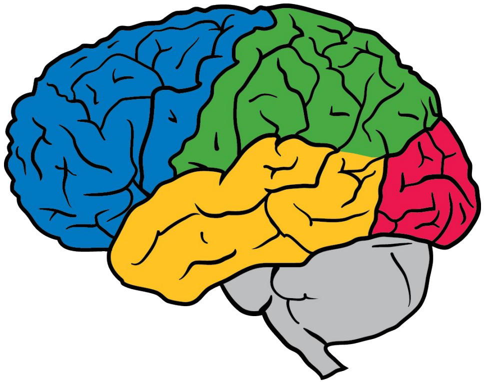


Large Scale Deep Learning with TensorFlow

Jeff Dean
Google Brain Team
g.co/brain

In collaboration with **many** other people at Google

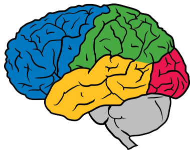
Important Property of Neural Networks

Results get better with

more data +
bigger models +
more computation

Large Datasets + Powerful Models

- Combination works incredibly well
- Poses interesting systems problems, though:
  - Need lots of computation
  - Want to train and do experiments quickly
  - Large-scale parallelism using distributed systems really only way to do this at very large scale
  - Also want to easily express machine learning ideas



Basics of Deep Learning

- Unsupervised cat
- Speech
- Vision
- General trend is towards more complex models:
    - Embeddings of various kinds
    - Generative models
    - Layered LSTMs
    - Attention

What do you want in a machine learning system?

- **Ease of expression**: for lots of crazy ML ideas/algorithms
- **Scalability**: can run experiments quickly
- **Portability**: can run on wide variety of platforms
- **Reproducibility**: easy to share and reproduce research
- **Production readiness**: go from research to real products

TensorFlow

http://tensorflow.org/

and

https://github.com/tensorflow/tensorflow

Open, standard software for general machine learning

Great for Deep Learning in particular

First released Nov 2015

Apache 2.0 license

TensorFlow: A system for large-scale machine learning

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng

Google Brain

Preprint: arxiv.org/abs/1605.08695

Updated version to appear in OSDI 2016

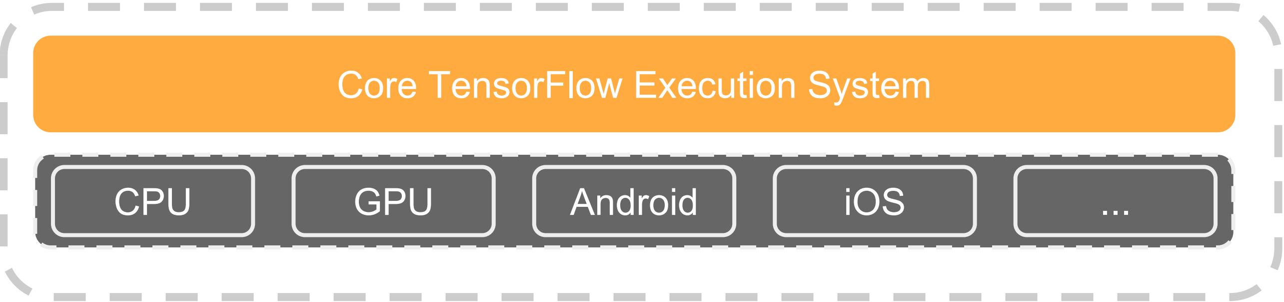
Motivations

TensorFlow

- DistBelief (our 1st system) was the first scalable deep learning system, but not as flexible as we wanted for research purposes
- Better understanding of problem space allowed us to make some dramatic simplifications

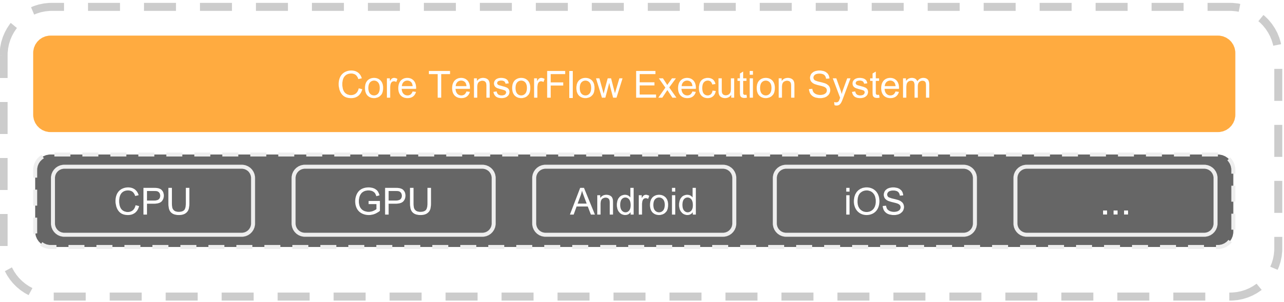
TensorFlow: Expressing High-Level ML Computations

- Core in C++
    - Very low overhead



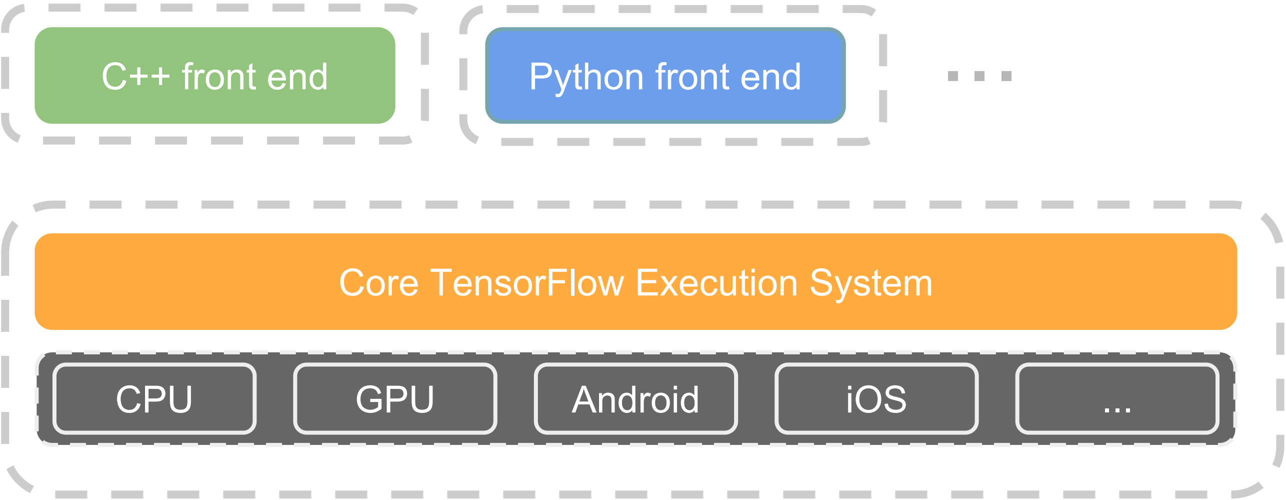
TensorFlow: Expressing High-Level ML Computations

- Core in C++
  - Very low overhead
- Different front ends for specifying/driving the computation
  - Python and C++ today, easy to add more

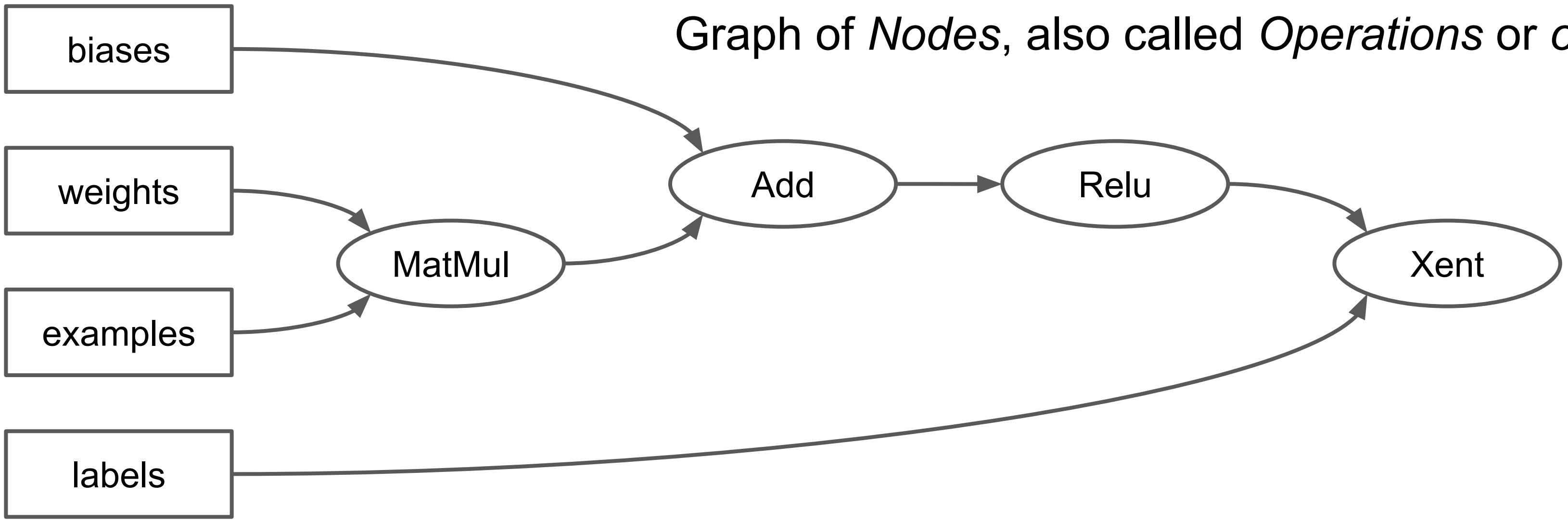


TensorFlow: Expressing High-Level ML Computations

- Core in C++
  - Very low overhead
- Different front ends for specifying/driving the computation
  - Python and C++ today, easy to add more



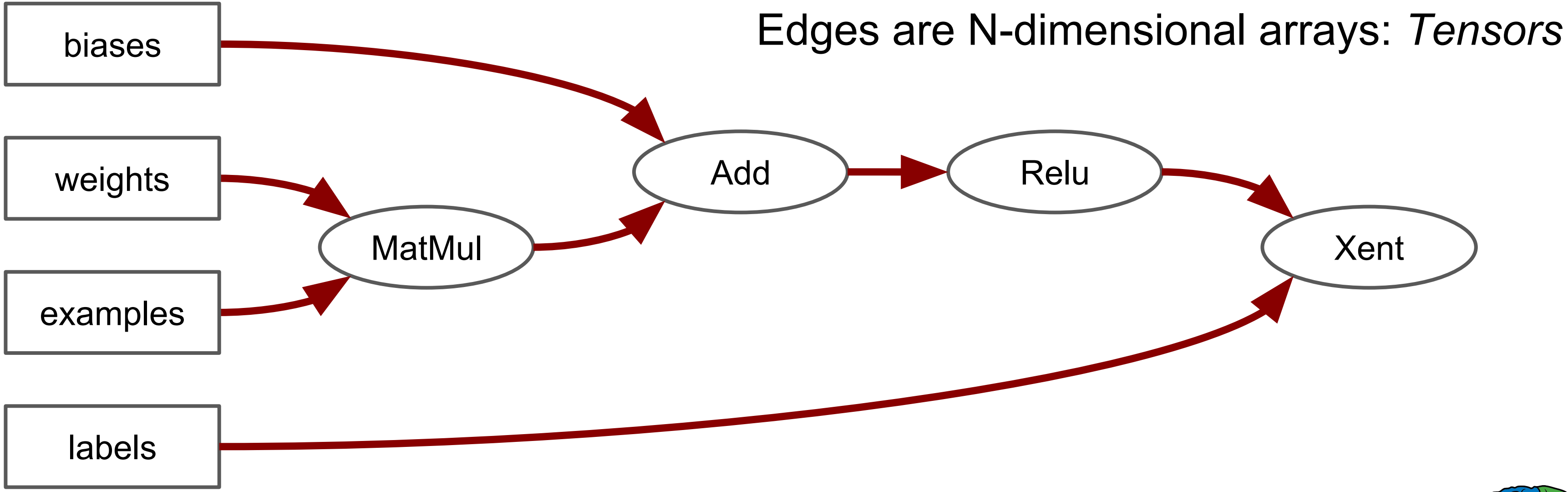
Computation is a dataflow graph



Graph of *Nodes*, also called *Operations* or *ops.*

Computation is a dataflow graph

with tensors



Example TensorFlow fragment

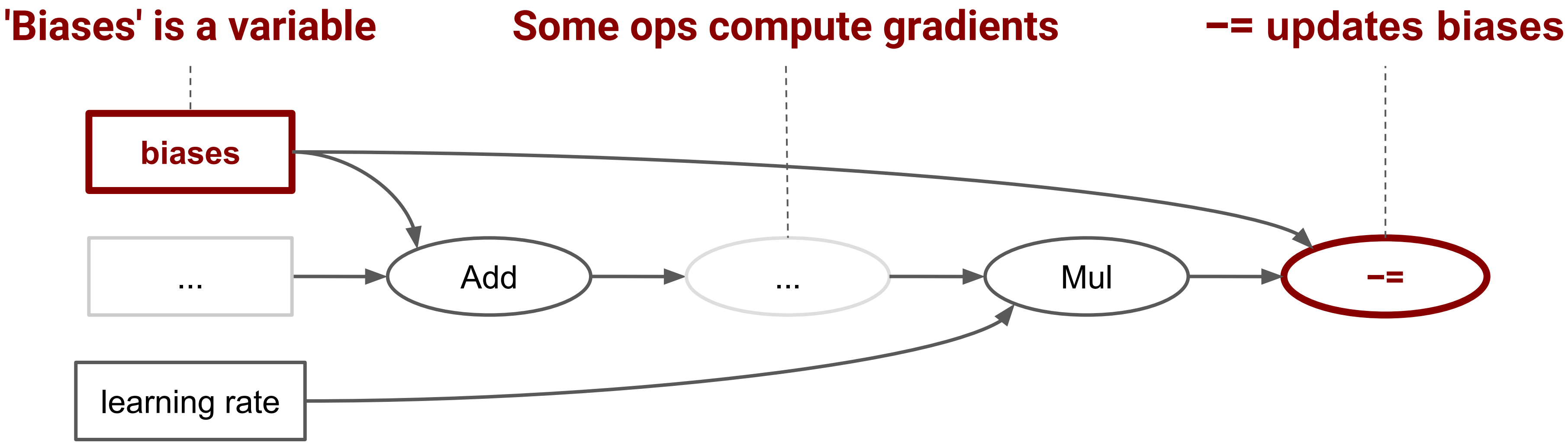
- Build a graph computing a neural net inference.

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets('MNIST_data', one_hot=True)
x = tf.placeholder("float", shape=[None, 784])
W = tf.Variable(tf.zeros([784,10]))
b = tf.Variable(tf.zeros([10]))
y = tf.nn.softmax(tf.matmul(x, W) + b)
```

Computation is a dataflow graph

with state



Symbolic Differentiation

- Automatically add ops to calculate symbolic gradients of variables w.r.t. loss function.
- Apply these gradients with an optimization algorithm

```
y_ = tf.placeholder(tf.float32, [None, 10])
cross_entropy = -tf.reduce_sum(y_ * tf.log(y))
opt = tf.train.GradientDescentOptimizer(0.01)
train_op = opt.minimize(cross_entropy)
```

Define graph and then execute it repeatedly

- Launch the graph and run the training ops in a loop

```
init = tf.initialize_all_variables()
sess = tf.Session()
sess.run(init)
for i in range(1000):
  batch_xs, batch_ys = mnist.train.next_batch(100)
  sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
```

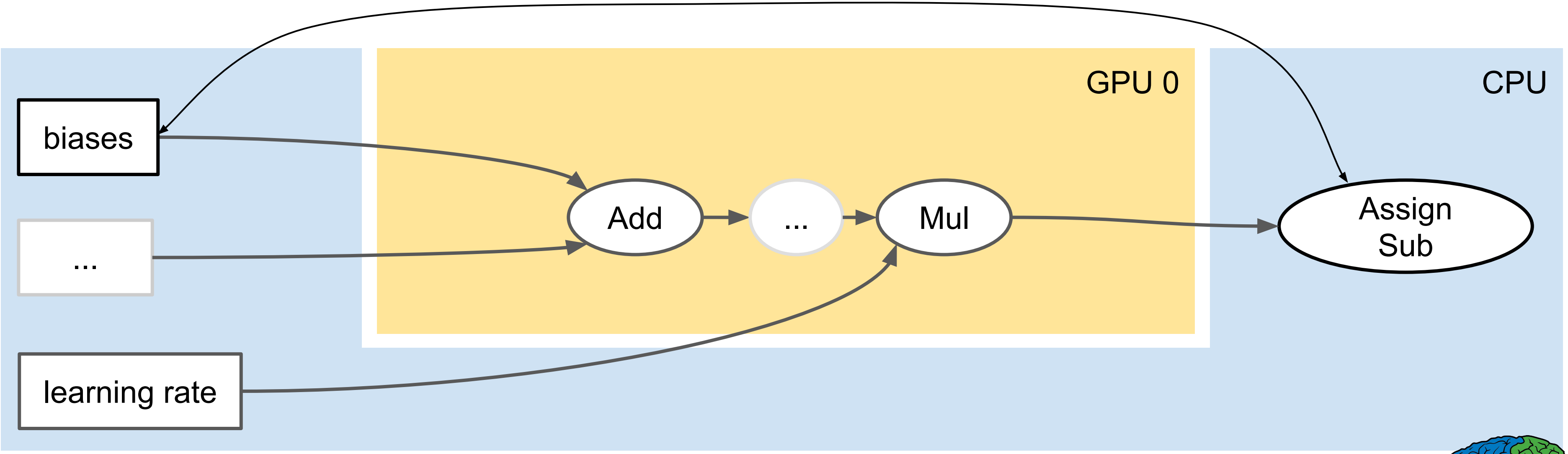
Session Interface

- `Extend`: add nodes to computation graph
- `Run`: execute an arbitrary subgraph
  - optionally feeding in Tensor inputs and retrieving Tensor output

Typically, setup a graph with one or a few `Extend` calls and then `Run` it thousands or millions or times

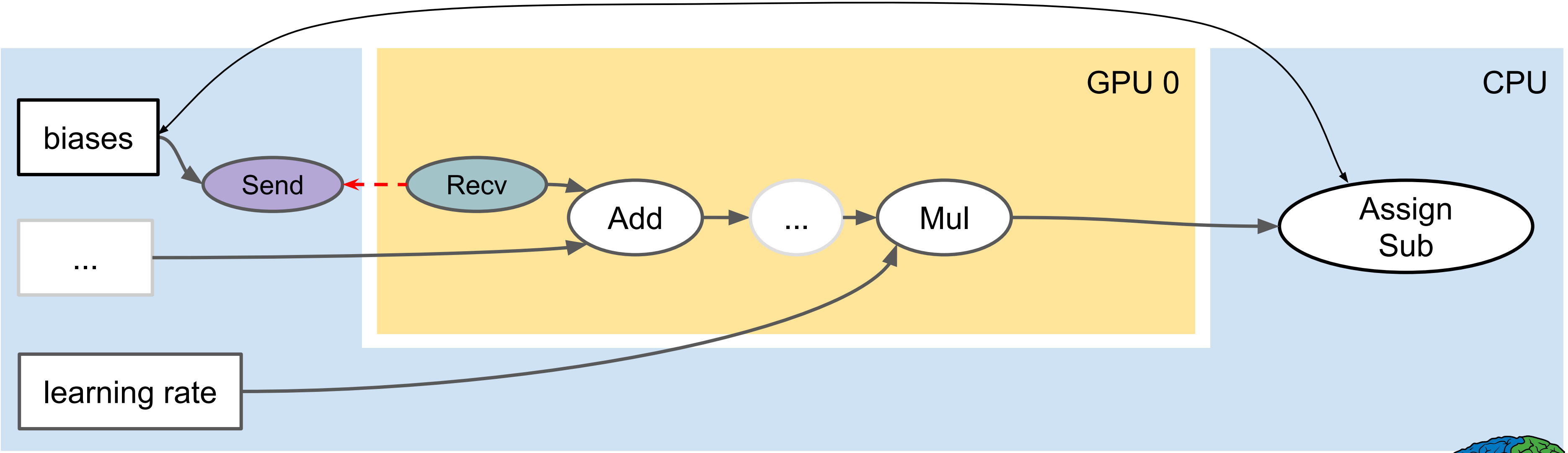
Computation is a dataflow graph

distributed



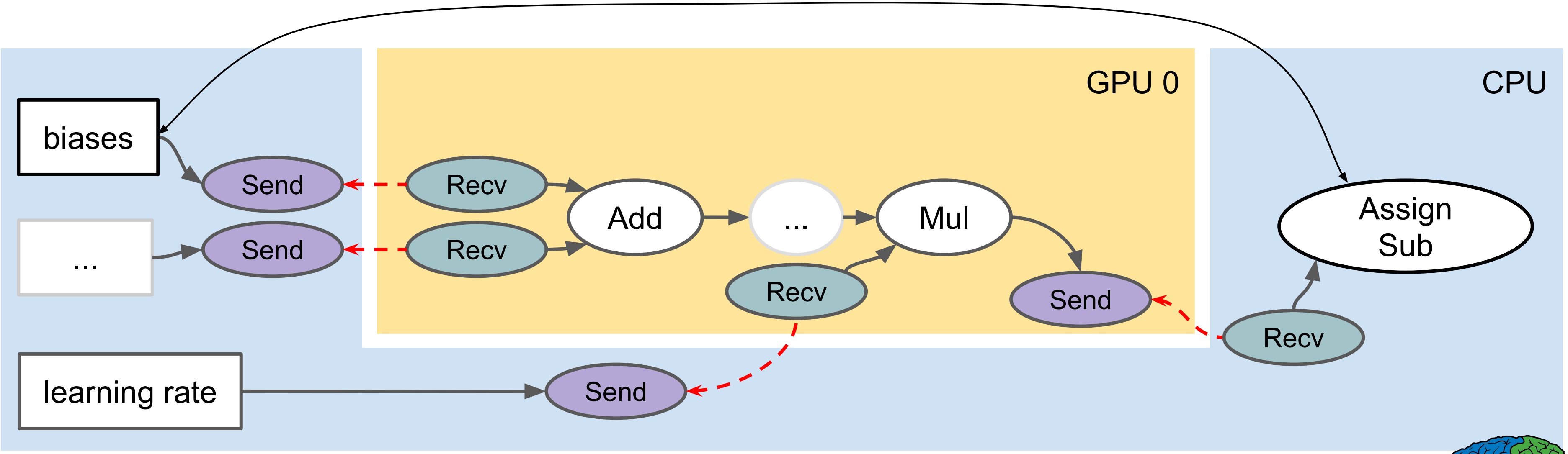
Assign *Devices* to Ops

- TensorFlow inserts *Send/Recv* Ops to transport tensors across devices
- *Recv* ops pull data from *Send* ops



Assign *Devices* to Ops

- TensorFlow inserts *Send/Recv* Ops to transport tensors across devices
- *Recv* ops pull data from *Send* ops

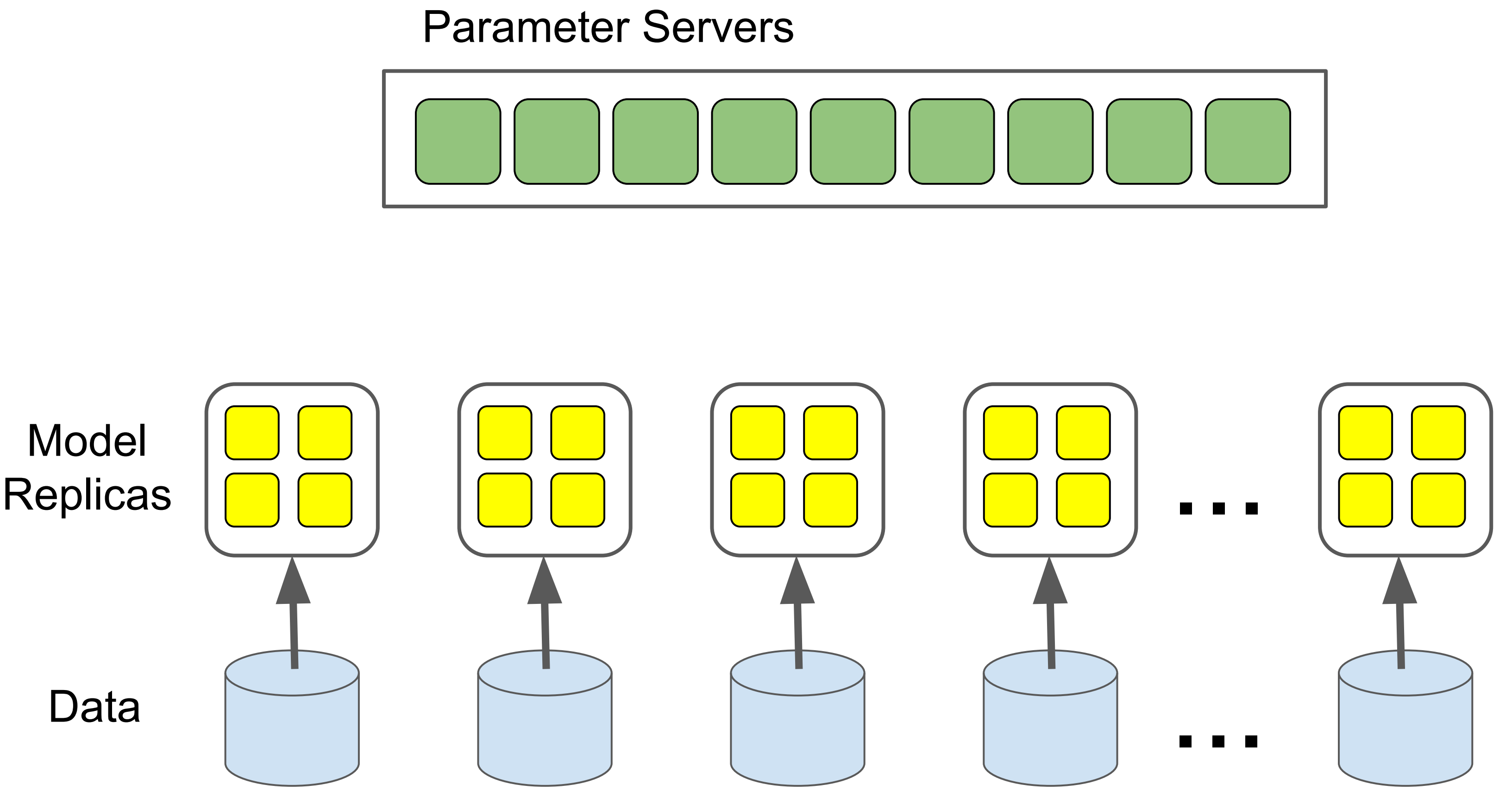


Send and Receive Implementations

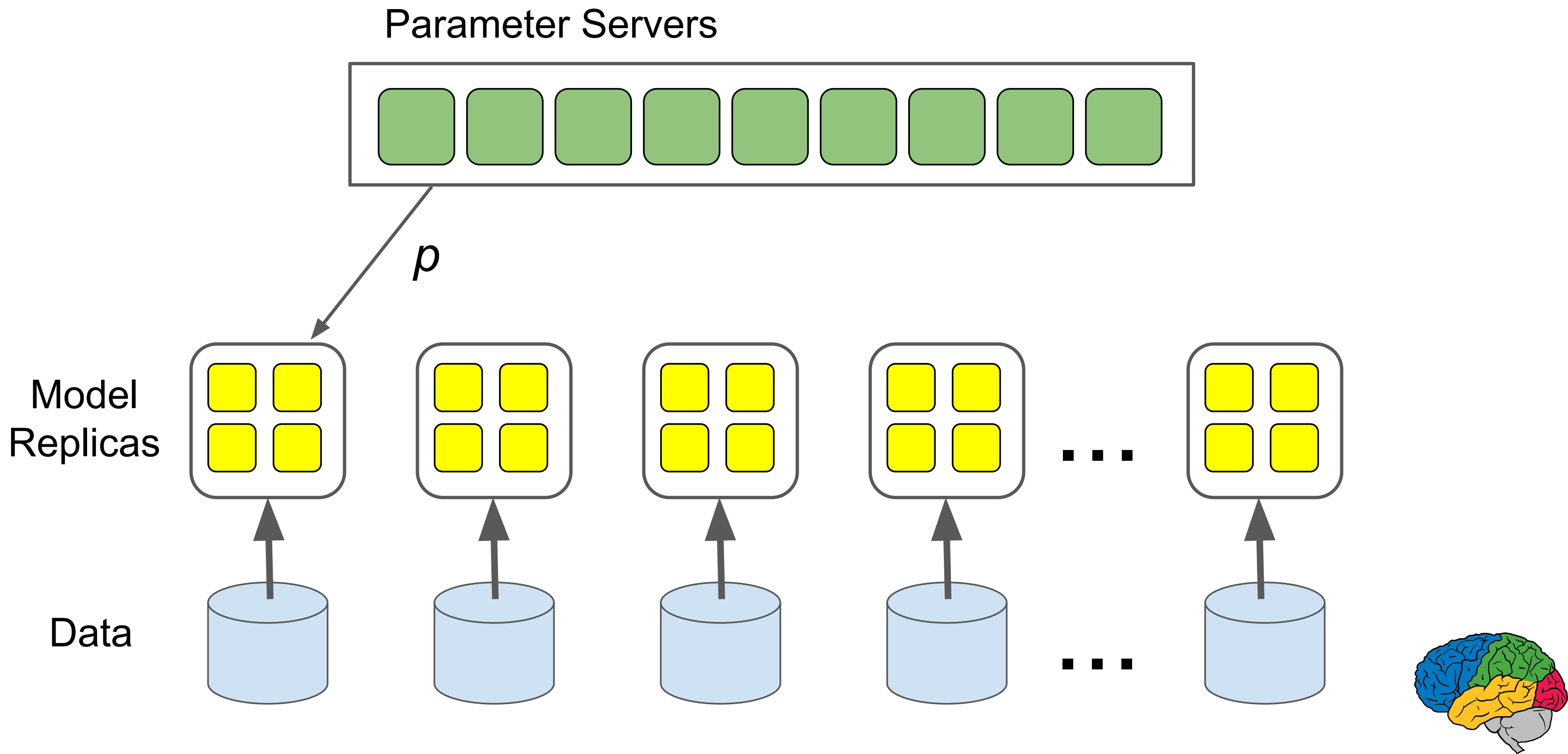
- Different implementations depending on source/dest devices
- e.g. GPUs on same machine: **local GPU → GPU copy**
- e.g. CPUs on different machines: **cross-machine RPC**
- e.g. GPUs on different machines: **RDMA**


Google
google.com/datacenters

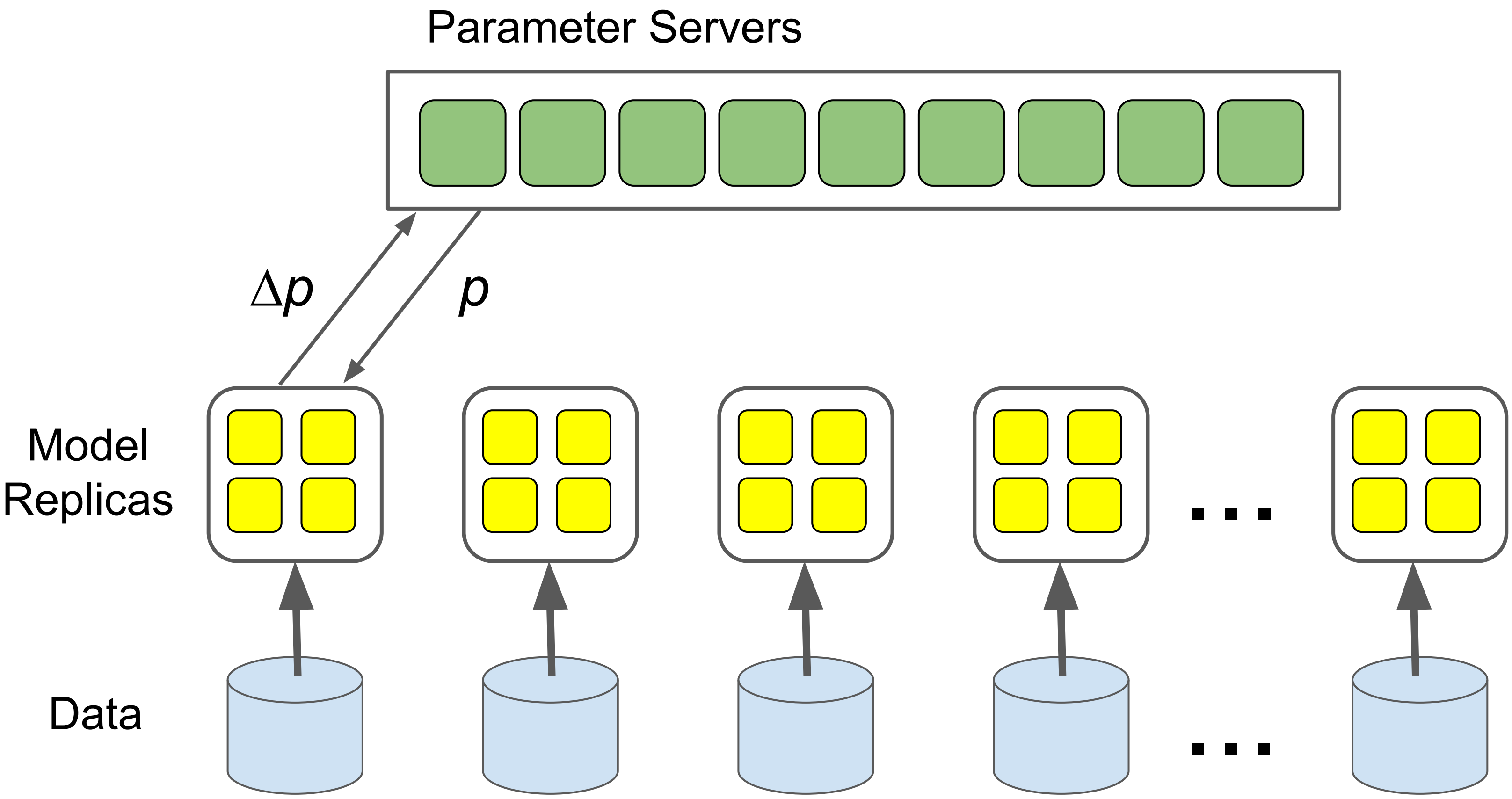
Data Parallelism



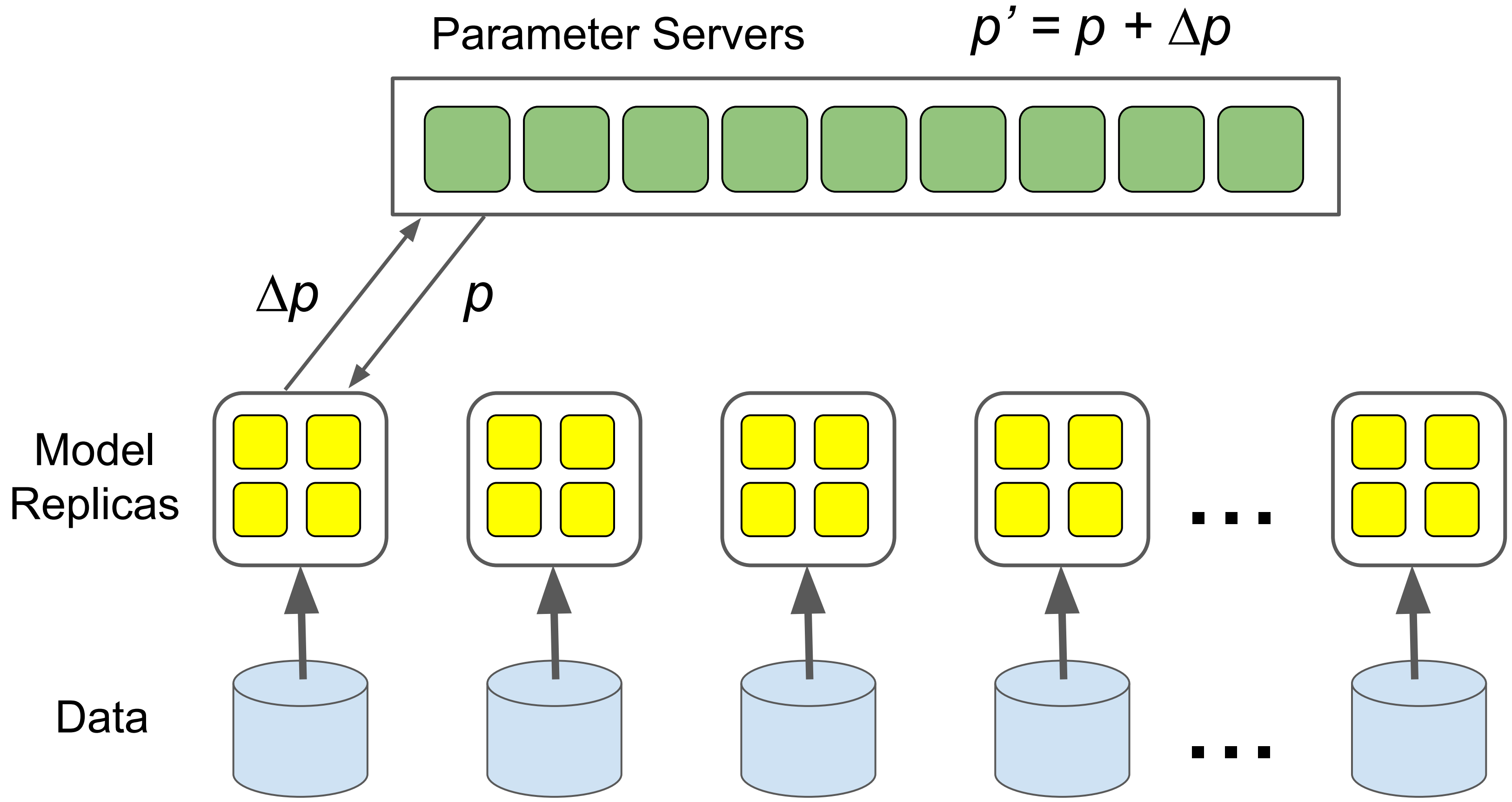
Data Parallelism



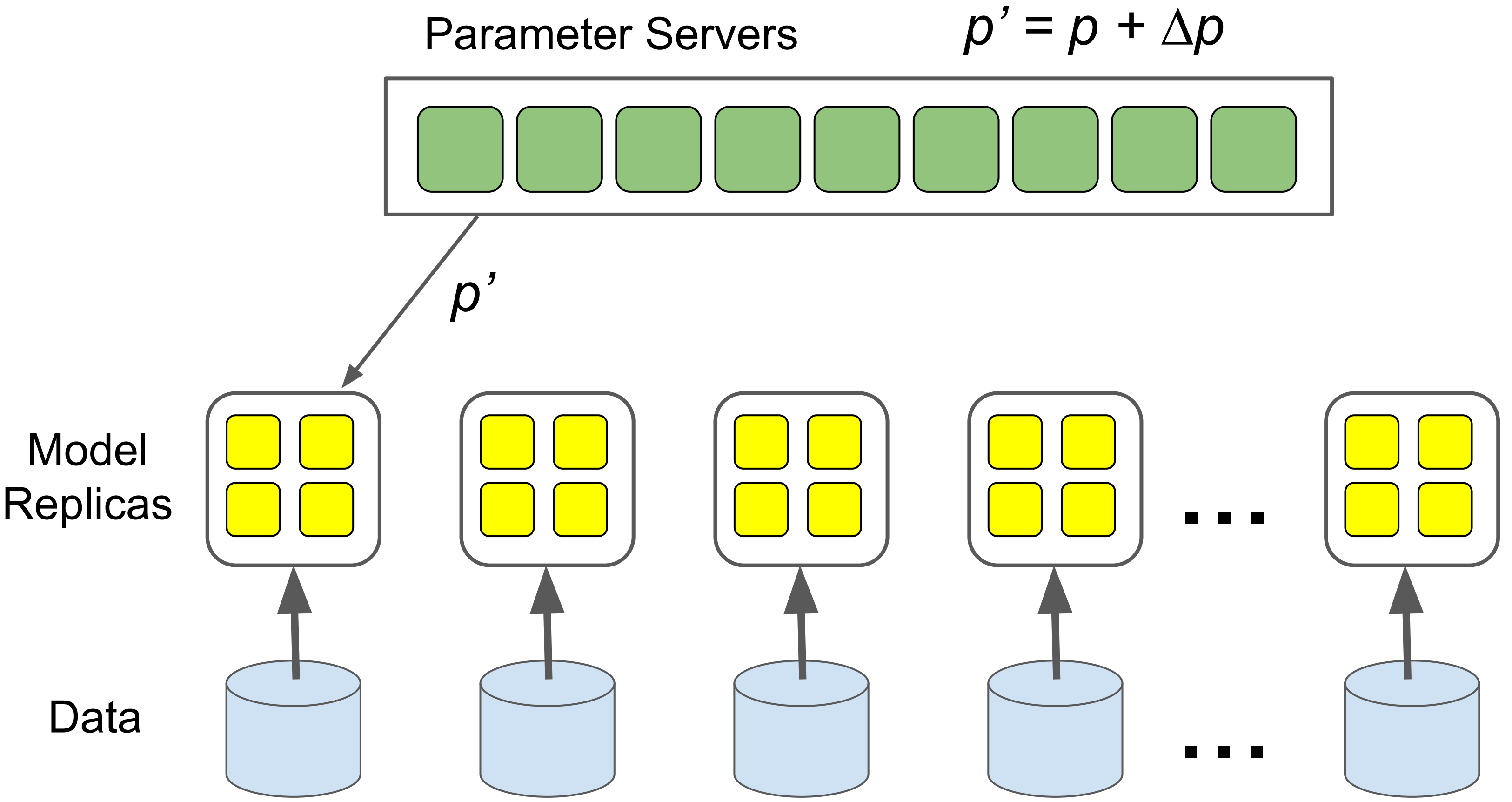
Data Parallelism



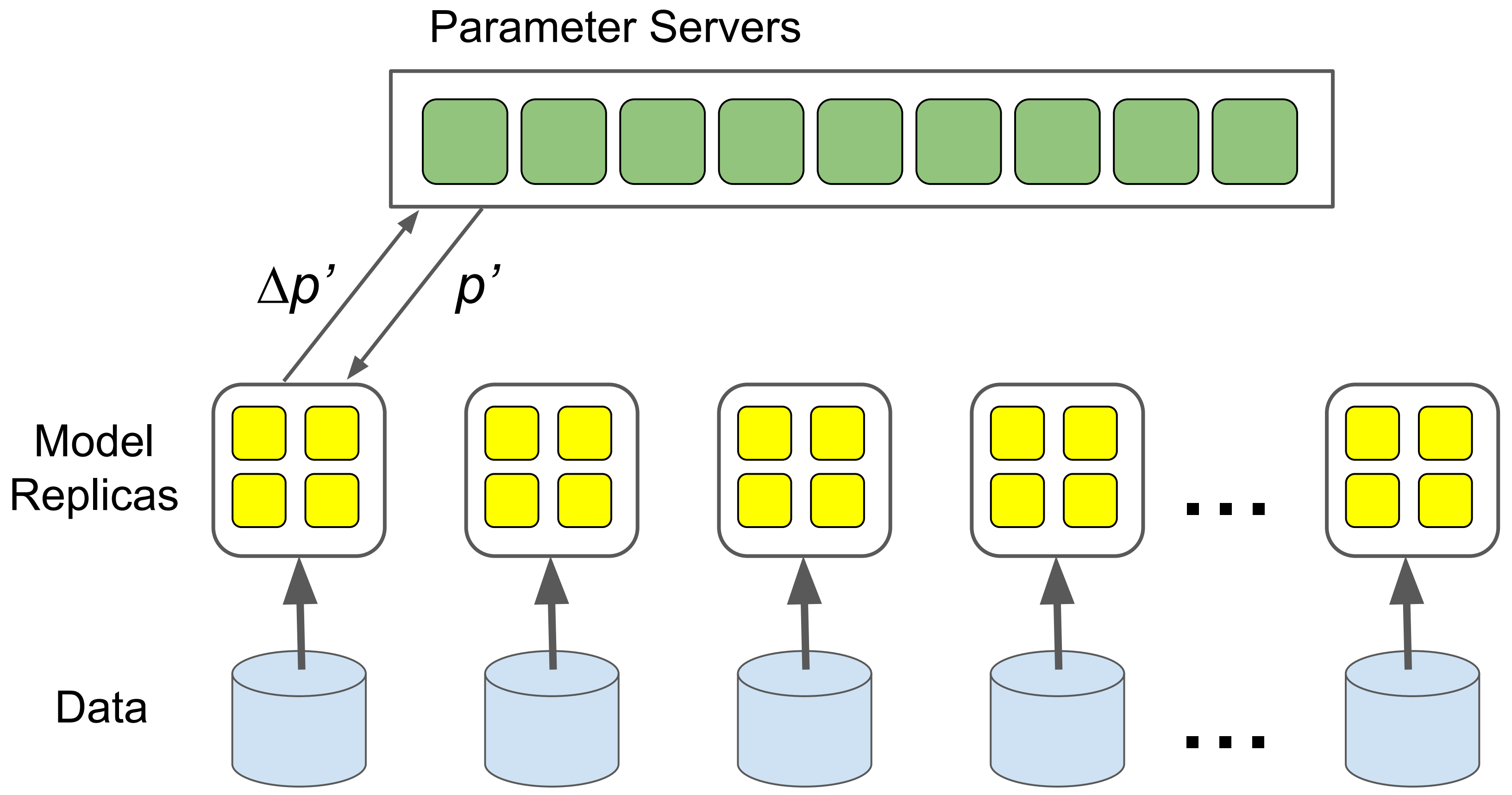
Data Parallelism



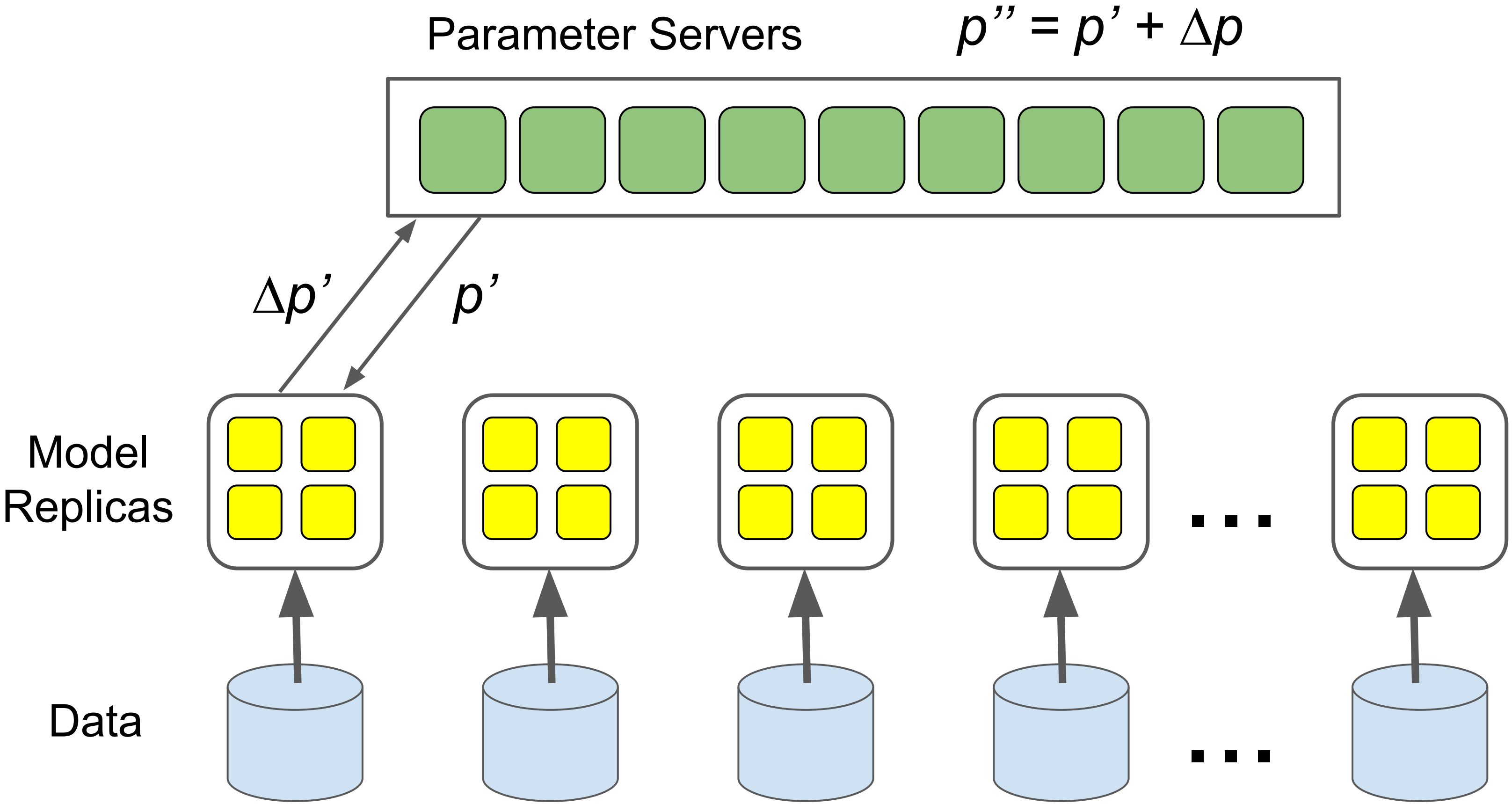
Data Parallelism



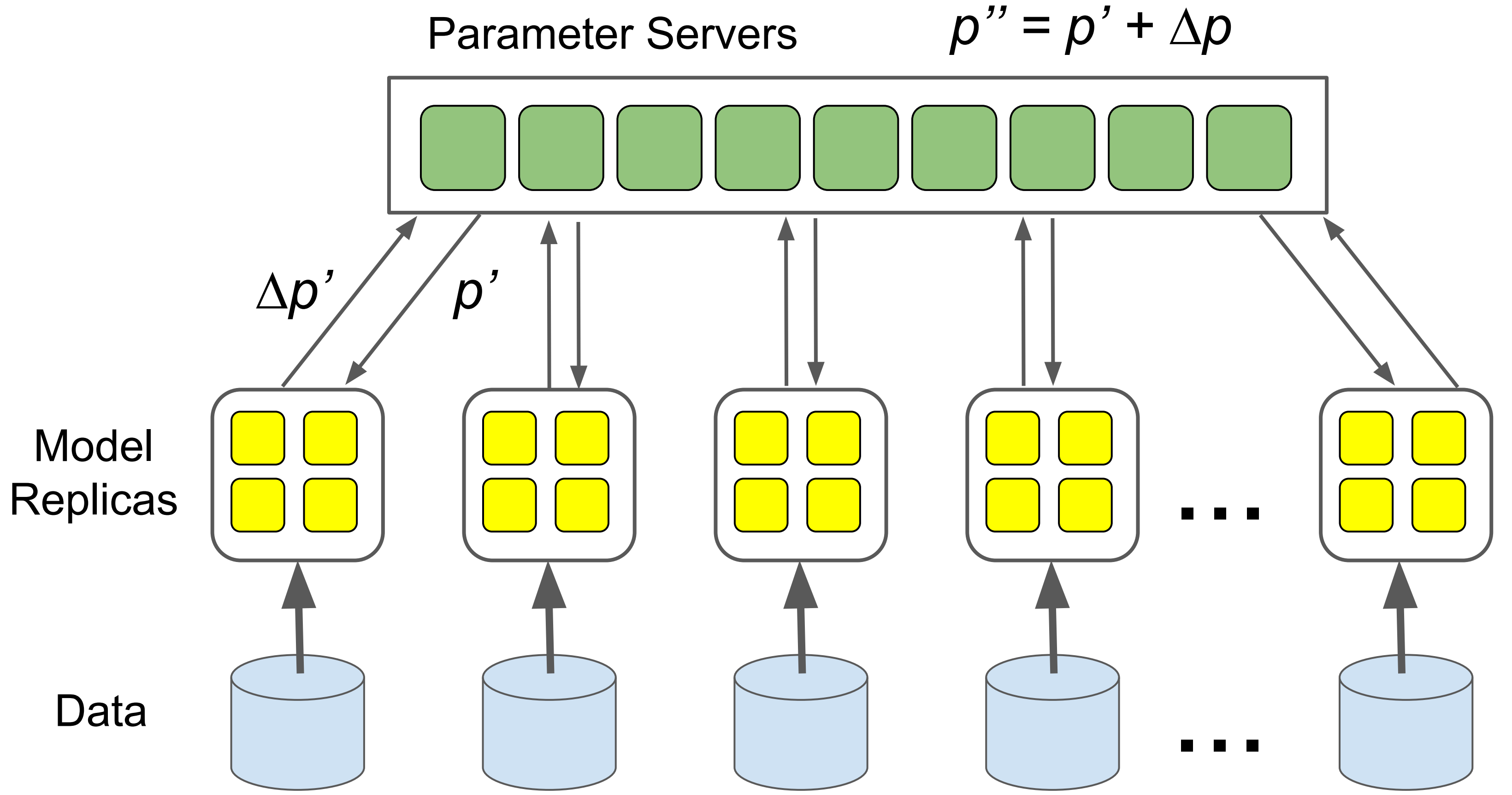
Data Parallelism



Data Parallelism



Data Parallelism



DistBelief: Separate Parameter Servers

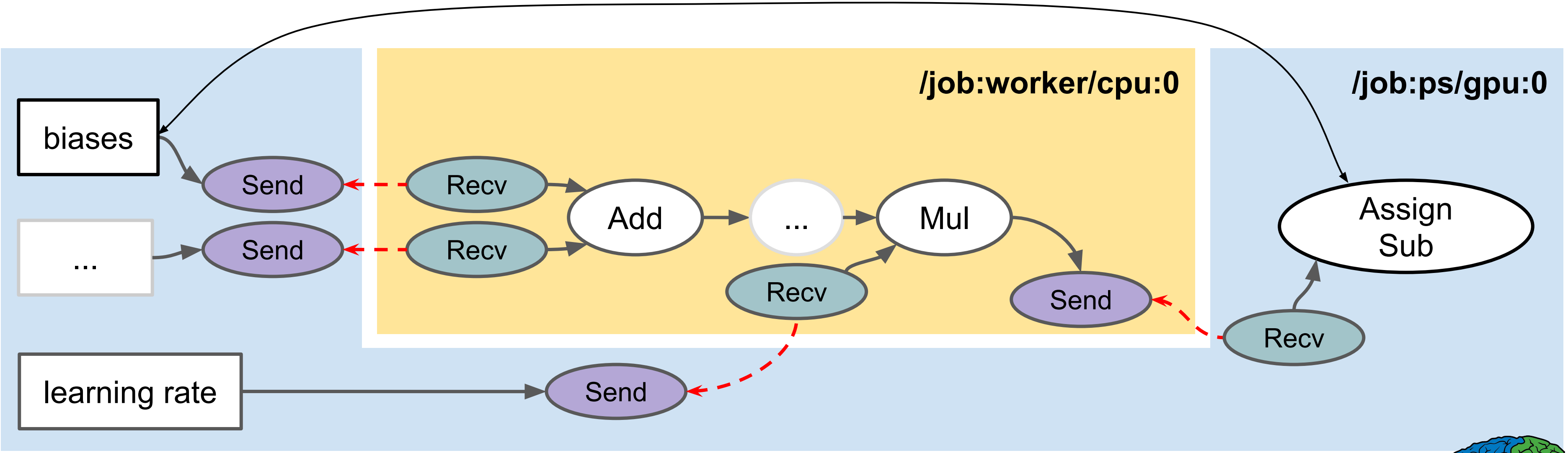
Parameter update rules not the same programming model as the rest of the system

Separate code for parameter servers vs. rest of system

Lacked uniformity & was more complicated

Cross process communication is the same!

- Communication across machines over the network abstracted identically to cross device communication.



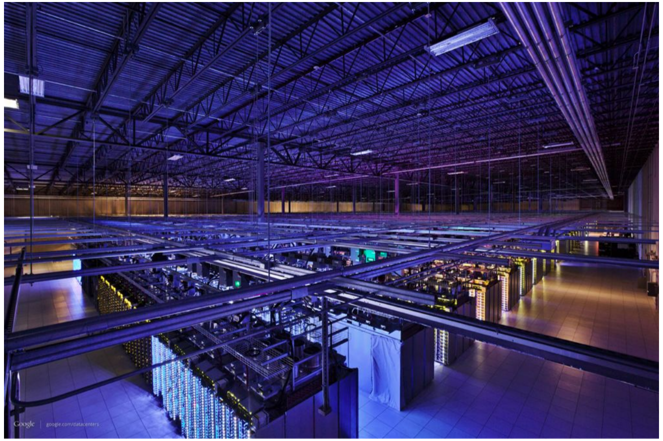
No specialized parameter server subsystem!

Runs on Variety of Platforms

phones

single machines (CPU and/or GPUs) …

distributed systems of 100s of machines and/or GPU cards



custom ML hardware

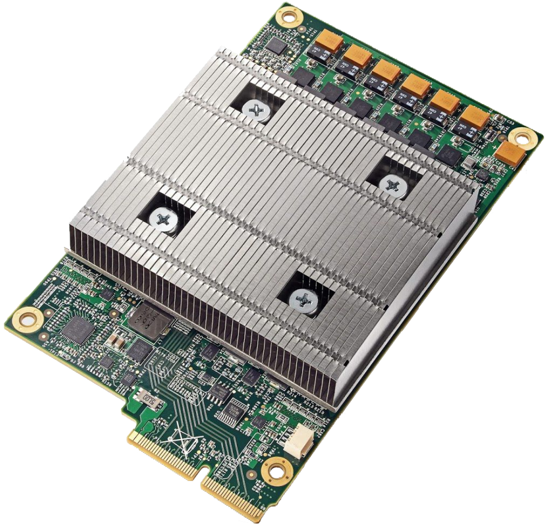
Trend: Much More Heterogeneous hardware

General purpose CPU performance scaling has slowed significantly

Specialization of hardware for certain workloads will be more important

Tensor Processing Unit

Custom machine learning ASIC



In production use for >16 months: used on every search query, used for AlphaGo match, many other uses, ...

See Google Cloud Platform blog: [Google supercharges machine learning tasks with TPU custom chip](), by Norm Jouppi, May, 2016

Extensible

- Core system defines a number of standard ***operations*** and ***kernels*** (device-specific implementations of operations)
- Easy to define new operators and/or kernels

Single device performance important, but

....

biggest performance improvements come from large-scale distributed systems with model and data parallelism