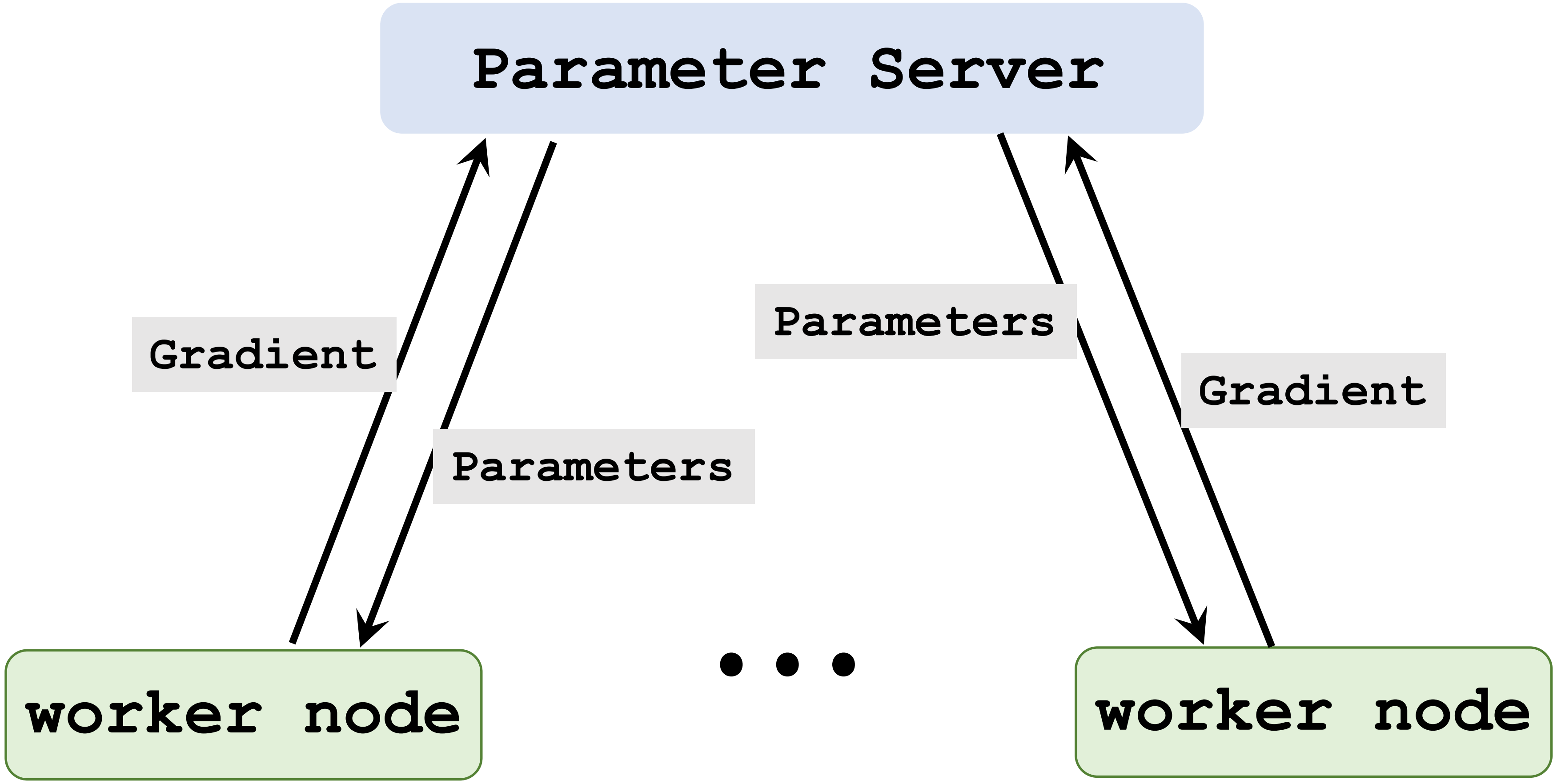
Parallel Computing for Machine Learning
(Part 2)

Shusen Wang

Asynchronous Parallel Gradient Descent

Using Parameter Server

Parameter Server's Architecture



The Parameter Server

- The parameter server was proposed by [1] for scalable machine learning.
- **Characters:** client-server architecture, message-passing communication, and asynchronous.
- (Note that MapReduce is bulk synchronous.)

Reference

1. Li and others: Scaling distributed machine learning with the parameter server. In *OSDI*, 2014.

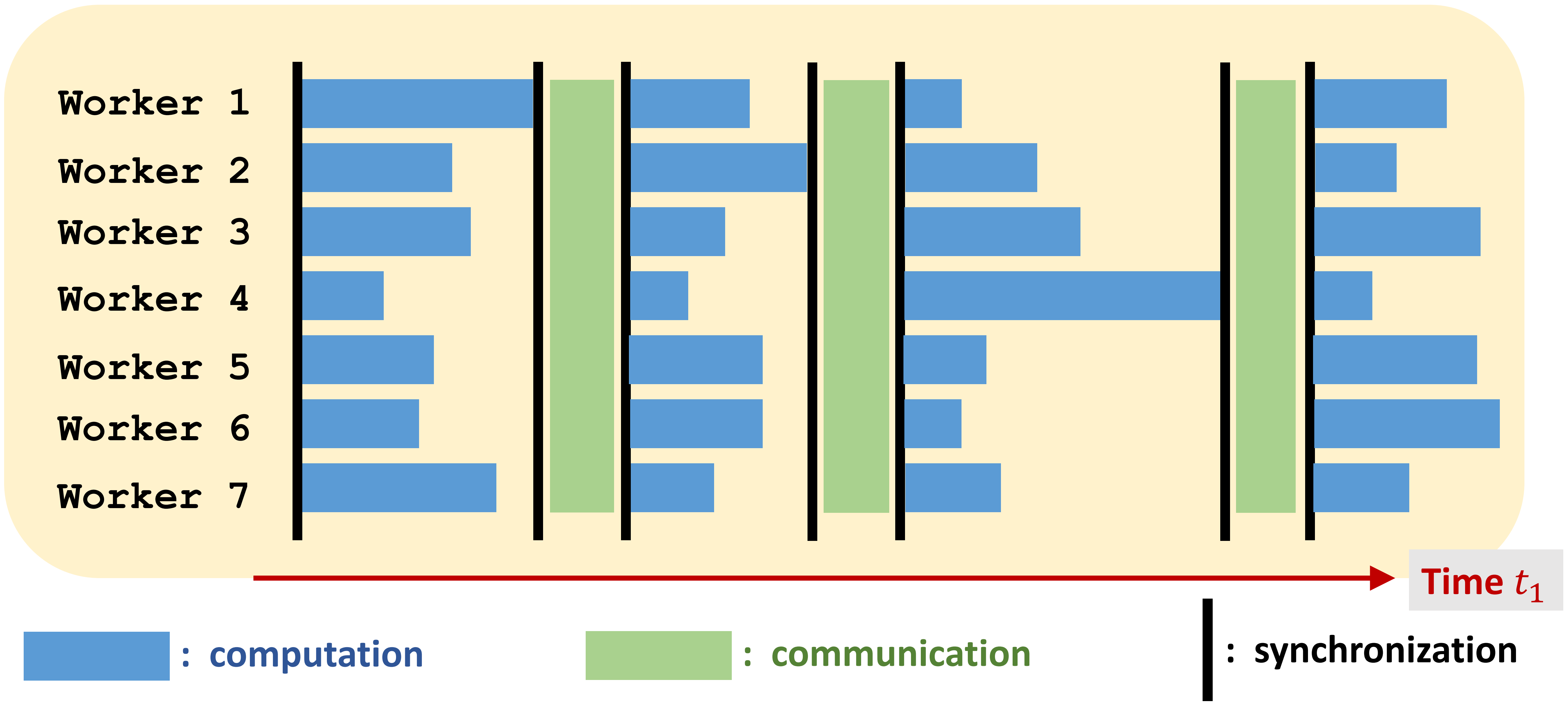
The Parameter Server

- The parameter server was proposed by [1] for scalable machine learning.
- **Characters:** client-server architecture, message-passing communication, and asynchronous.
- (Note that MapReduce is bulk synchronous.)
- Ray [2], an open-source software system, supports parameter server.

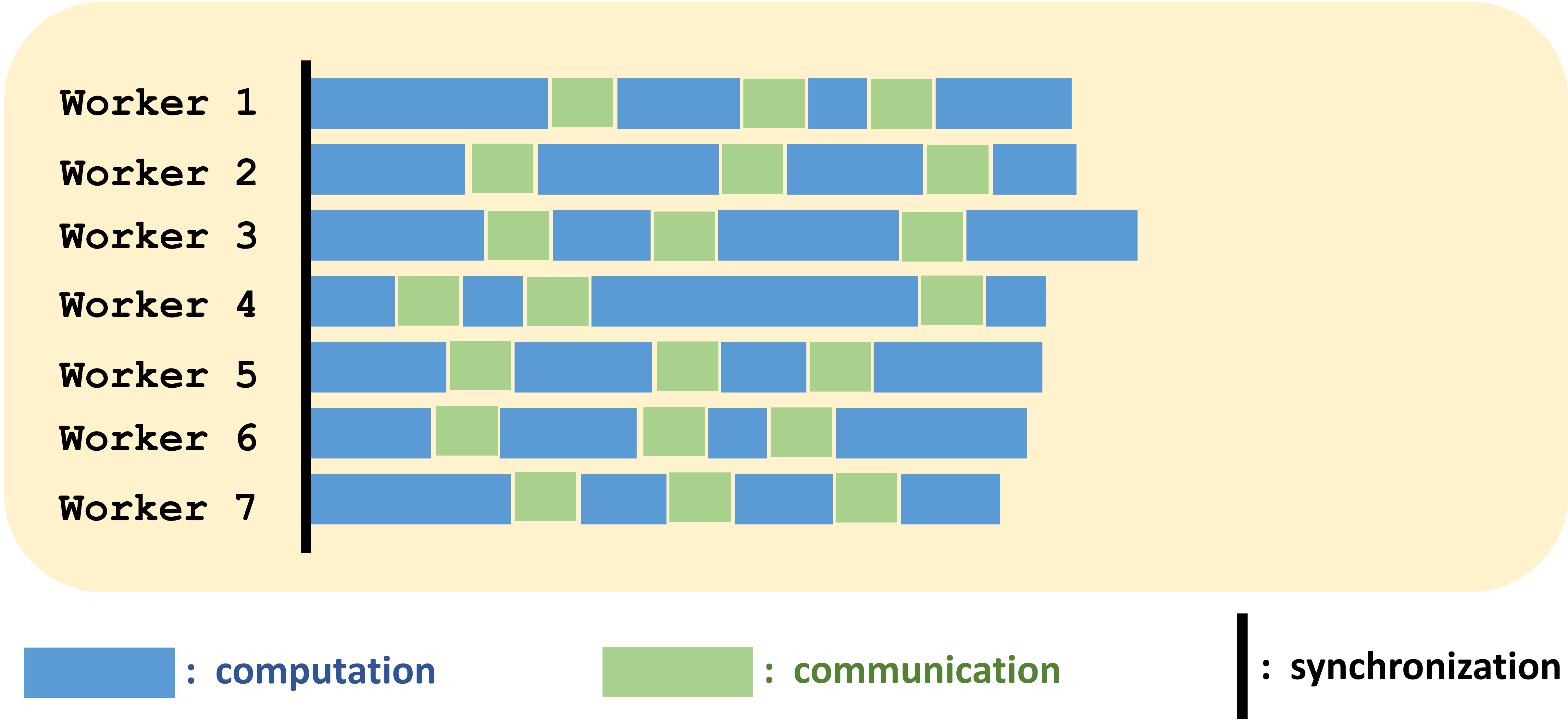
Reference

1. Li and others: Scaling distributed machine learning with the parameter server. In *OSDI*, 2014.
2. Moritz and others: Ray: A distributed framework for emerging AI applications. In *OSDI*, 2018.

Let us recall synchronous algorithm

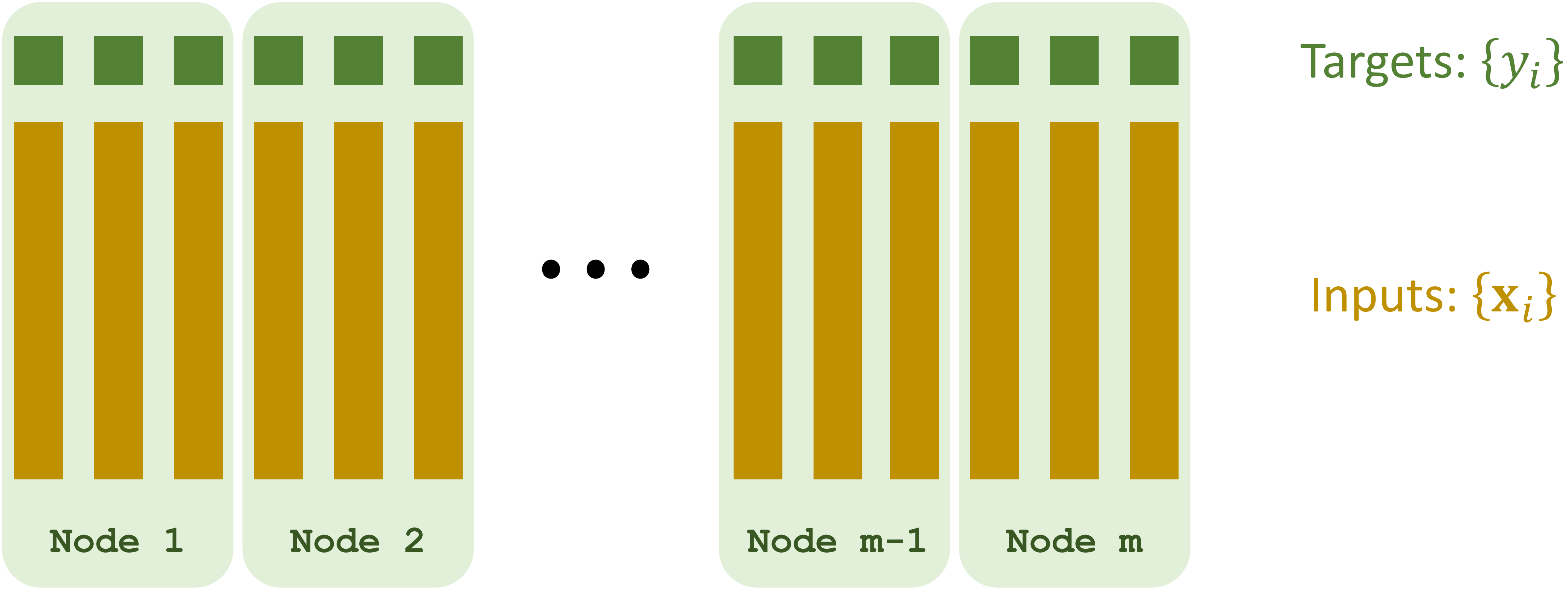


Asynchronous algorithm



Asynchronous Gradient Descent

- Partition the data among worker nodes. (A node has a subset of data.)



Asynchronous Gradient Descent

The i-th worker repeats:

1. Pull the up-to-date model parameters $\mathbf{w}$ from the server.
2. Compute gradient $\tilde{\mathbf{g}}_i$ using its local data and $\mathbf{w}$.
3. Push $\tilde{\mathbf{g}}_i$ to the server.

The server performs:

1. Receive gradient $\tilde{\mathbf{g}}_i$ from a worker.
2. Update the parameters by:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \tilde{\mathbf{g}}_i .$$

Reference

1. Niu and others: Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, 2011.

Asynchronous Gradient Descent

The i-th worker repeats:

1. Pull the up-to-date model parameters $\mathbf{w}$ from the server.
2. Compute gradient $\tilde{\mathbf{g}}_i$ using its local data and $\mathbf{w}$.
3. Push $\tilde{\mathbf{g}}_i$ to the server.

The server performs:

1. Receive gradient $\tilde{\mathbf{g}}_i$ from a worker.
2. Update the parameters by:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \tilde{\mathbf{g}}_i .$$

Reference

1. Niu and others: Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, 2011.

Pro and Con of Asynchronous Algorithms

- In practice, asynchronous algorithms are faster than the synchronous.
- In theory, asynchronous algorithms has slower convergence rate.
- Asynchronous algorithms have restrictions, e.g., a worker cannot be much slower than the others. (Why?)

Pro and Con of Asynchronous Algorithms

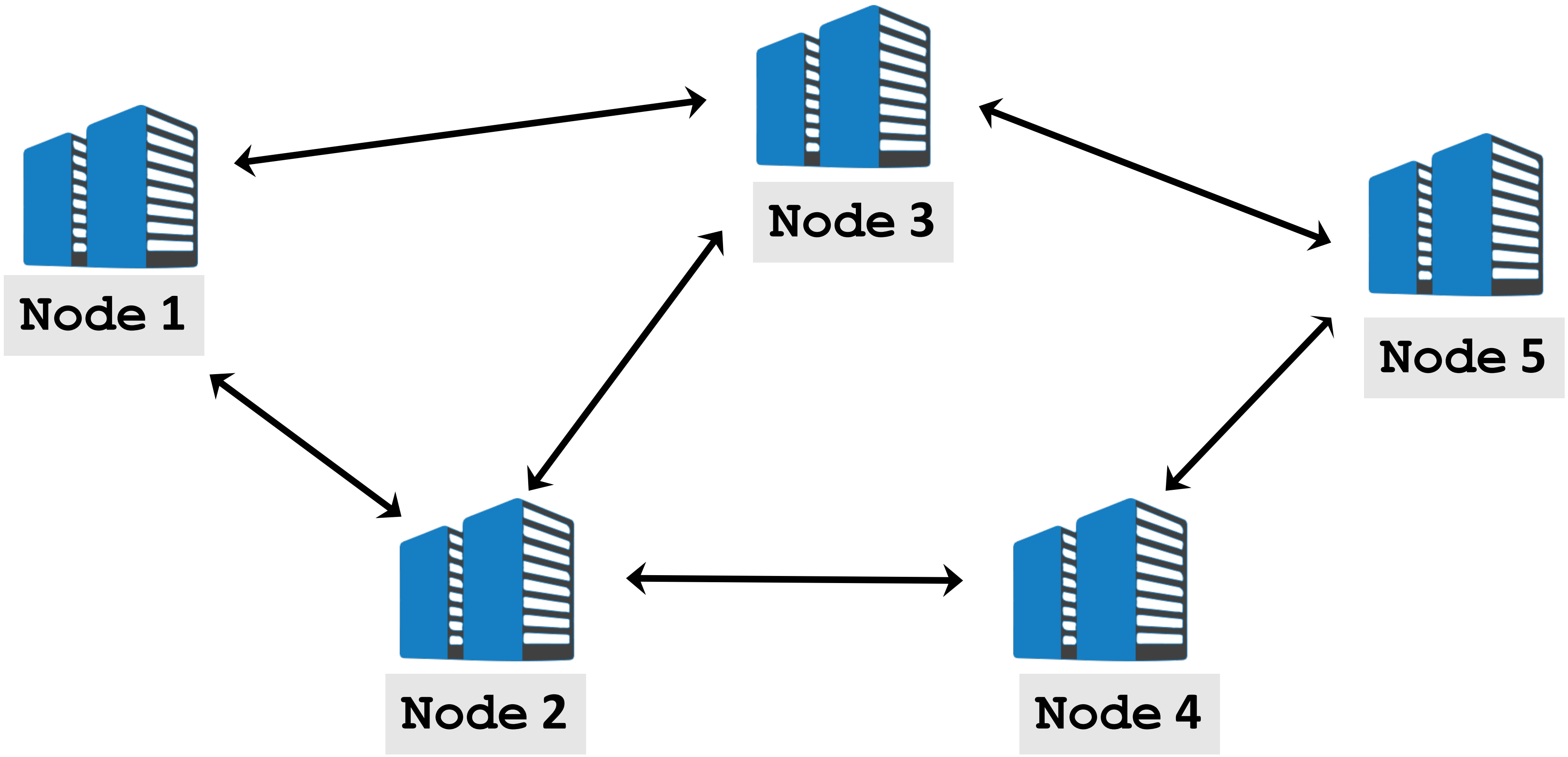
Question: What if a worker is too slow?

- At time t_1, the parameters in the server have been updated many times.
- Worker 3's gradient is based on very old parameters (at time t_0)
- ➔ Worker 3's gradient is harmful!

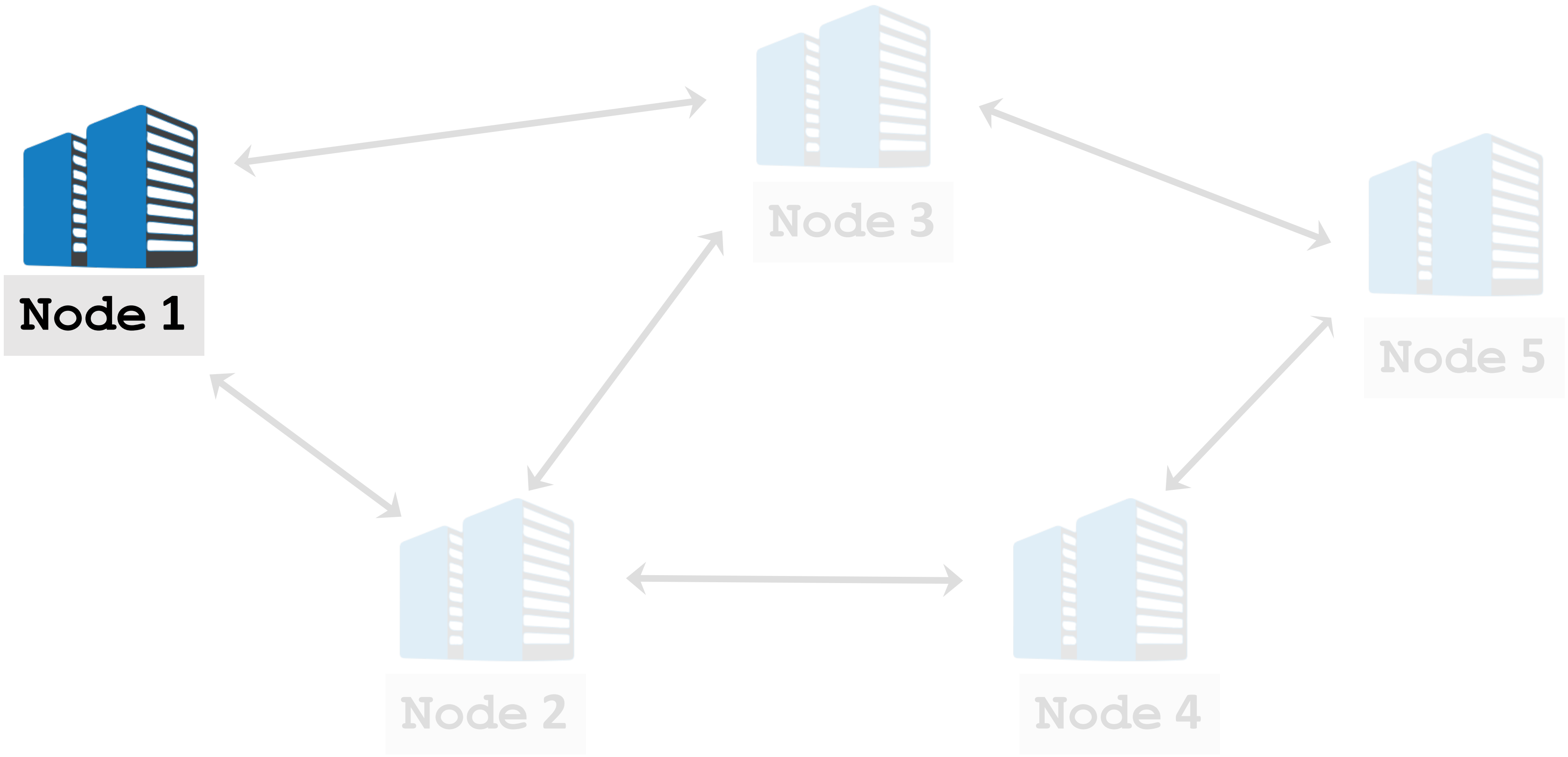
Parallel Gradient Descent in Decentralized Network

Decentralized Network

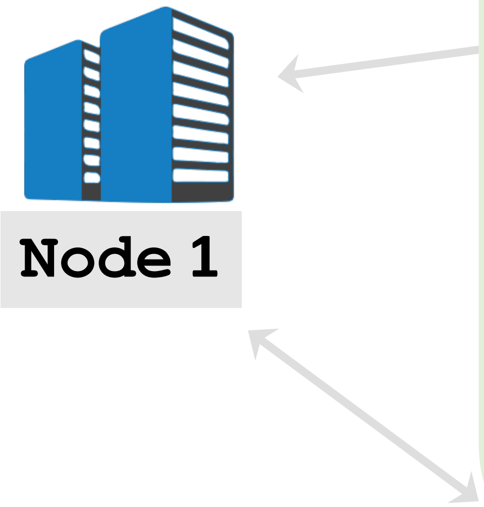
- Characters: peer-to-peer architecture (no central server), message-passing communication, a node communicate with its neighbors.



Decentralized Gradient Descent

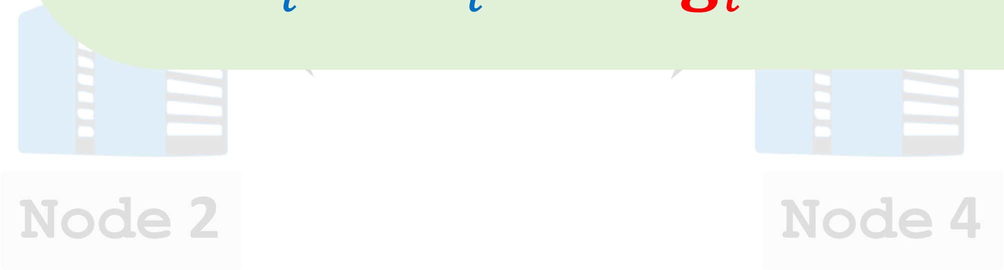


Decentralized Gradient Descent



The i-th node repeats:

1. Compute gradient $\tilde{\mathbf{g}}_i$ using its local data and current parameters $\tilde{\mathbf{w}}_i$.
2. Pull the parameters from its neighbors, denote $\{\tilde{\mathbf{w}}_k\}$.
3. $\tilde{\mathbf{w}}_i \leftarrow$ weighted average of $\tilde{\mathbf{w}}_i$ and $\{\tilde{\mathbf{w}}_k\}$.
4. $\tilde{\mathbf{w}}_i \leftarrow \tilde{\mathbf{w}}_i - \alpha \cdot \tilde{\mathbf{g}}_i$.



Theories of Decentralized Algorithms

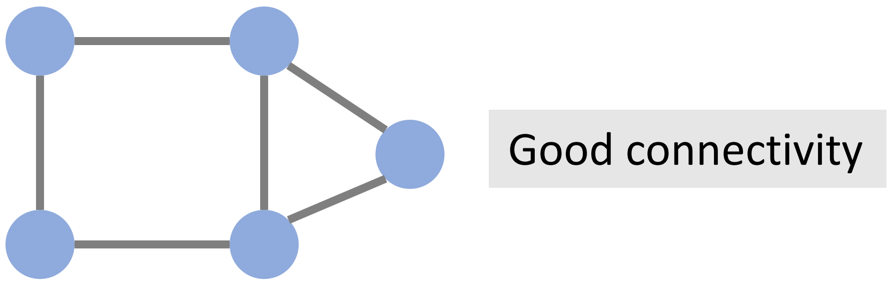
- Decentralized GD and SGD are guaranteed to converge, e.g., [1].

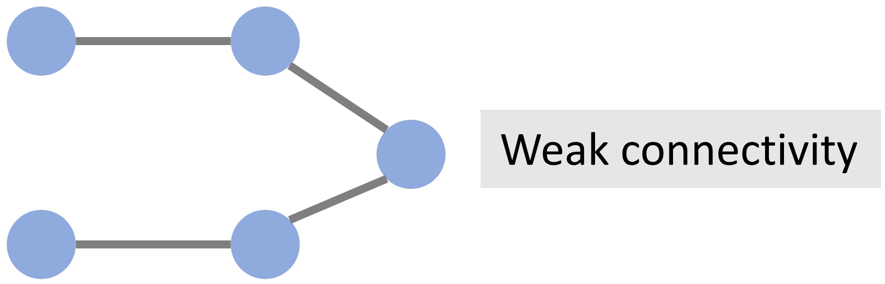
Reference

1. Lian and others: Can decentralized algorithms outperform centralized algorithms? In *NIPS*, 2017.

Theories of Decentralized Algorithms

- Decentralized GD and SGD are guaranteed to converge, e.g., [1].
- Convergence rate depends on how well the nodes are connected.
  - If the nodes are well connected, then it has fast convergence.



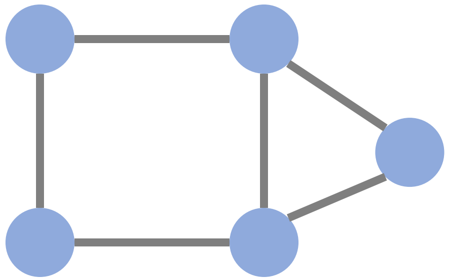


Reference

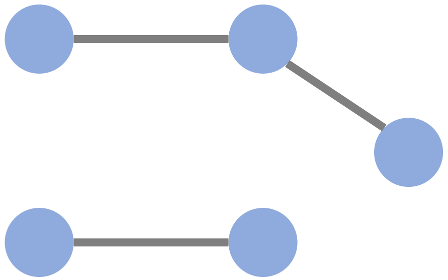
1. Lian and others: Can decentralized algorithms outperform centralized algorithms? In *NIPS*, 2017.

Theories of Decentralized Algorithms

- Decentralized GD and SGD are guaranteed to converge, e.g., [1].
- Convergence rate depends on how well the nodes are connected.
    - If the nodes are well connected, then it has fast convergence.
    - If the graph is not strongly connected, then it does not converge.



Good connectivity



Not strongly connected

Reference

1. Lian and others: Can decentralized algorithms outperform centralized algorithms? In *NIPS*, 2017.

Summary

Parallel Computing

- Why? To make the wall-clock runtime shorter.
- How? Use multiple processors and/or multiple nodes.

Important Concepts

- Communication: sharing memory **V.S.** message passing.
- Architecture: client-server **V.S.** peer-to-peer.

Important Concepts

- Communication: sharing memory **V.S.** message passing.
- Architecture: client-server **V.S.** peer-to-peer.
- Synchronization: bulk synchronous **V.S.** asynchronous.

Important Concepts

- Communication: sharing memory **V.S.** message passing.
- Architecture: client-server **V.S.** peer-to-peer.
- Synchronization: bulk synchronous **V.S.** asynchronous.
- Parallelism: data parallelism (more popular) **V.S.** model parallelism.

Parallel Programming Models

- MapReduce: Message passing, client-server, and synchronous.
- Parameter Server: Message passing, client-server, and asynchronous.
- Decentralized: Message passing, peer-to-peer, synchronous or asynchronous.

Parallel Computing v.s. Distributed Computing

Distributed computing is a kind of parallel computing.

Question: What is the difference?

- It is not black and white. No consensus in the academia.

Parallel Computing v.s. Distributed Computing

Distributed computing is a kind of parallel computing.

Question: What is the difference?

- It is not black and white. No consensus in the academia.
- HPC people's opinion:
  - When the compute nodes are not in the physical locations, parallel computing is called distributed computing.
- ML people's opinion:
  - When the data or model are partitioned among multiple nodes, parallel computing is called distributed computing.
  - In contrast, computation in one node (which has many processors) is not distributed computing.

Thank you!