Distributed TensorFlow

Stony Brook University

CSE545, Fall 2017

Goals

- Understand TensorFlow as a workflow system.
- Know the key components of TensorFlow.
- Understand the key concepts of *distributed* TensorFlow.
- Do basic analysis in distributed TensorFlow.

Will not know but will be easier to pick up

- How deep learning works
- What is a CNN
- What is an RNN (or LSTM, GRU)

TensorFlow

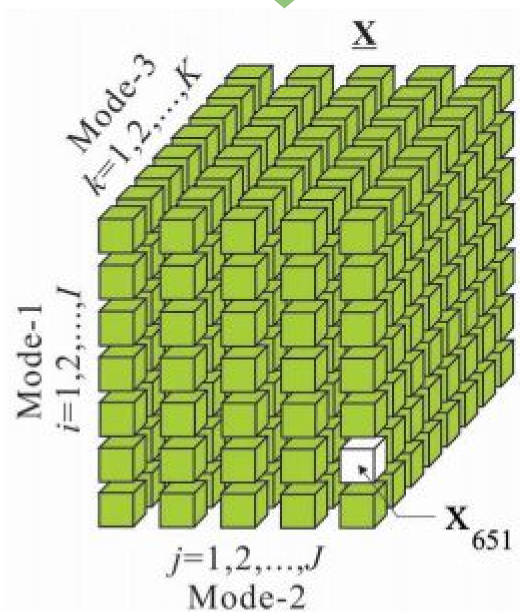
A workflow system catered to numerical computation.

Like Spark, but uses *tensors* instead of *RDDs*.

TensorFlow

A workflow system catered to numerical computation.

Like Spark, but uses *tensors* instead of *RDDs*.



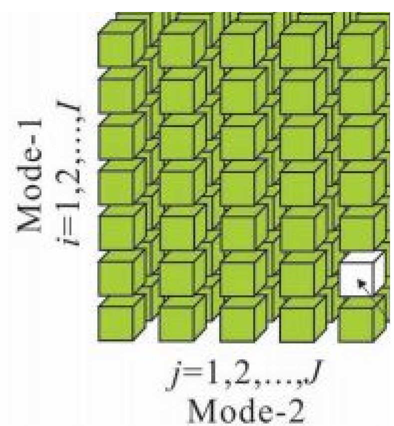
A multi-dimensional matrix

(i.stack.imgur.com)

TensorFlow

A workflow system catered to numerical computation.

Like Spark, but uses *tensors* instead of *RDDs*.



(i.stack.imgur.com)

A 2-d tensor is just a matrix.

1-d: vector

0-d: a constant / scalar

Note: Linguistic ambiguity:
Dimensions of a Tensor =/=
Dimensions of a Matrix

TensorFlow

A workflow system catered to numerical computation.

Like Spark, but uses *tensors* instead of *RDDs*.

Example: Image definitions from assignment 2:

image[*row*][*column*][*rgbx*]

TensorFlow

A workflow system catered to numerical computation.

Like Spark, but uses *tensors* instead of *RDDs*.

Technically, less abstract than *RDDs* which could hold tensors as well as many other data structures (dictionaries/HashMaps, Trees, ...etc…).

Then, what is valuable about TensorFlow?

TensorFlow

Efficient, high-level built-in **linear algebra** and **machine learning** *operations* (i.e. transformations).

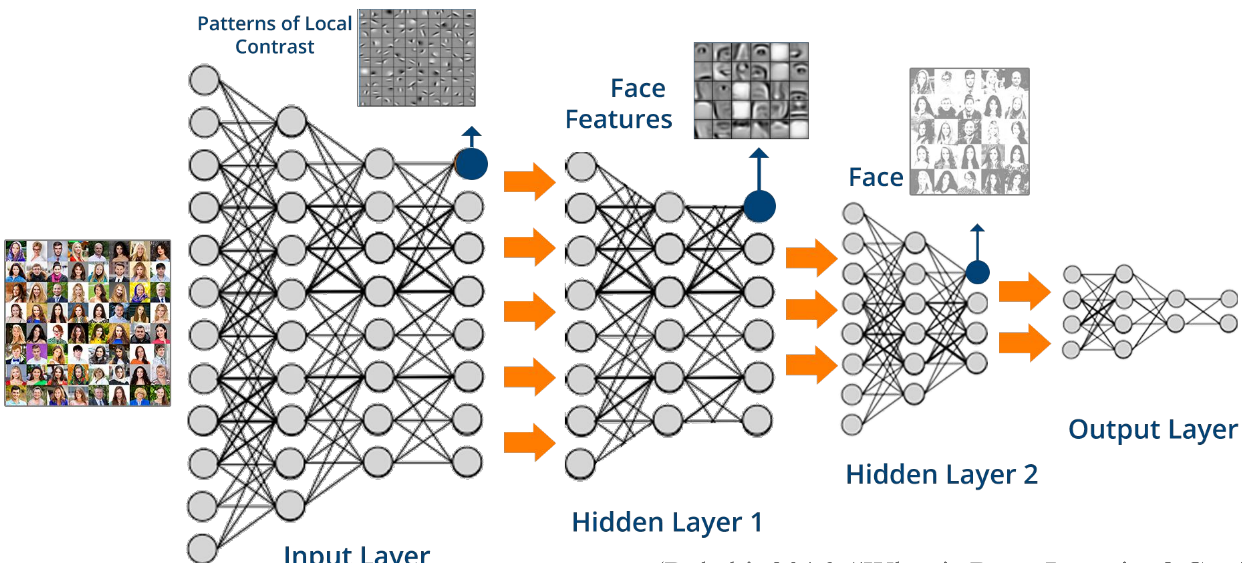
enables complex models, like deep learning

Then, what is valuable about TensorFlow?

TensorFlow

Efficient, high-level built-in **linear algebra** and **machine learning** *operations*.

enables complex models, like deep learning



(Bakshi, 2016, “What is Deep Learning? Getting Started With Deep Learning”)

TensorFlow

Efficient, high-level built-in **linear algebra** and **machine learning** *operations*.

```
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))                   # 100-d vector, init to zeroes
W = tf.Variable(tf.random_uniform([784,100],-1,1)) # 784x100 matrix w/rnd vals
x = tf.placeholder(name="x")                       # Placeholder for input
relu = tf.nn.relu(tf.matmul(W, x) + b)             # Relu(Wx+b)
C = [...]                                          # Cost computed as a function
                                                   # of Relu

s = tf.Session()
for step in xrange(0, 10):
  input = ...construct 100-D input array ...       # Create 100-d vector for input
  result = s.run(C, feed_dict={x: input})          # Fetch cost, feeding x=input
  print step, result
```

(Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Ghemawat, S. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.)

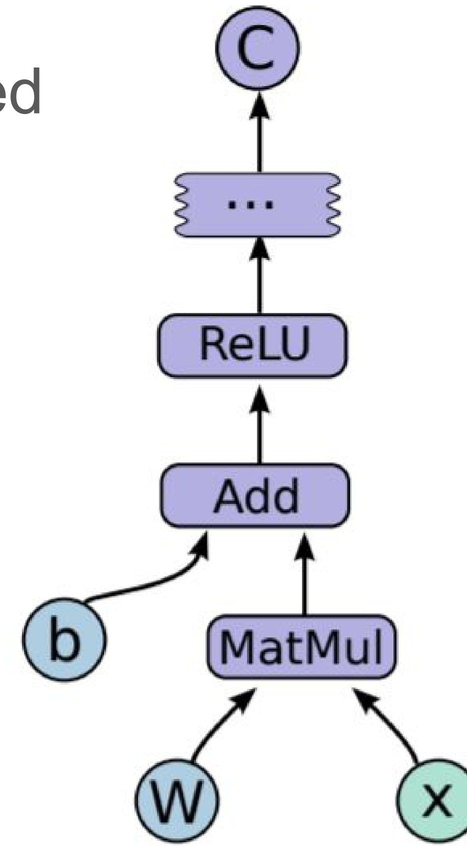
TensorFlow

Operations on tensors are often conceptualized as graphs:

```
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784,100],-1,1))
x = tf.placeholder(name="x")
relu = tf.nn.relu(tf.matmul(W, x) + b)
C = [...]

s = tf.Session()
for step in xrange(0, 10):
  input = ...construct 100-D input array ...
  result = s.run(C, feed_dict={x: input})
  print step, result
```



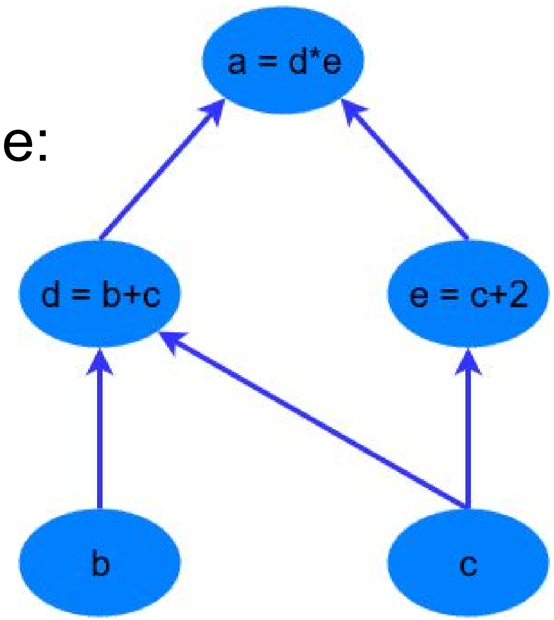
(Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Ghemawat, S. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.)

Tensor**Flow**

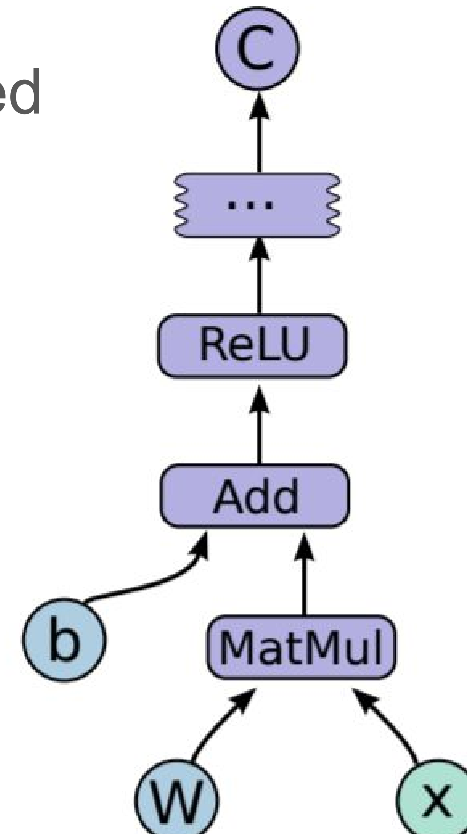
Operations on tensors are often conceptualized as graphs:

A simpler example:

d=b+c
e=c+2
a=d∗e



(Adventures in Machine Learning. *Python TensorFlow Tutorial*, 2017)



Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Ghemawat, S. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Ingredients of a TensorFlow

* technically, *operations* that work with tensors.

tensors*
variables - persistent mutable tensors
constants - constant
placeholders - from data

operations
an abstract computation (e.g. matrix multiply, add) executed by device *kernels*

graph

session
defines the environment in which operations *run*. (like a Spark context)

devices
the specific devices (cpus or gpus) on which to run the session.

* technically, *operations* that work with tensors.

Ingredients of a TensorFlow

tensors*

variables - persistent mutable tensors

constants - constant

placeholders - from data

- tf.Variable(initial_value, name)
- tf.constant(value, type, name)
- tf.placeholder(type, shape, name)

operations

an abstract computation (e.g. matrix multiply, add) executed by device *kernels*

graph

session

defines the environment in which operations *run*. (like a Spark context)

devices

the specific devices (cpus or gpus) on which to run the session.

Operations

tensors*
variables - persistent
mutable tensors
constants - constant
placeholders - from data

operations
an abstract computation
(e.g. matrix multiply, add)
executed by device *kernels*

Category	**Examples**
Element-wise mathematical operations	Add, Sub, Mul, Div, Exp, Log, Greater, Less, Equal, ...
Array operations	Concat, Slice, Split, Constant, Rank, Shape, Shuffle, ...
Matrix operations	MatMul, MatrixInverse, MatrixDeterminant, ...
Stateful operations	Variable, Assign, AssignAdd, ...
Neural-net building blocks	SoftMax, Sigmoid, ReLU, Convolution2D, MaxPool, ...
Checkpointing operations	Save, Restore
Queue and synchronization operations	Enqueue, Dequeue, MutexAcquire, MutexRelease, ...
Control flow operations	Merge, Switch, Enter, Leave, NextIteration

Sessions

- Places operations on devices
- Stores the values of variables (when not distributed)
- Carries out execution: eval() or run()

tensors*

variables - persistent mutable tensors

constants - constant

placeholders - from data

operations

an abstract computation (e.g. matrix multiply, add) executed by device *kernels*

graph

session

defines the environment in which operations *run*. (like a Spark context)

devices

the specific devices (cpus or gpus) on which to run the session.

Ingredients of a TensorFlow

* technically, *operations* that work with tensors.

tensors*

variables - persistent mutable tensors

constants - constant

placeholders - from data

operations

an abstract computation (e.g. matrix multiply, add) executed by device *kernels*

graph

session

defines the environment in which operations *run*. (like a Spark context)

devices

the specific devices (cpus or gpus) on which to run the session.

Demo

Ridge Regression (L2 Penalized linear regression, $\lambda||\beta||_2^2$)

$$\hat{\beta}^{ridge} = argmin_{\beta}\{\sum_{i=1}^{N}(y_i - \sum_{j=1}^{m} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{m} \beta_j^2\}$$

Matrix Solution:

$$\hat{\beta}^{ridge} = (X^T X + \lambda I)^{-1} X^T y$$

Demo

Ridge Regression (L2 Penalized linear regression, $\lambda||\beta||_2^2$)

$$\hat{\beta}^{ridge} = argmin_{\beta}\{\sum_{i=1}^{N}(y_i - \sum_{j=1}^{m} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{m} \beta_j^2\}$$

Gradient descent needs to solve.
(Mirrors many parameter optimization problems.)

Matrix Solution:

$$\hat{\beta}^{ridge} = (X^T X + \lambda I)^{-1} X^T y$$

Gradients

Ridge Regression (L2 Penalized linear regression, $\lambda||\beta||_2^2$)

$$\hat{\beta}^{ridge} = argmin_\beta \{ \boxed{\sum_{i=1}^{N}(y_i - \sum_{j=1}^{m} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{m} \beta_j^2 } \}$$

Gradient descent needs to solve.
(Mirrors many parameter optimization problems.)

TensorFlow has built-in ability to derive gradients given a $\boxed{\text{cost function}}$.

Gradients

Ridge Regression (L2 Penalized linear regression, $\lambda||\beta||_2^2$)

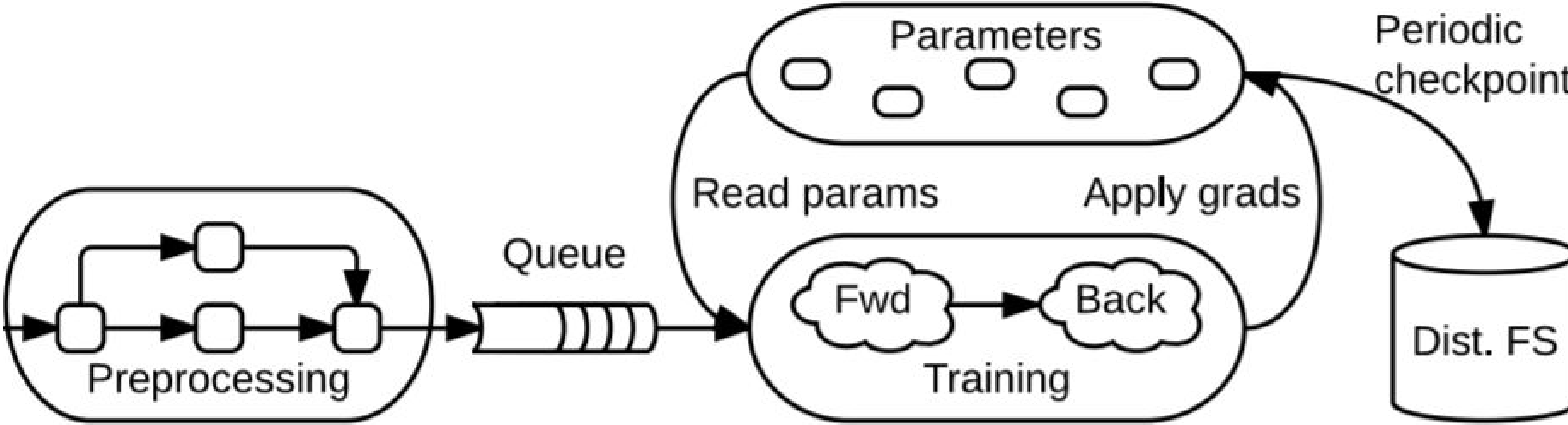
$$\hat{\beta}^{ridge} = argmin_{\beta}\{\boxed{\sum_{i=1}^{N}(y_i - \sum_{j=1}^{m} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{m}\beta_j^2}\}$$

Gradient descent needs to solve.
(Mirrors many parameter optimization problems.)

TensorFlow has built-in ability to derive gradients given a cost function.

```
tf.gradients(cost, [params])
```

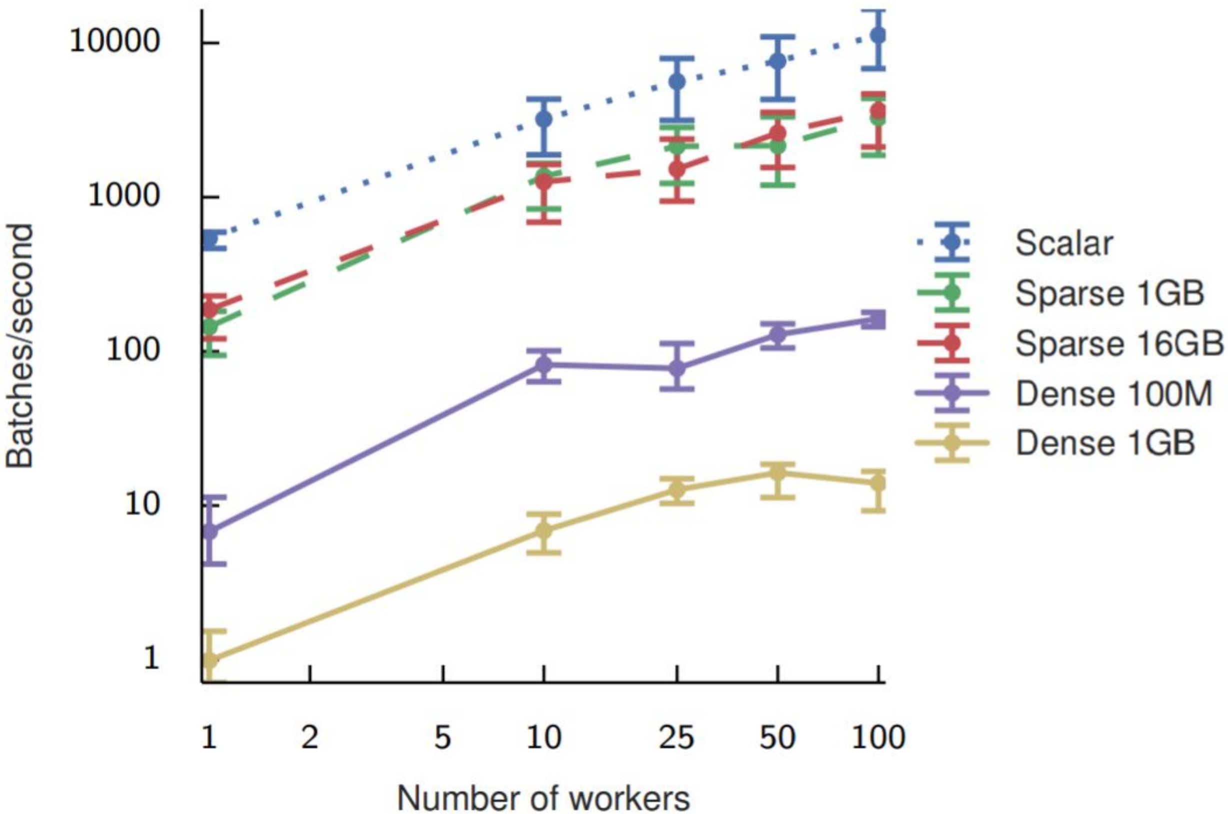
Gradients



TensorFlow has built-in ability to derive gradients given a cost function.

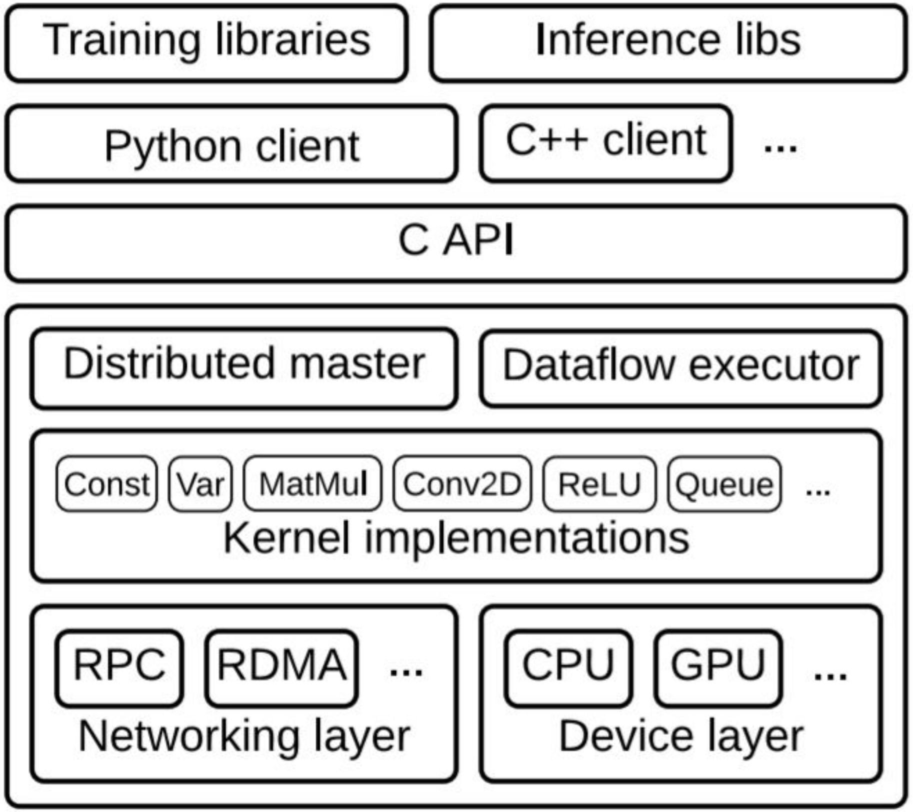
```
tf.gradients(cost, [params])
```

Distributed TensorFlow



Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016, November). TensorFlow: A System for Large-Scale Machine Learning. In *OSDI* (Vol. 16, pp. 265-283).

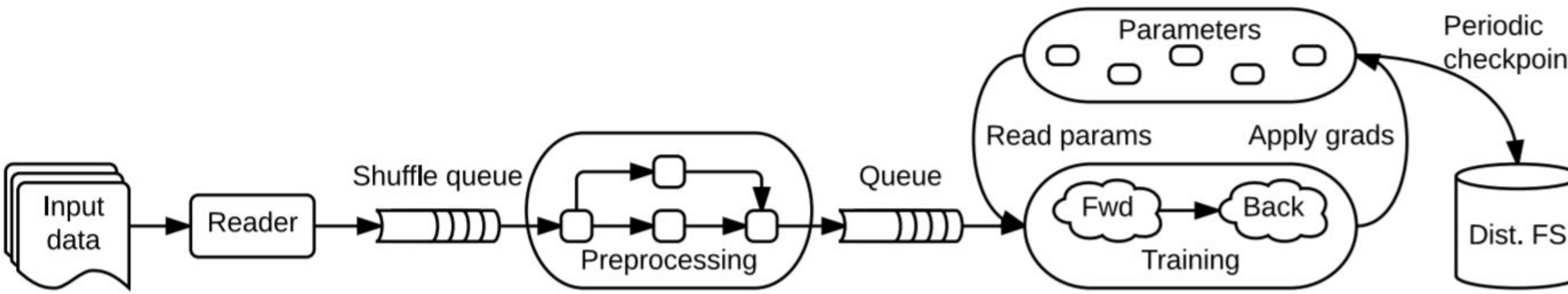
Distributed TensorFlow



The layered TensorFlow architecture.

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016, November). TensorFlow: A System for Large-Scale Machine Learning. In *OSDI* (Vol. 16, pp. 265-283).

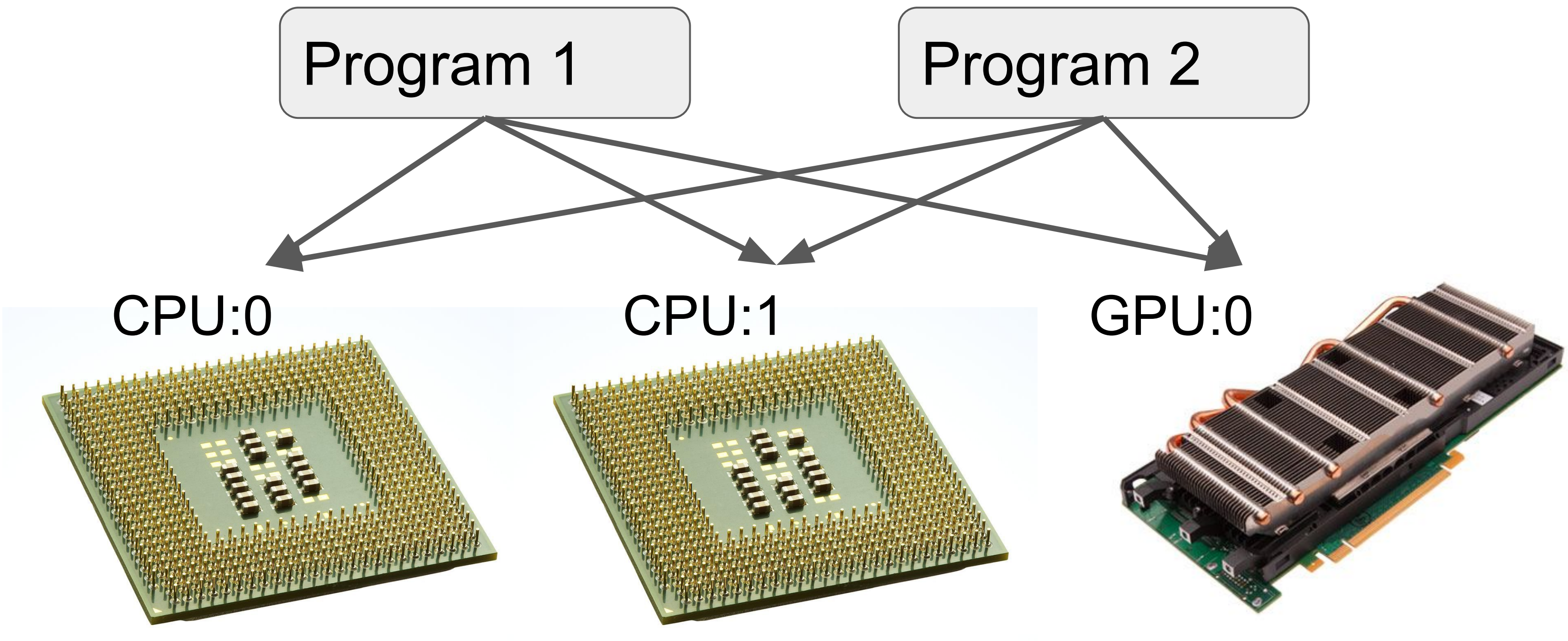
Distributed TensorFlow: Full Pipeline



Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016, November). TensorFlow: A System for Large-Scale Machine Learning. In *OSDI* (Vol. 16, pp. 265-283).

Local Distribution

Multiple devices on single machine



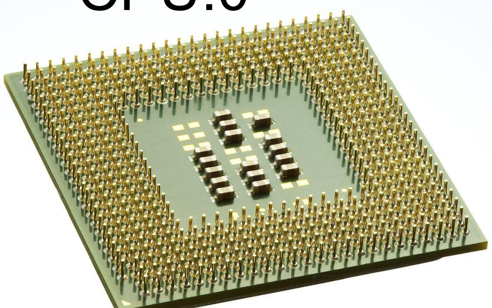
Local Distribution

Multiple devices on single machine

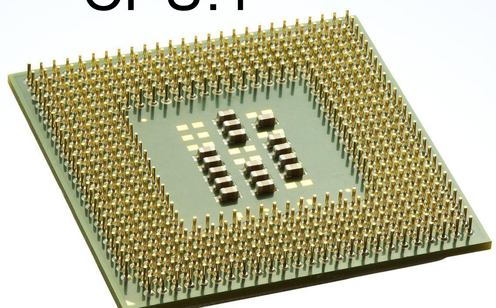
```
with tf.device("/cpu:1")
   beta=tf.Variable(...)
```

```
with tf.device("/gpu:0")
   y_pred=tf.matmul(beta,X)
```

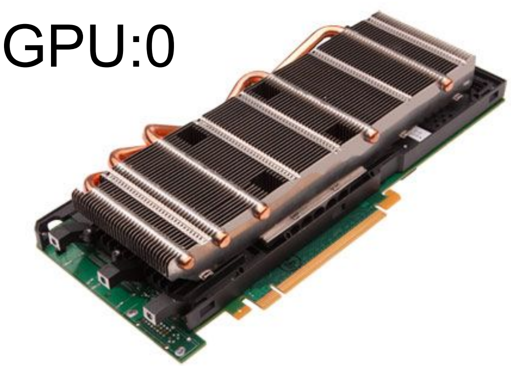
CPU:0



CPU:1



GPU:0

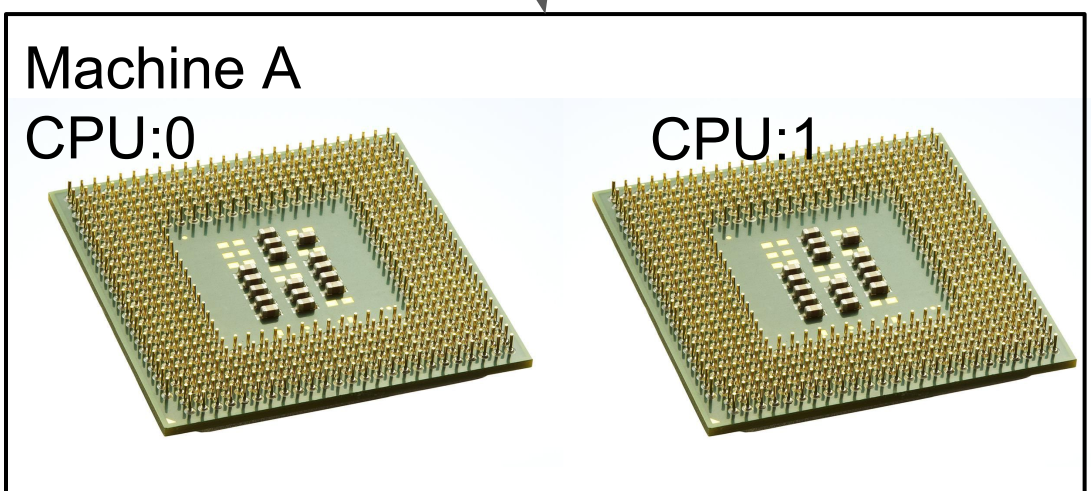


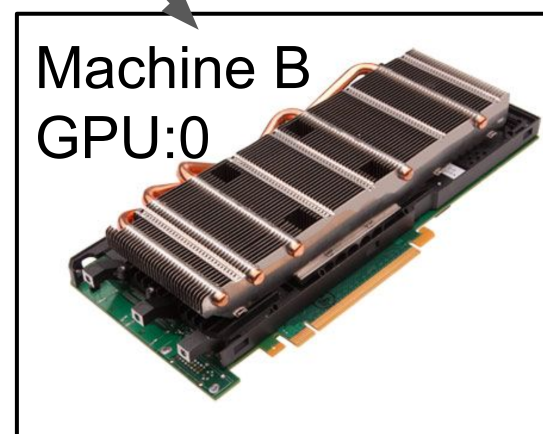
Cluster Distribution

Multiple devices on multiple machines

```
with tf.device("/cpu:1")
   beta=tf.Variable(...)
```

```
with tf.device("/gpu:0")
   y_pred=tf.matmul(beta,X)
```



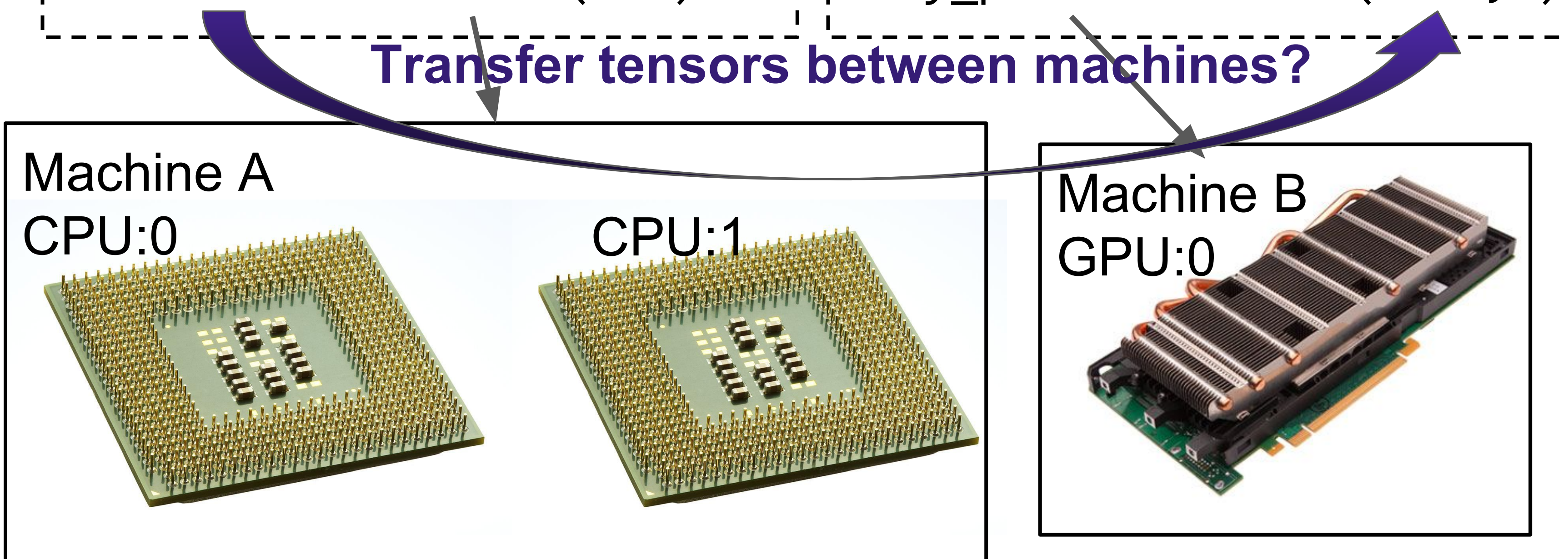


Cluster Distribution

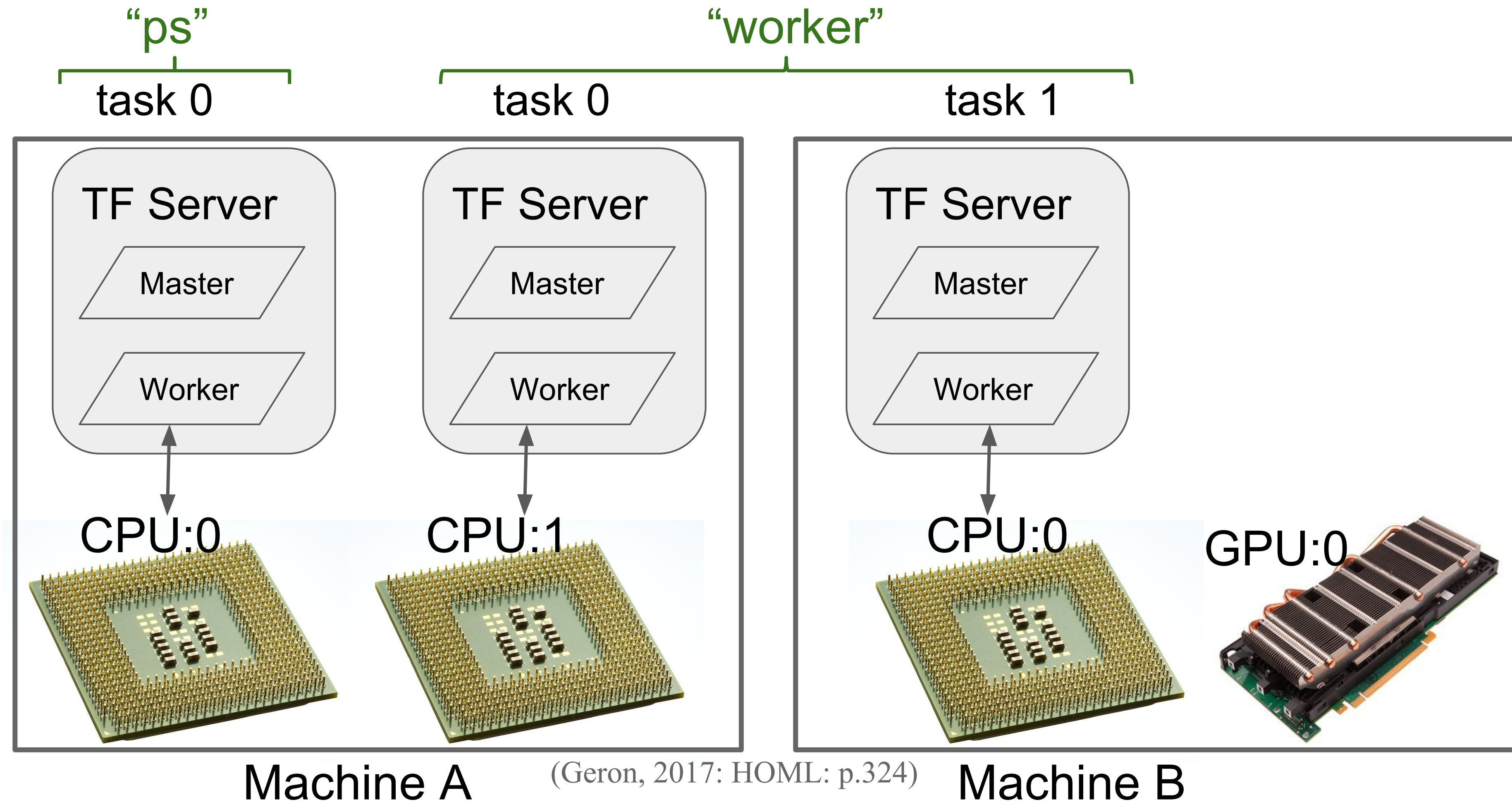
Multiple devices on multiple machines

```
with tf.device("/cpu:1")
   beta=tf.Variable(...)
```

```
with tf.device("/gpu:0")
   y_pred=tf.matmul(beta,X)
```

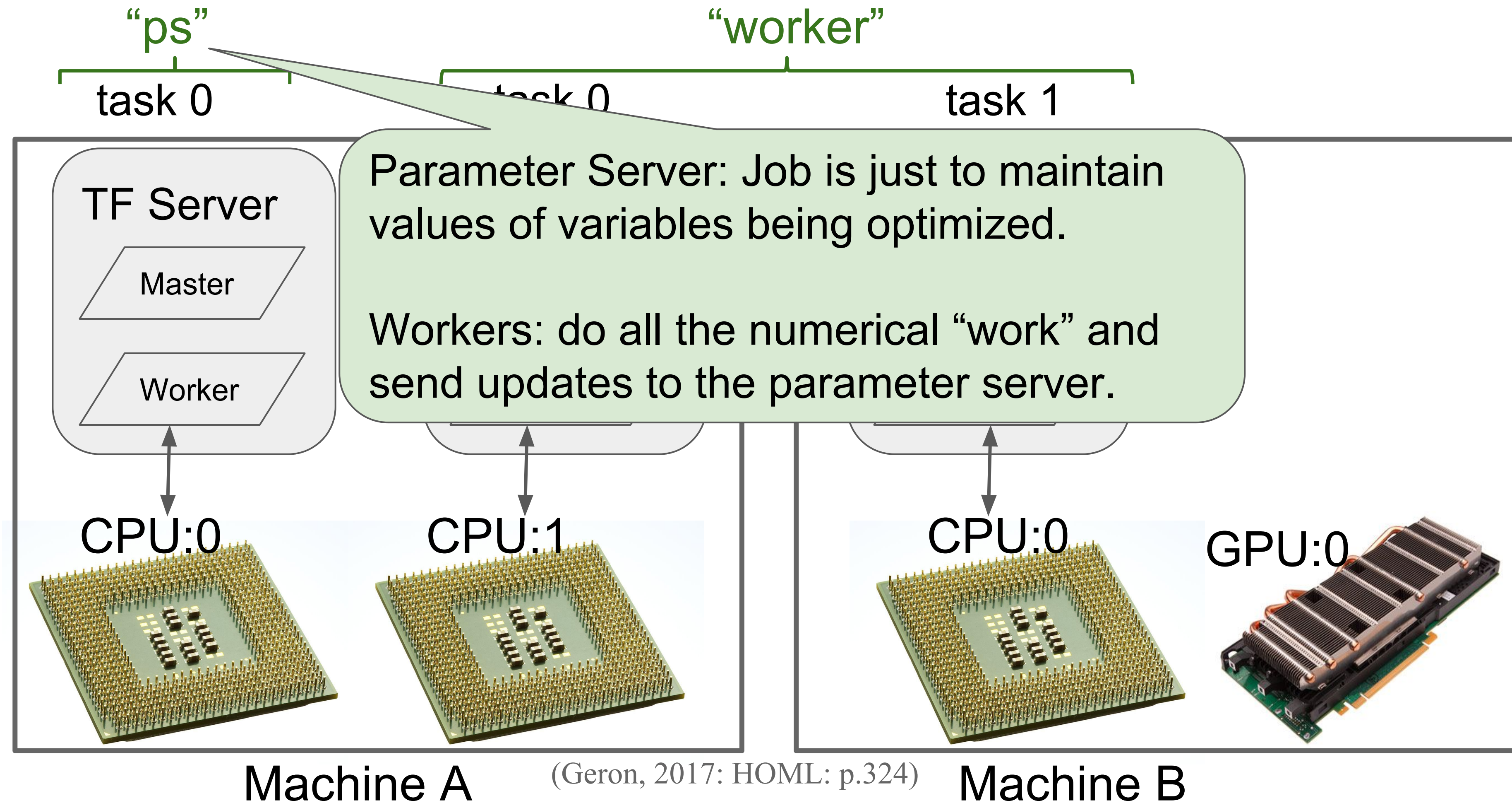


Cluster Distribution



(Geron, 2017: HOML: p.324)

Cluster Distribution



(Geron, 2017: HOML: p.324)

Summary

- TF is a workflow system, where records are always tensors
  - *operations* applied to tensors (as either Variables, constants, or placeholder)
- Optimized for numerical / linear algebra
  - automatically finds gradients
  - custom kernels for given devices
- “Easily” distributes
  - Within a single machine (local: many devices))
  - Across a cluster (many machines and devices)
  - Jobs broken up as parameter servers / workers makes coordination of data efficient