Scaling Distributed Machine Learning with the Parameter Server

Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su

Presented by: *Liang Gong*

CS294-110 Class Presentation

Fall 2015

Machine Learning in Industry

- Large training dataset (1TB to 1PB)
- Complex models (10^9 to 10^{12} parameters)
- → ML must be done in distributed environment

- Challenges:
  - Many machine learning algorithms are proposed for sequential execution
  - Machines can fail and jobs can be preempted

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Motivation

Balance the need of performance, flexibility and generality of machine learning algorithms, and the simplicity of systems design.

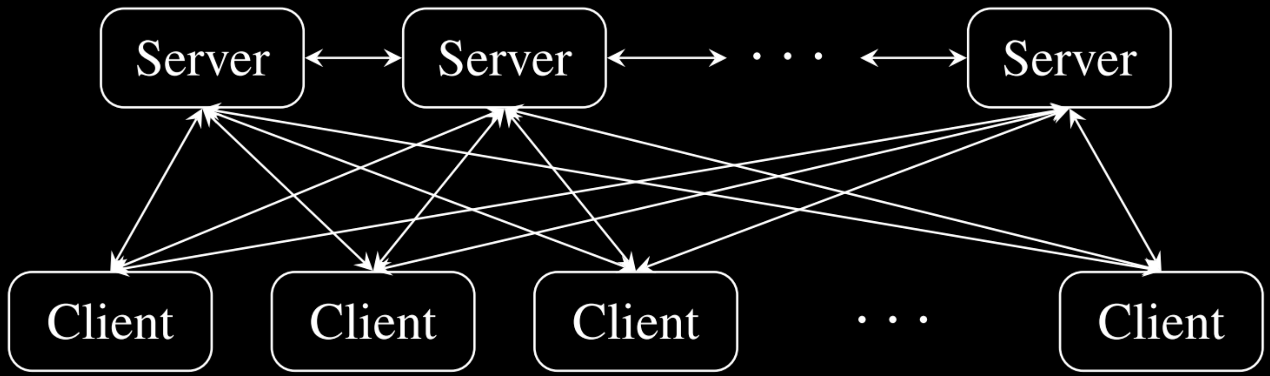
How to:

- Distribute workload
- Share the model among all machines
- Parallelize sequential algorithms
- Reduce communication cost

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Main Idea of Parameter Server

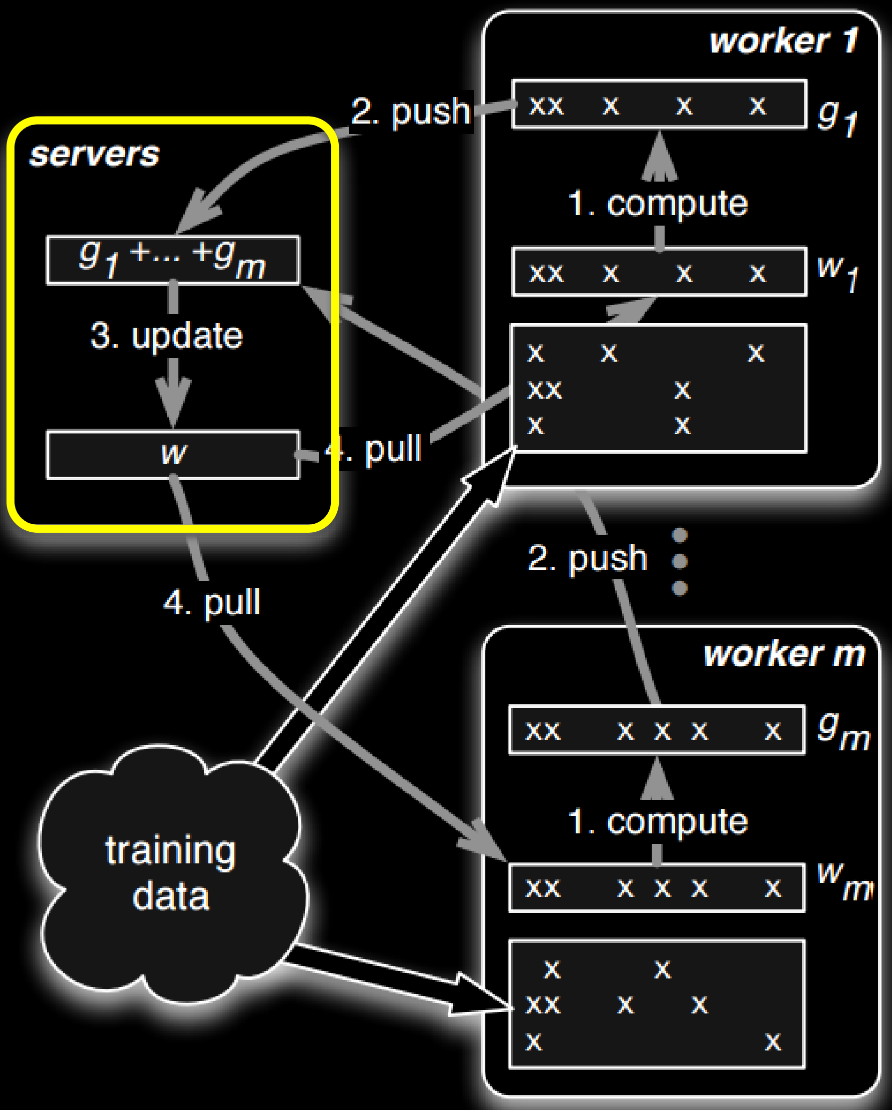
- Servers manage parameters
- Worker Nodes are responsible for computing updates (training) for parameters based on part of the training dataset
- Parameter updates derived from each node are pushed and aggregated on the server.



Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

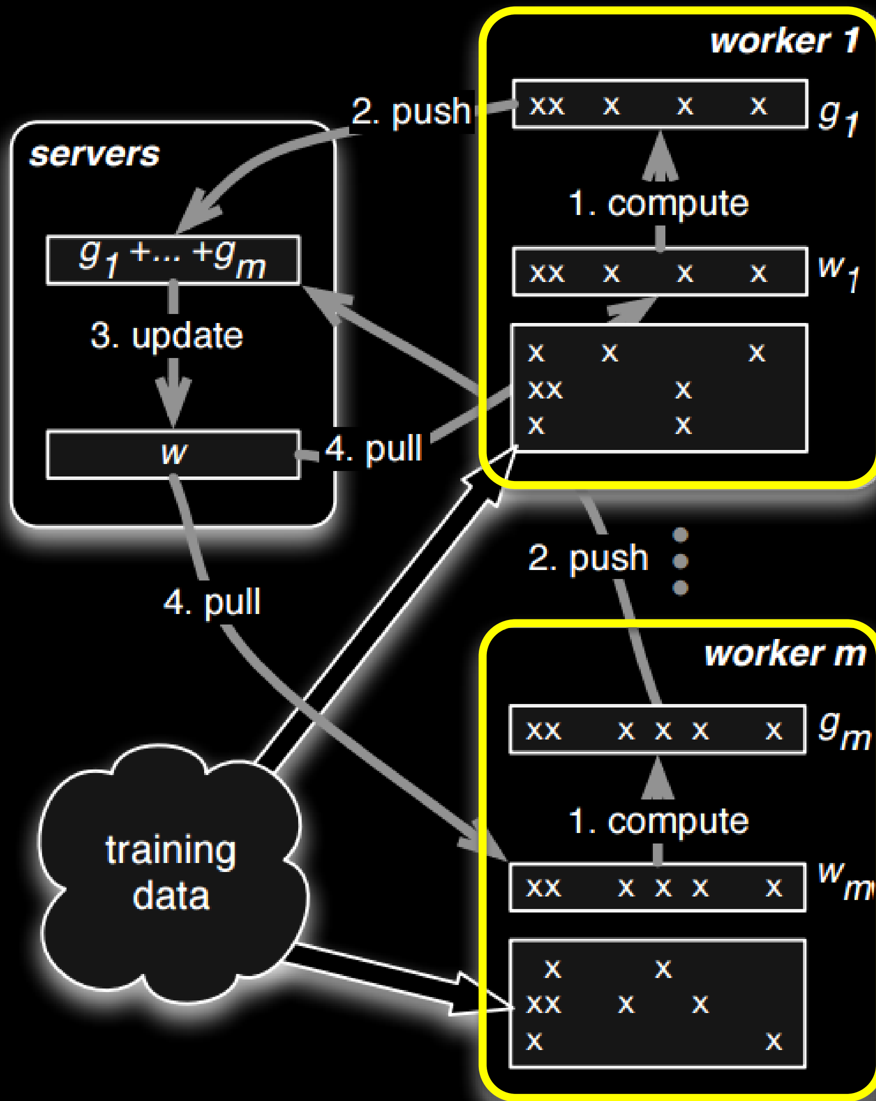
A Simple Example

- Server node



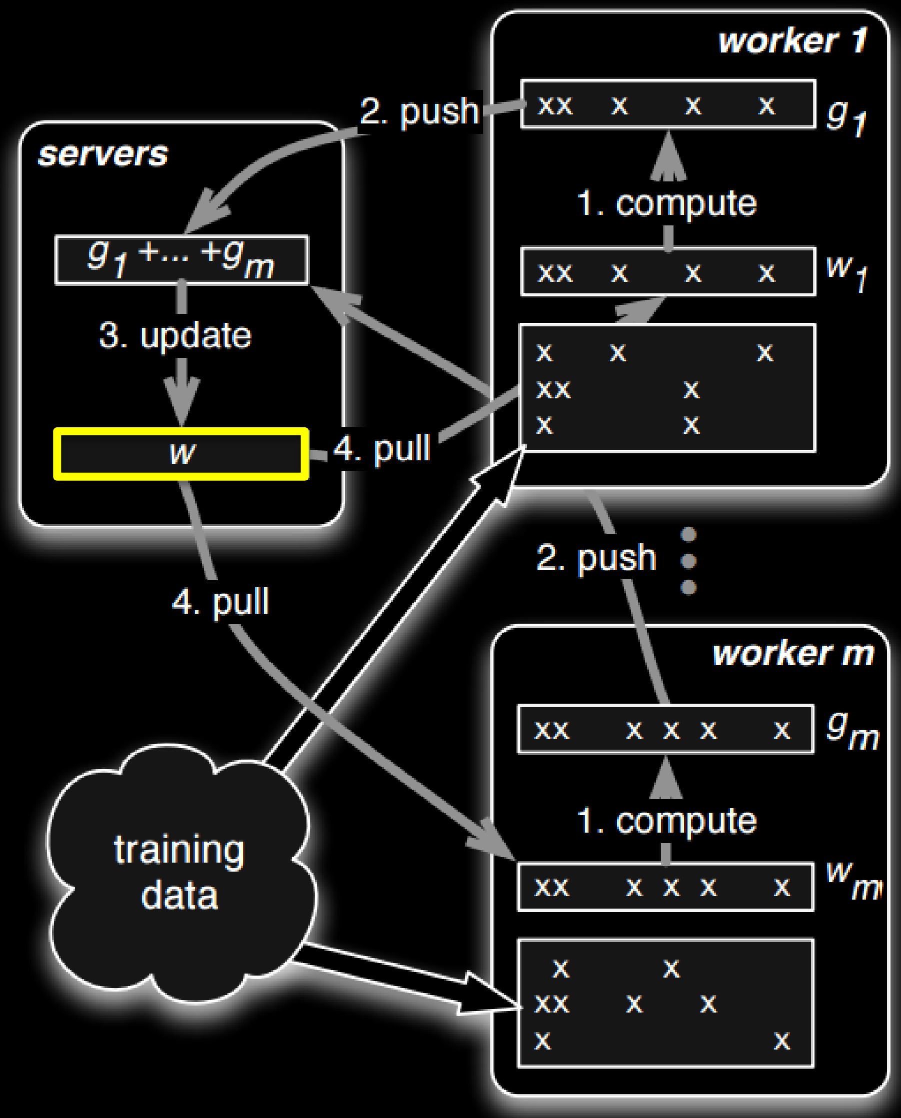
A Simple Example

- Server node + worker nodes



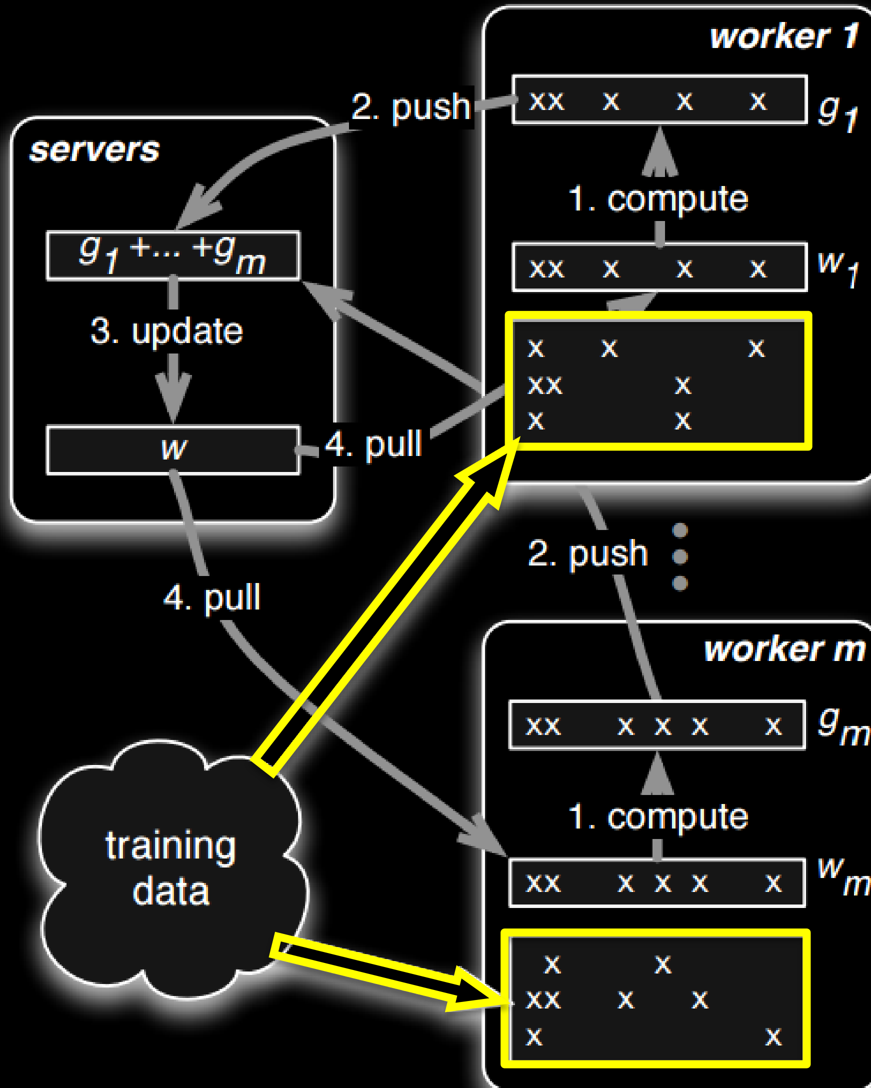
A Simple Example

- Server node + worker nodes
- Server node: all parameters



A Simple Example

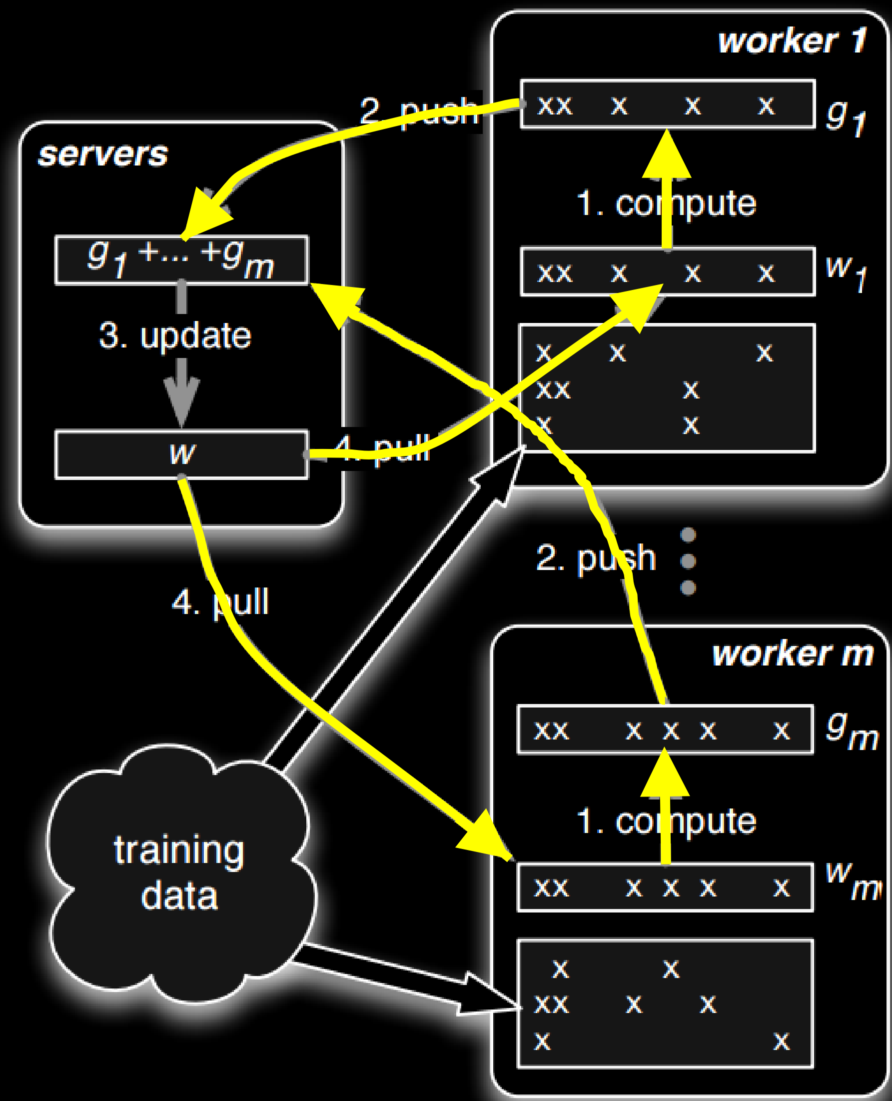
- Server node + worker nodes
- Server node: all parameters
- Worker node: owns part of the training data



A Simple Example

- Server node + worker nodes
- Server node: all parameters
- Worker node: owns part of the training data

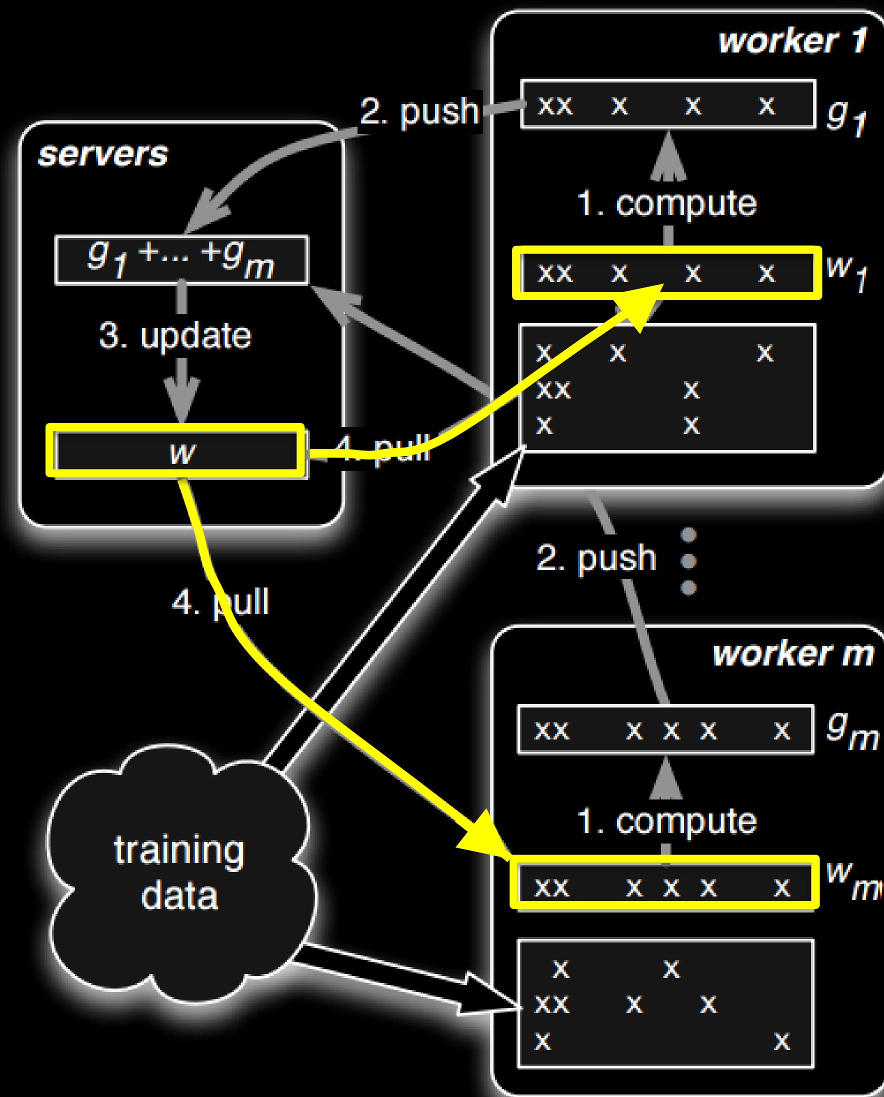
- Operates in iterations



A Simple Example

- Server node + worker nodes
- Server node: all parameters
- Worker node: owns part of the training data

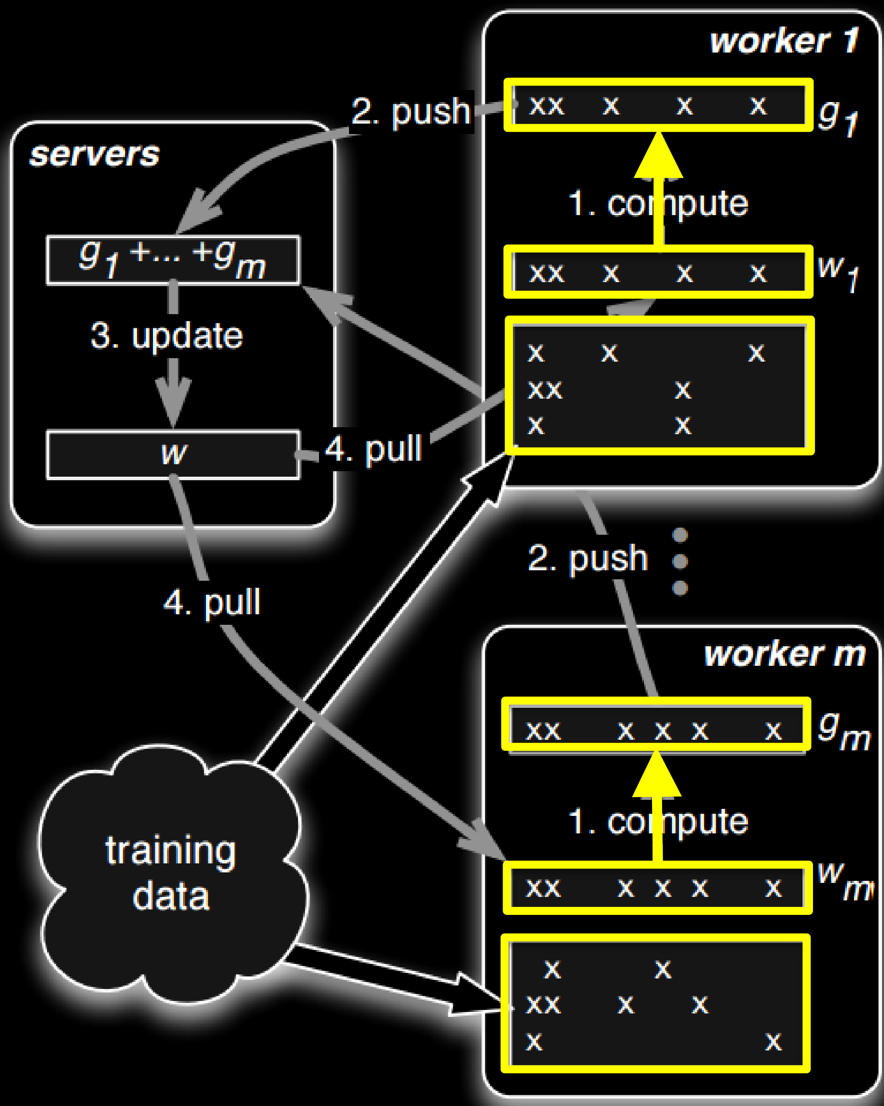
- Operates in iterations
- Worker nodes pull the updated w



A Simple Example

- Server node + worker nodes
- Server node: all parameters
- Worker node: owns part of the training data

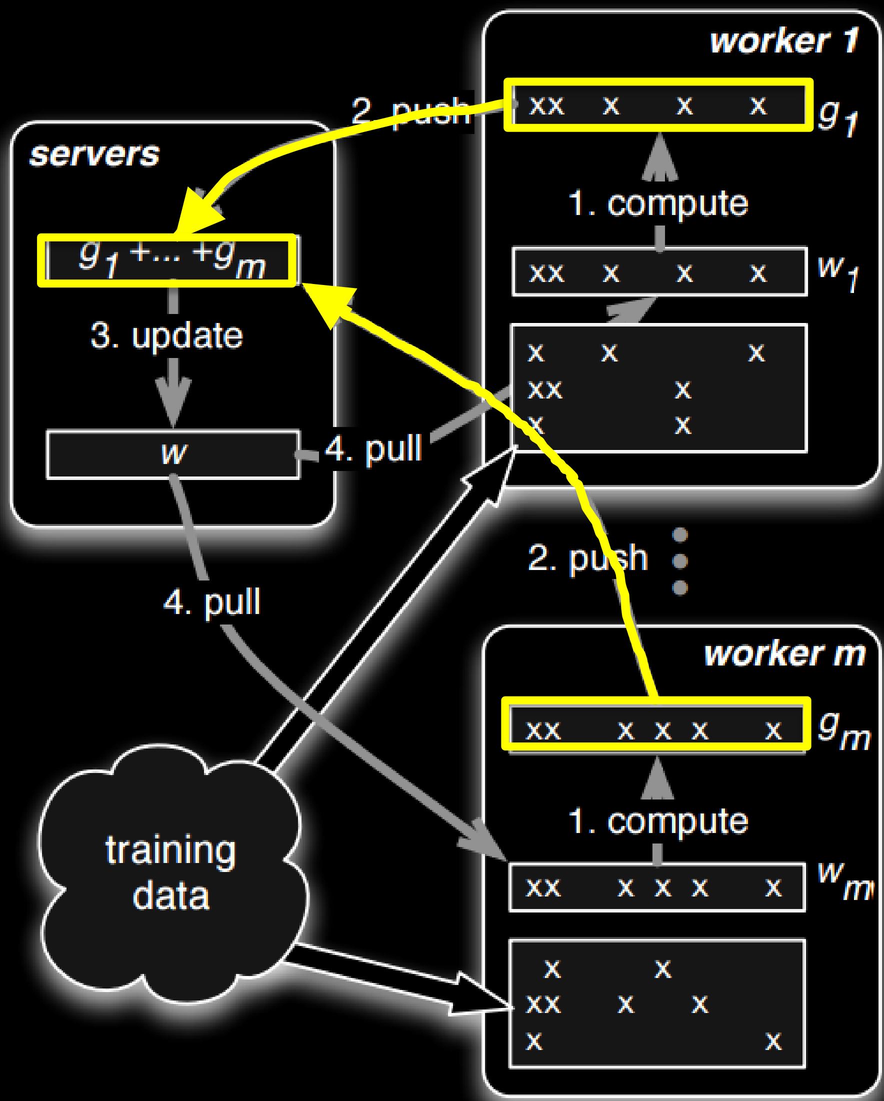
- Operates in iterations
- Worker nodes pull the updated w
- Worker node computes updates to w (local training)



A Simple Example

- Server node + worker nodes
- Server node: all parameters
- Worker node: owns part of the training data

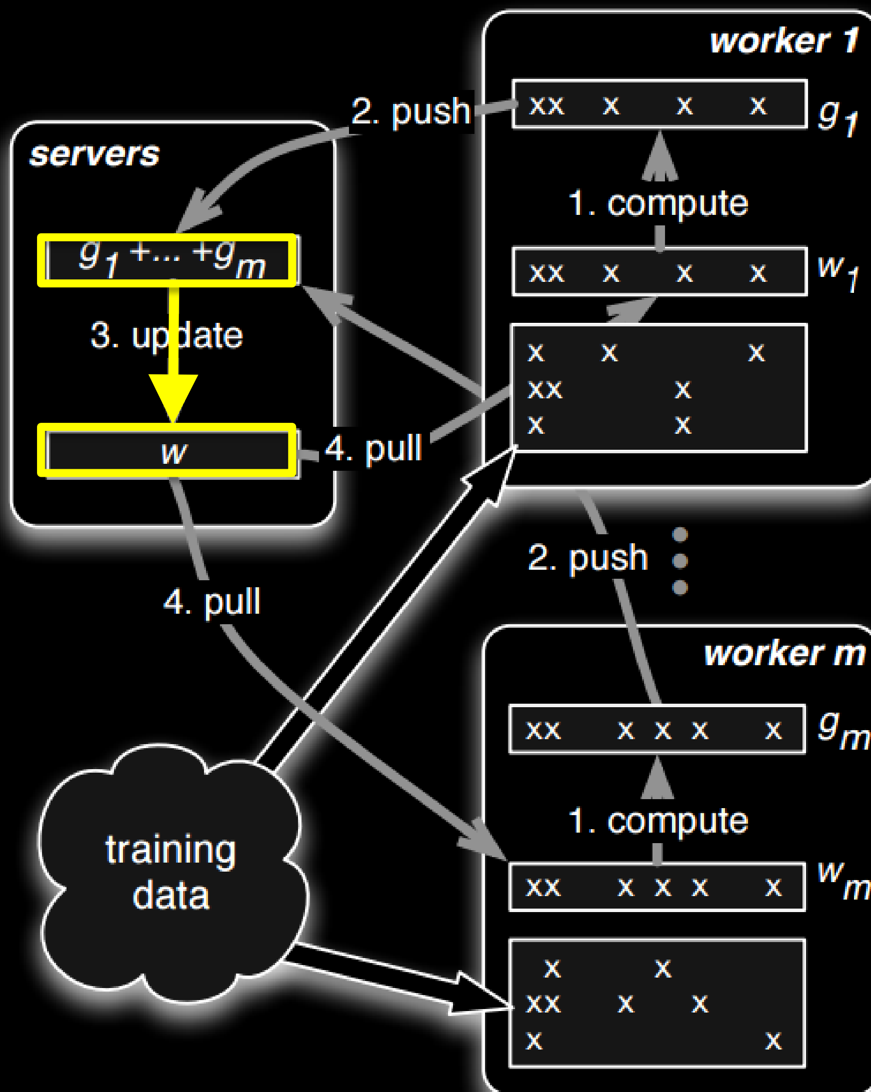
- Operates in iterations
- Worker nodes pull the updated w
- Worker node computes updates to w (local training)
- Worker node pushes updates to the server node



A Simple Example

- Server node + worker nodes
- Server node: all parameters
- Worker node: owns part of the training data

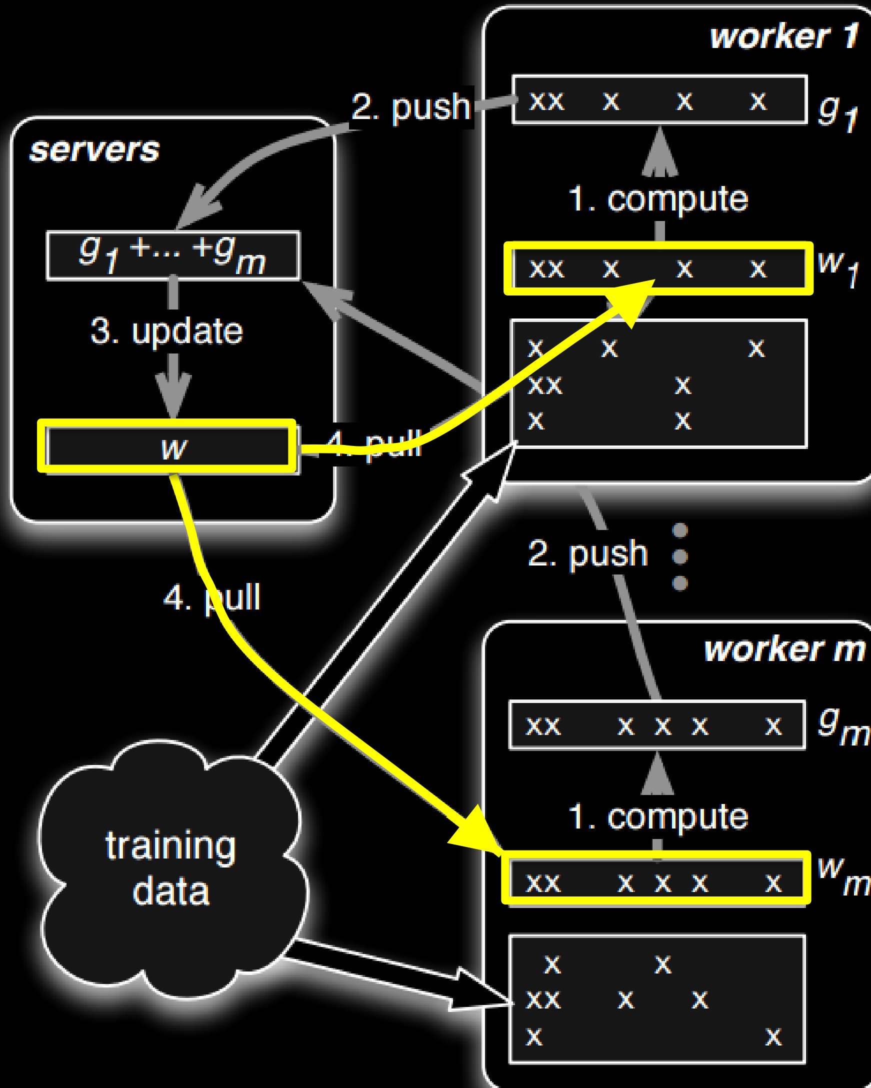
- Operates in iterations
- Worker nodes pull the updated w
- Worker node computes updates to w (local training)
- Worker node pushes updates to the server node
- Server node updates w



A Simple Example

- Server node + worker nodes
- Server node: all parameters
- Worker node: owns part of the training data

- Operates in iterations
- Worker nodes pull the updated w
- Worker node computes updates to w (local training)
- Worker node pushes updates to the server node
- Server node updates w

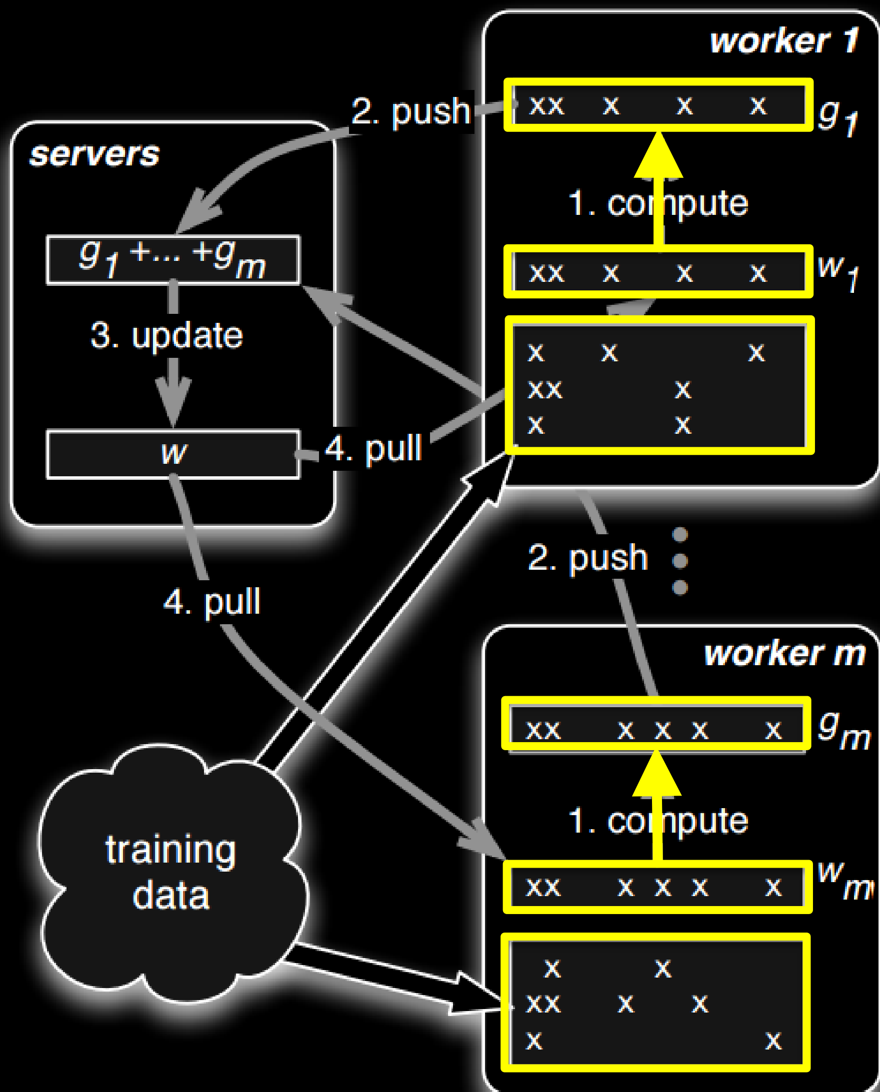


Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

A Simple Example

- Server node + worker nodes
- Server node: all parameters
- Worker node: owns part of the training data

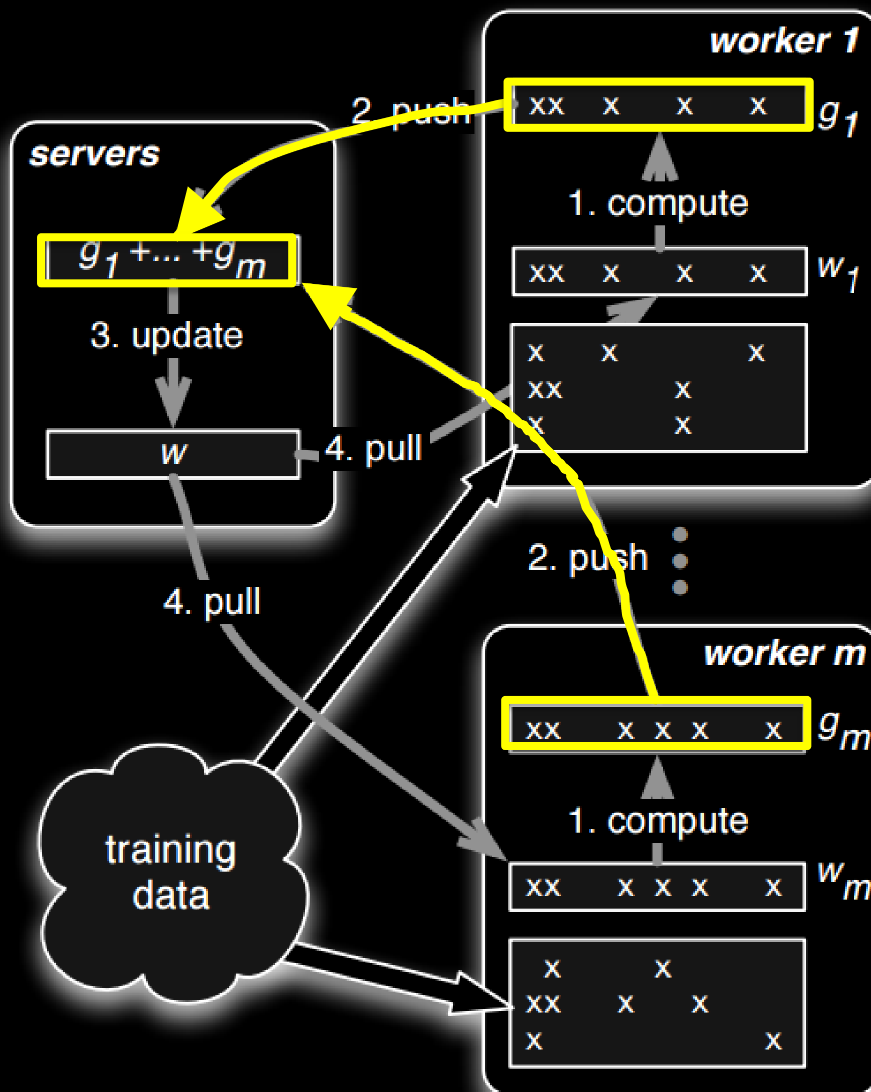
- Operates in iterations
- Worker nodes pull the updated w
- Worker node computes updates to w (local training)
- Worker node pushes updates to the server node
- Server node updates w



A Simple Example

- Server node + worker nodes
- Server node: all parameters
- Worker node: owns part of the training data

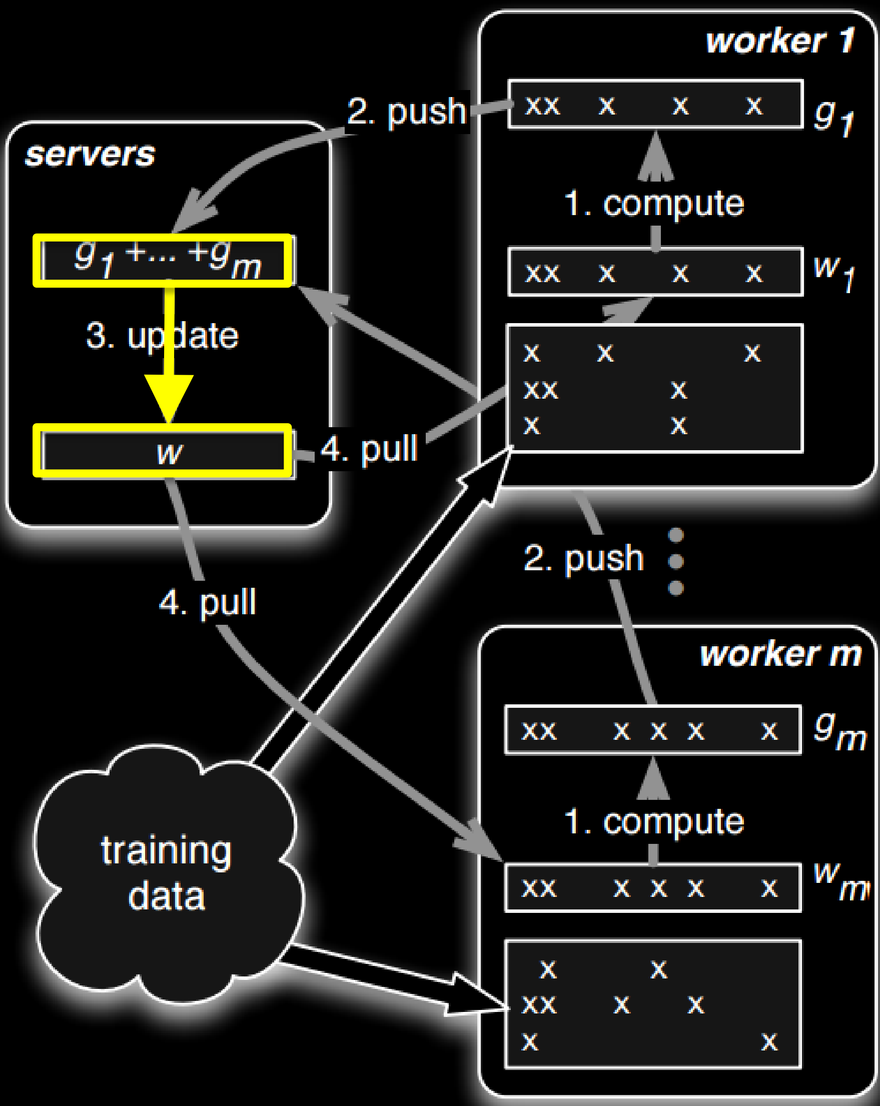
- Operates in iterations
- Worker nodes pull the updated w
- Worker node computes updates to w (local training)
- Worker node pushes updates to the server node
- Server node updates w



A Simple Example

- Server node + worker nodes
- Server node: all parameters
- Worker node: owns part of the training data

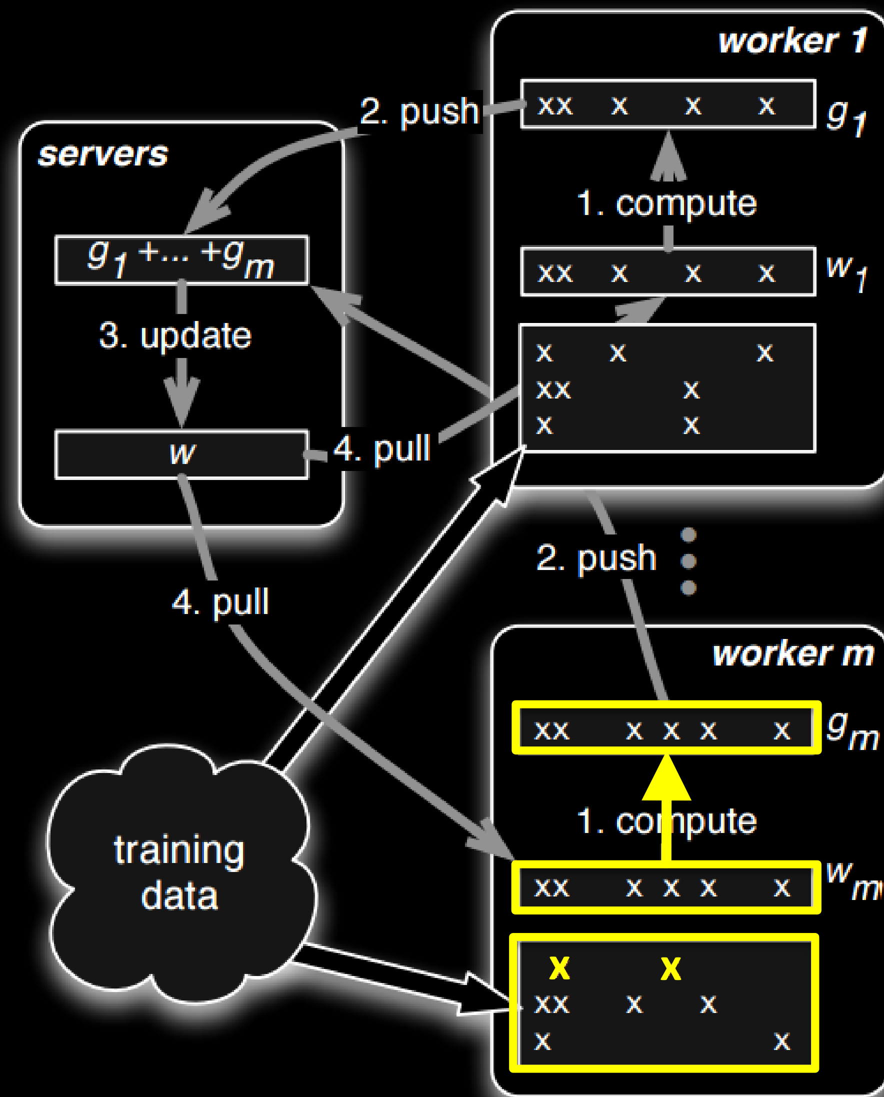
- Operates in iterations
- Worker nodes pull the updated w
- Worker node computes updates to w (local training)
- Worker node pushes updates to the server node
- Server node updates w



A Simple Example

- Server node + worker nodes
- Server node: all parameters
- Worker node: owns part of the training data

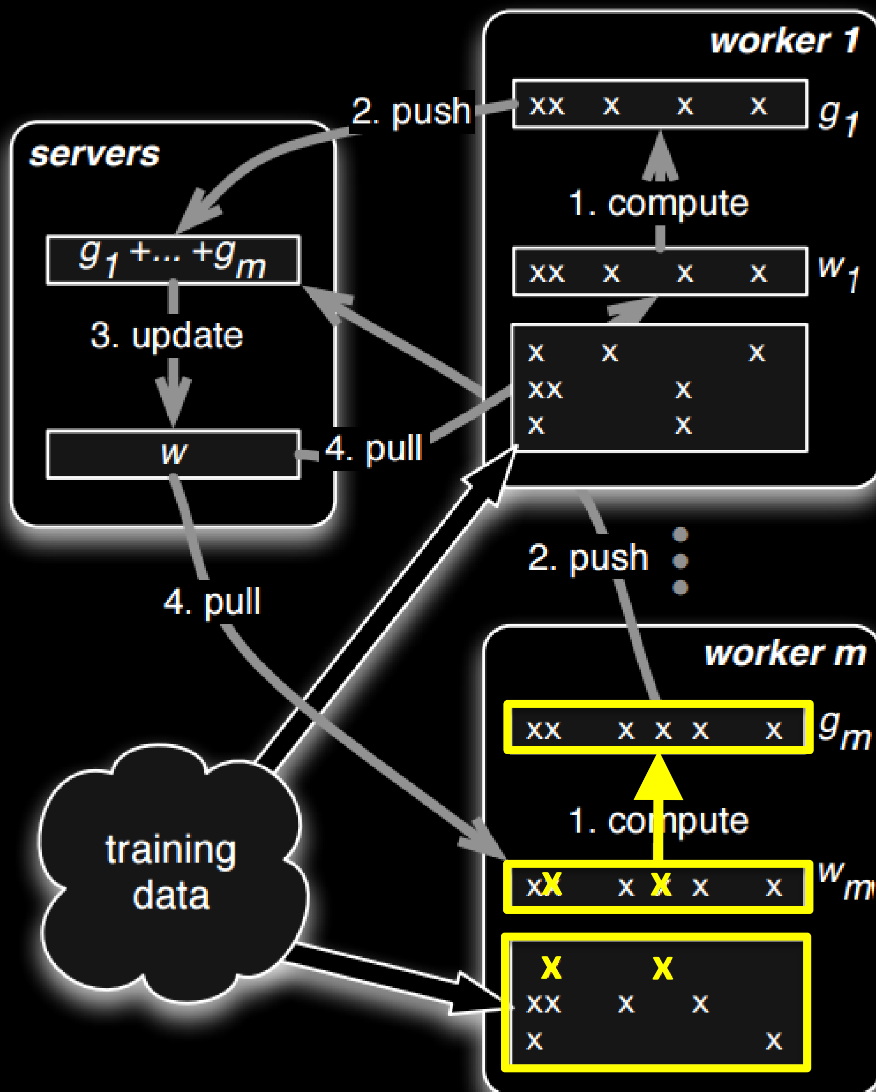
- Operates in iterations
- Worker nodes pull the updated w
- Worker node computes updates to w (local training)
- Worker node pushes updates to the server node
- Server node updates w



A Simple Example

- Server node + worker nodes
- Server node: all parameters
- Worker node: owns part of the training data

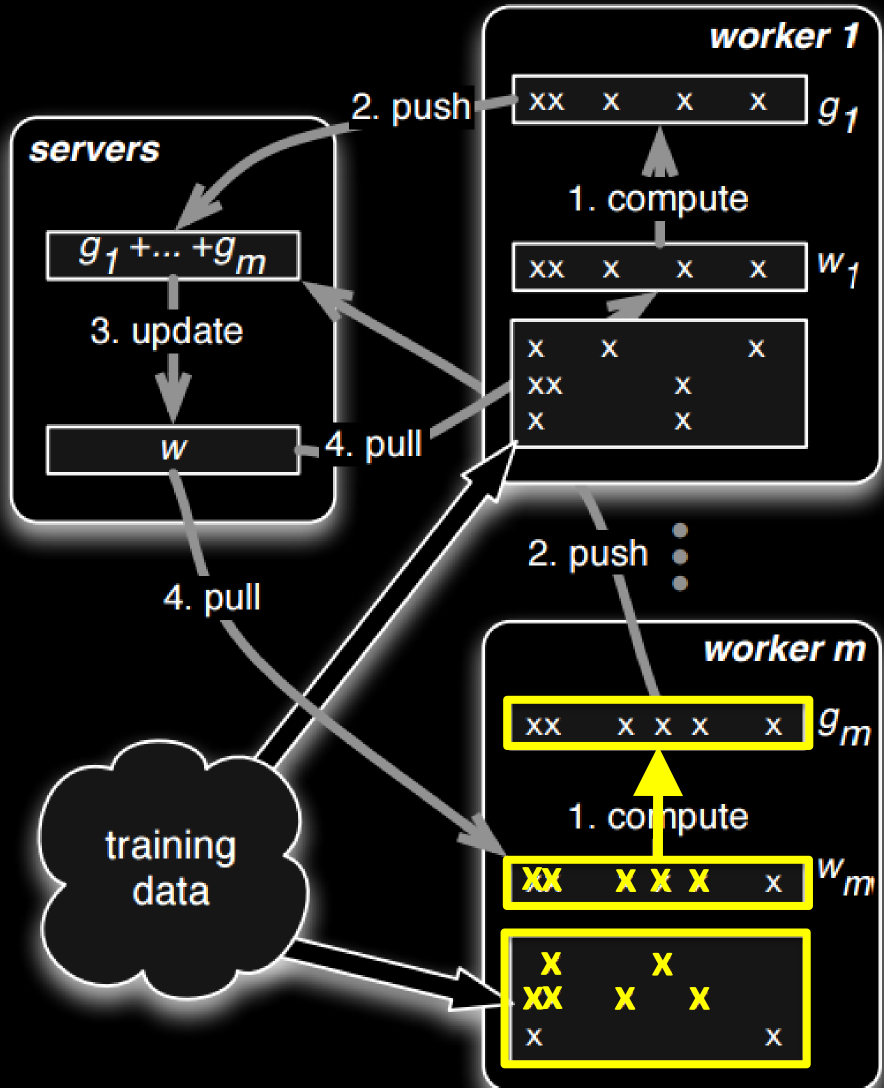
- Operates in iterations
- Worker nodes pull the updated w
- Worker node computes updates to w (local training)
- Worker node pushes updates to the server node
- Server node updates w



A Simple Example

- Server node + worker nodes
- Server node: all parameters
- Worker node: owns part of the training data

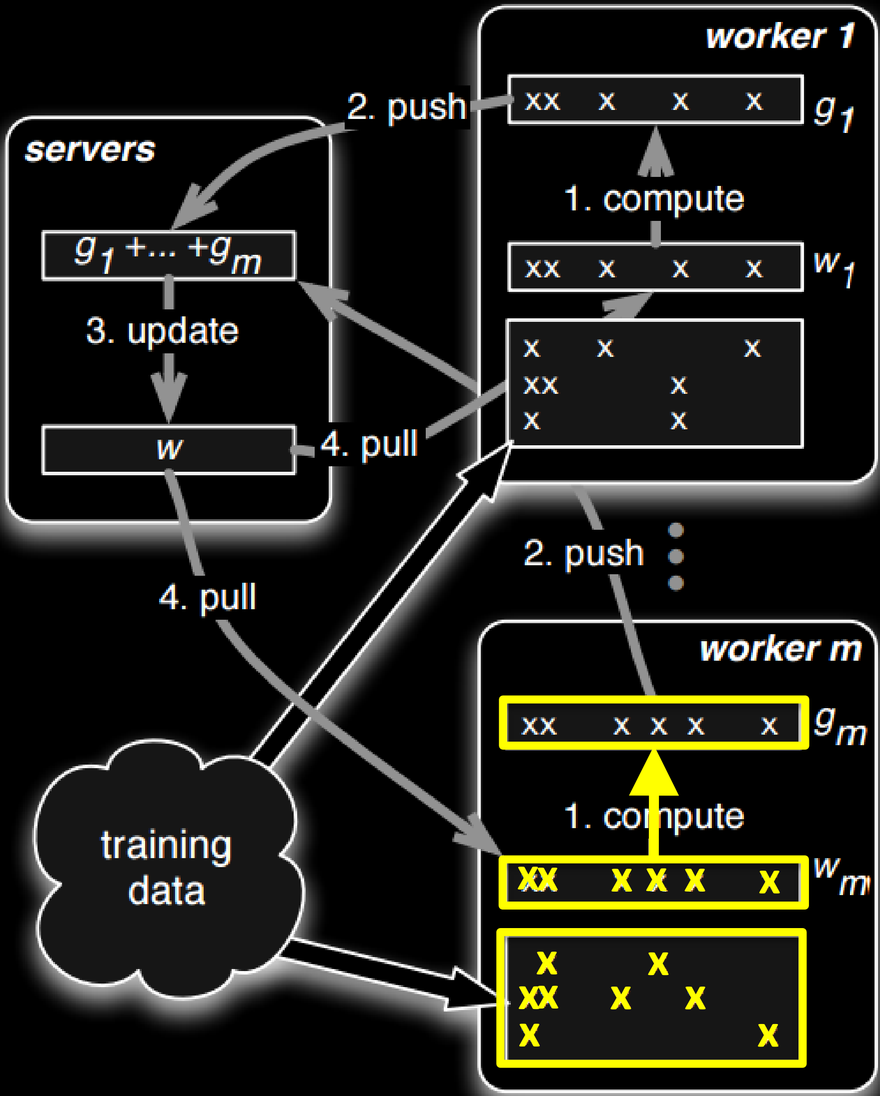
- Operates in iterations
- Worker nodes pull the updated w
- Worker node computes updates to w (local training)
- Worker node pushes updates to the server node
- Server node updates w



A Simple Example

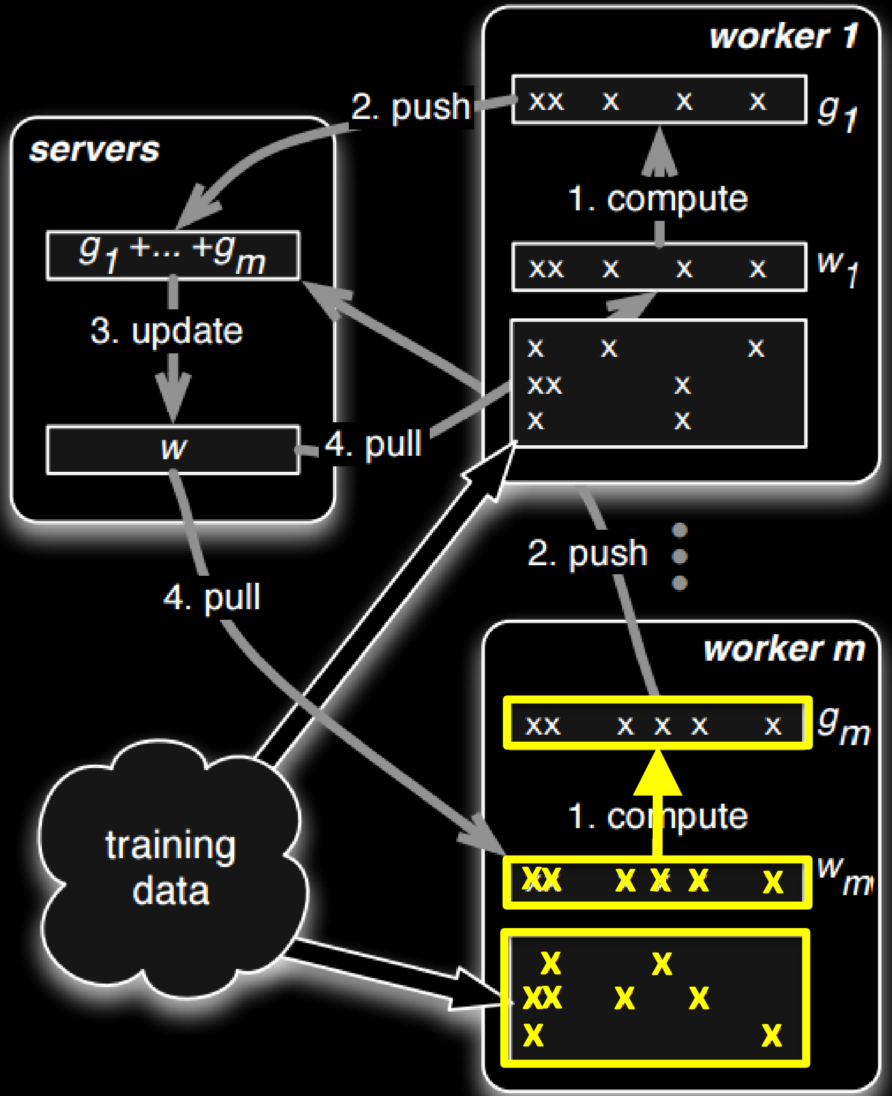
- Server node + worker nodes
- Server node: all parameters
- Worker node: owns part of the training data

- Operates in iterations
- Worker nodes pull the updated w
- Worker node computes updates to w (local training)
- Worker node pushes updates to the server node
- Server node updates w

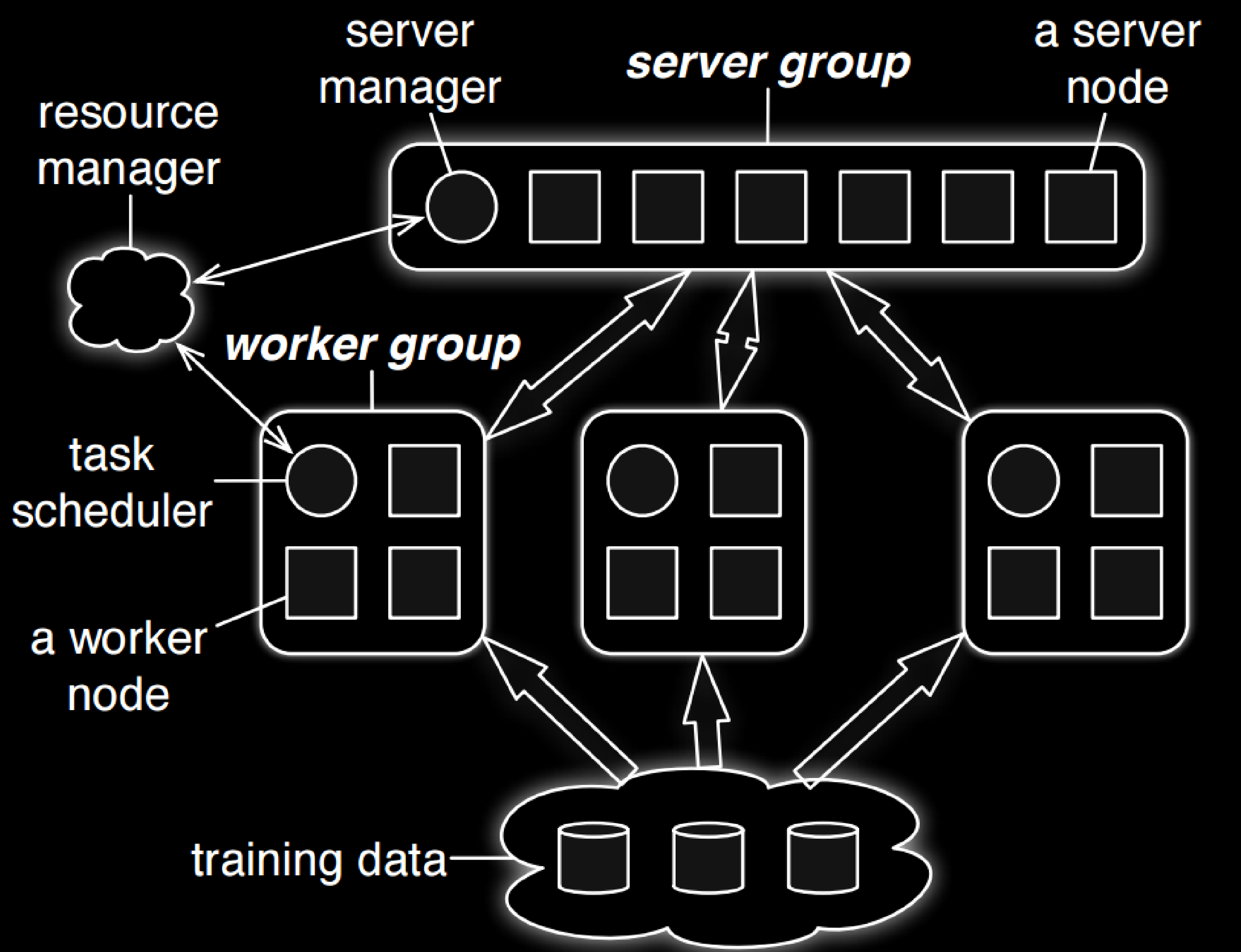


A Simple Example

- 100 nodes → 7.8% of w are used on one node (avg)
- 1000 nodes → 0.15% of w are used on one node (avg)



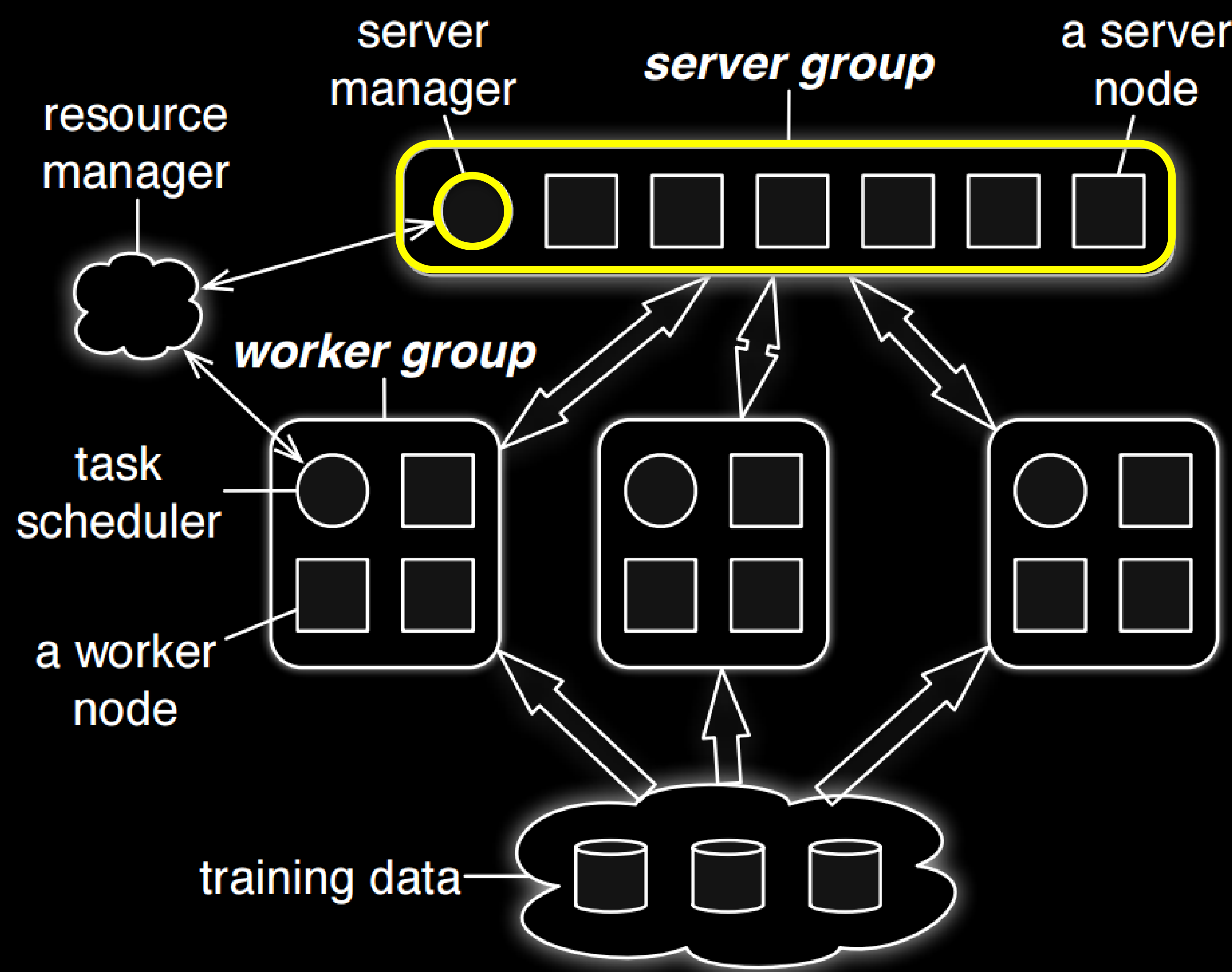
Architecture



Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Architecture

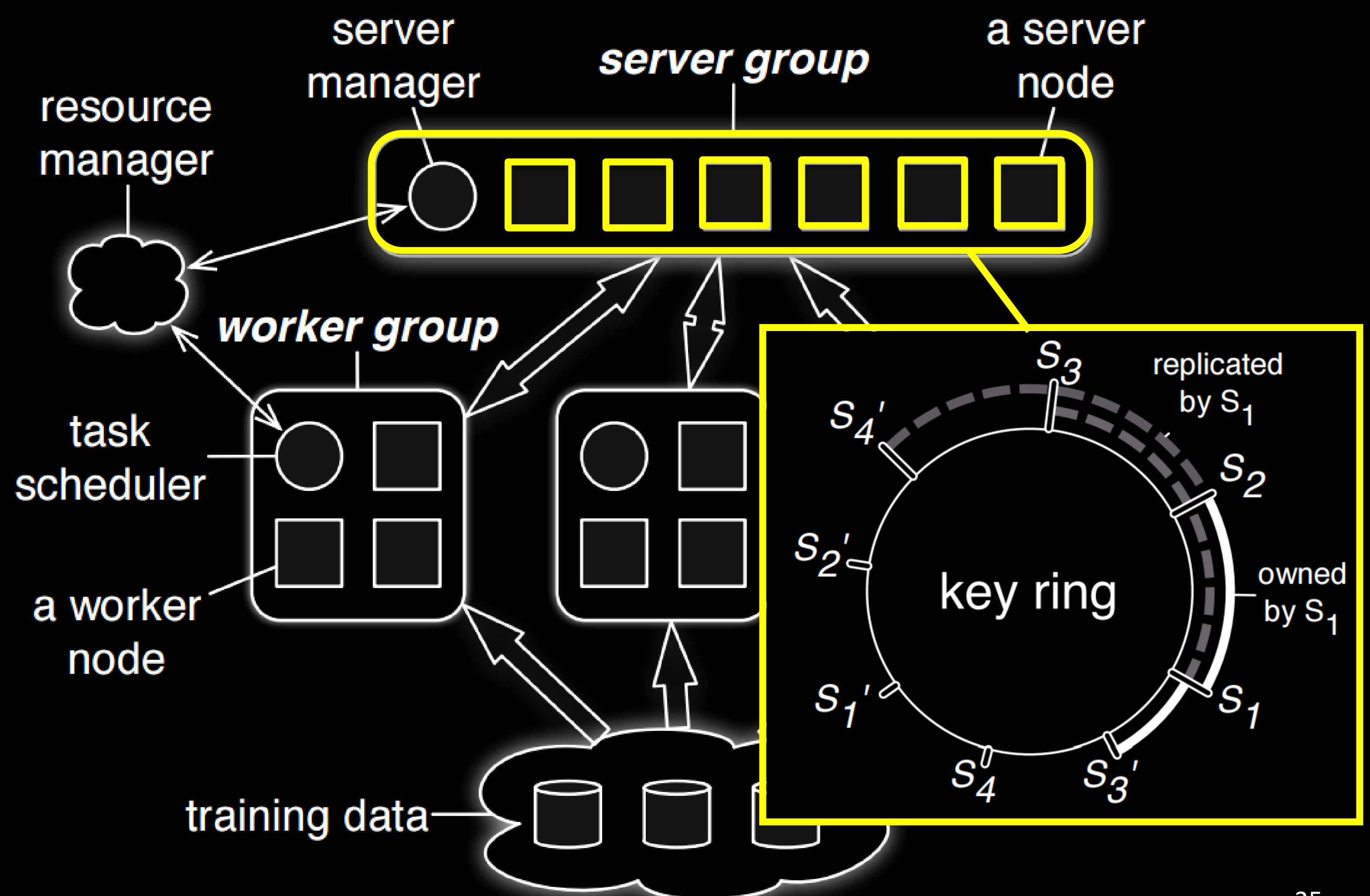
Server manager: Liveness and parameter partition of server nodes



Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Architecture

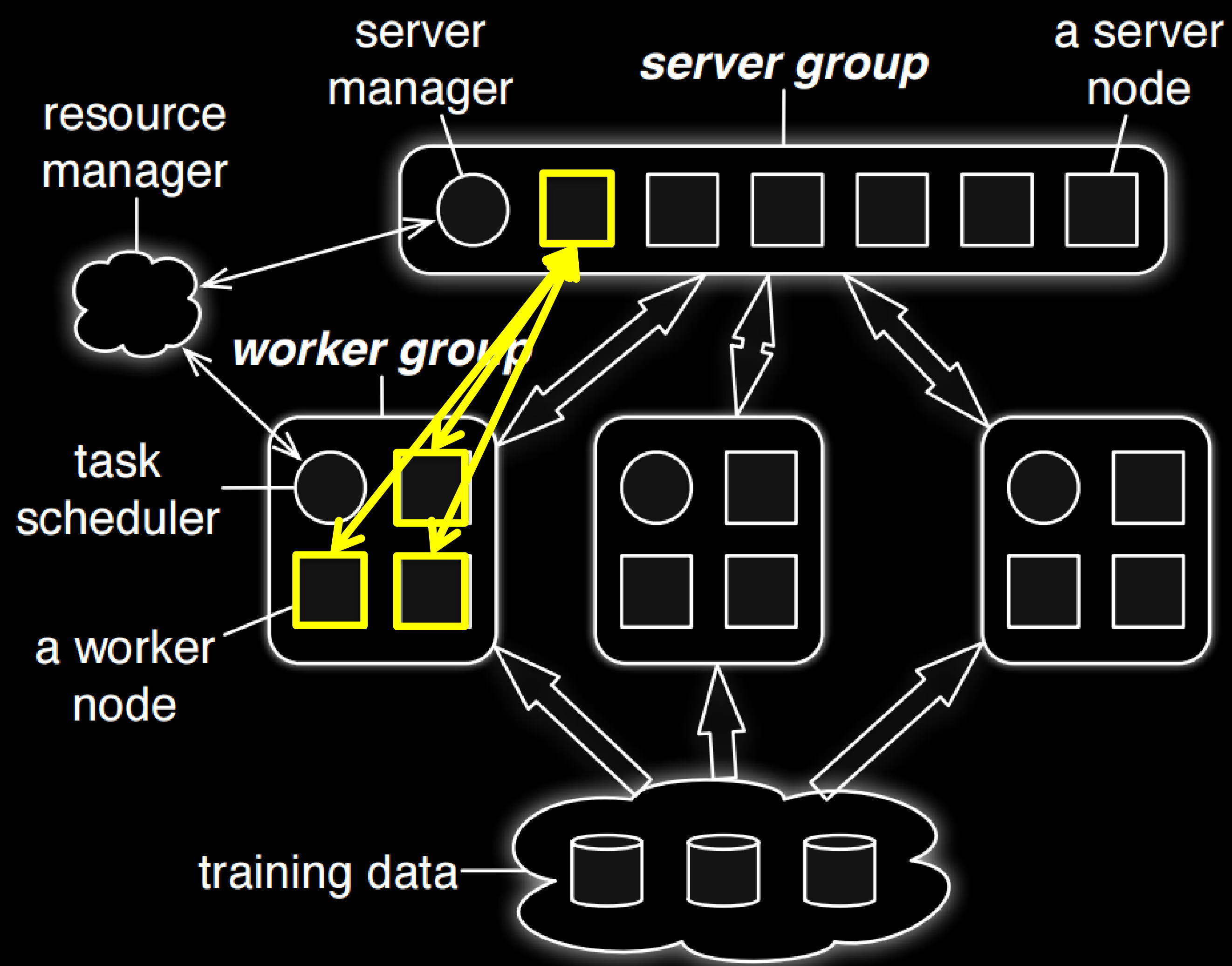
All server nodes partition parameters keys with consistent hashing.



Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Architecture

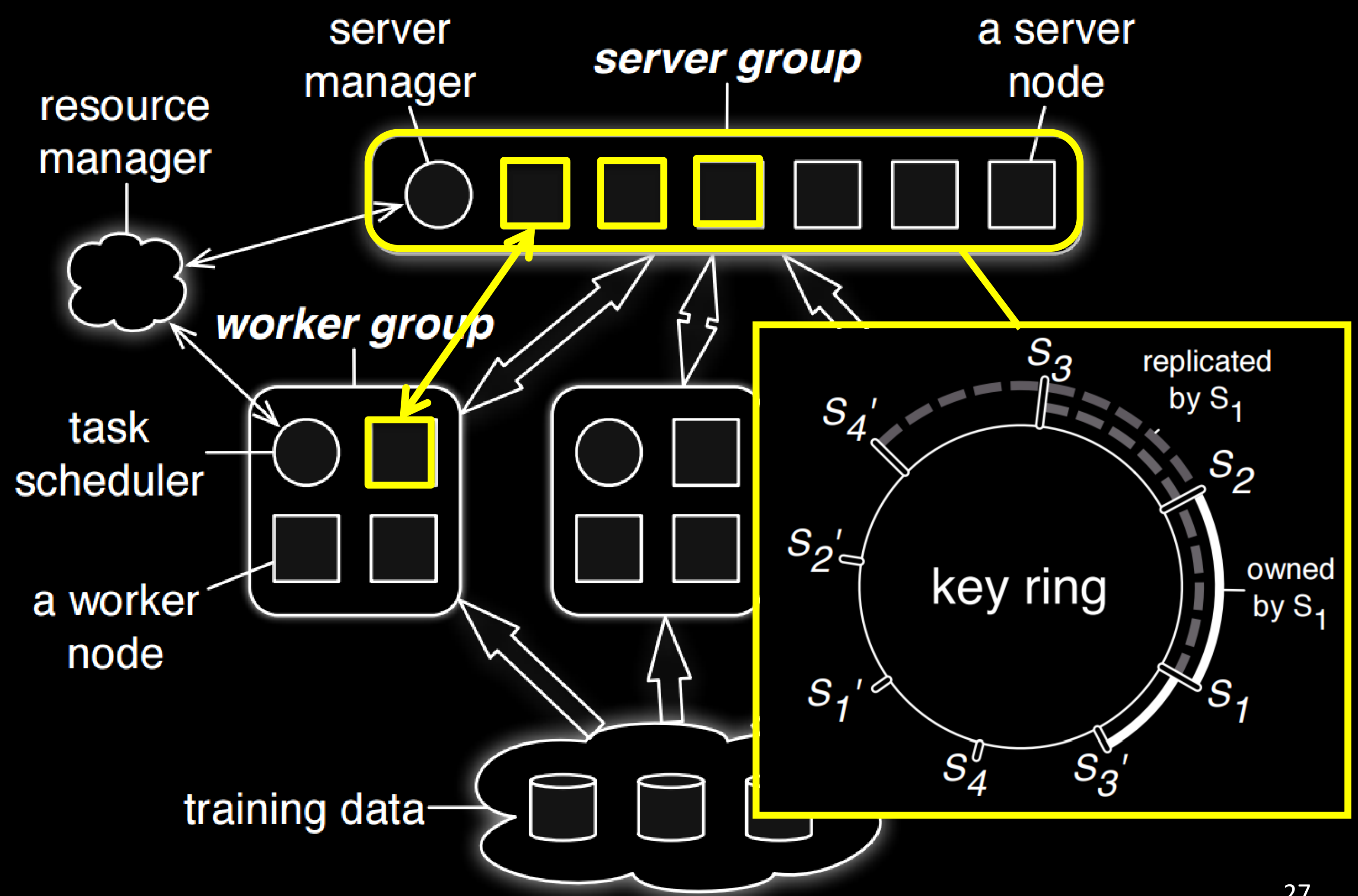
Worker node: communicate only with its server node



Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Architecture

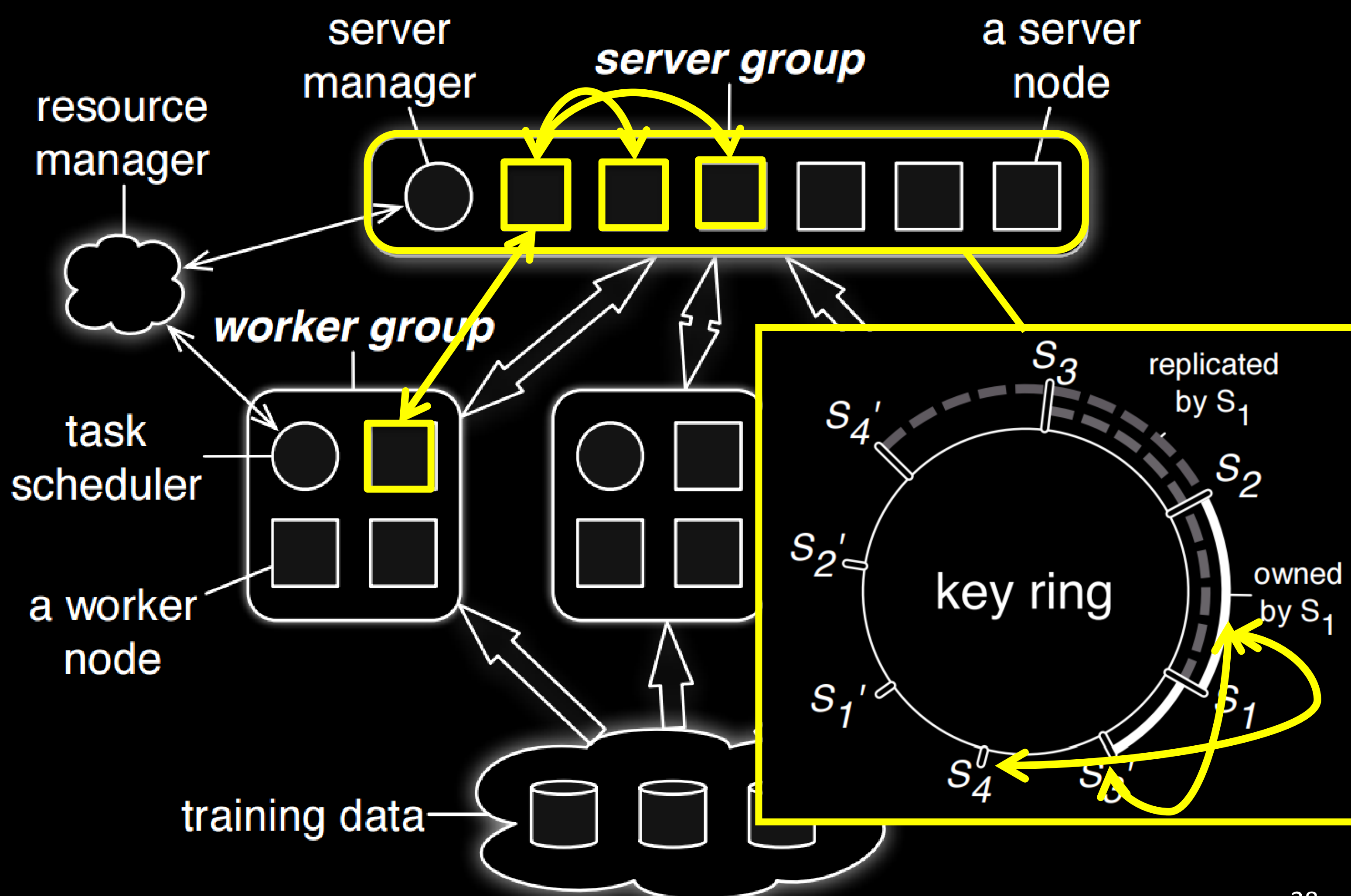
Updates are replicated to slave server nodes synchronously.



Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Architecture

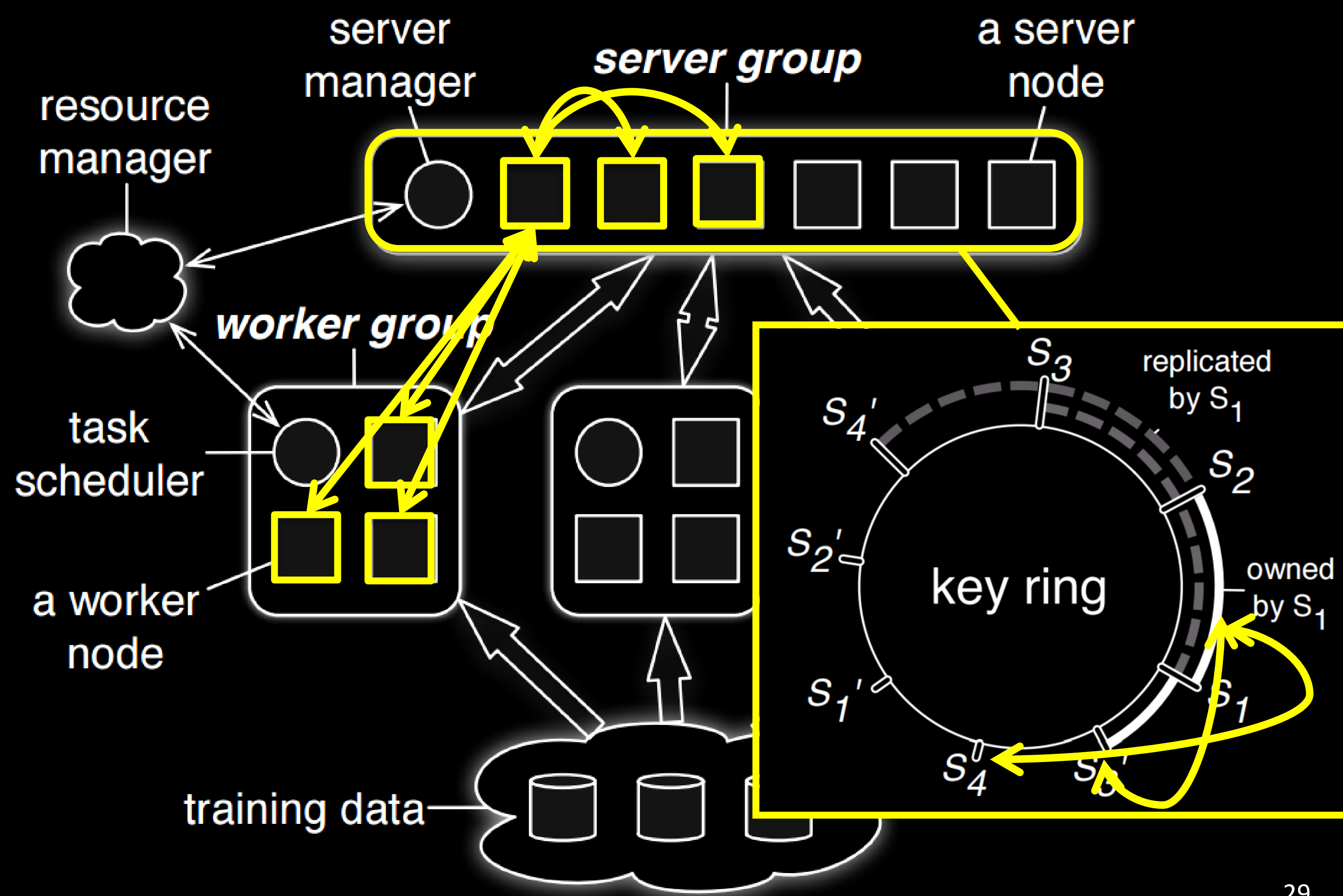
Updates are replicated to slave server nodes synchronously.



Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Architecture

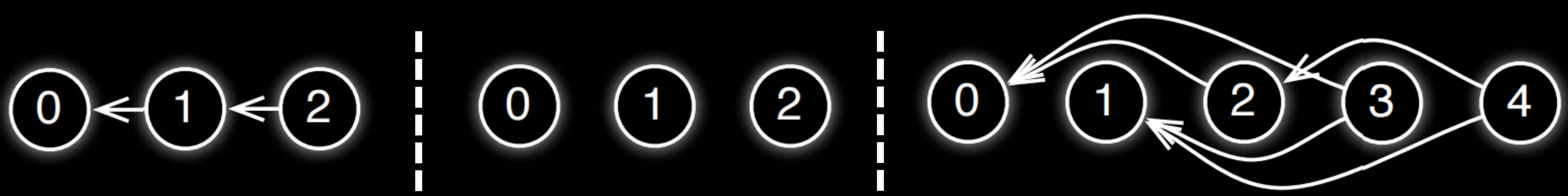
Optimization: replication after aggregation



Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Data Transmission / Calling

- The shared parameters are presented as **`<key, value>`** vectors.
- Data is sent by pushing and pulling key range.
- Tasks are issued by RPC.
- Tasks are executed asynchronously.
  - Caller executes without waiting for a return from the callee.
- Caller can specify dependencies between callees.

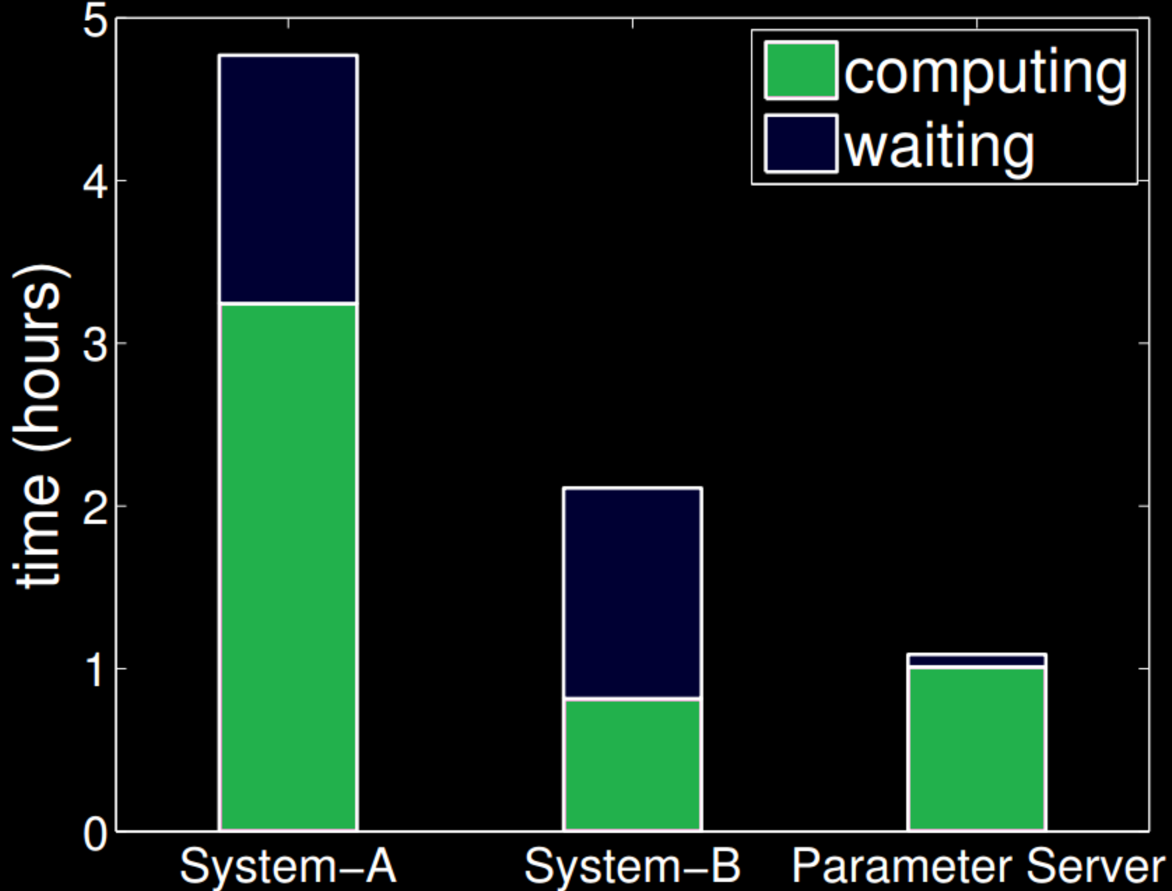


Sequential Consistency | Eventual Consistency | 1 Bounded Delay Consistency

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Trade-off: Asynchronous Call

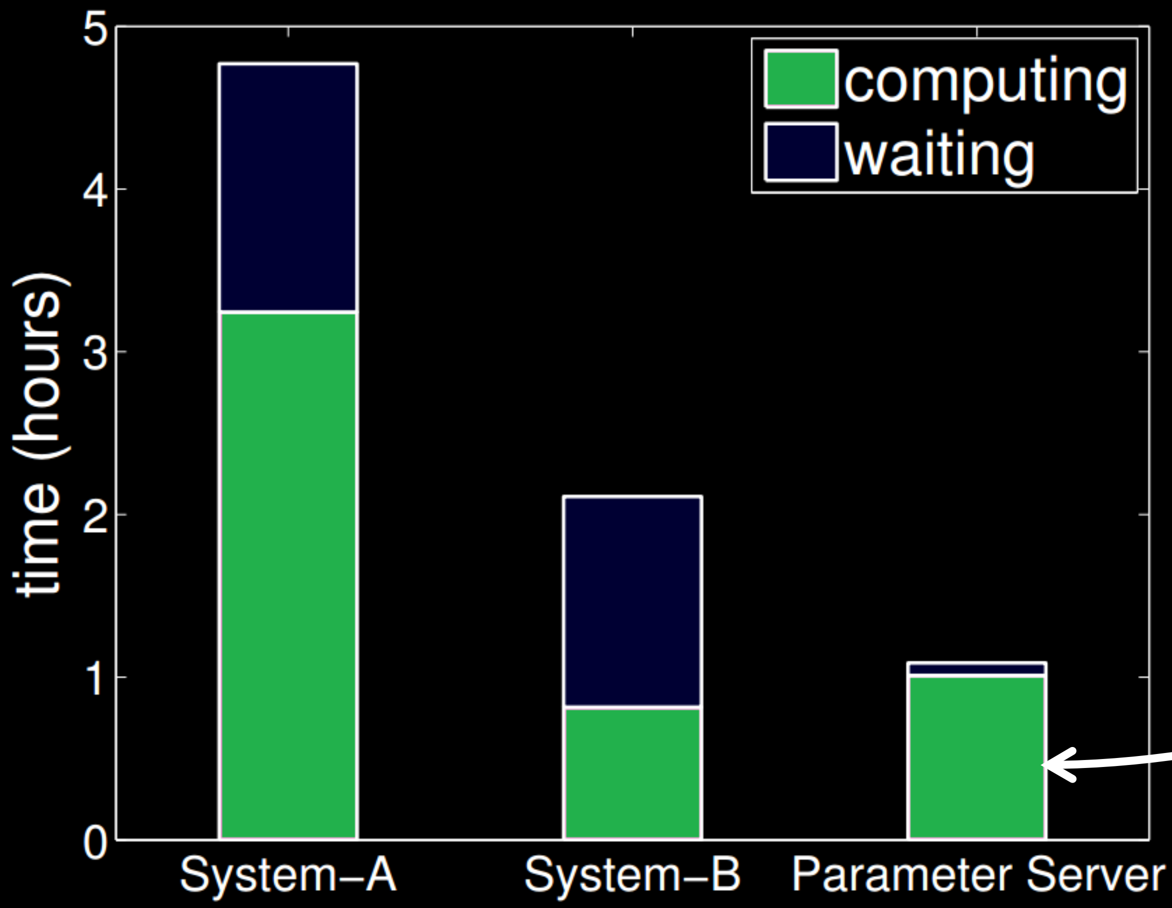
- 1000 machines, 800 workers, 200 parameter servers.
- 16 physical cores, 192G DRAM, 10Gb Ethernet.



Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Trade-off: Asynchronous Call

- 1000 machines, 800 workers, 200 parameter servers.
- 16 physical cores, 192G DRAM, 10Gb Ethernet.



Asynchronous updates require more iterations to achieve the same objective value.

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Assumptions

- It is OK to lose part of the training dataset.
  → Not urgent to recover a fail worker node
  → Recovering a failed server node is critical

- An approximate solution is good enough
  → Limited inaccuracy is tolerable
  → Relaxed consistency (as long as it converges)

Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

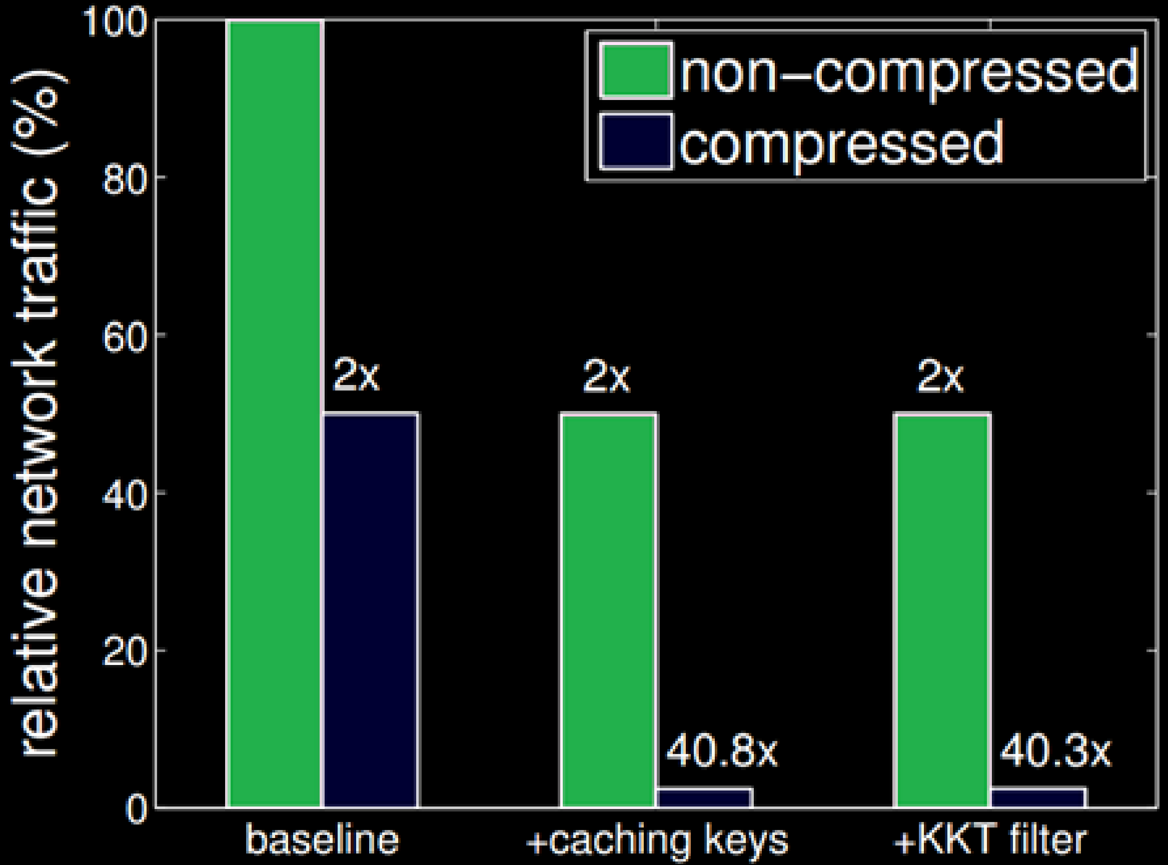
Optimizations

- Message compression → save bandwidth
- Aggregate parameter changes before synchronous replication on server node
- Key lists for parameter updates are likely to be the same as last iteration
  - → cache the list, send a hash

    <~~1,~~ 3>, <~~2,~~ 4>, <~~6,~~ 7.5>, <~~7,~~ 4.5> …
- Filter before transmission:
  - gradient update that is less than a threshold.

Network Saving

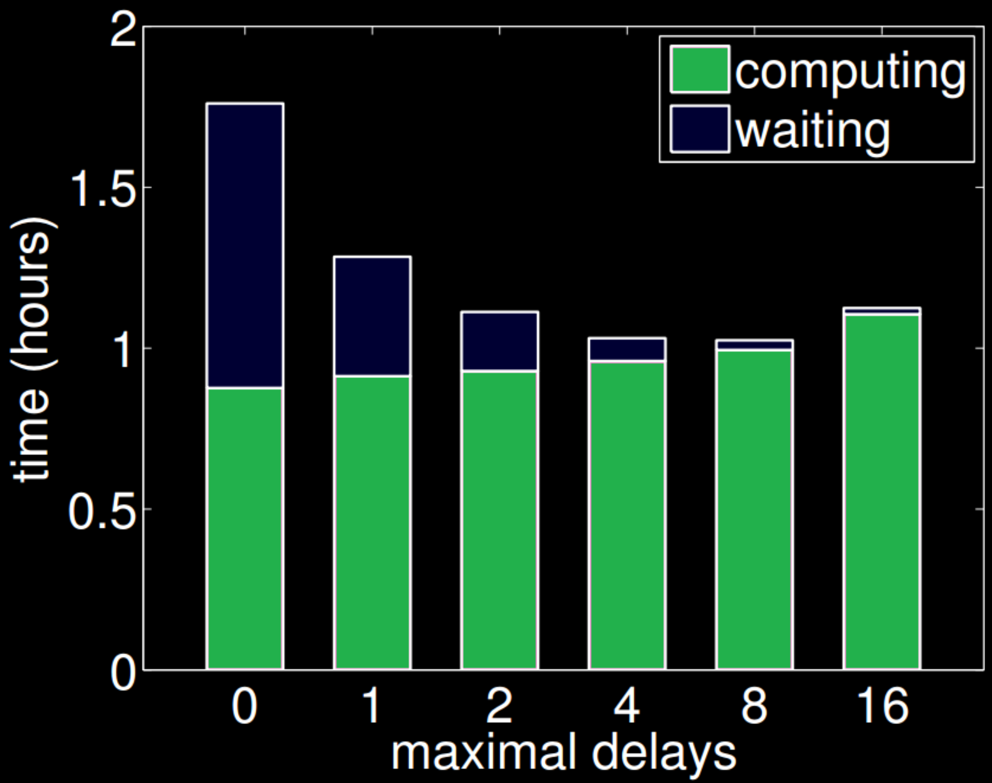
- 1000 machines, 800 workers, 200 parameter servers.
- 16 physical cores, 192G DRAM, 10Gb Ethernet.



Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

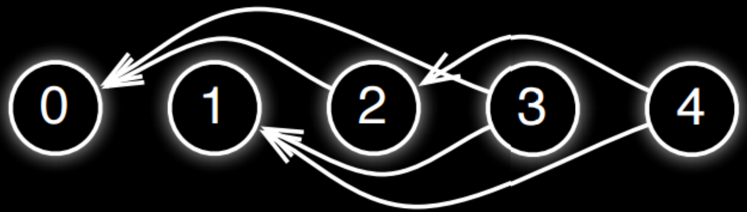
Trade-offs

- Consistency model vs Computing Time + Waiting Time



Sequential Consistency (τ=0)

Eventual Consistency (τ=∞)



1 Bounded Delay Consistency (τ=1)

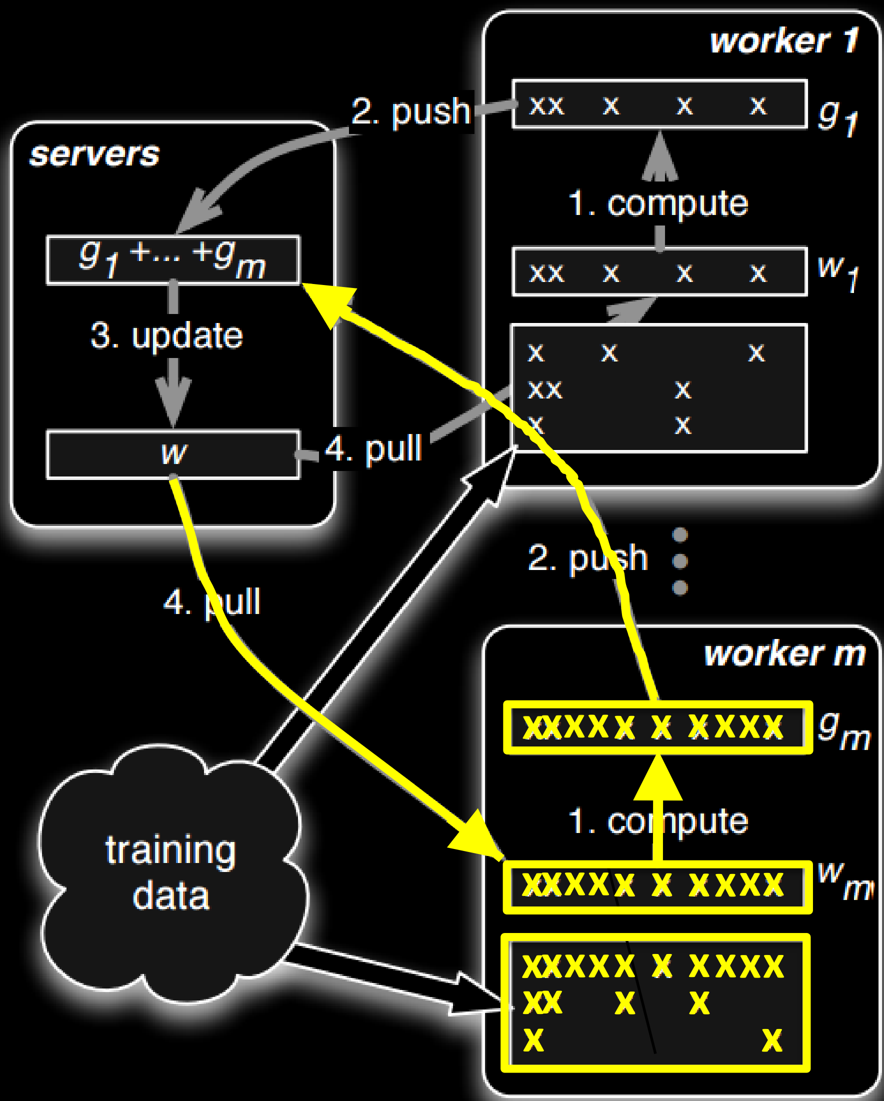
Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Discussions

- Feature selection? Sampling?
- Trillions of features and trillions of examples in the training dataset → overfitting?
- Each worker do multiple iterations before push?
- Diversify the labels each node is assigned > Random?
- If one worker only pushes trivial parameter changes, probably its training dataset are not very useful → remove and re-partition.
- A hierarchy of server node

Assumption / Problem

Fact: The total size of parameters (features) may exceed the capacity of a single machine.

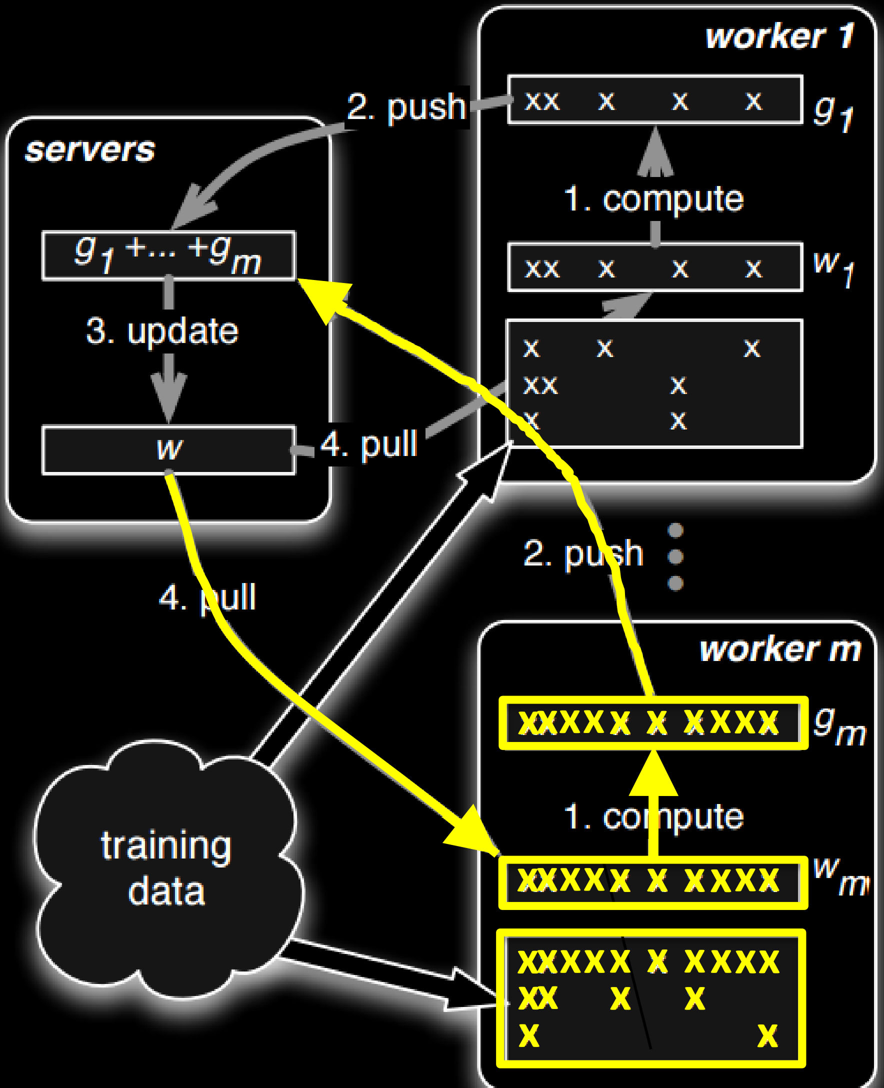


Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Assumption / Problem

Fact: The total size of parameters (features) may exceed the capacity of a single machine.

Assumption: Each instance in the training set only contains a small portion of all features.



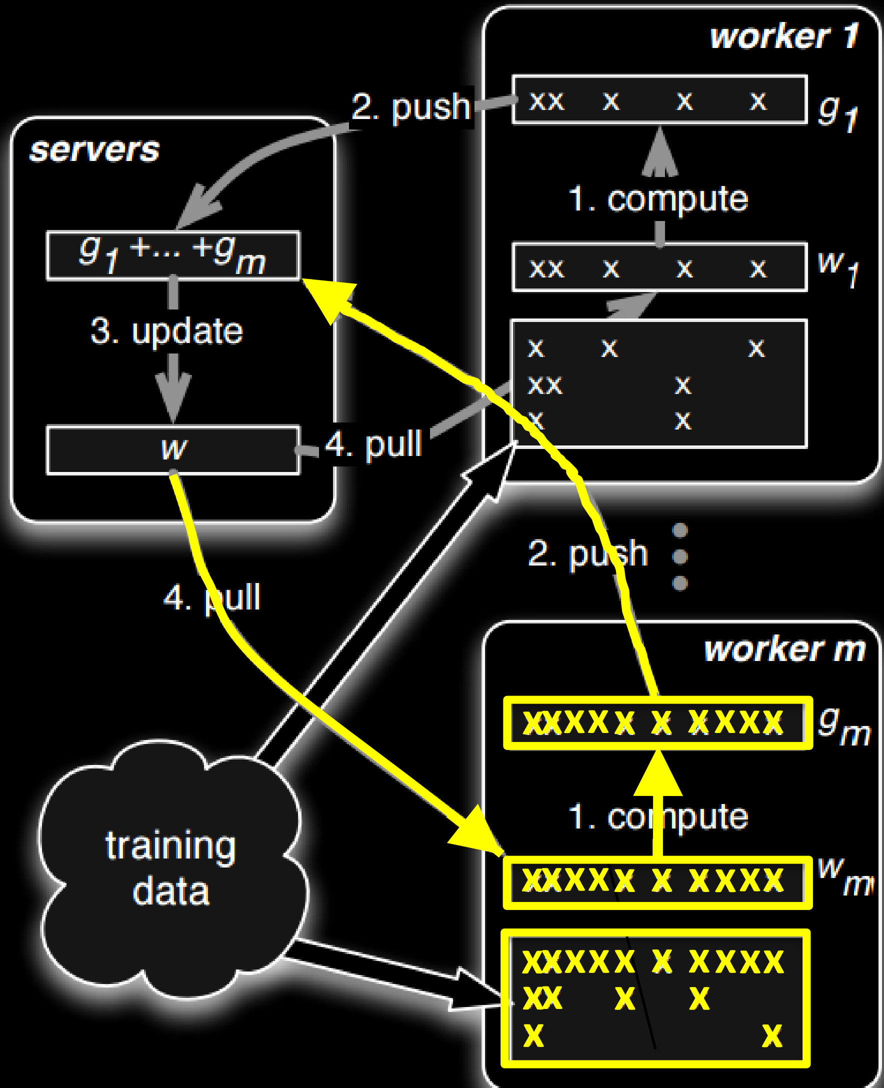
Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Assumption / Problem

Fact: The total size of parameters (features) may exceed the capacity of a single machine.

Assumption: Each instance in the training set only contains a small portion of all features.

Problem: What if one example contains 90% of features (trillions of features in total)?



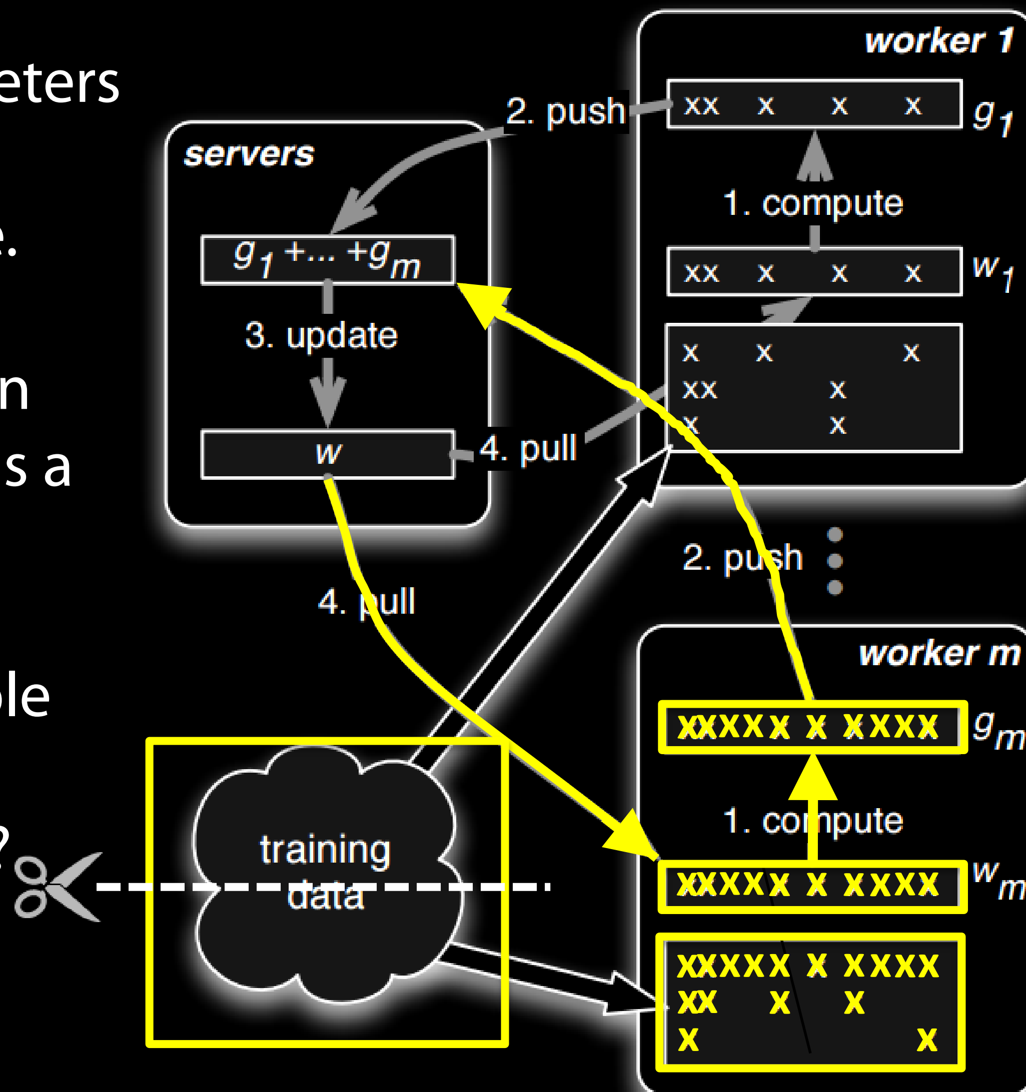
Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Assumption / Problem

Fact: The total size of parameters (features) may exceed the capacity of a single machine.

Assumption: Each instance in the training set only contains a small portion of all features.

Problem: What if one example contains 90% of features (trillions of features in total)?



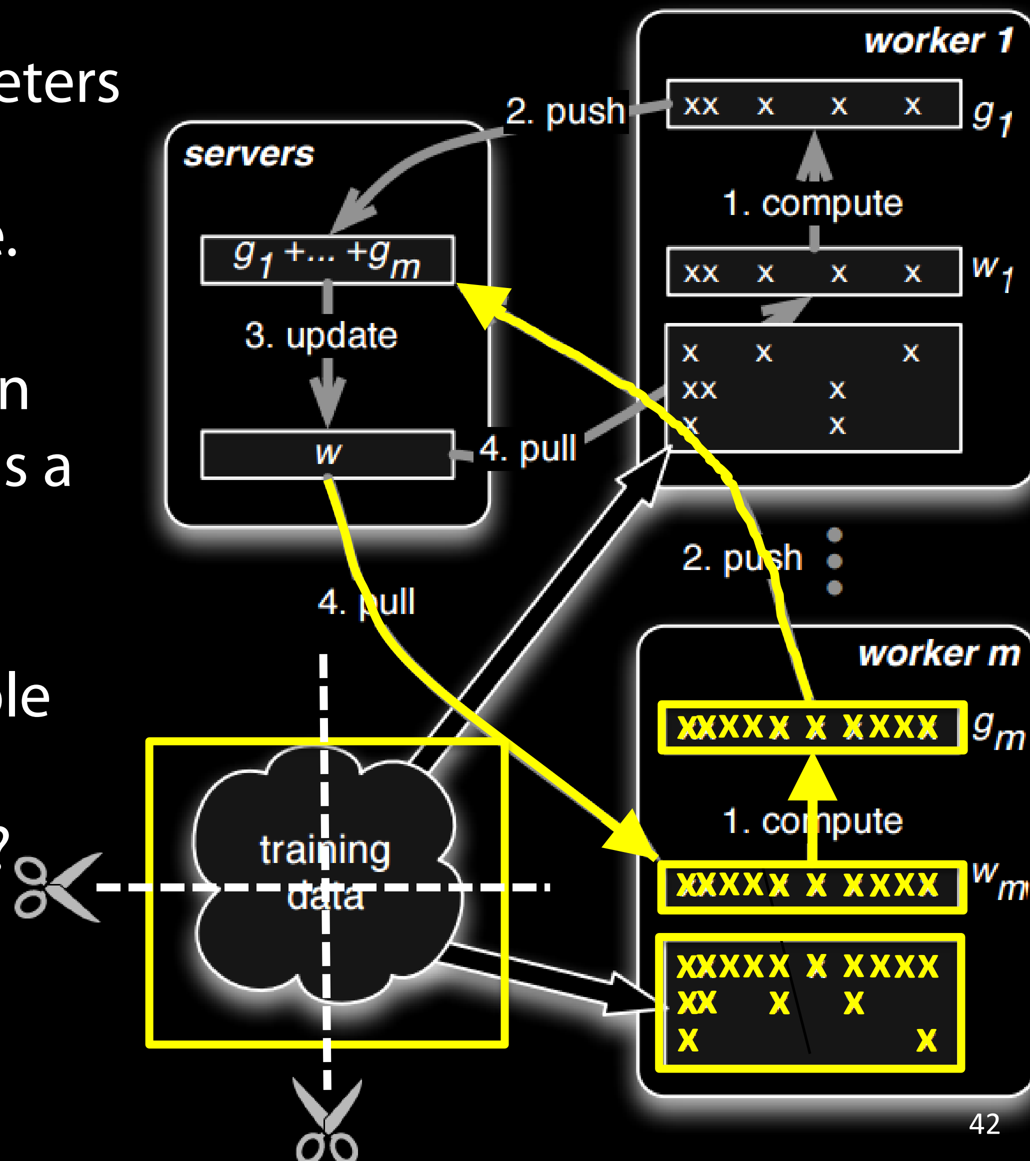
Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.

Assumption / Problem

Fact: The total size of parameters (features) may exceed the capacity of a single machine.

Assumption: Each instance in the training set only contains a small portion of all features.

Problem: What if one example contains 90% of features (trillions of features in total)?



Sketch Based Machine Learning Algorithms

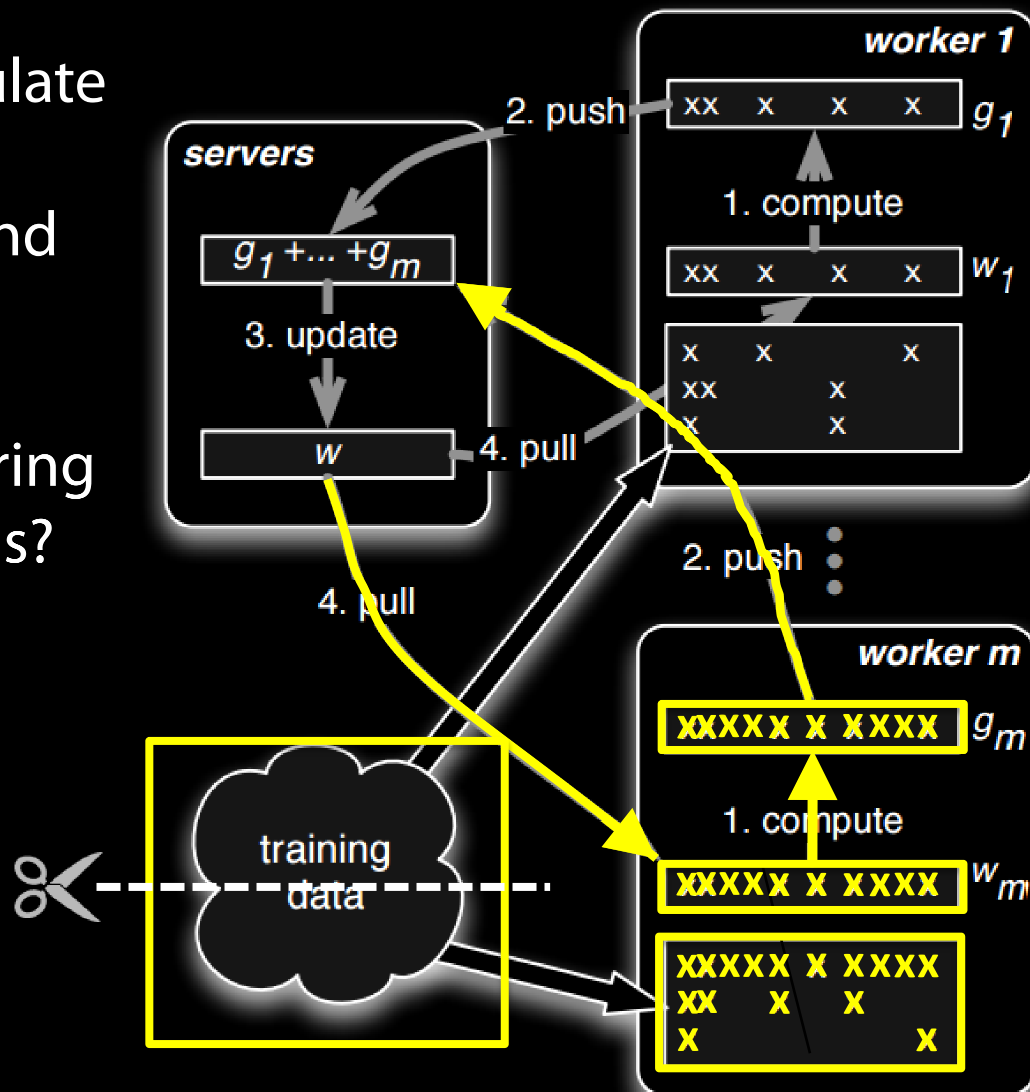
- Sketches are a class of data stream summaries
- Problem: An infinite number of data items arrive continuously, whereas the memory capacity is bounded by a small size
  - Every item is seen once
- Approach: Typically formed by linear projections of source data with appropriate (pseudo) random vectors
- Goal: use small memory to answer interesting queries with strong precision guarantees

http://web.engr.illinois.edu/~vvnktrm2/talks/sketch.pdf

Assumption / Problem

Assumption: It is OK to calculate updates for models on each portion of data separately and aggregate the updates.

Problem: What about clustering and other ML/DM algorithms?



Liang Gong, Electric Engineering & Computer Science, University of California, Berkeley.