11868 LLM Systems
Deep Learning Framework Design

Lei Li

Carnegie Mellon University
Language Technologies Institute

Recap

- Learning algorithm for Neural Network
    - stochastic gradient descent
- Computation Graph
    - topological traversal along the DAG
- Auto Differentiation
    - building backward computation graph

Today's Topic

- How to design a deep learning framework
  - Design ideas in TensorFlow
    - Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning", OSDI 2016

Need for DL Programming System

- Deep learning already claiming big successes
- Huge need for high-productivity tools for developing machine learning solutions for various applications
- Instead of writing cuda and differentiation code for each specific model

Deep Learning Programming Framework

- Open source library for machine learning computation using data flow graphs

- TensorFlow is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms

- PyTorch is a programming framework for tensor computation, deep learning, and auto differentiation

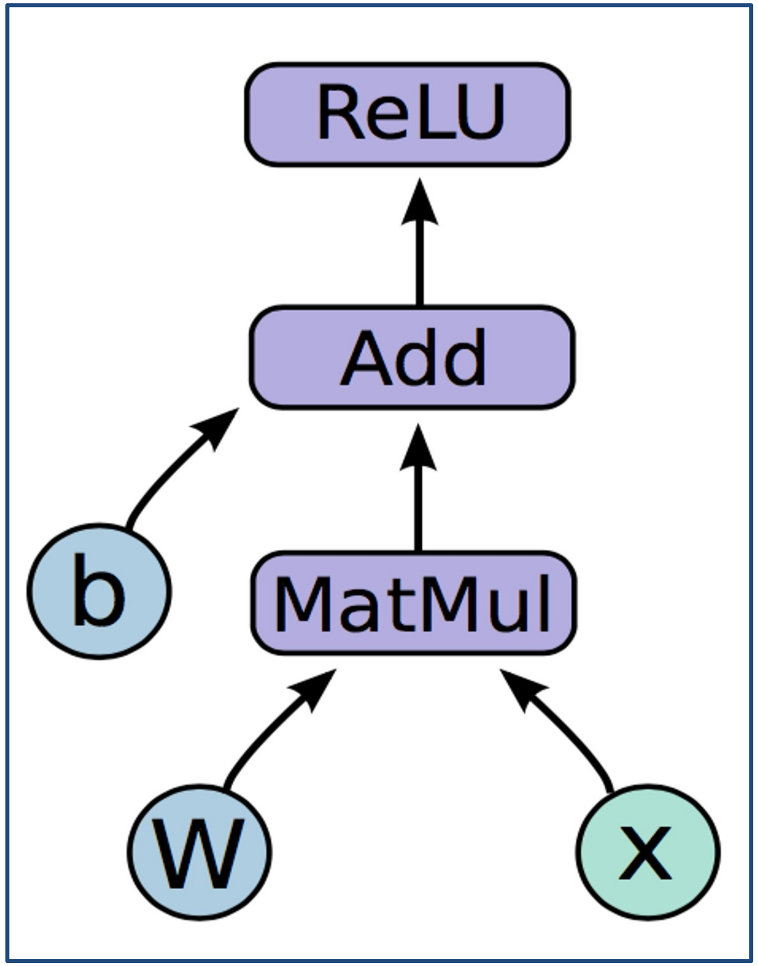
TensorFlow

- Key idea: express a numeric computation as a computation graph
  - following what we described in last lecture
- Graph nodes are operations with any number of inputs and outputs
- Graph edges are tensors which flow between nodes
  - tensor: multidimensional array

Programming Model

Computation graph in tensorflow

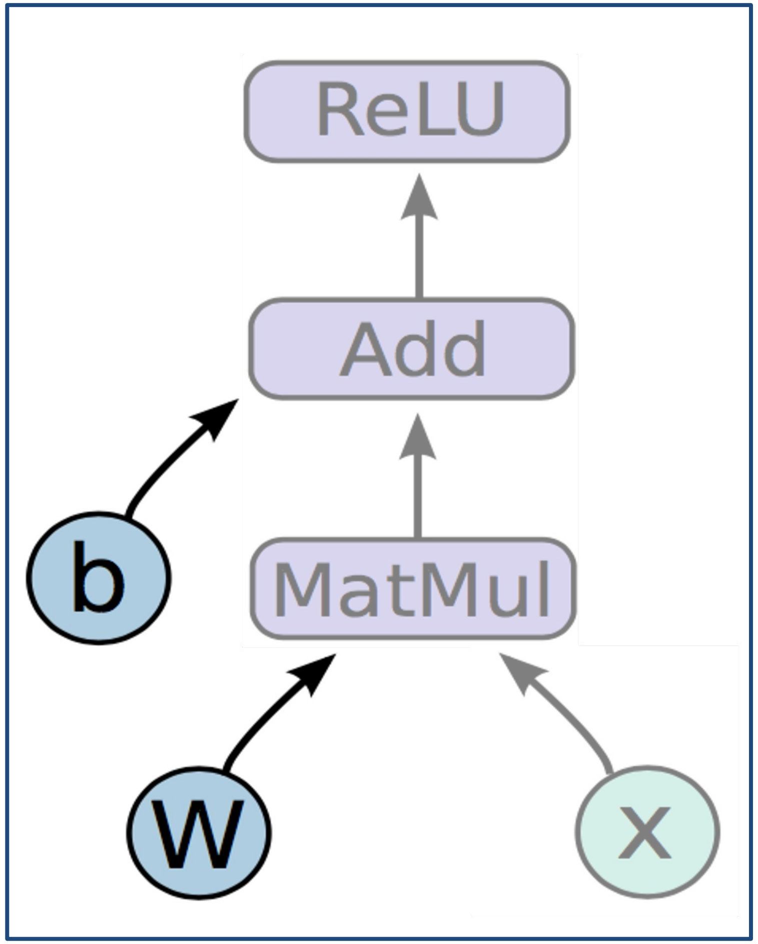
$$h = RELU(Wx + b)$$



Variables

$$h = RELU(Wx + b)$$

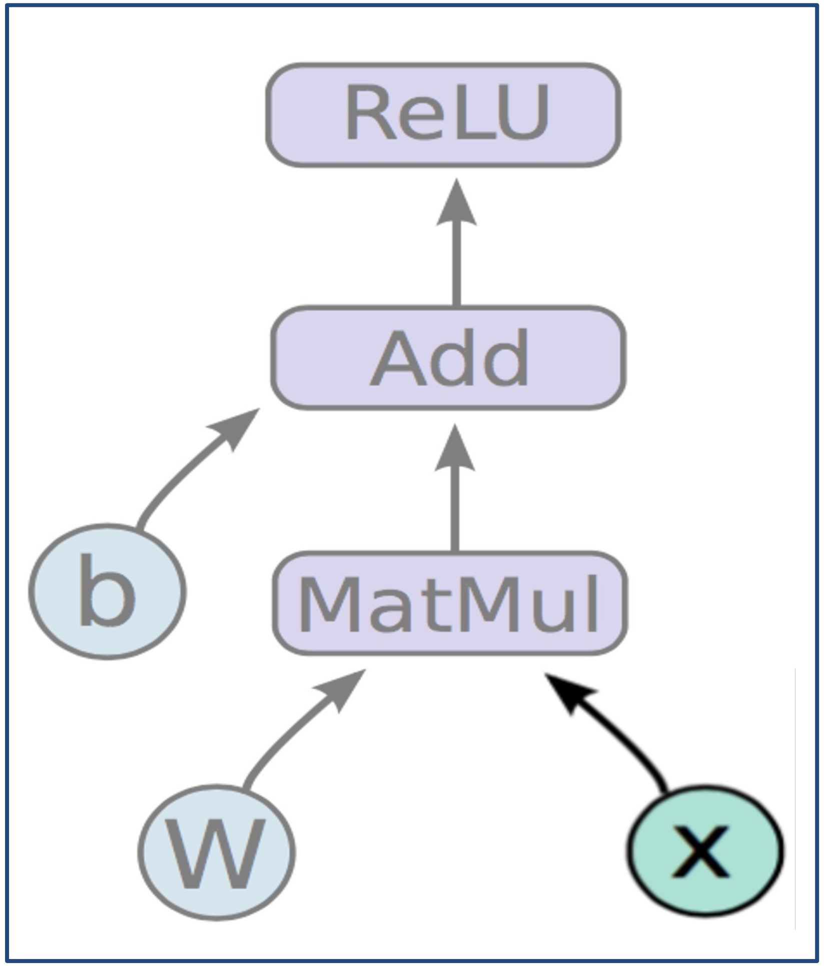
- Variables are stateful nodes which output their current value.
- State is retained across multiple executions of a graph
- mostly parameters



Placeholders

$$h = RELU(Wx + b)$$

- Placeholders are nodes whose value is fed in at execution time
- Inputs, Labels, …

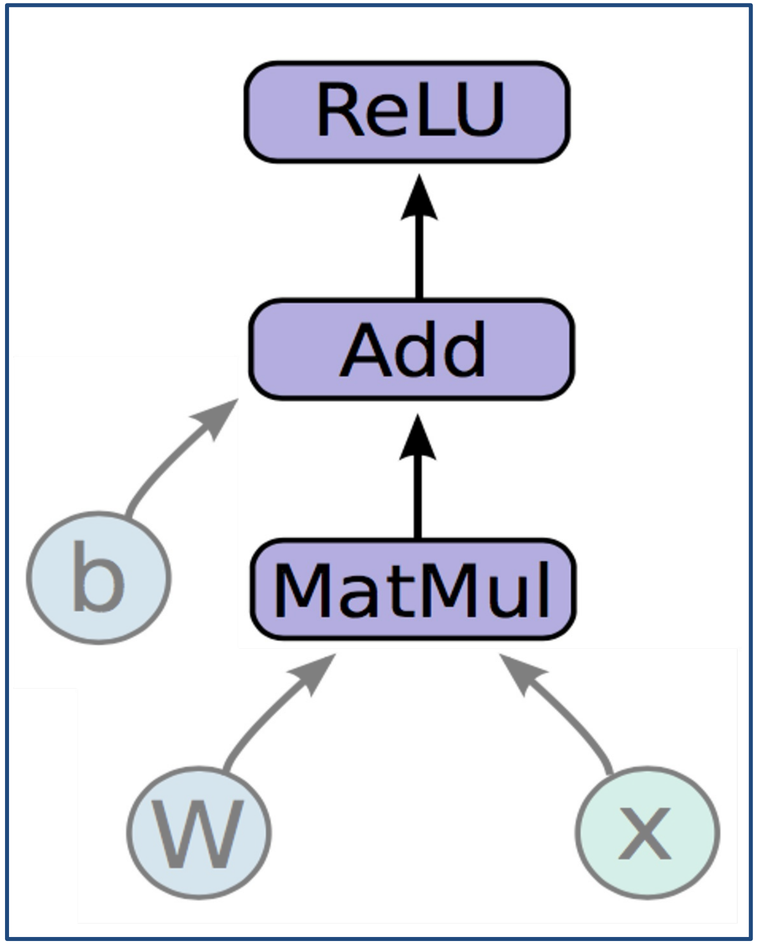


Mathematical Operations

$$h = RELU(Wx + b)$$

MatMul: Multiply two matrices
Add: Add elementwise
ReLU: Activate with elementwise rectified linear function

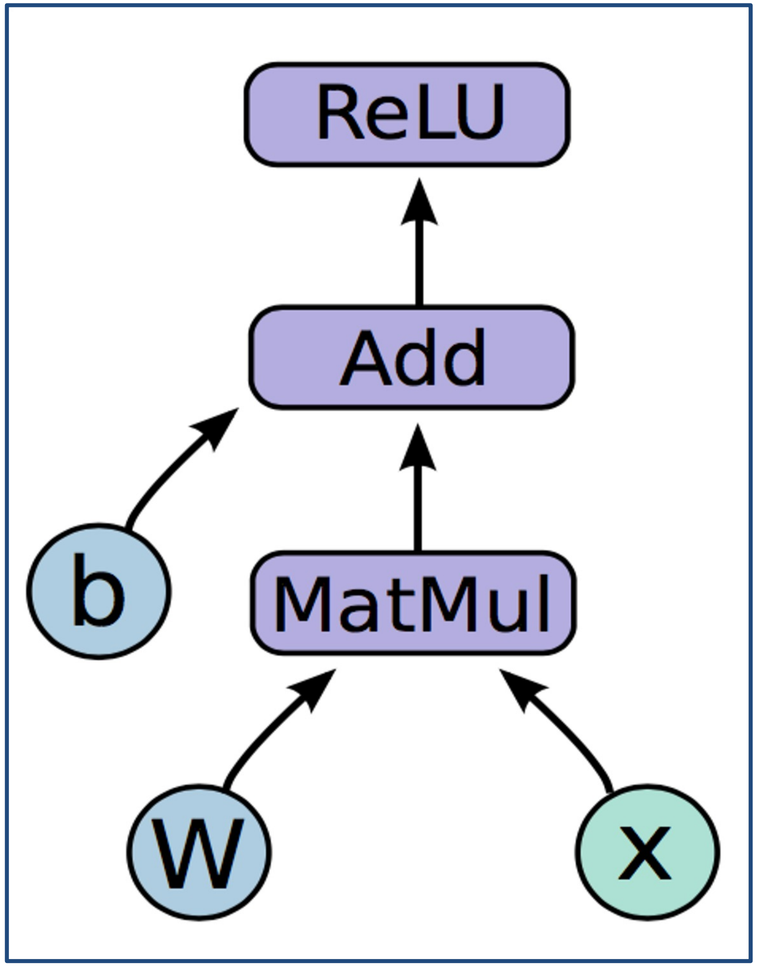
$$ReLu(x) = \begin{cases} 0, & x <= 0 \\ x, & x > 0 \end{cases}$$



Programming the Graph

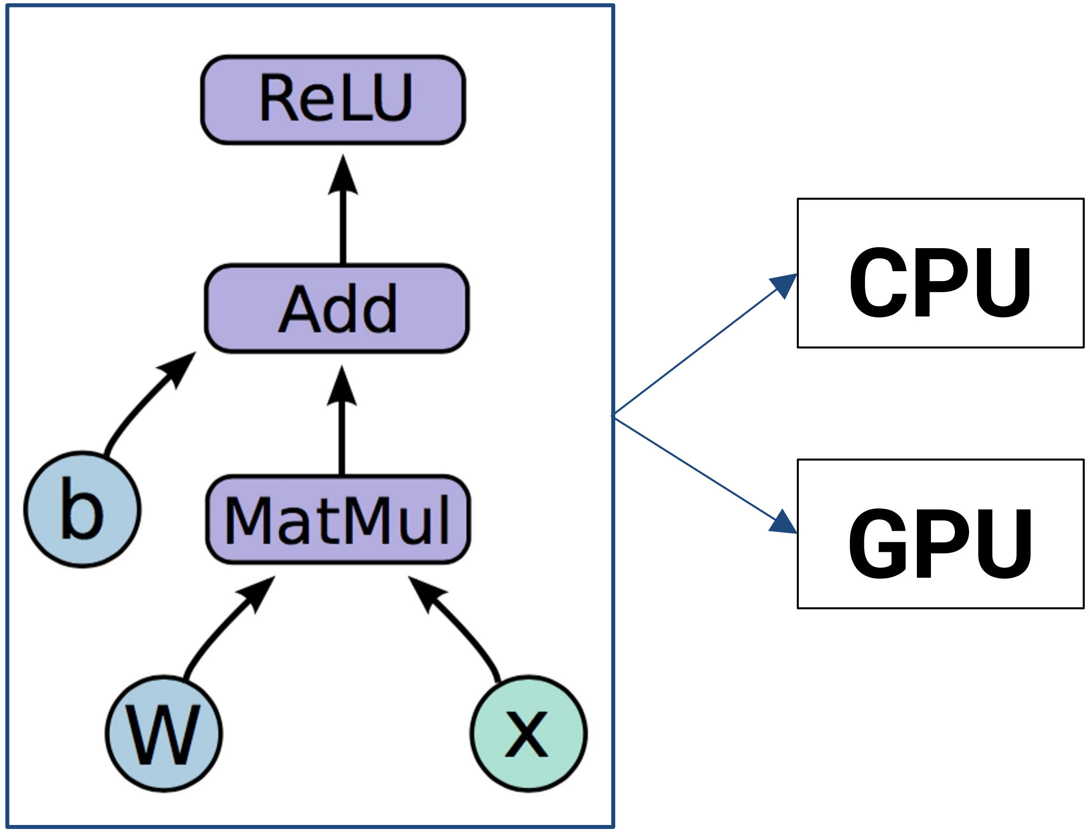
```
import tensorflow as tf

b = tf.Variable(tf.zeros((100,)))
W = tf.Variable(tf.random_uniform((784, 100), -
1, 1))

x = tf.placeholder(tf.float32, (1, 784))

h = tf.nn.relu(tf.matmul(x, W) + b)
```

$$h = RELU(Wx + b)$$



Running the Graph

Deploy graph with a session: a binding to a particular execution context (e.g. CPU, GPU)



```python
import tensorflow as tf

with tf.Session() as sess:
  # Phase 1: constructing the graph
  a = tf.constant(15, name="a")
  b = tf.constant(5, name="b")
  prod = tf.multiply(a, b, name="Multiply")
  sum = tf.add(a, b, name="Add")
  res = tf.divide(prod, sum, name="Divide")

  # Phase 2: running the session
  out = sess.run(res)
  print(out)
```

Defining Loss

- Use **placeholder** for **labels**
- Build loss node using labels and **prediction**

```
prediction = tf.nn.softmax(...)  #Output of neural
network
label = tf.placeholder(tf.float32, [100, 10])

cross_entropy = -tf.reduce_sum(label *
tf.log(prediction), axis=1)
```

Gradient Computation

```
train_step =
tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
```

- tf.train.GradientDescentOptimizer is an Optimizer object
- tf.train.GradientDescentOptimizer(lr).minimize(cross_entropy) adds optimization operation to computation graph
- TensorFlow graph nodes have attached gradient operations
- Gradient with respect to parameters computed with Auto Differentiation (recall previous lecture)

Core TensorFlow Constructs

- All nodes return tensors, or higher-dimensional matrices
- How a node computes is indistinguishable to TensorFlow
- You are metaprogramming - constructing the graph for the real computation. No computation occurs yet!

Implementing Graph Nodes

Design Principles

- Dataflow graphs of primitive operators
- Deferred execution (two phases)
    1. Define program i.e., symbolic dataflow graph w/ placeholders
    2. Executes optimized version of program on set of available devices

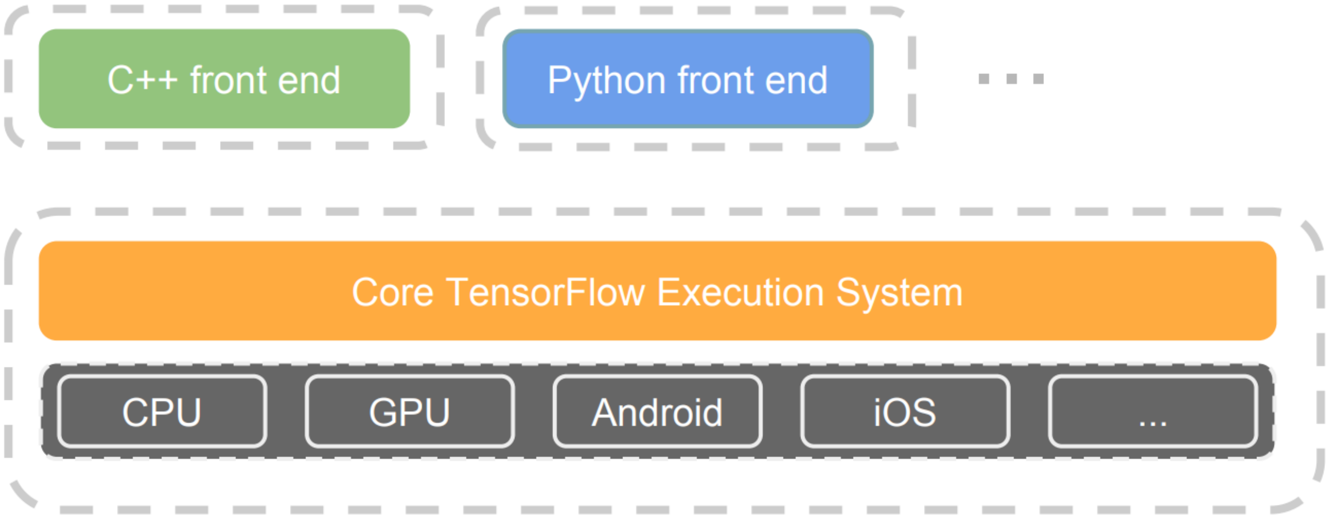
Dynamic Flow Control

- Problem: support ML algos that contain conditional and iterative control flow, e.g.
  - Recurrent Neural Networks (RNNs) and LSTMs
  - Autoregressive decoder
- Solution: Add conditional (if statement) and iterative (while loop) programming constructs

TensorFlow Architecture

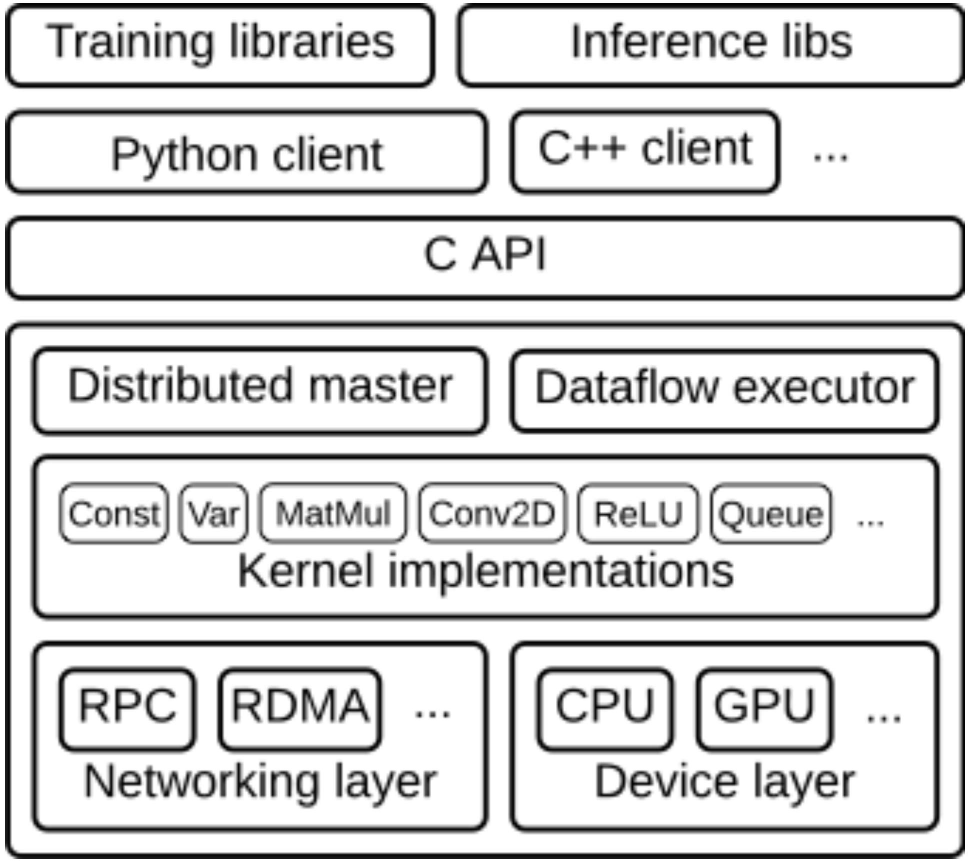
- Core in C++
  - Very low overhead

- Different front ends for specifying/driving the computation
  - Python and C++, easy to add more



TensorFlow Implementation

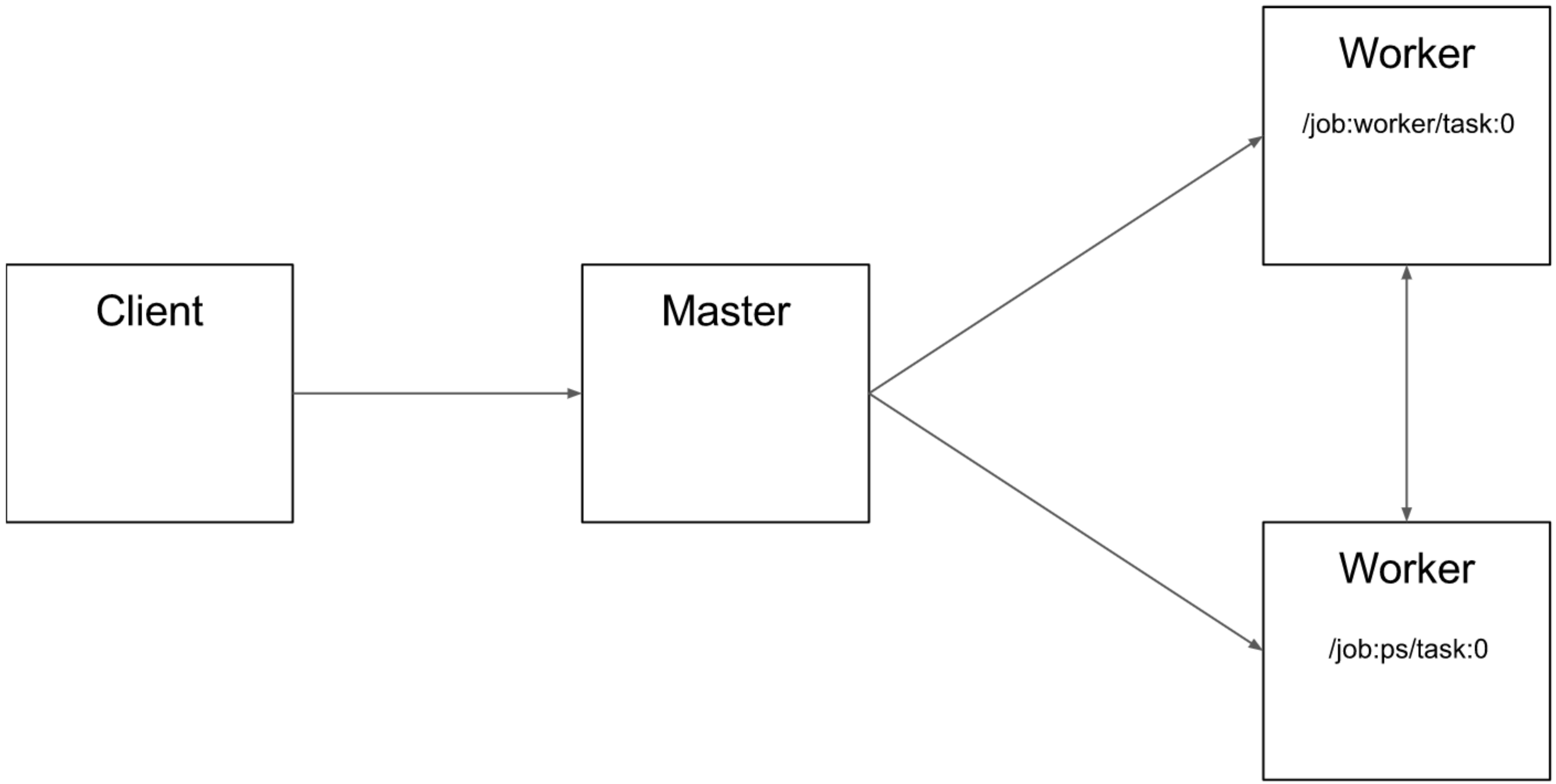
- Semi-interpreted
- Call to kernel per primitive operation
- Can batch operations with custom C++
- Basic type-safety within dataflow graph (error at graph construction time)



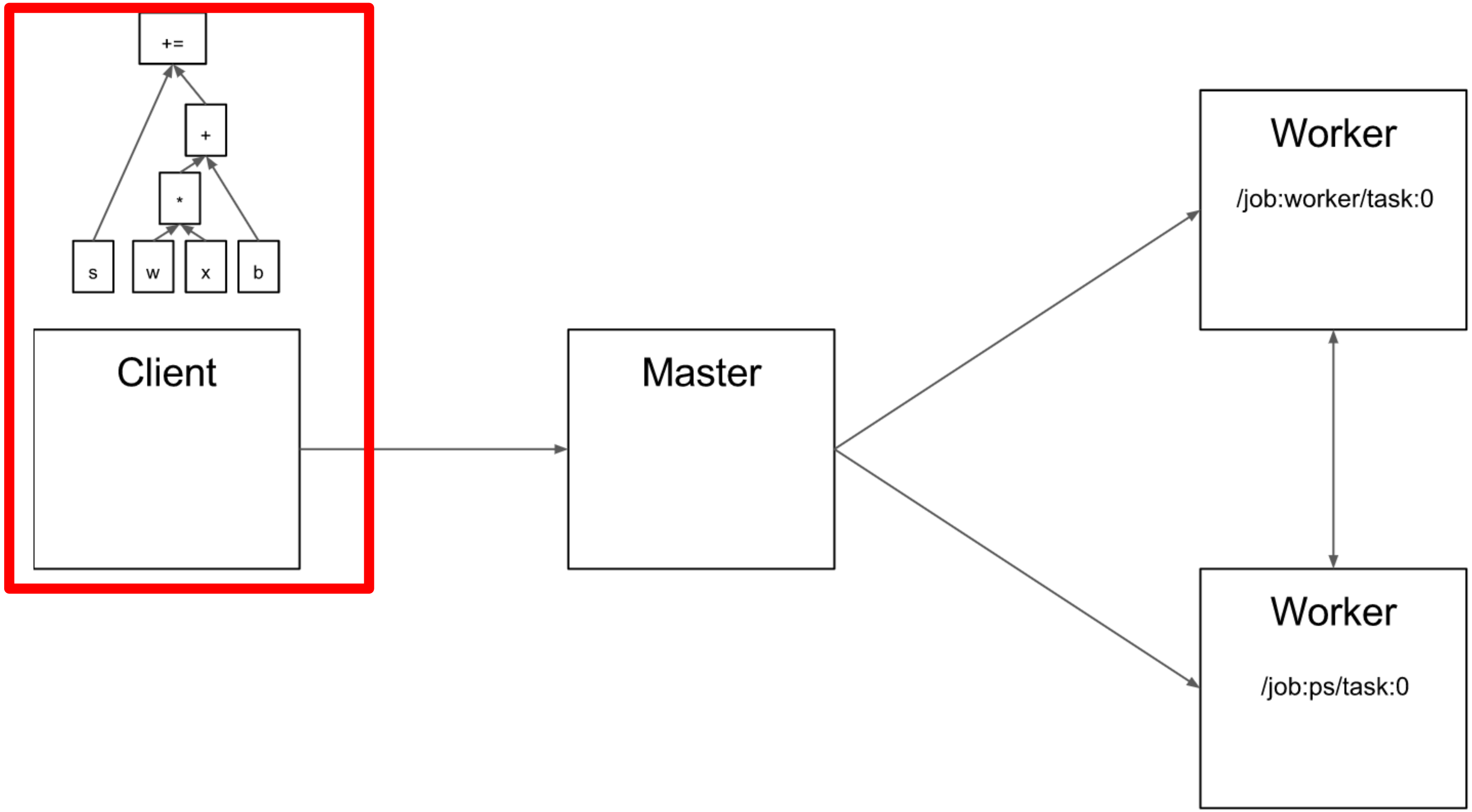
https://www.tensorflow.org/extend/architecture

Key Components

- Similar to MapReduce, Apache Hadoop, Apache Spark, …

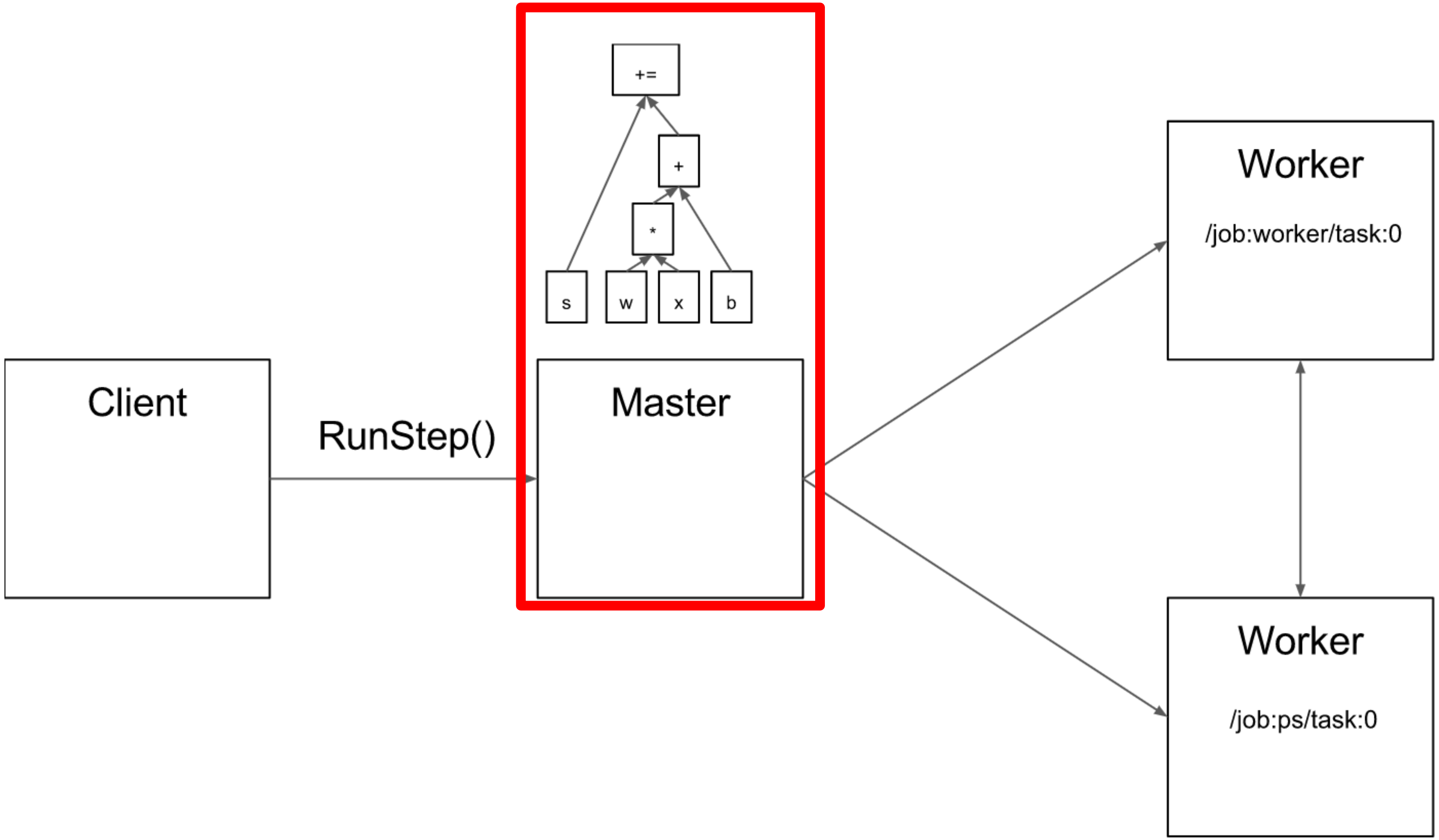


Client



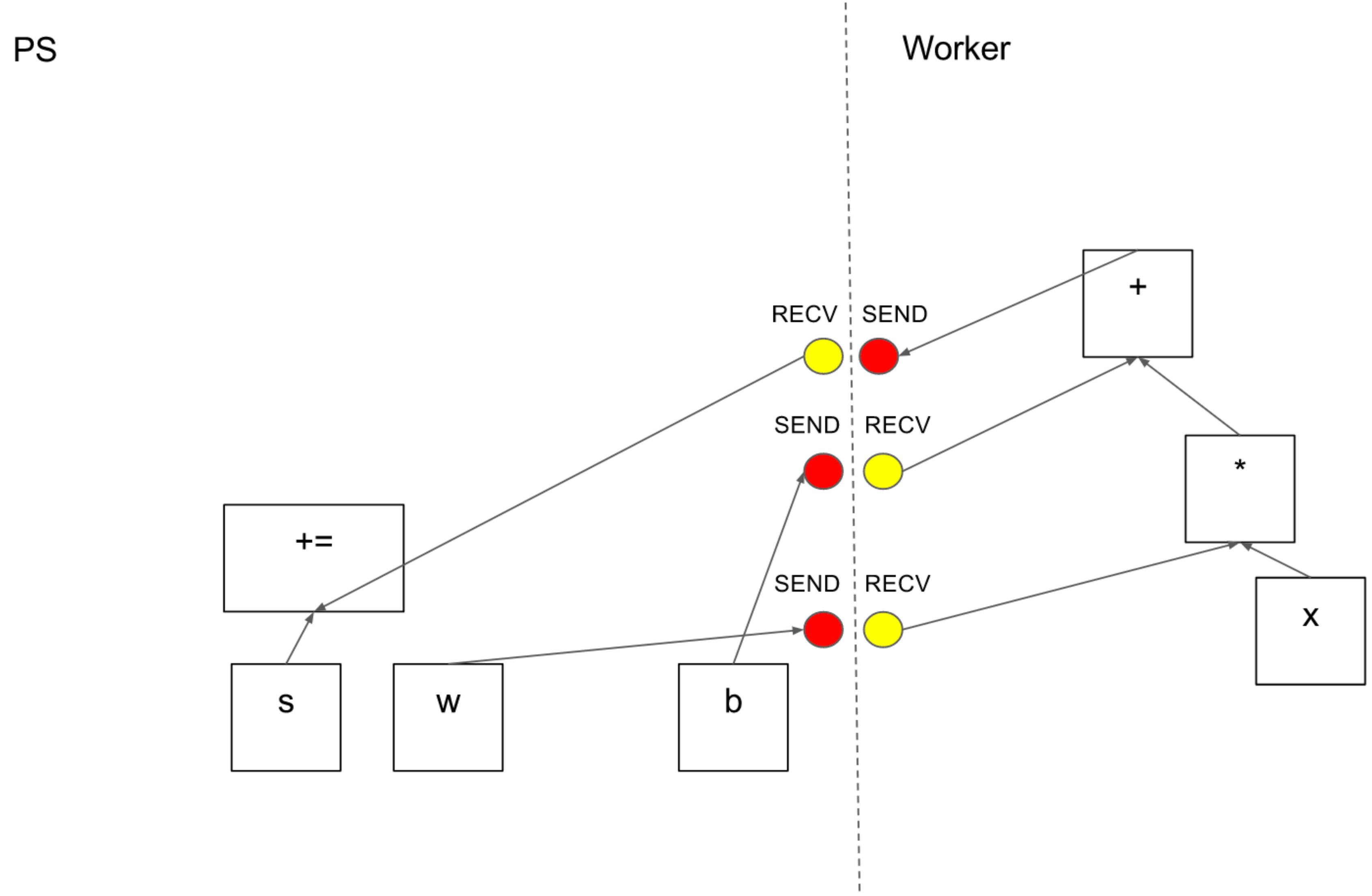
+=
+
*
s
w
x
b
Client
Master
Worker
/job:worker/task:0
Worker
/job:ps/task:0

Master

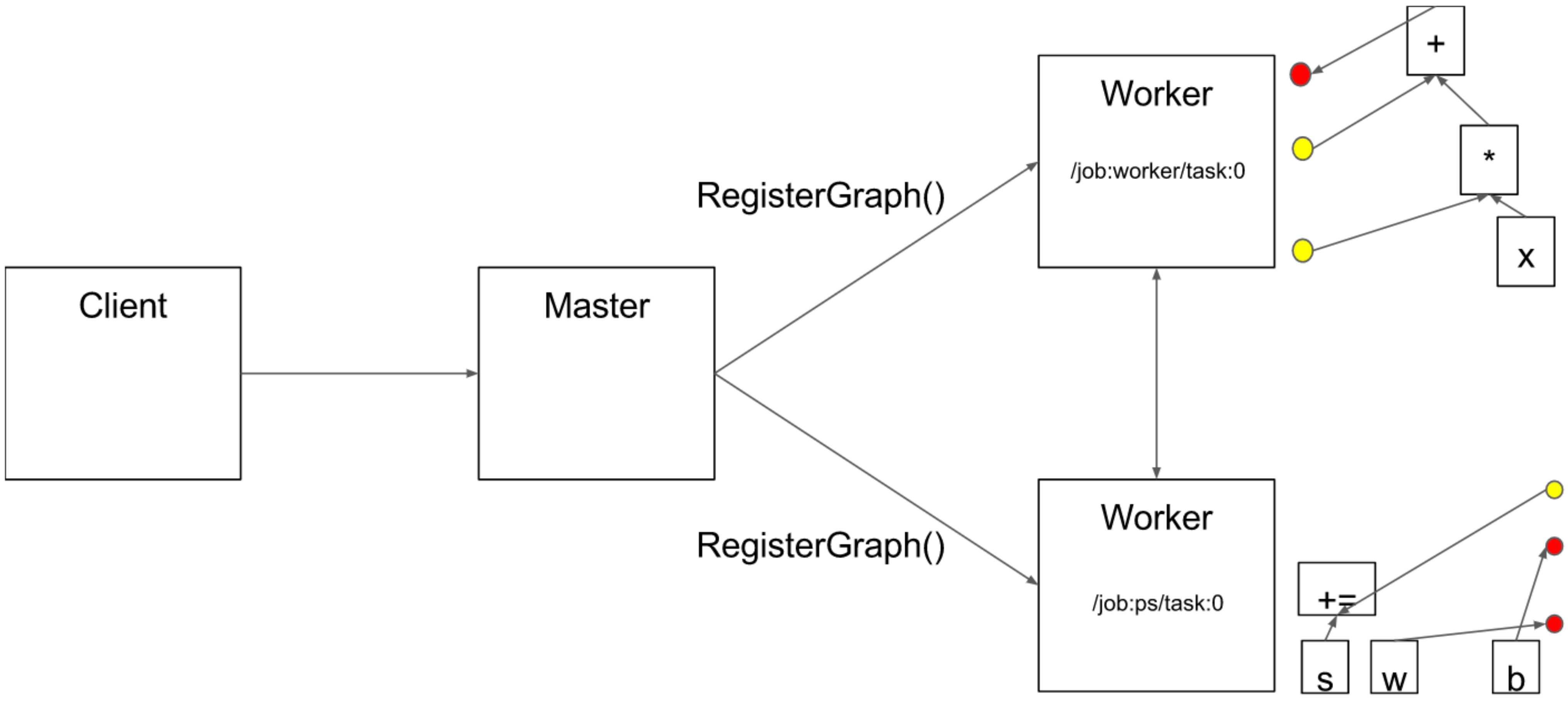


+=
+
*
s
w
x
b
Client
RunStep()
Master
Worker
/job:worker/task:0
Worker
/job:ps/task:0

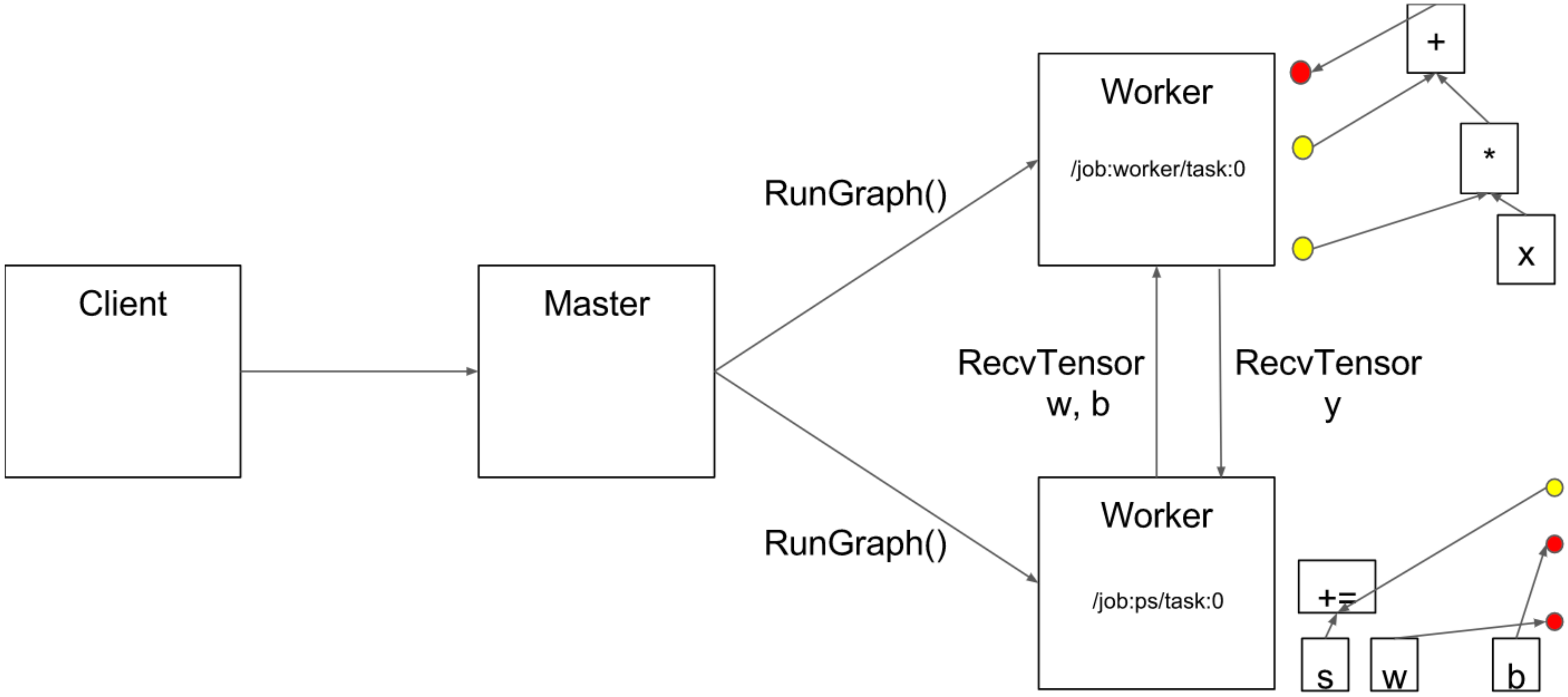
Computation Graph Partition



Computation Graph Partition



Execution



Client
Master
RunGraph()
RunGraph()
Worker
/job:worker/task:0
Worker
/job:ps/task:0
RecvTensor
w, b
RecvTensor
y
+
*
x
+=
s
w
b

Synchronous vs Asynchronous

- Determined by node: Queue nodes used for barriers
- Synchronous nearly as fast as asynchronous
- Default model is asynchronous

Fault Tolerance

- Assumptions:
  - Fine grain operations: “It is unlikely that tasks will fail so often that individual operations need fault tolerance” ;-)
  - “Many learning algorithms do not require strong consistency”
- Solution: user-level checkpointing (provides 2 ops)
  - save(): writes one or more tensors to a checkpoint file
  - restore(): reads one or more tensors from a checkpoint file

Performance

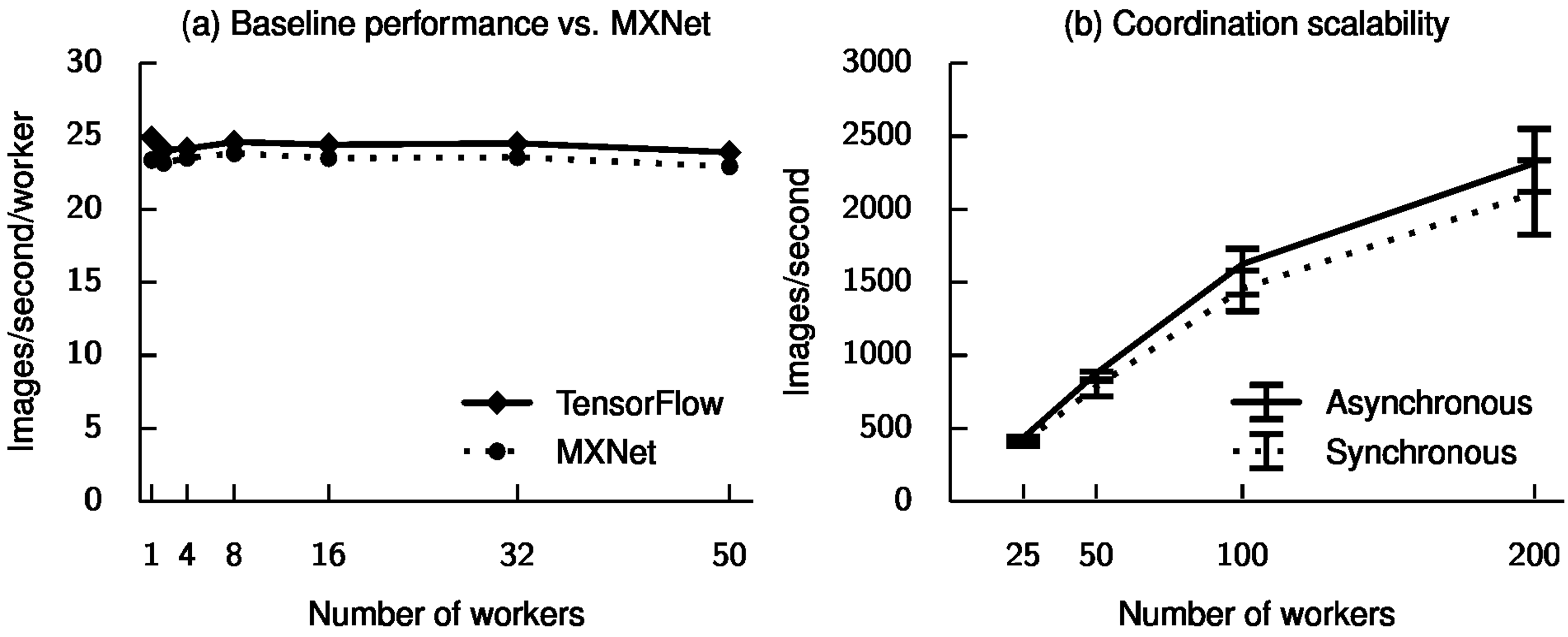
- Single Node

Library	Training step time (ms)			
	AlexNet	Overfeat	OxfordNet	GoogleNet
Caffe [38]	324	823	1068	1935
Neon [58]	87	**211**	**320**	**270**
Torch [17]	**81**	268	529	470
TensorFlow	**81**	279	540	445

Abadi et al., “TensorFlow: A System for Large-Scale Machine Learning”, OSDI 2016

Performance

- Distributed Throughput



Abadi et al., “TensorFlow: A System for Large-Scale Machine Learning”, OSDI 2016

Summary

- Key Contributions
  - Programmability
  - Accessibility / ease of use
  - Richness of Libraries
  - Ready-made community

Abadi et al., “TensorFlow: A System for Large-Scale Machine Learning”, OSDI 2016

MiniTorch Code Explanation

Reading for Next Class

- Attention is all you need. 2017