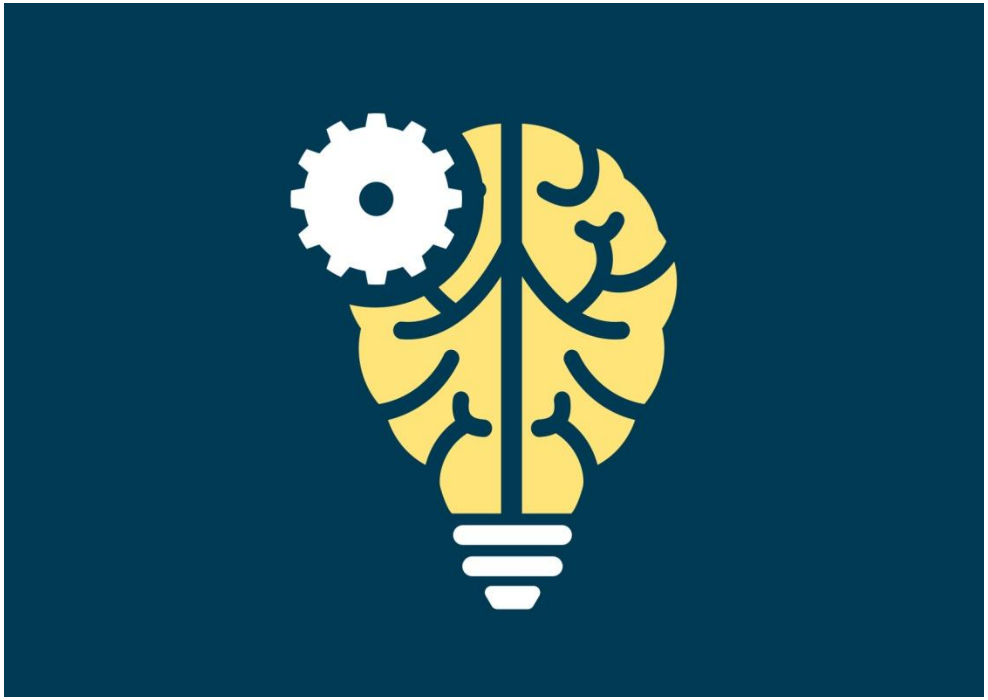
Machine Learning with Big Data

Specialized distributed systems for machine learning purposes

Overview

- Why distribute machine learning?
- Systems
  - Map-Reduce – Hadoop
  - Resilient Distributed Datasets (RDD) – Spark
  - **Parameter Server**
  - **Tensor Flow**
- Evaluation



WHY DISTRIBUTE MACHINE LEARNING?

Why distribute?

- Machine learning? The more data, the better
  - But, required resources increase constantly
- Big companies gather *bytes of data per day
  - Processable for data mining, not machine learning
- Not all tasks are eligible for data mining, such as classification

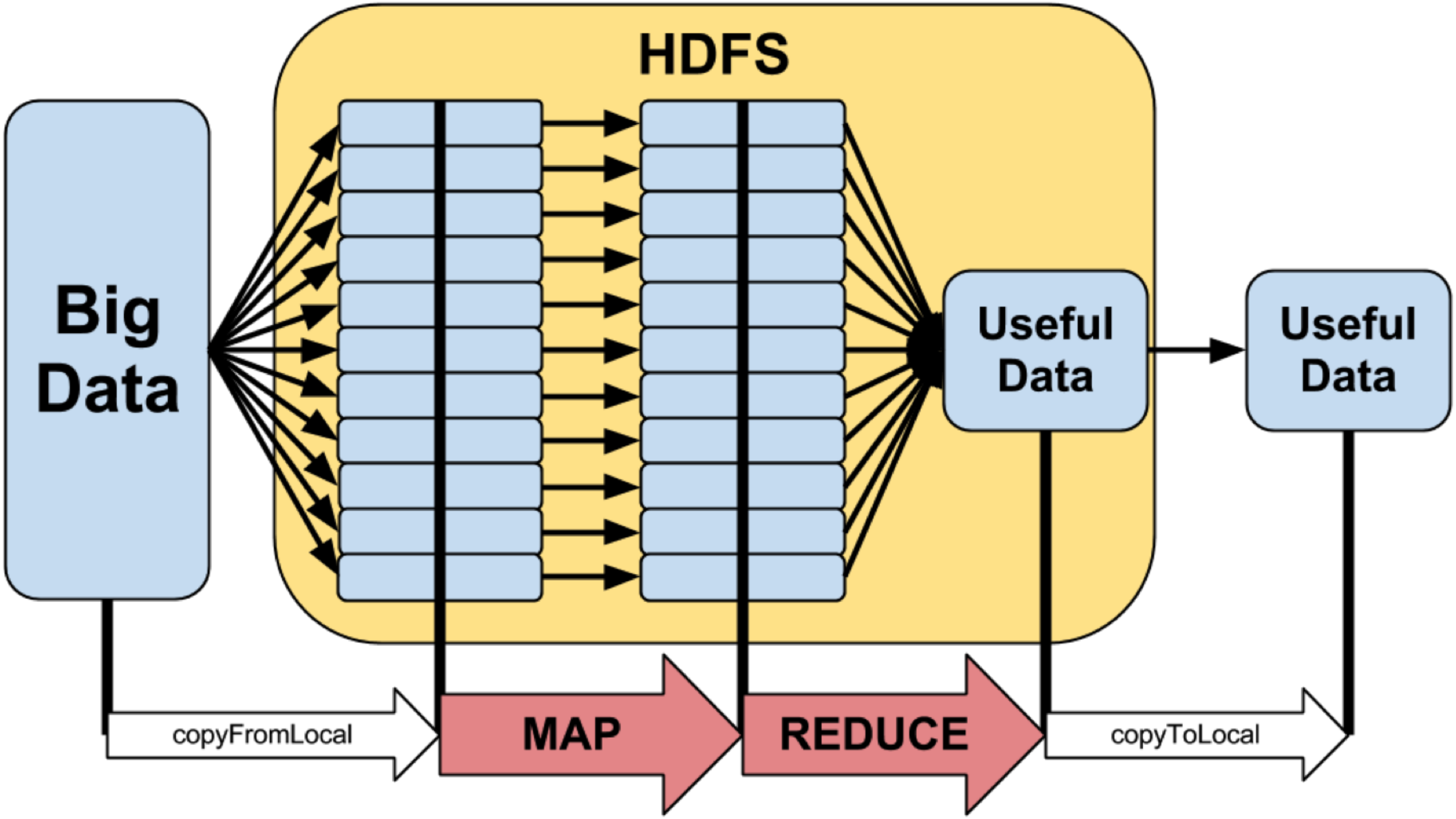
Distributed approach

- Machine learning is typically a sequential task
  - The “model” is a centralized object
  - Each item it learns affects its state
- Crunching this data is unfeasible in a single machine – use a distributed approach
- How to distribute machine learning?

Systems

HADOOP

Map-Reduce architecture



Map-Reduce ideas

- Split “splittable” problem to workers (map)
- Gather results from workers (reduce)

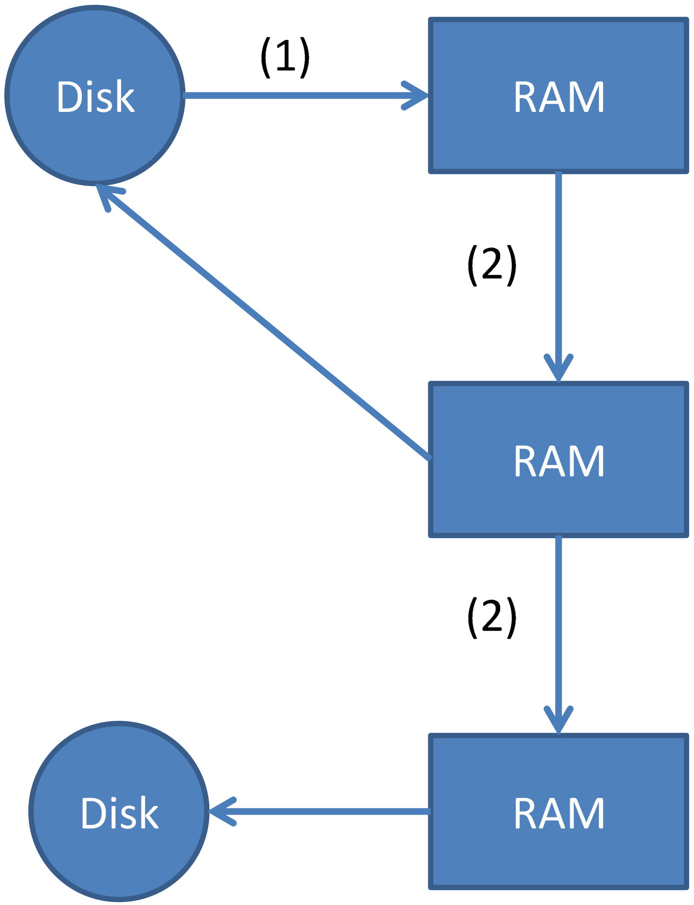
- Main innovation is on the algorithm itself
  - Particularly good for text processing, but not thought for a very specialized task

Systems

SPARK

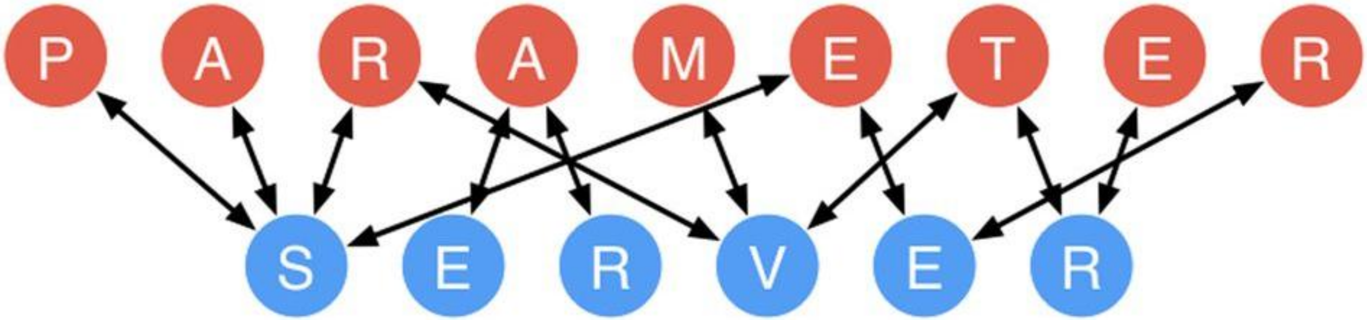
Resilient Distributed Datasets

- “Formally, an RDD is a read-only, partitioned collection of records.” [1]
- RDDs can only be created through deterministic operations on (1) data in storage or (2) other RDDs
- Operations include map, filter, and join
- Operations are stored in RAM
- An RDD has enough information to be reconstructed after a failure
- Persistence on disk automatically or on demand



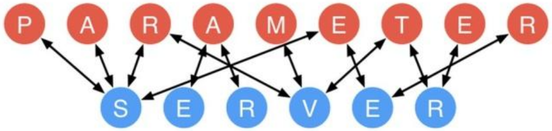
RDDs vs Map-Reduce

- Allows data reuse for (e.g.) iterative machine learning and graph algorithms
- Recovery after failures is faster
- Requires more RAM (expensive machines)



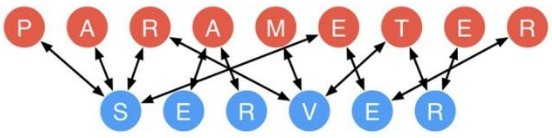
Systems

PARAMETER SERVER



Parameter Server

- Specific for machine learning
- Assumes features are already extracted



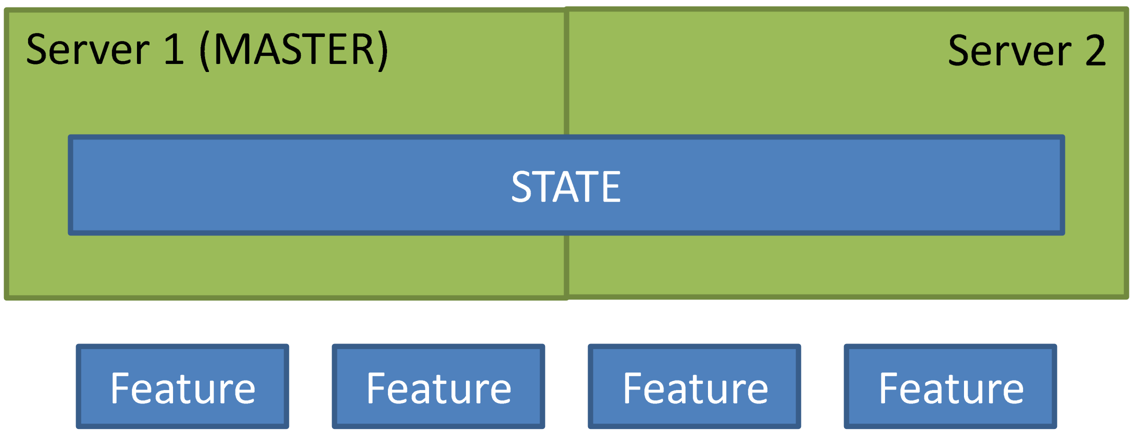
Map-Reduce

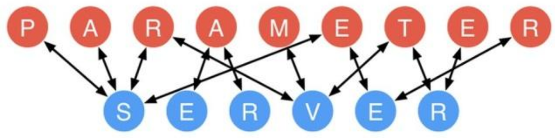
- For one job
  - One master
  - *N* slaves
- Master tracks job state
- Jobs advance in map-reduce rounds

Parameter Server

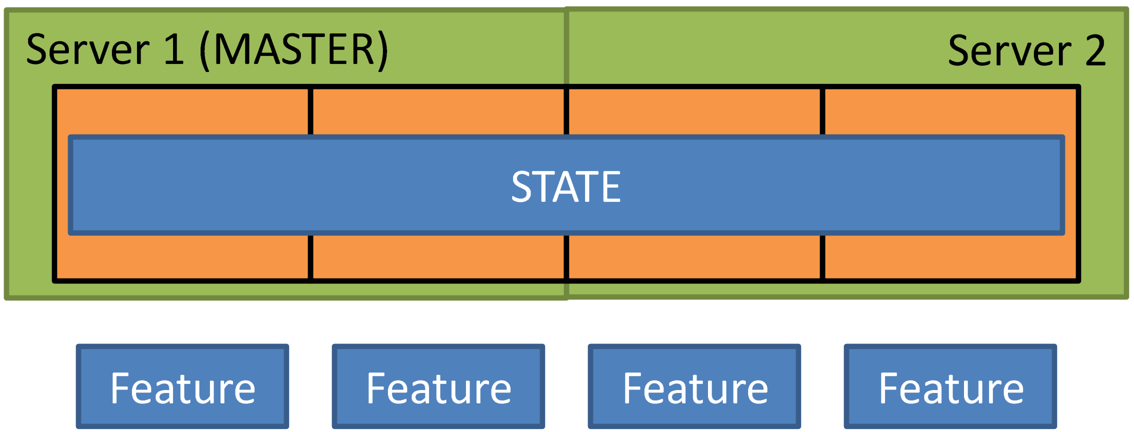
- For one job
  - *K* masters
  - *J* slaves, $j > k$
- State is shared among servers
- Features are learned continuously, in a distributed fashion

Architecture

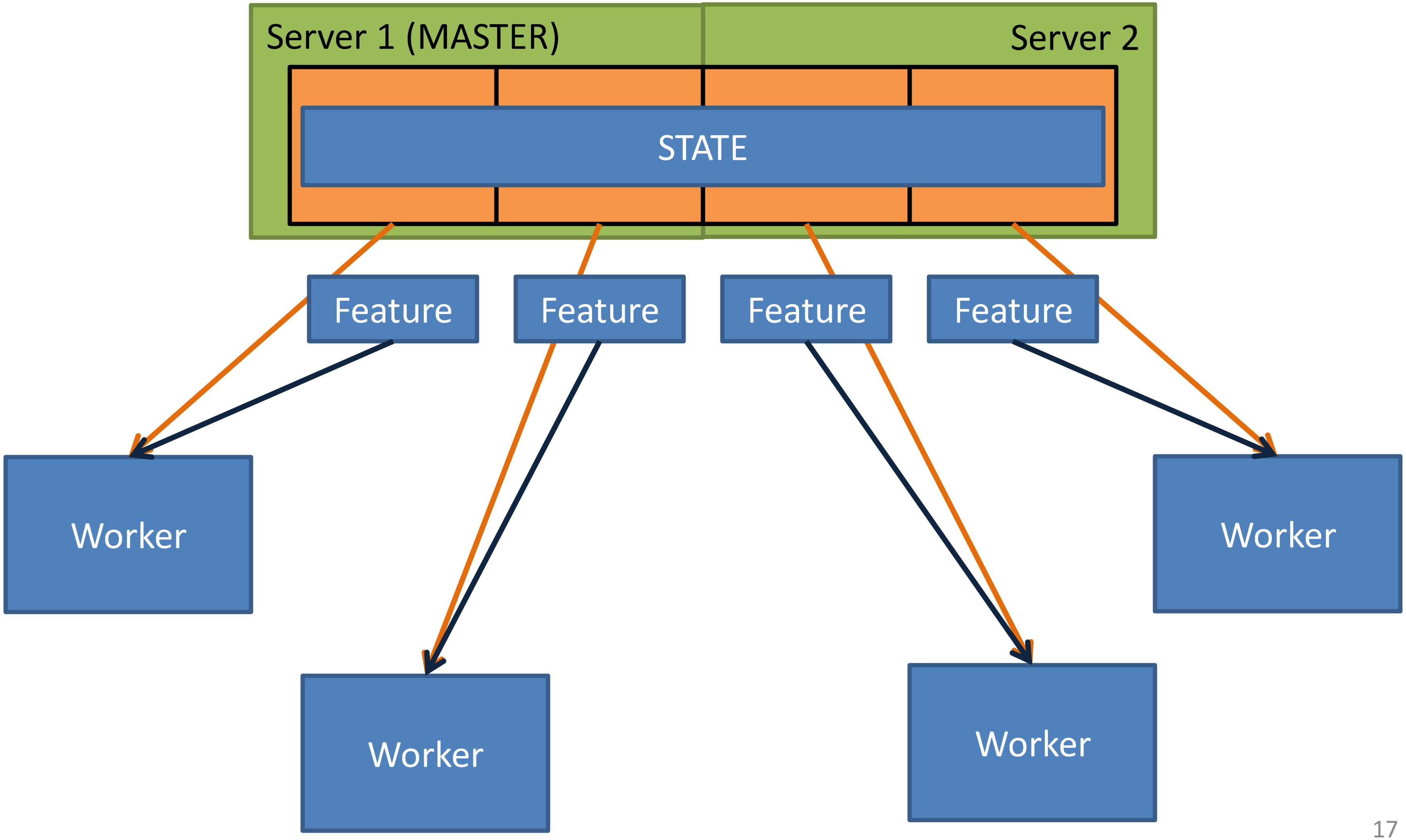




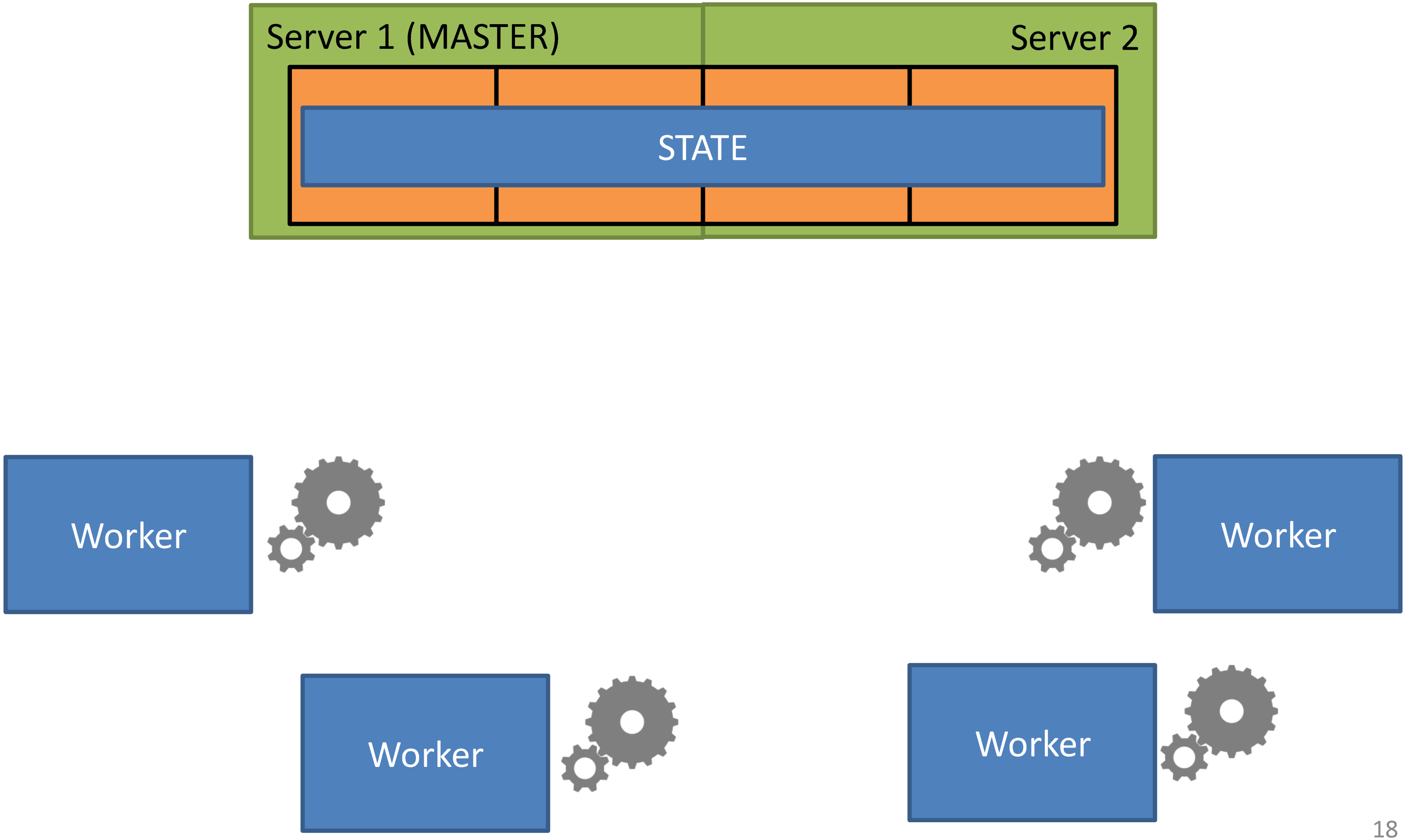
Architecture



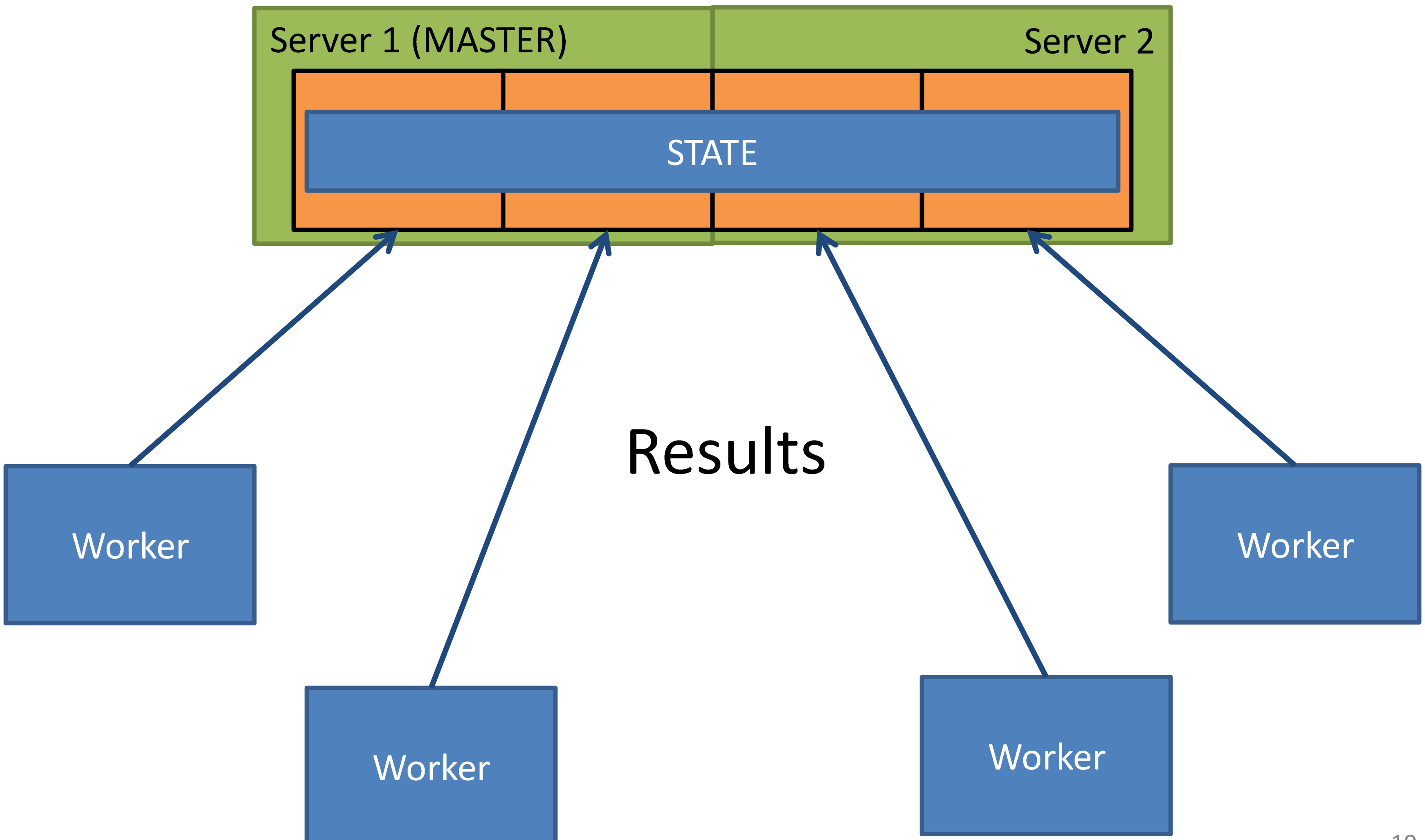
Architecture



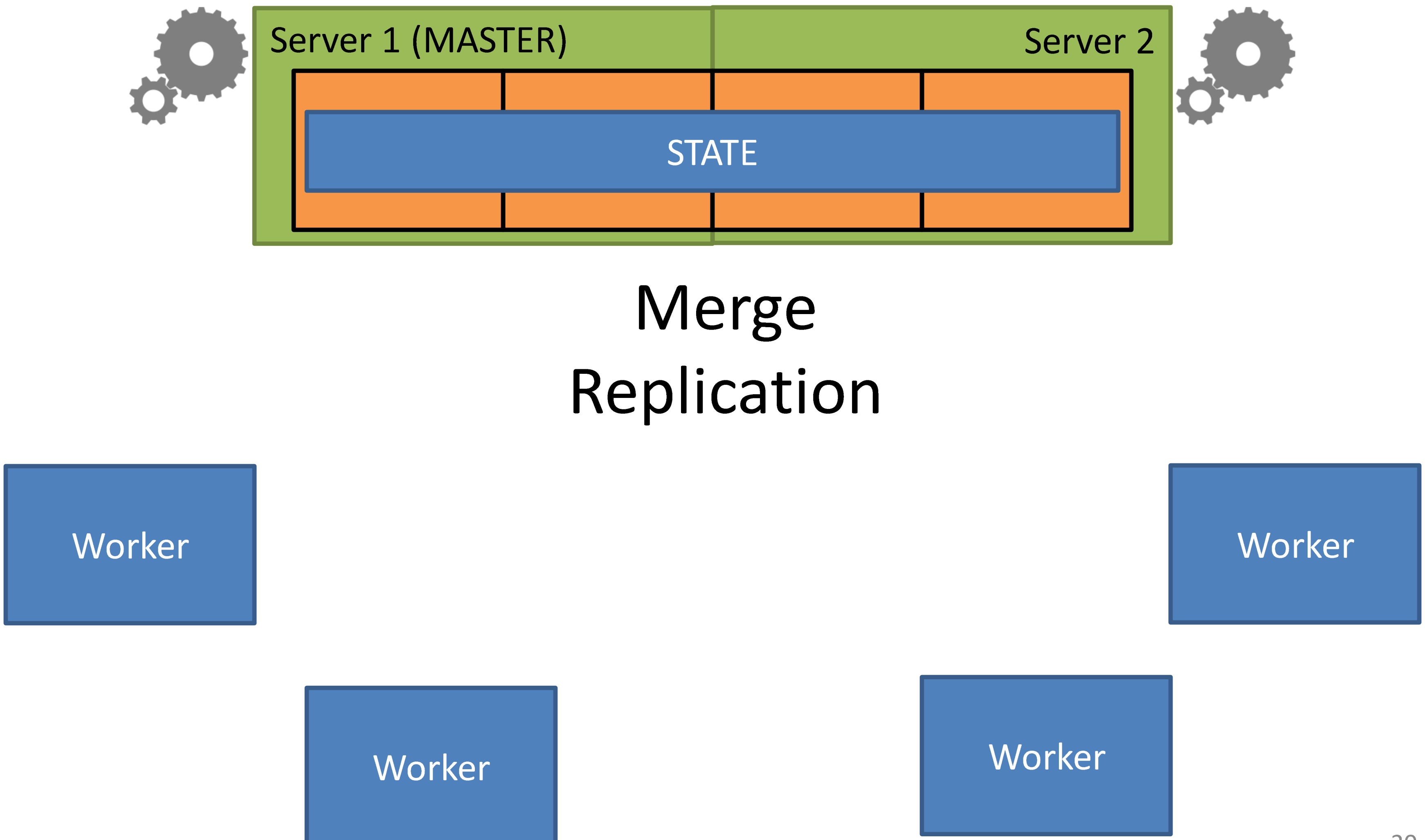
Architecture

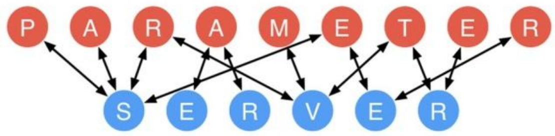


Architecture



Architecture





Job attribution

- Job attribution and result gathering can be sync or async
  - Sync: learning converges in fewer steps
  - Async: more steps can be performed vs sync
- Effectively, it parallelizes learning at cost of converging the state

Fault tolerance

- Fault tolerance through rescheduling jobs
- Replication by duplicating *k* neighbors

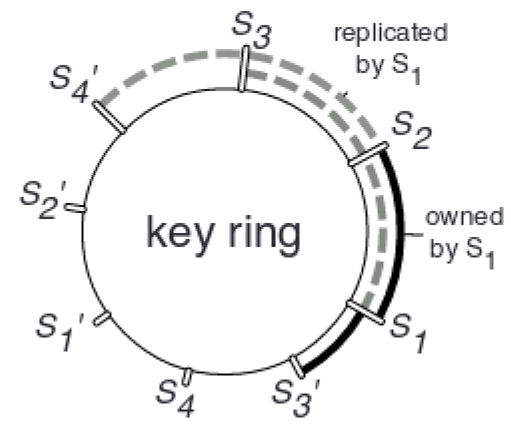


Figure 7: Server node layout. [2]

Systems

TENSOR FLOW

Tensor Flow

- Specific for machine learning
- Similar to/based on parameter server, but adds efficiency mechanisms
  - Plans required jobs
  - Jobs are distributed to the most adequate hardware available
  - Uses specialized, efficient software
- How?

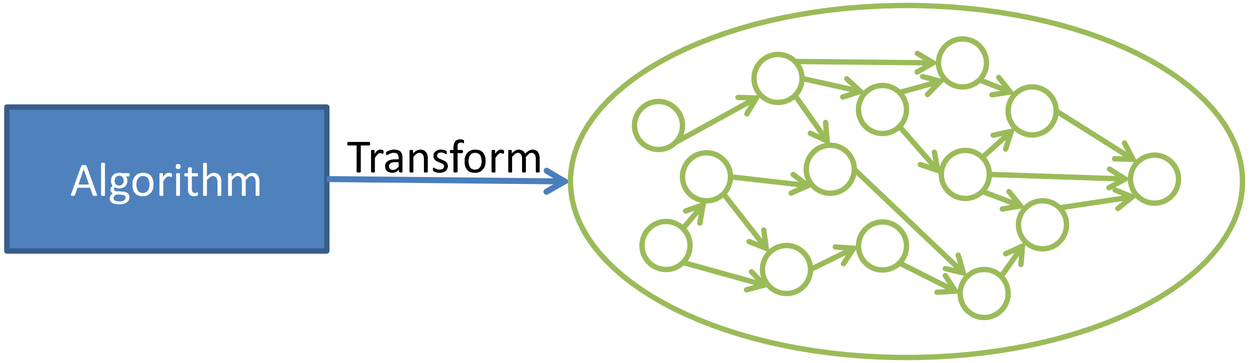
Job attribution

- Transform algorithm (task) into a graph format
- Evaluates the available resources
  - CPUs
  - GPUs (for acceleration)
- Attributes the nodes (jobs) of the graph to the resources

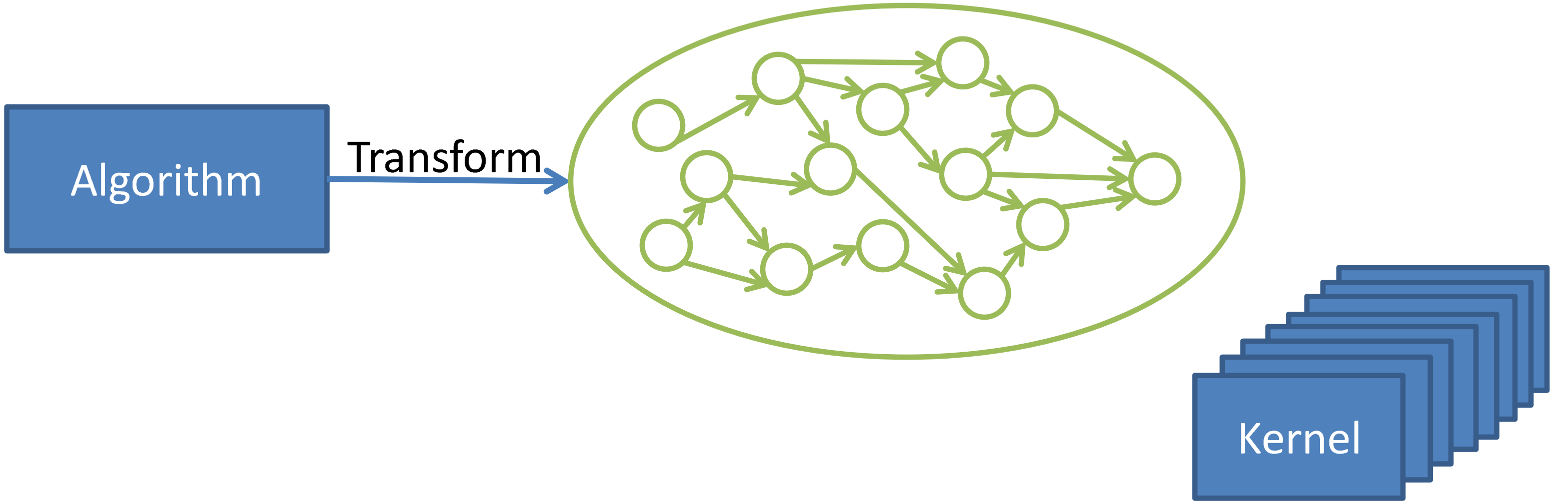
Job attribution

- Each job is a set of operations
- Operations are mathematical, such as matrix addition
- Each operation is implemented in a kernel
- A large set of kernels is available
  - The set is expansible

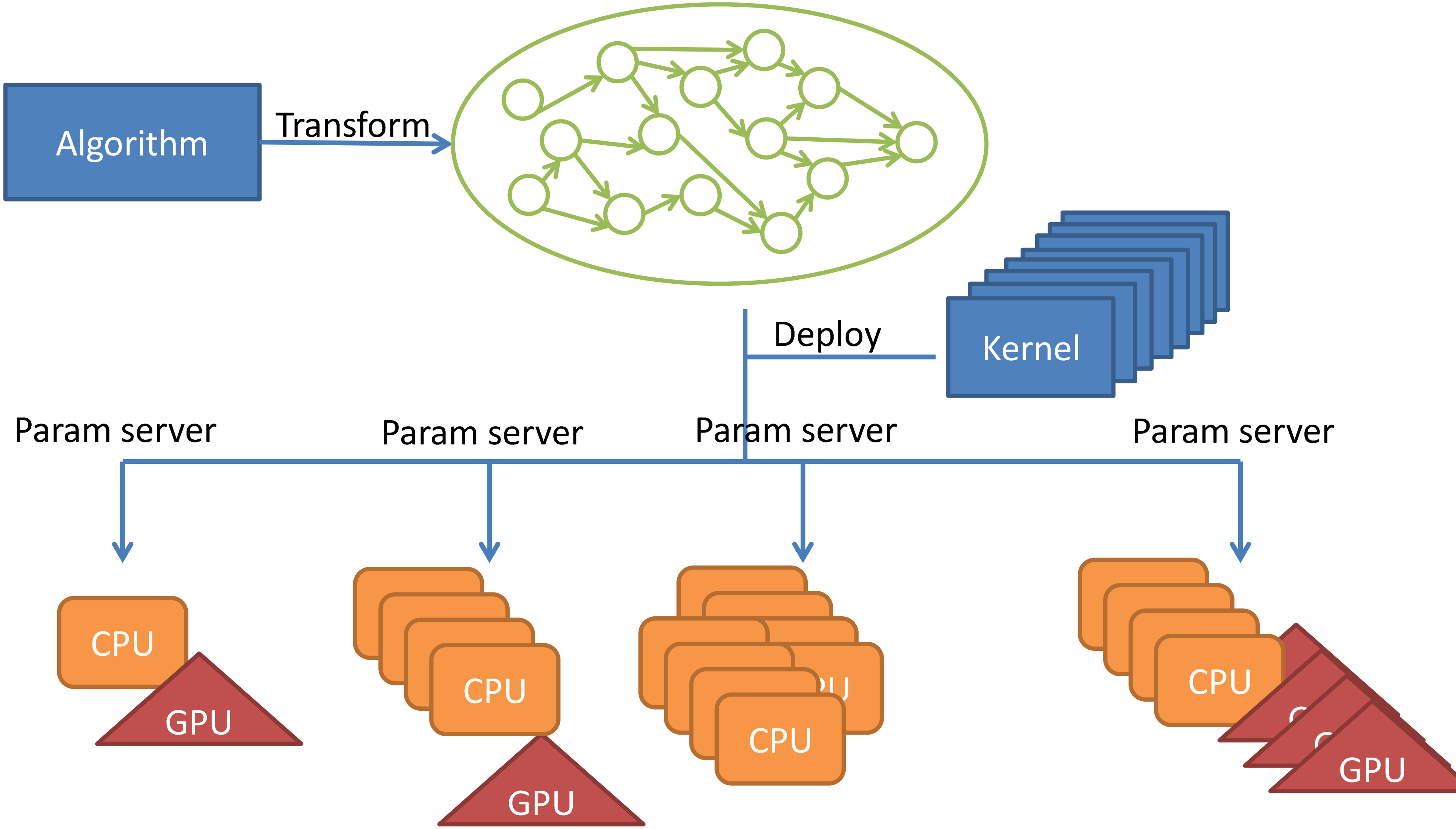
Architecture



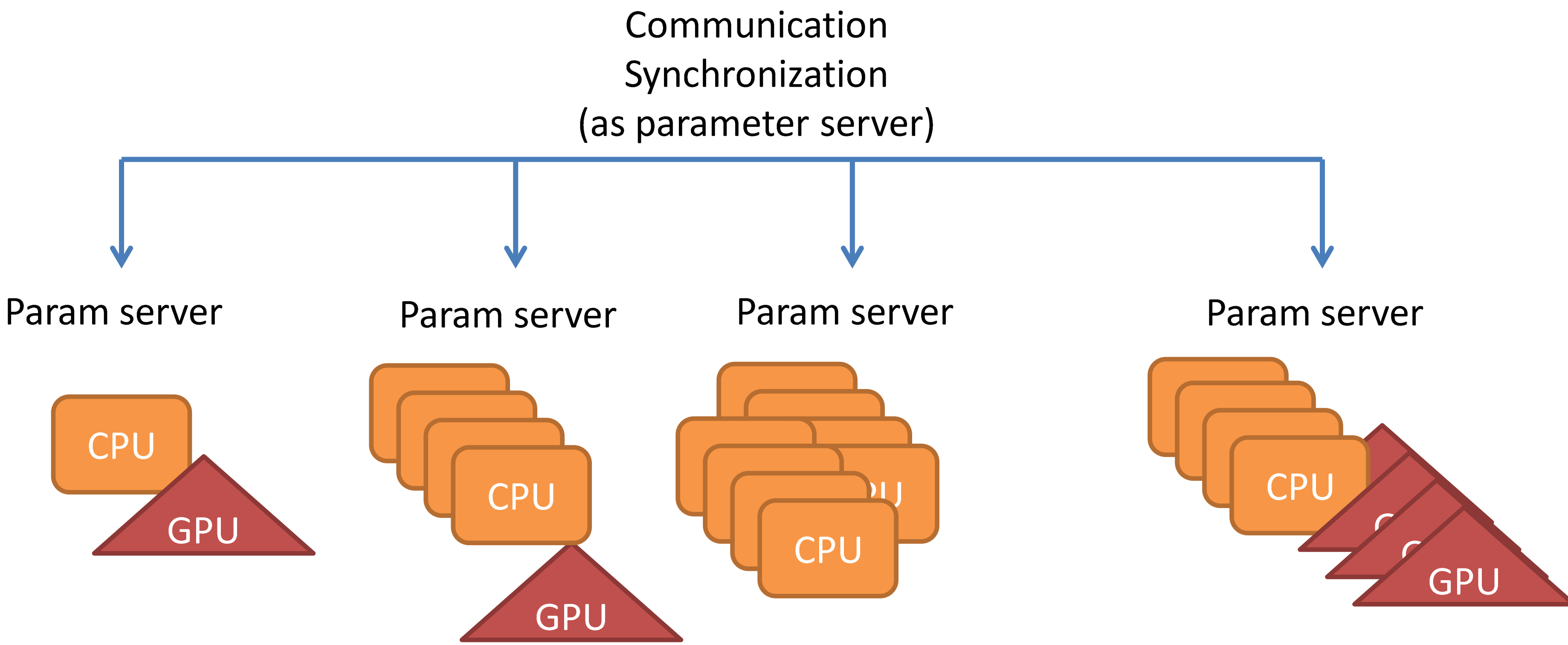
Architecture



Architecture



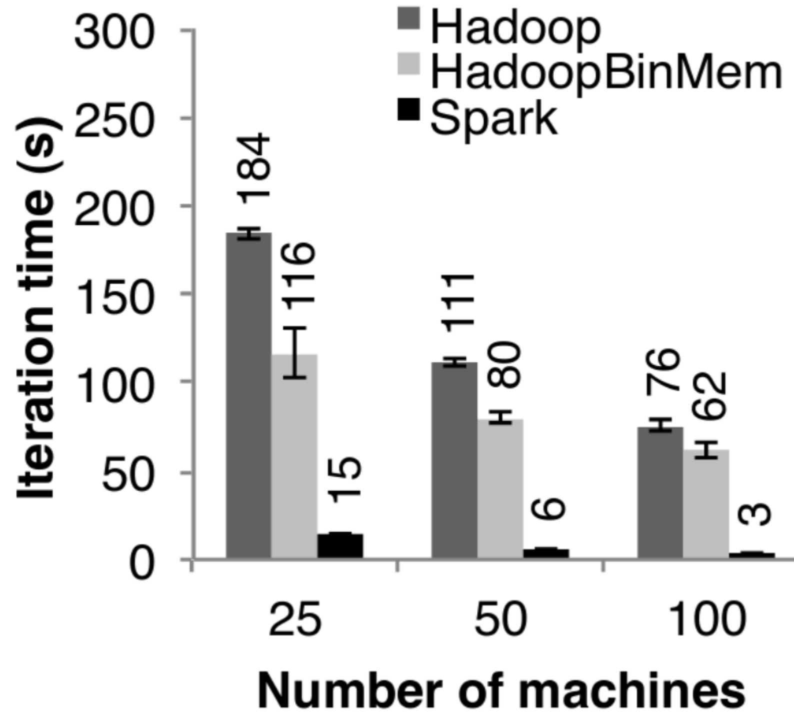
Architecture



EVALUATION

Hadoop and Spark

- Amazon EC2 m1.xlarge machines
  - 4 cores
  - 15GB RAM



(a) Logistic Regression

Parameter Server

- Sparse Logistic Regression
  - Ad click prediction dataset with 170 billion examples and 65 billion unique features
  - This dataset is 636 TB
  - Parameter server on 1000 machines:
    - 16 cores, 192GB DRAM, connected by 10 Gb Ethernet
    - 800 workers, and 200 parameter servers
  - The cluster was in concurrent use by other (unrelated) tasks during operation.

Parameter Server

- Sparse Logistic Regression

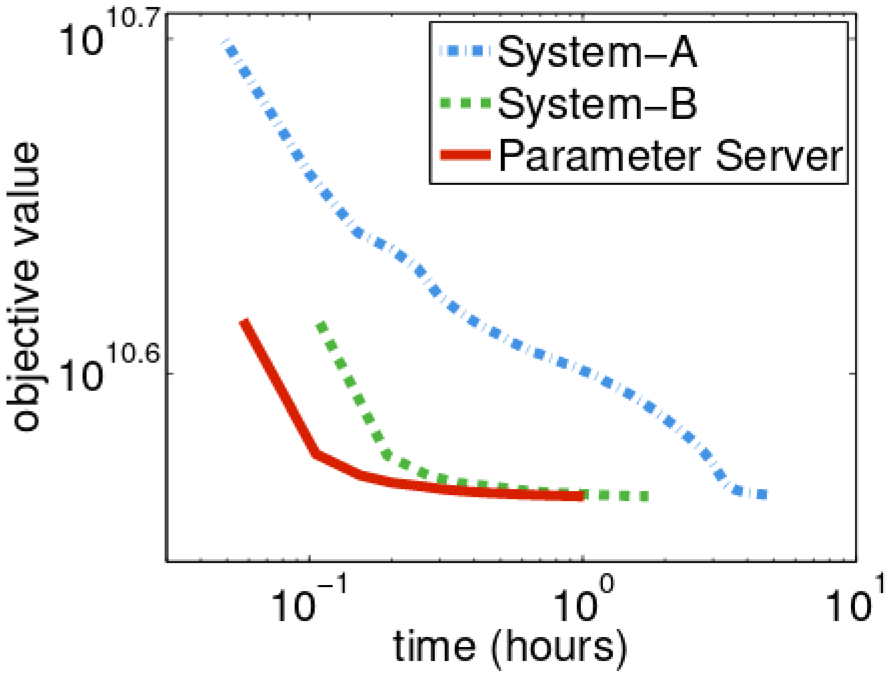


Figure 9: Convergence of sparse logistic regression. The goal is to minimize the objective rapidly.

Parameter Server

- Sparse Logistic Regression

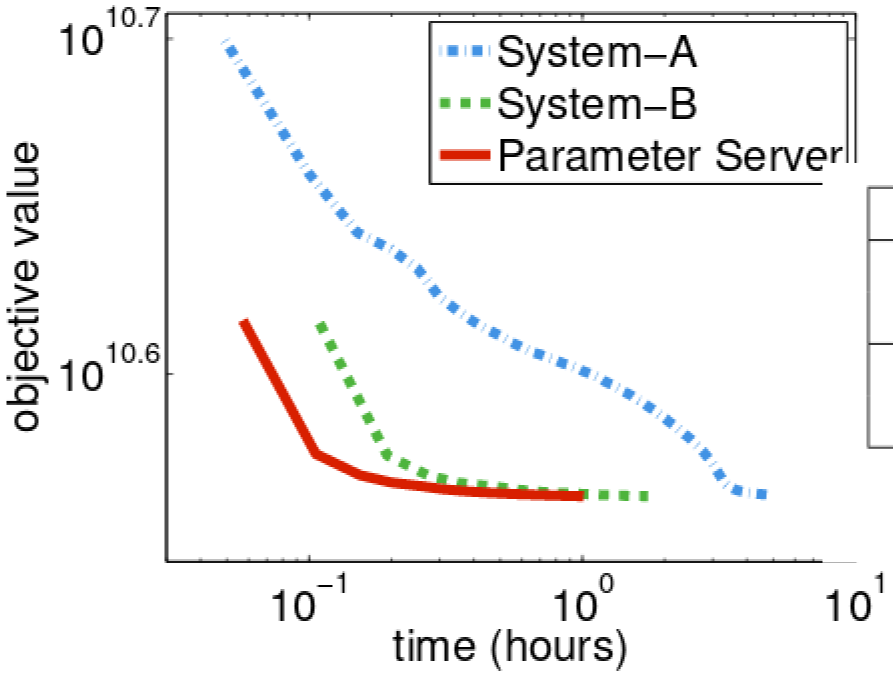


Figure 9: Convergence of sparse logistic regression. The goal is to minimize the objective rapidly.

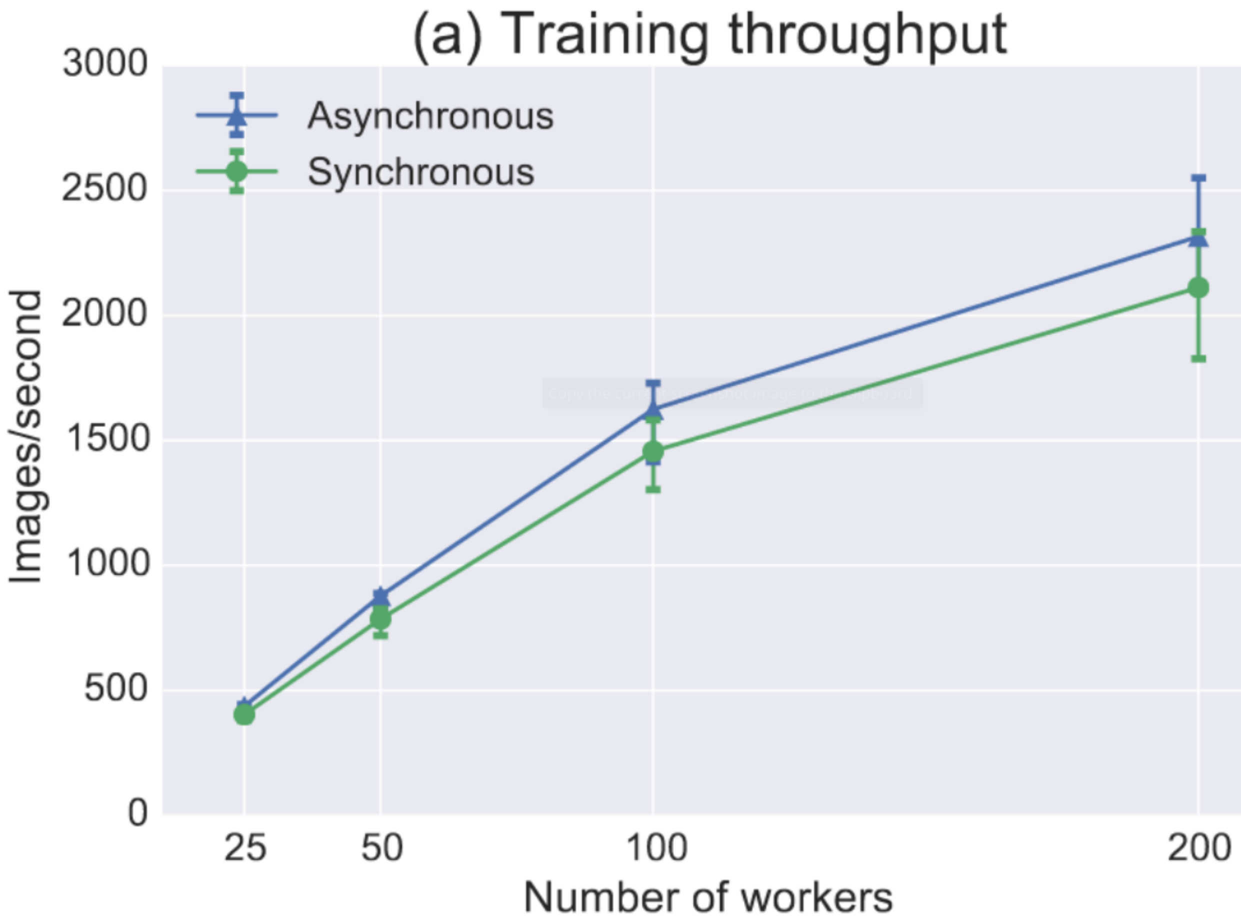
	Method	Consistency	LOC
System A	L-BFGS	Sequential	10,000
System B	Block PG	Sequential	30,000
Parameter Server	Block PG	Bounded Delay KKT Filter	300

Table 3: Systems evaluated.

Tensor Flow

- Google's Inception-v3 model (Google image recognition system using neural networks)
- 17 Param. servers, each with 8 IvyBridge cores
- Variable number of workers, each with
  - NVIDIA K40 GPU (12GB GDDR5, 1.43 double-precision Tflops, 4.29 single-precision Tflops)
  - 5 IvyBridge cores

Tensor Flow



(a) Training throughput
Asynchronous
Synchronous
Images/second
3000
2500
2000
1500
1000
500
0
25
50
100
200
Number of workers

Conclusions

- Map-Reduce covers a large set of problems, but…
- Specific problems require specialized approaches
- Parameter Servers and Tensor Flow specialize in math-based problems, with clear benefits

Bibliography

[1] Zaharia, Matei, et al. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing." Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association, 2012.

[2] Li, Mu, et al. "Scaling distributed machine learning with the parameter server." 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14). 2014.

[3] Abadi, Martín, et al. "TensorFlow: A system for large-scale machine learning." arXiv preprint arXiv:1605.08695 (2016).

Appendix

- Spark
  - https://github.com/apache/spark
  - Built with Maven

- Parameter server
  - https://github.com/dmlc/ps-lite
  - Built with Make build system

- TensorFlow
  - https://github.com/tensorflow/tensorflow/
  - Install with python package manager *pip*