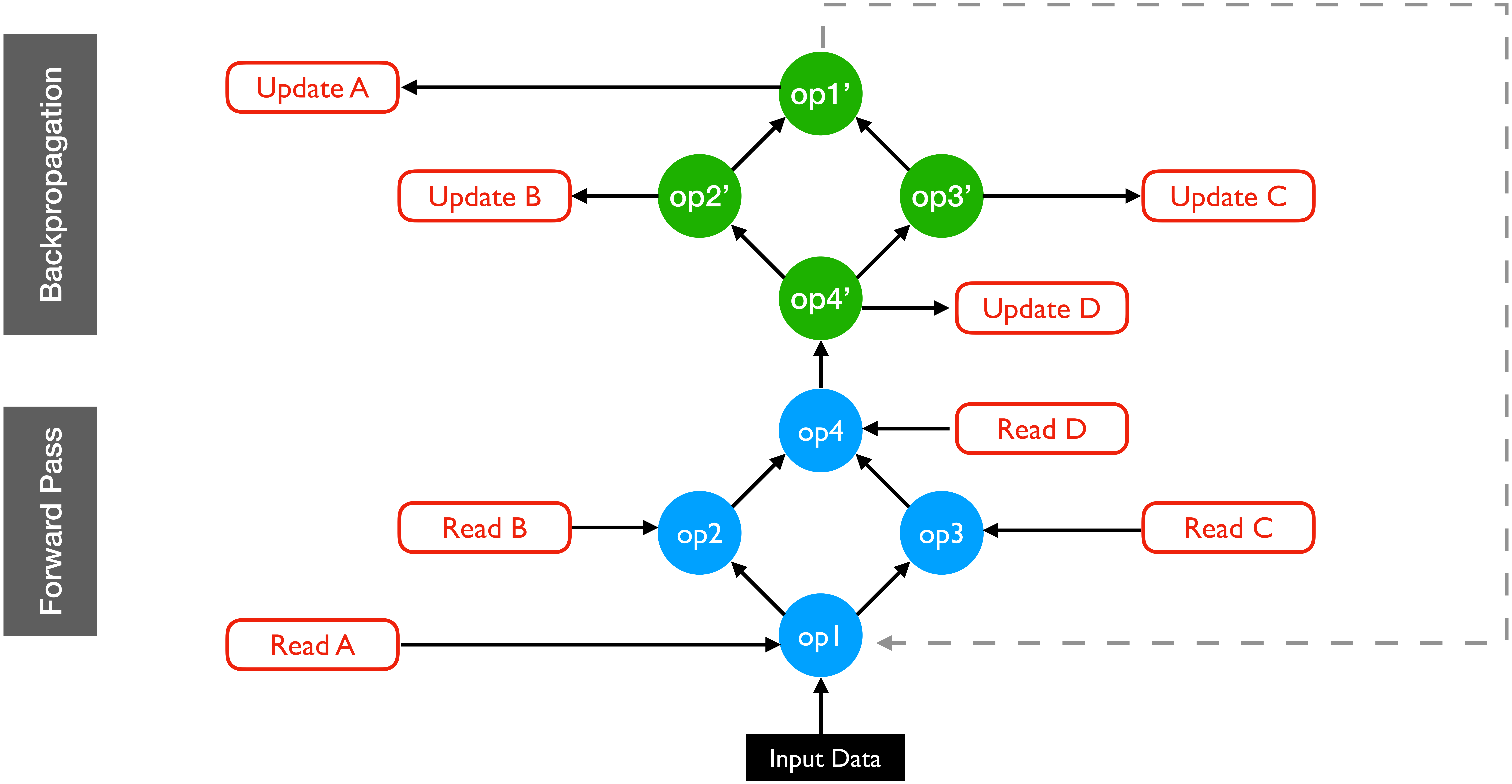




Datasets and Models are rapidly growing in size ➡️ Distributed training is necessary

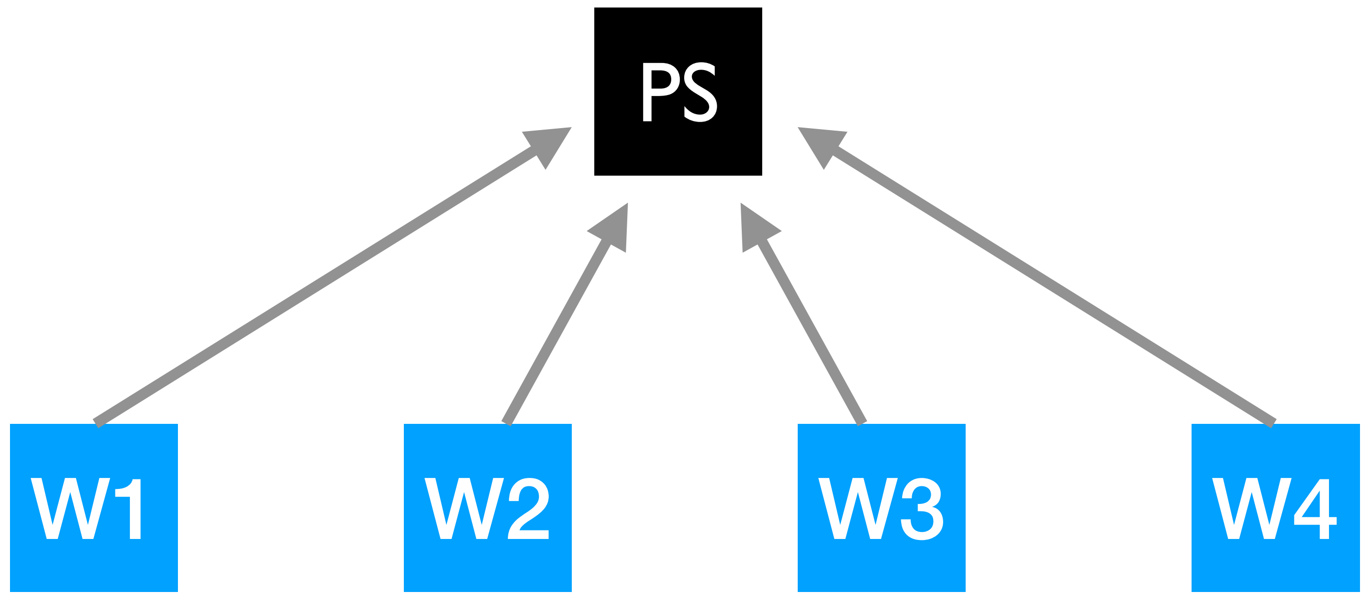
# Distribution Patterns



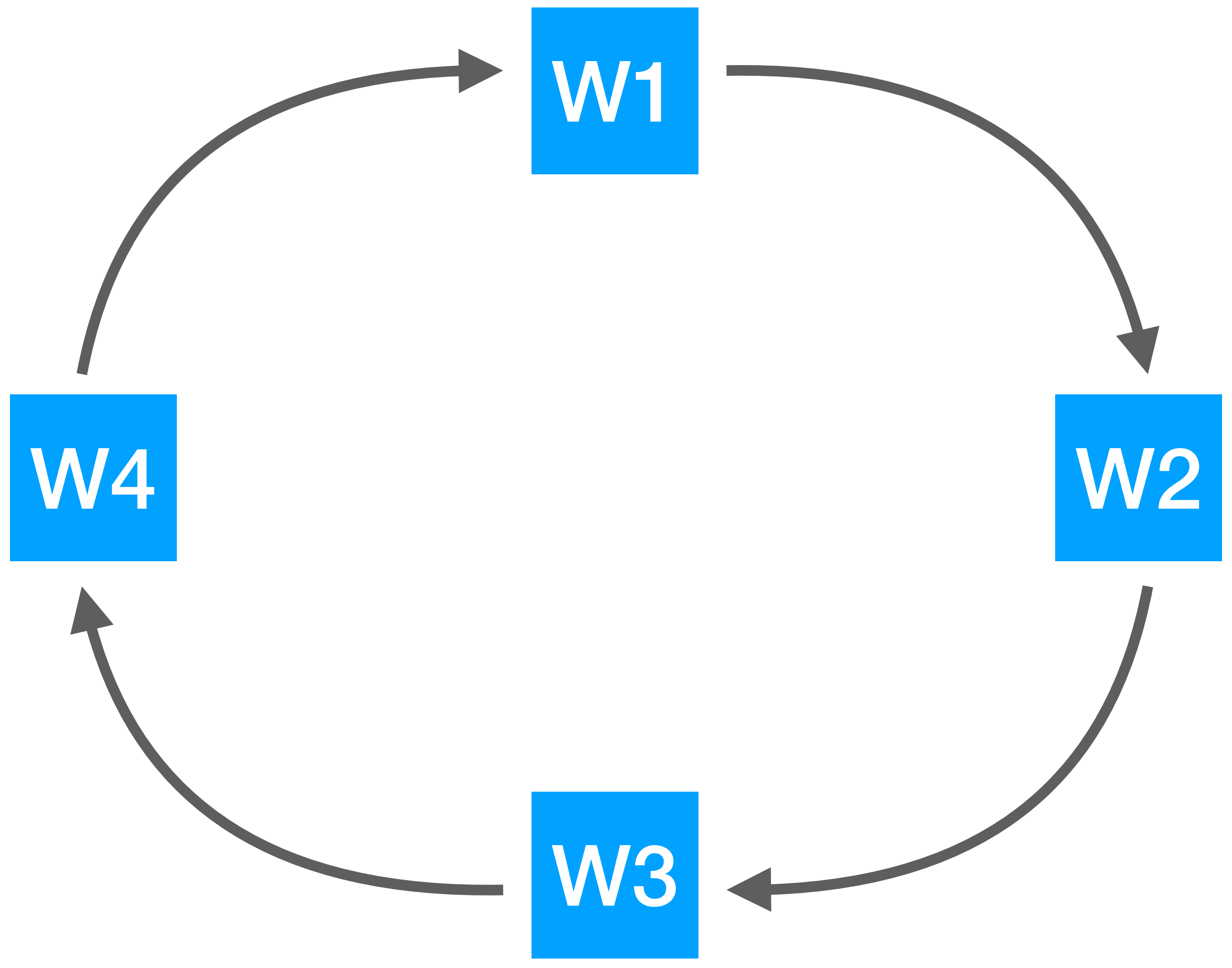
# Training Process



# Popular Modes of Network Aggregation



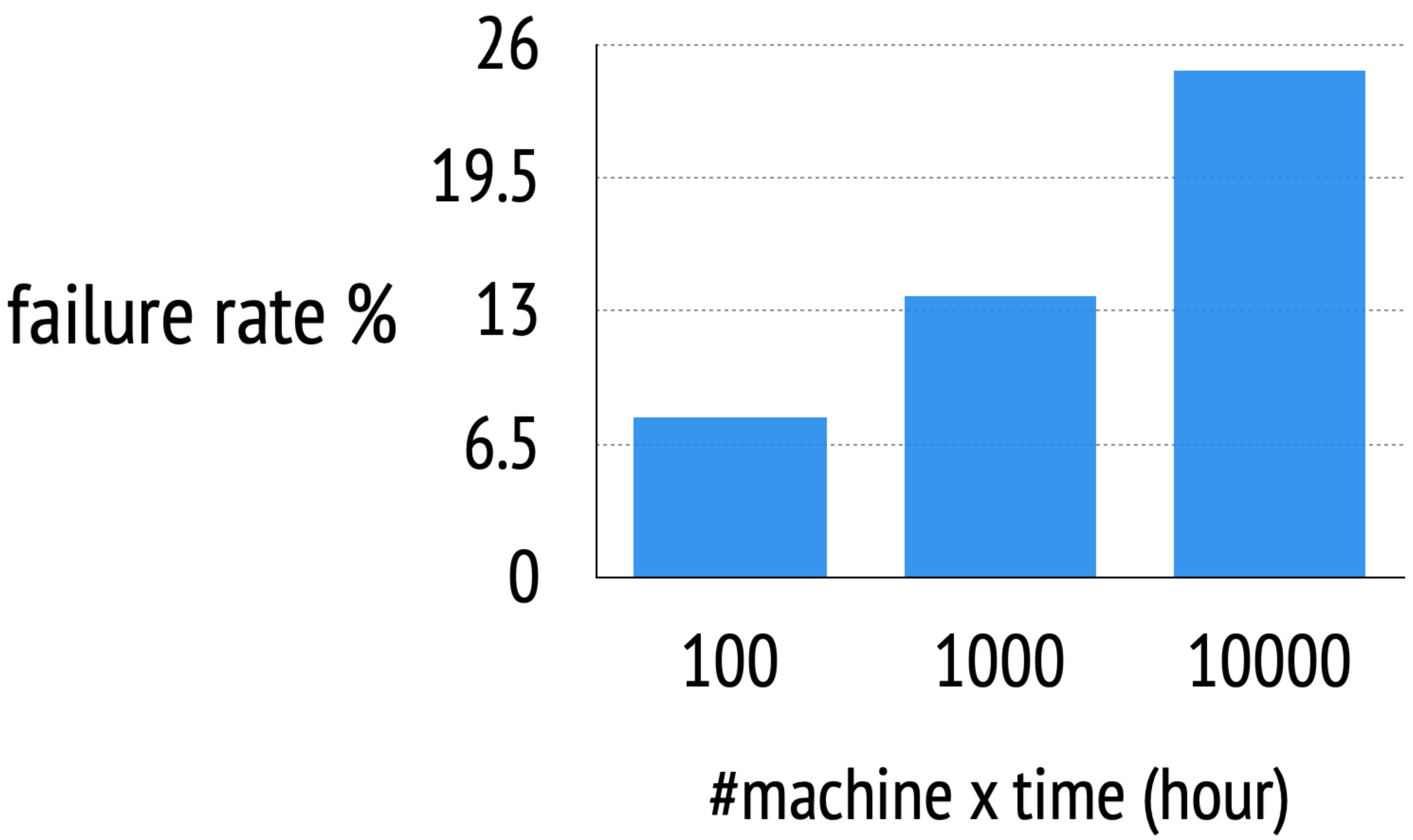

Parameter Server




Decentralized Aggregation

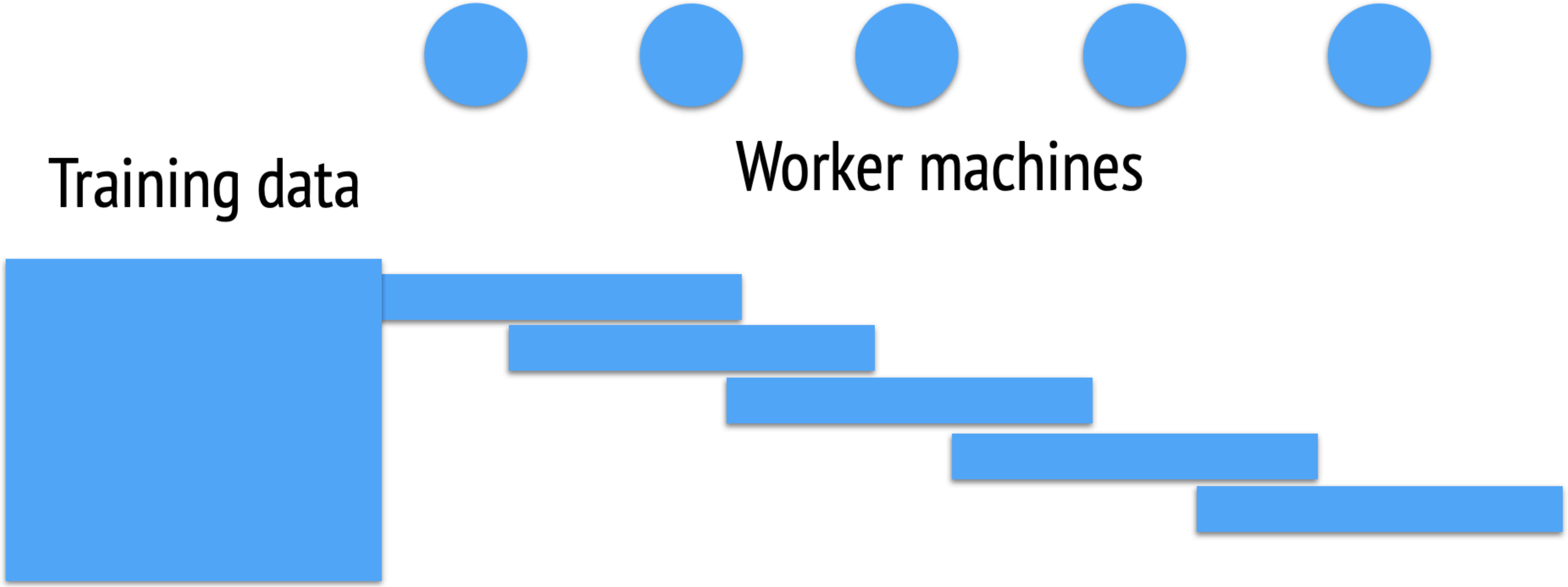
# Parameter Server [OSDI'14]

- Goals
  - Scale to industry-scale problems
    - billions of samples and features
    - hundreds of machines
  - Enable efficient communication
  - Fault tolerance
  - Easy to use

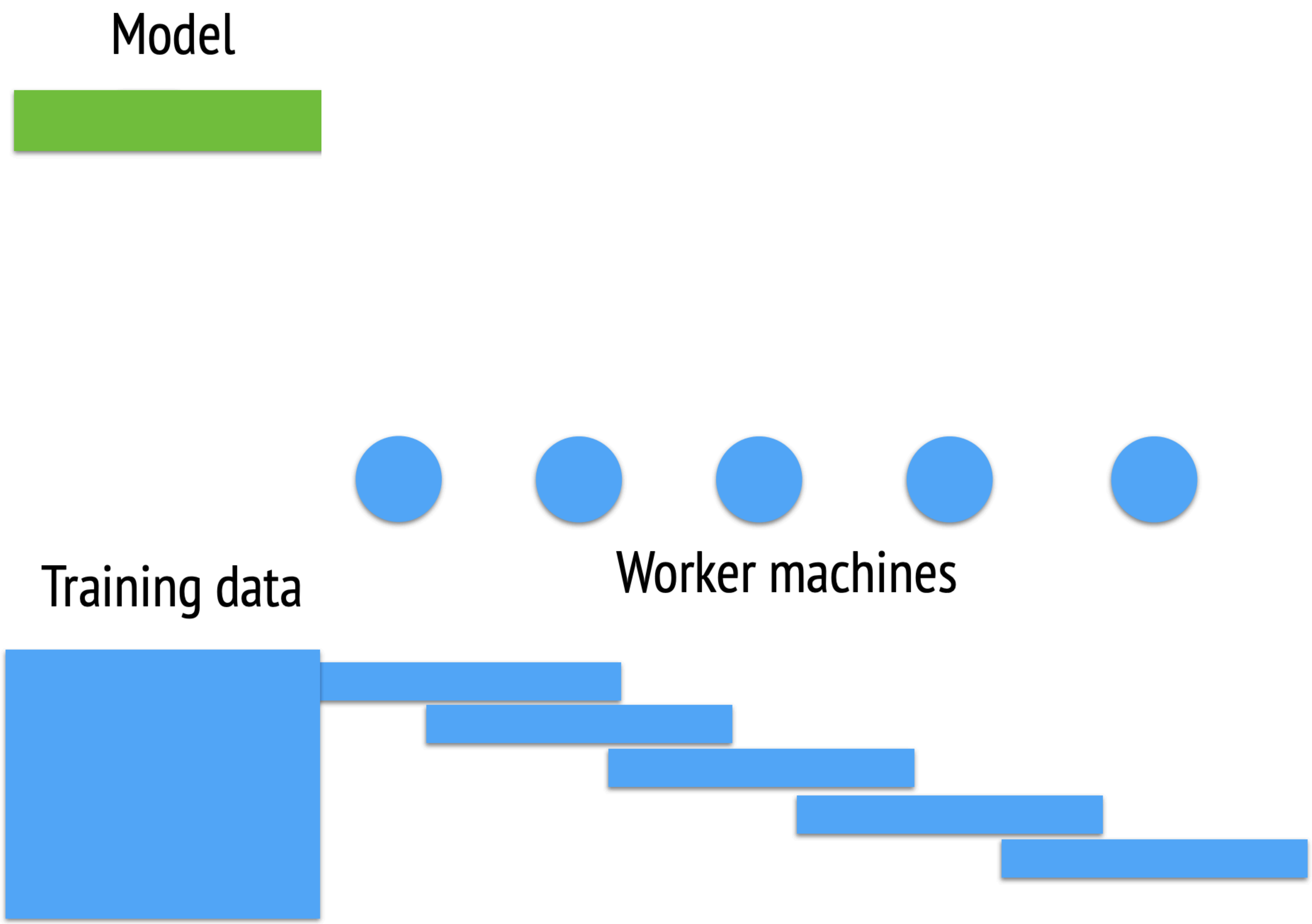


# Data and Model Partitioning

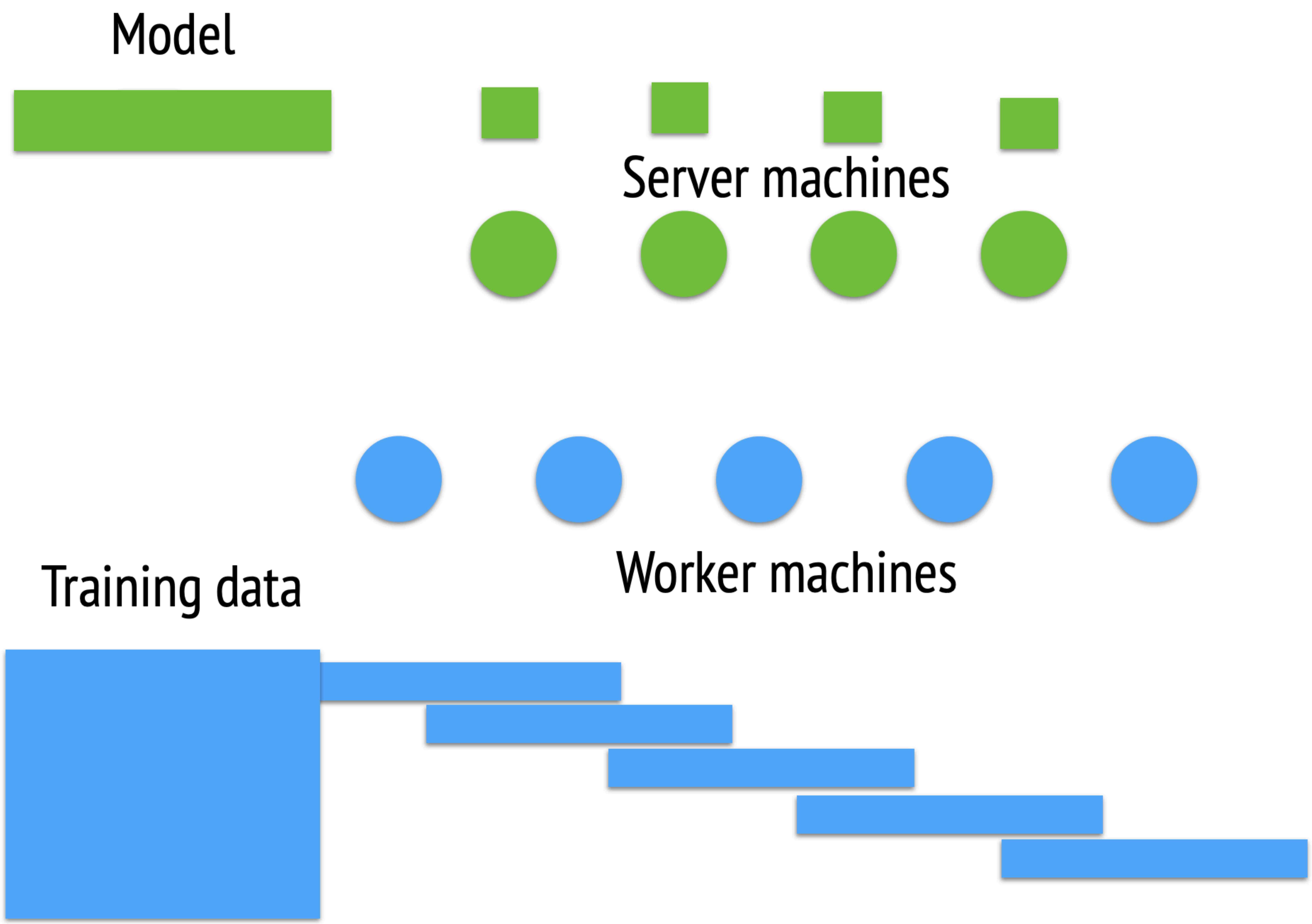
# Data and Model Partitioning



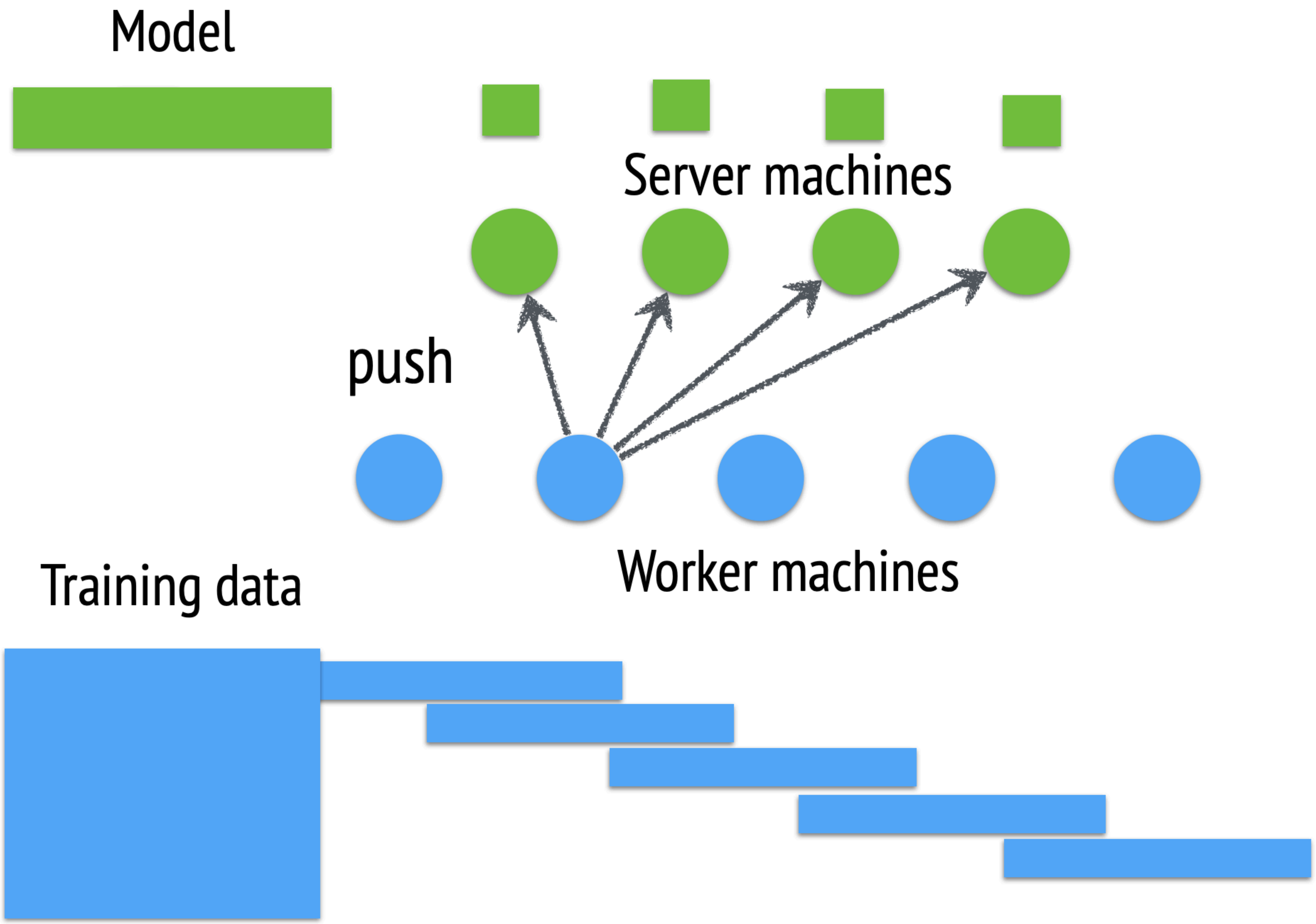
# Data and Model Partitioning



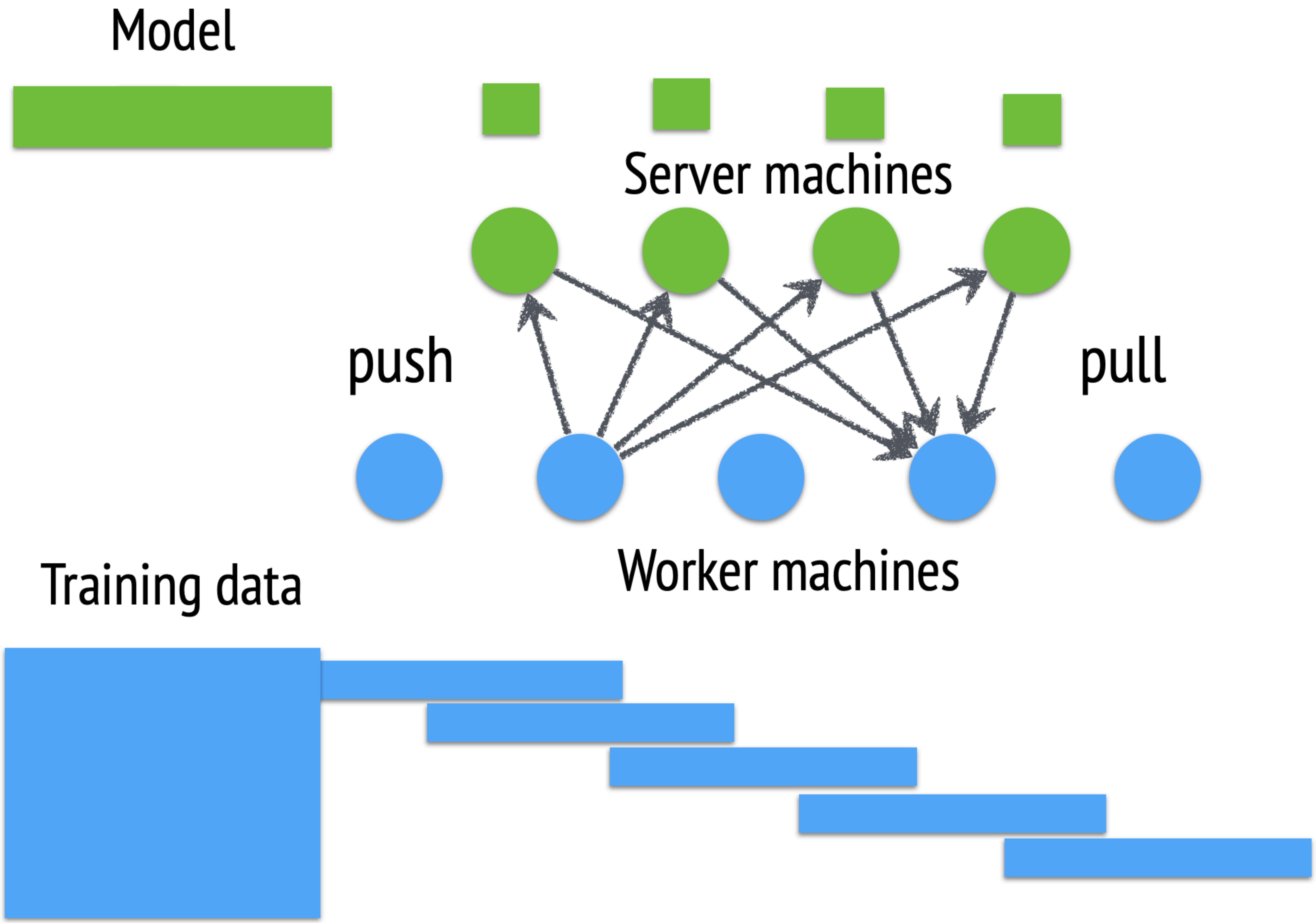
# Data and Model Partitioning



# Communication Operations



# Communication Operations



# Issue with Parameter Server

Even with distributed PS architecture,

there can be network congestion at the parameter servers

Solution: Decentralized Aggregation

# Ring AllReduce - Decentralized Aggregation

# Ring AllReduce

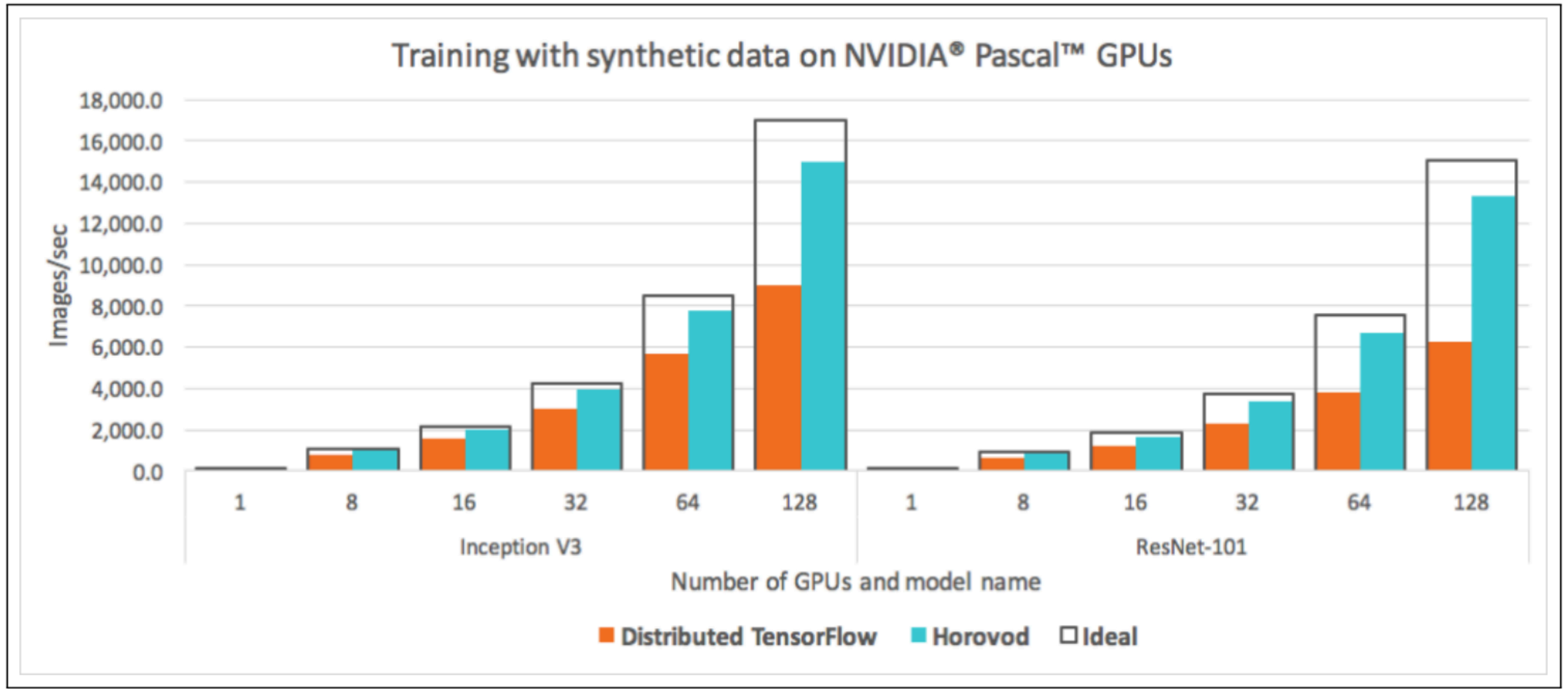
# Ring AllReduce

# Ring AllReduce

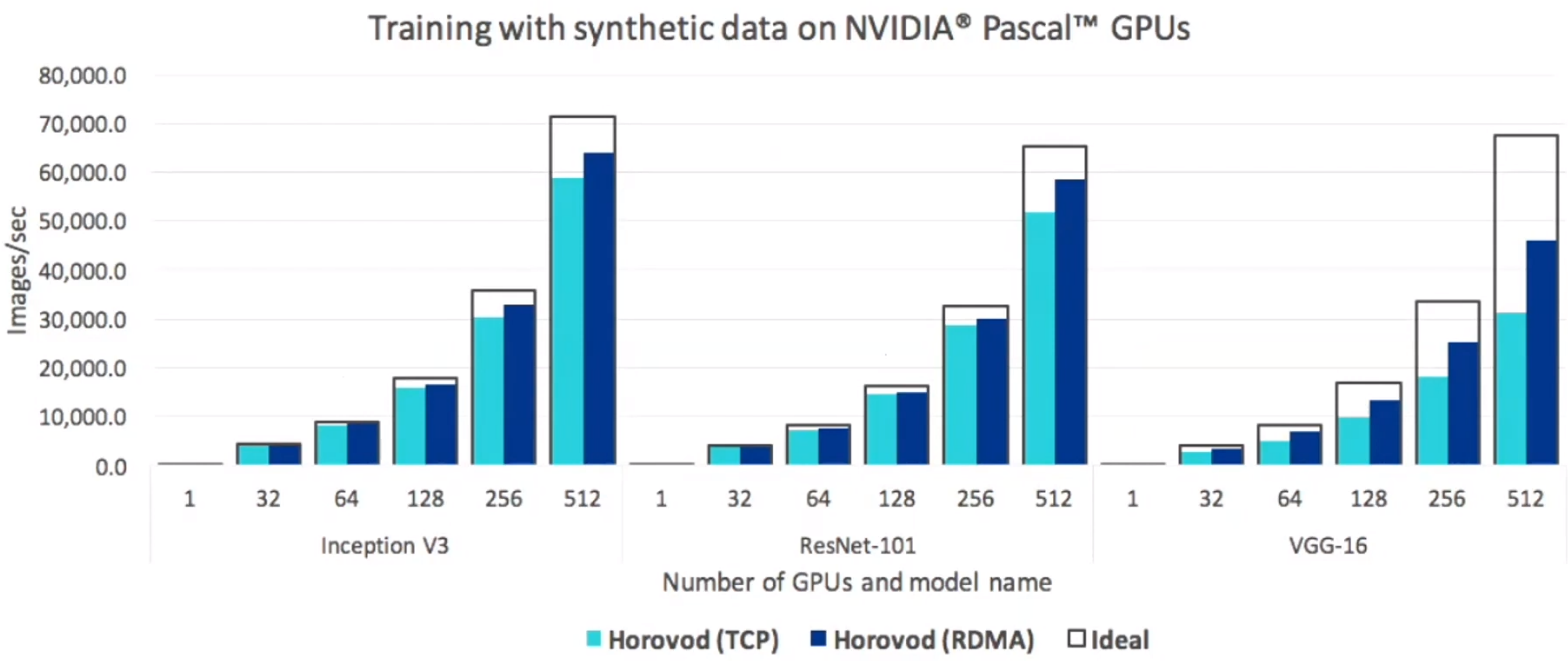
# Ring AllReduce

# Ring AllReduce

# Performance



# Performance



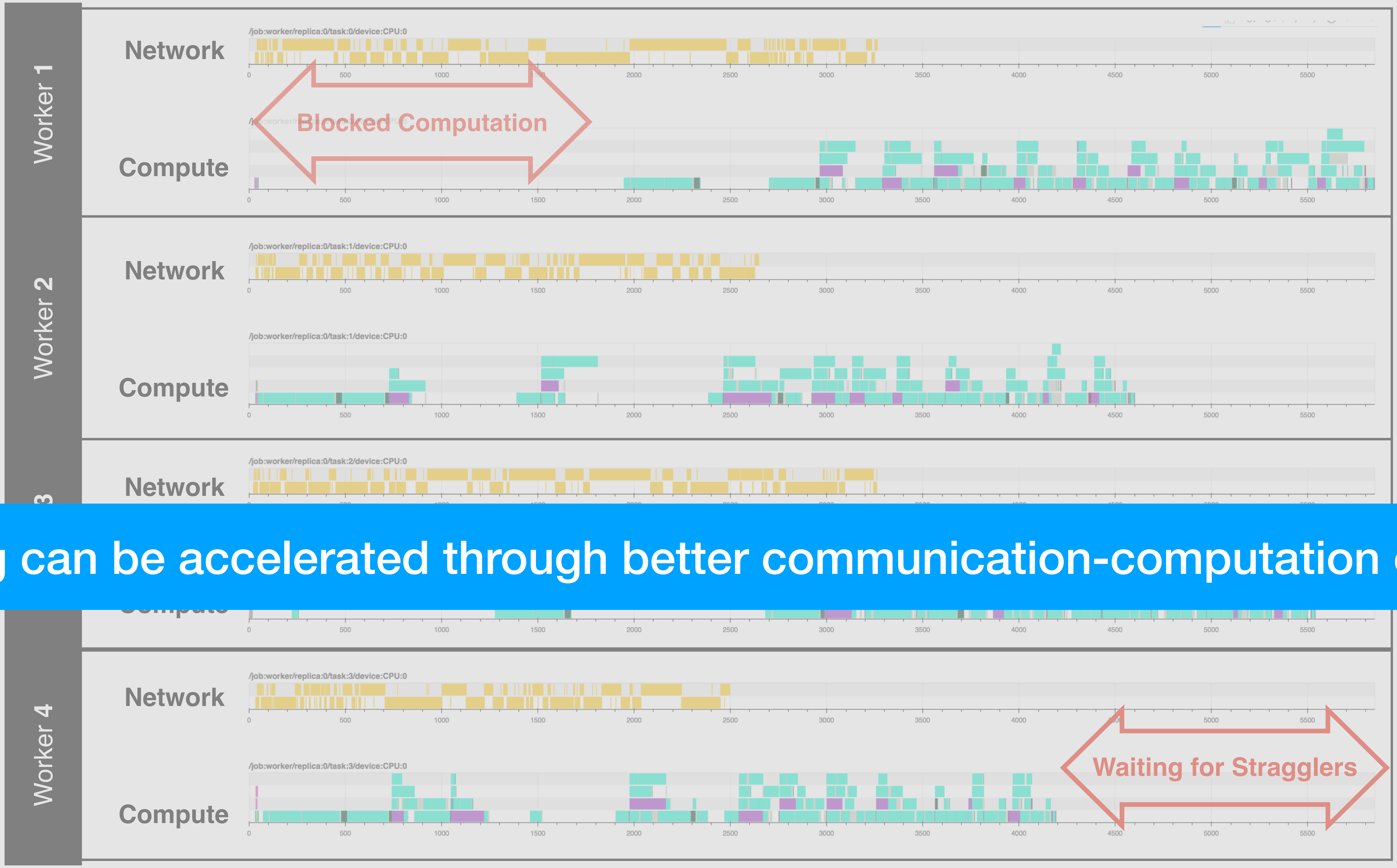
# AllReduce advantages

- Better performance
- More scalable
- Fits well with Torus topology

# An issue with both PS and AllReduce

Compute under-utilization

# Understanding Compute Underutilization

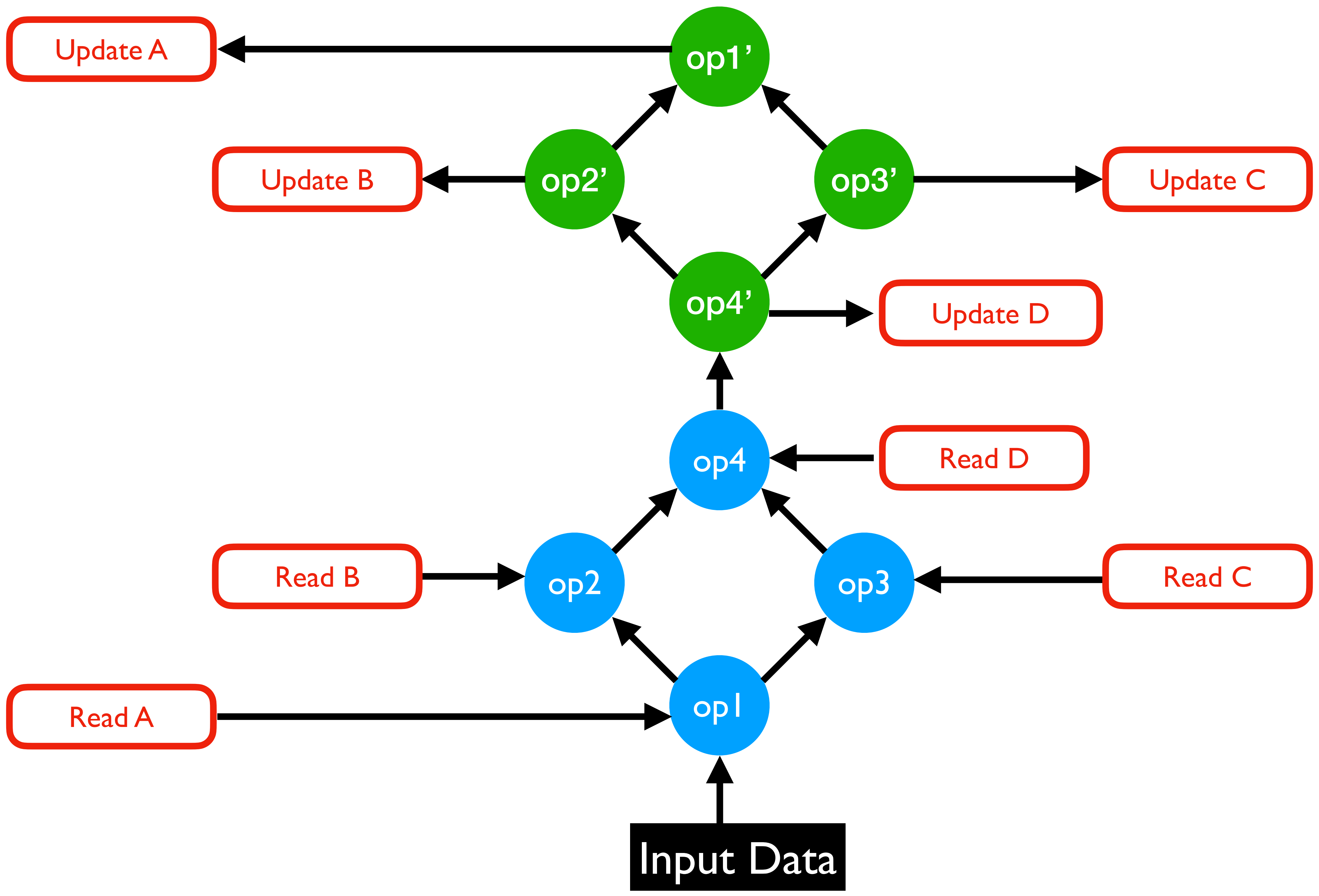



Training can be accelerated through better communication-computation overlap

Inception v3
Data-Parallel with Parameter Server
TensorFlow
Mustang: CPU

# Cause: Random Order of Parameter Transfers

- In this example, the computation cannot start until parameter A is received
- B, C, or D may be transferred before A, thereby blocking the computation
- To make things worse, parameters that are updated last are consumed first



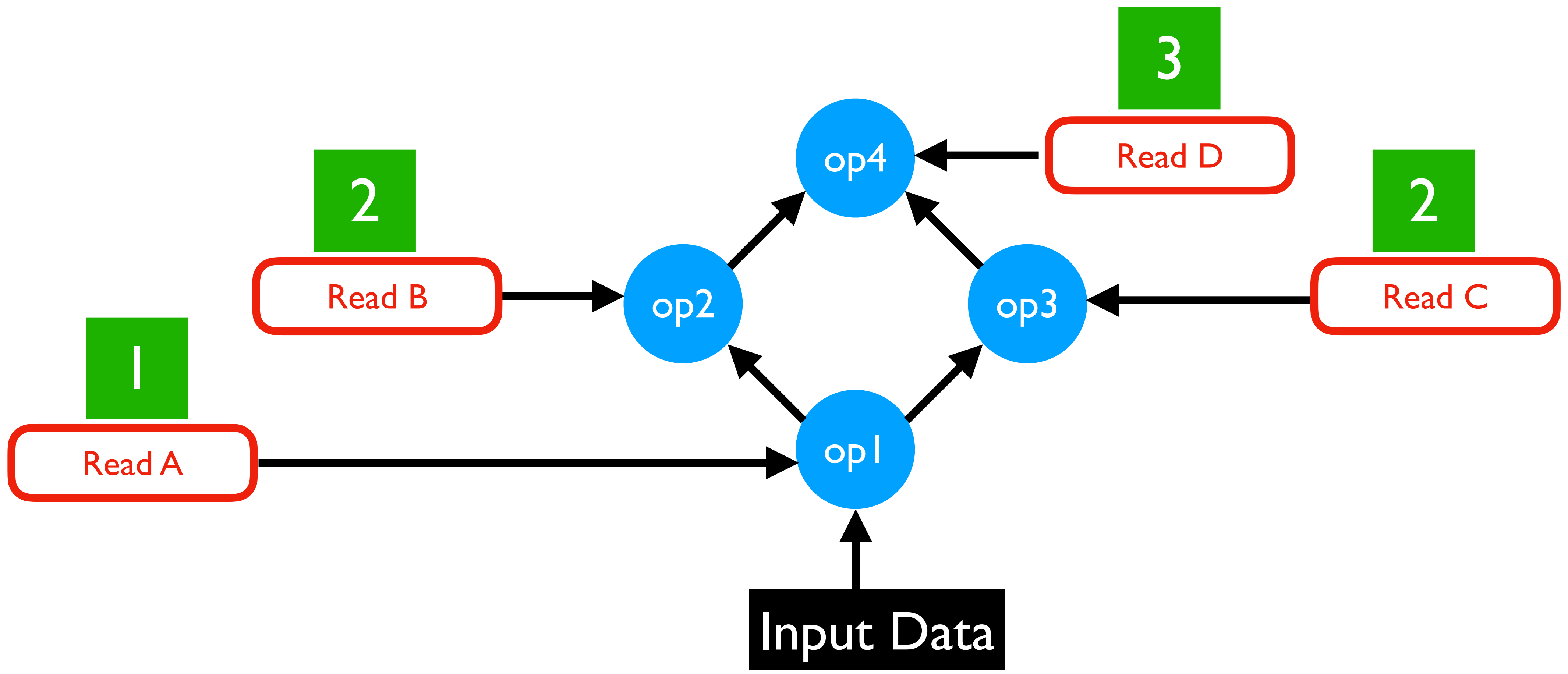
# TicTac and P3 [MLSys'19] High-level idea

- Improve iteration time through better communication-computation overlap in Parameter Server based aggregation

- Achieved through parameter transfer scheduling

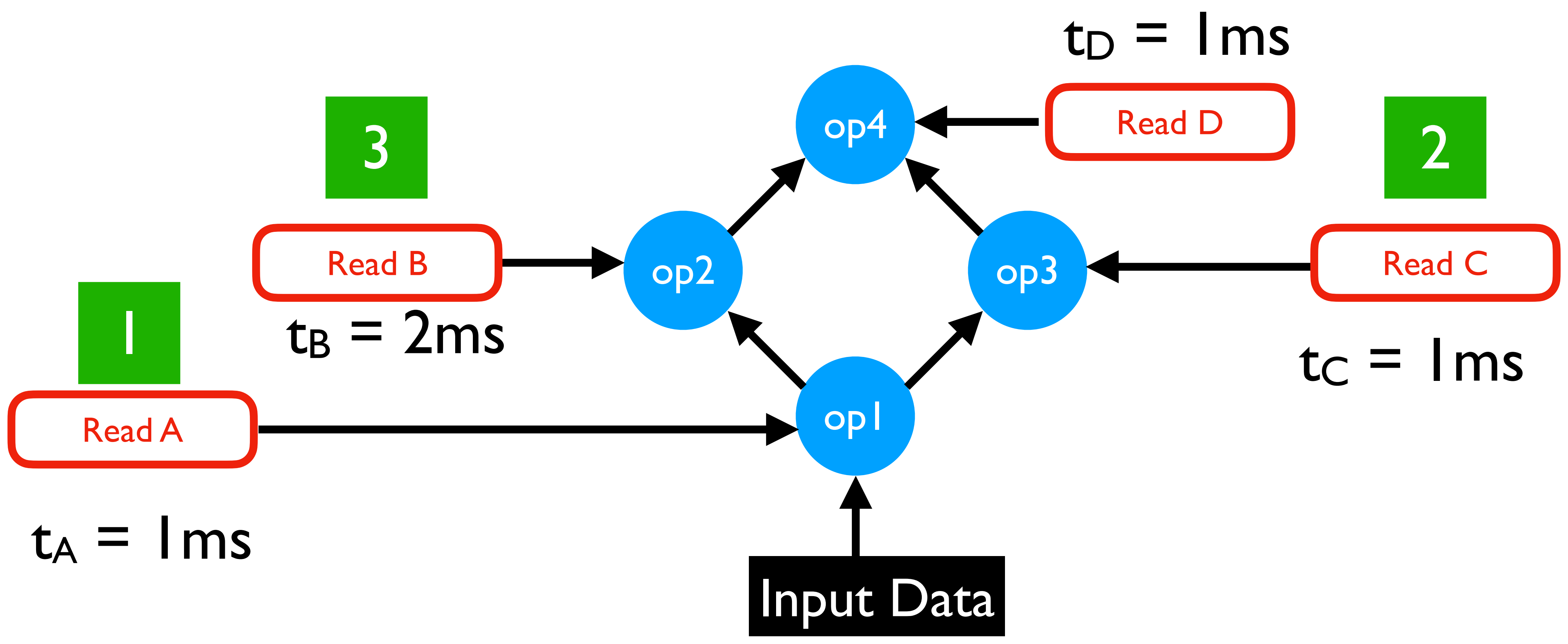
# Timing Independent Computation Scheduling

- Uses DAG structure only
- Assign priorities based on the number of communication operations *dependent* on a given transfer
- In the e.g, A has no other transfers dependent on it. Hence, it gets the highest priority
- B and C each have one dependency. Hence, the next priority
- D assigned lowest priority



# Timing Aware Computation Scheduling

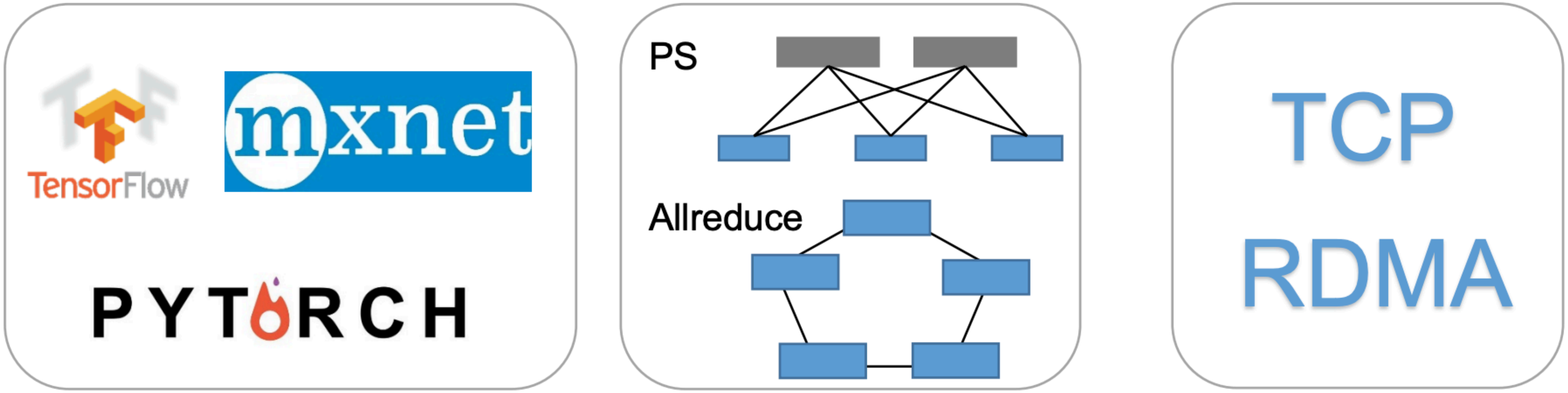
- Uses DAG structure and time taken by each operation
- Reduce blocking on the critical path
- A assigned highest priority
- C is the next smallest blocking transfer
- Followed by B, then D



# Limitations of Past Work

- Coupled with specific framework implementations, e.g., P3 for MXNet PS and TicTac for TensorFlow PS

Many different setups in distributed DNN training:



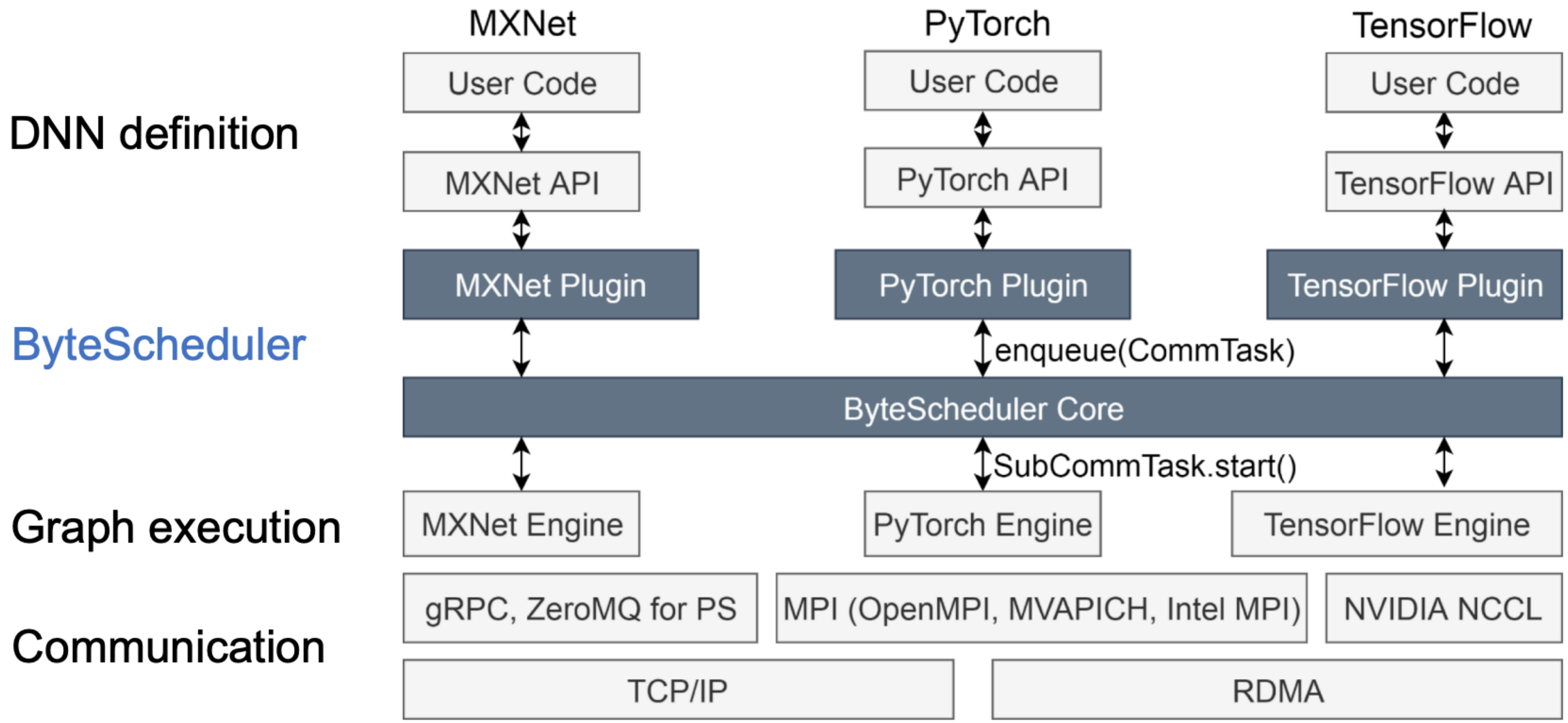

ML frameworks | Communication architectures | Network protocols

# ByteScheduler [SOSP'19]

- Observation: The dependency graph structure is intrinsic for DNN training (regardless of training frameworks, communication architectures, or network protocols)

- ByteScheduler: A generic tensor scheduling framework for deep learning DAGs

# Unified Scheduler Across Frameworks




Plugin: Wrap each communication operation as a CommTask

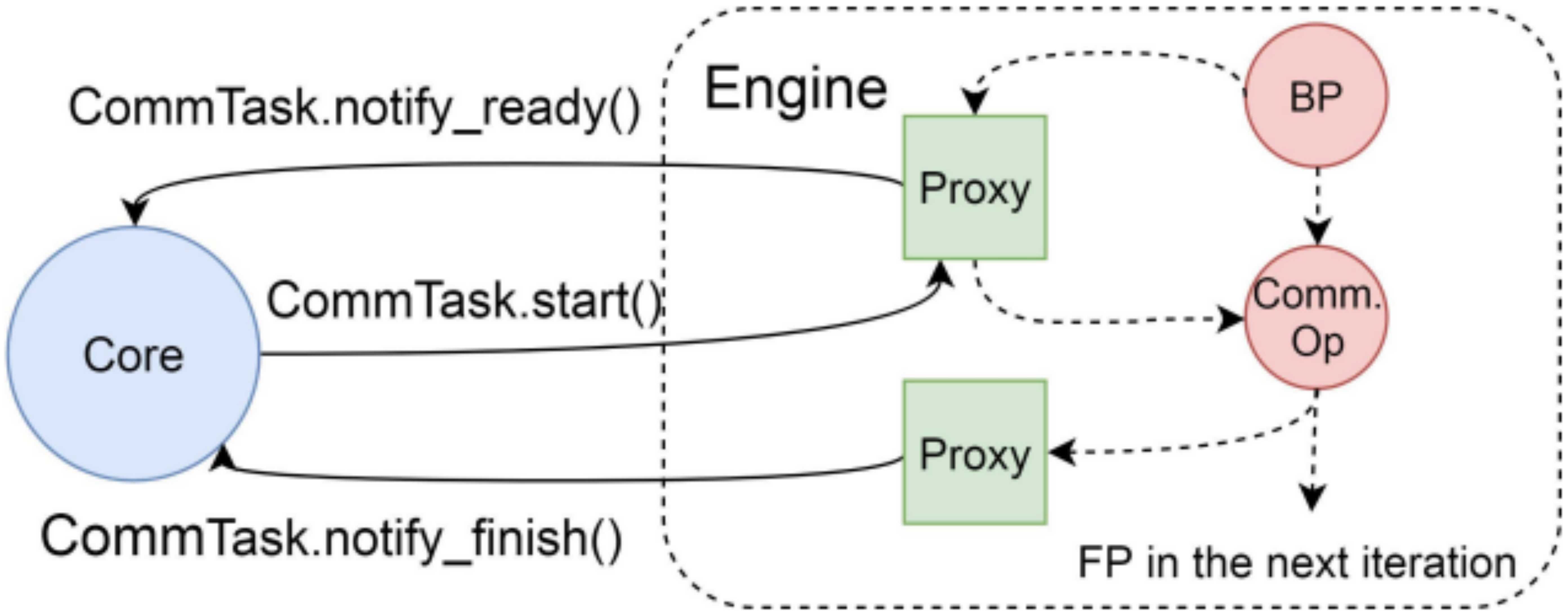
Core: Partition and schedule CommTasks

# CommTask: A Unified Abstraction

- CommTask: A wrapped communication operation, e.g., push one tensor, allreduce one tensor
- CommTask APIs implemented in framework plugins:
    - partition(size): partition a CommTask into SubCommTasks with tensors no larger than a threshold size
    - notify_ready(): notify Core about the readiness of a CommTask
    - start(): start a CommTask by calling the underlying push/pull/all-reduce
    - notify_finish(): notify Core about the completion of a CommTask

# Dependency Proxy: Get the Scheduling Control

- An operator to get the scheduling control from the frameworks to the Core



# Optimal Scheduling Theorem

- For PS, prioritize $push_i$ over $push_j$, and $pull_i$ over $pull_j$, $\forall i < j$
- For all-reduce, prioritize $allreduce_i$ over $allreduce_j$, $\forall i < j$

# Other Research Directions / Challenges

- Training in Heterogeneous GPU/CPU Clusters
- Federated Machine Learning over the Wide Area Network
- In-Network Aggregation
- Topology adaptation for DNN training workloads

Thanks!