#ODSC

OPEN
DATA
SCIENCE
CONFERENCE

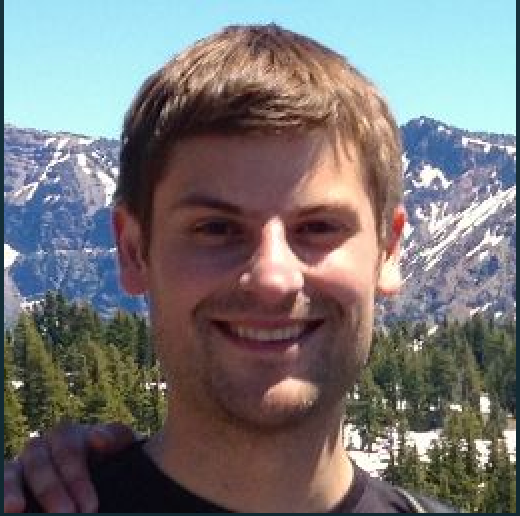
Boston
May 20-22nd 2016

@ODSC

A tour through the TensorFlow codebase

Kevin Robinson
@krob
Teaching Systems Lab, MIT

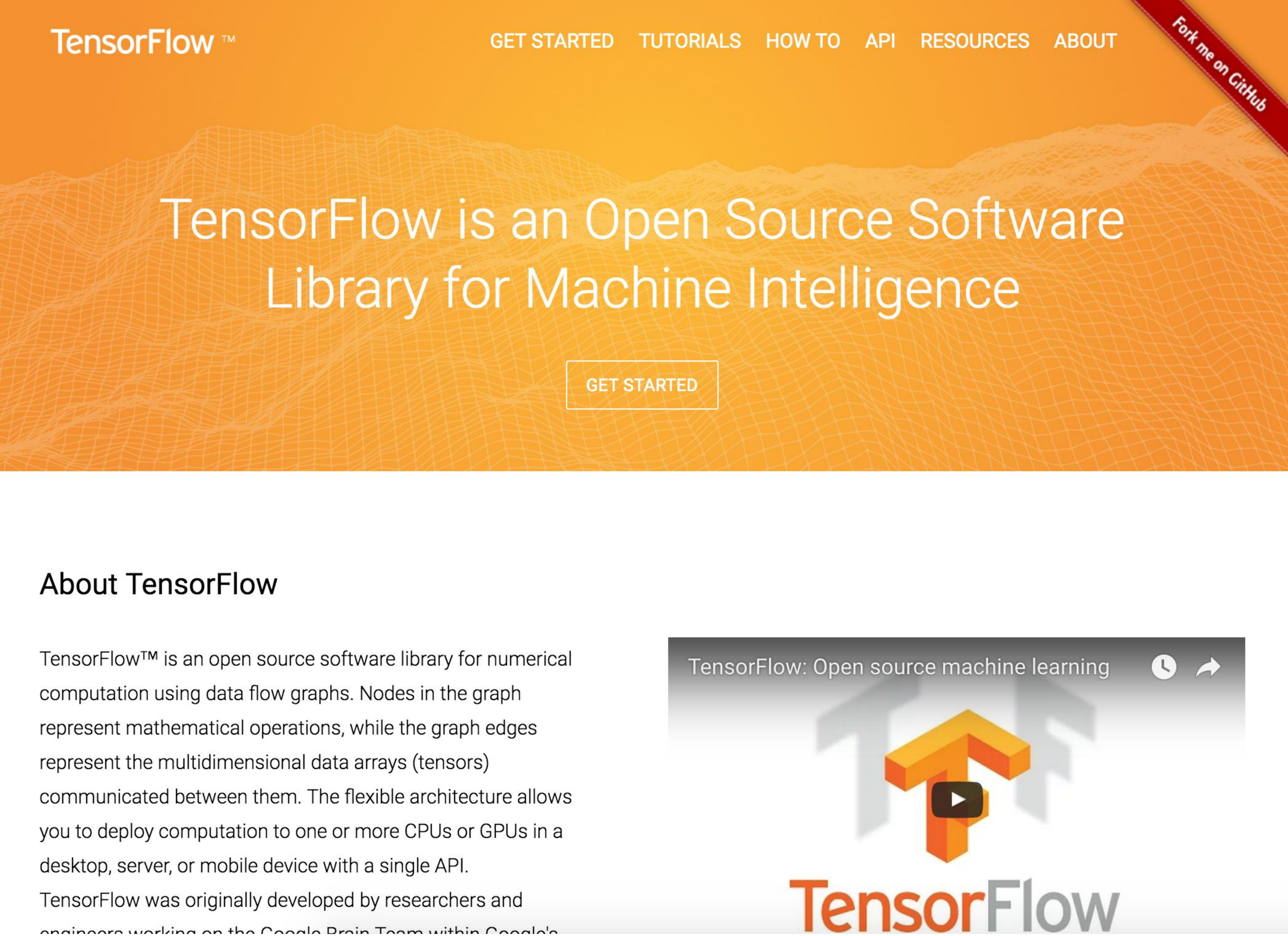
hello!



Kevin Robinson

@krob

Teaching Systems Lab, MIT

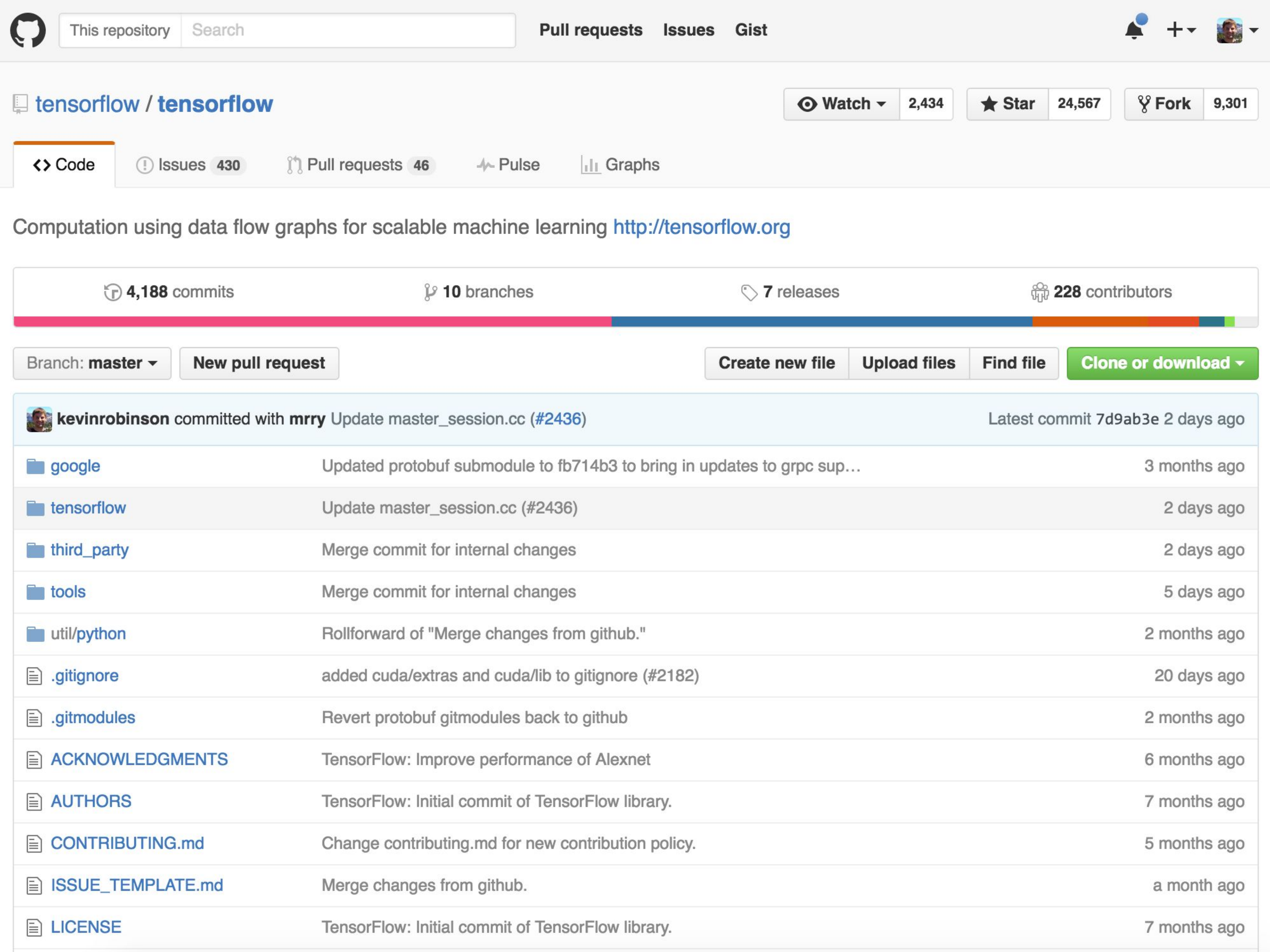


TensorFlow is an Open Source Software Library for Machine Intelligence

About TensorFlow

TensorFlow™ is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API.

TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's



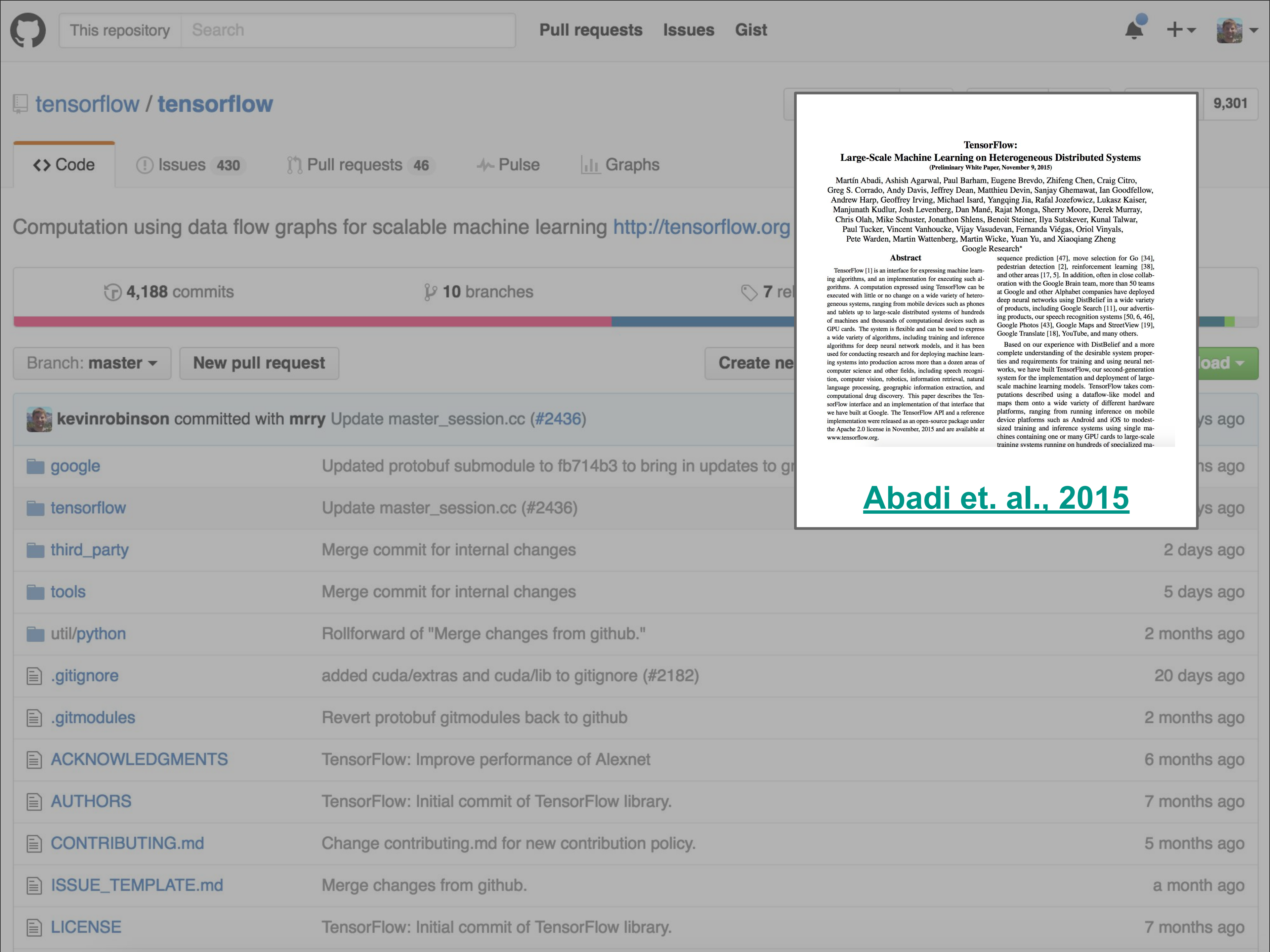
tensorflow / tensorflow

Computation using data flow graphs for scalable machine learning http://tensorflow.org

4,188 commits | 10 branches | 7 releases | 228 contributors

kevinrobinson committed with mrry Update master_session.cc (#2436) Latest commit 7d9ab3e 2 days ago

google	Updated protobuf submodule to fb714b3 to bring in updates to grpc sup…	3 months ago
tensorflow	Update master_session.cc (#2436)	2 days ago
third_party	Merge commit for internal changes	2 days ago
tools	Merge commit for internal changes	5 days ago
util/python	Rollforward of "Merge changes from github."	2 months ago
.gitignore	added cuda/extras and cuda/lib to gitignore (#2182)	20 days ago
.gitmodules	Revert protobuf gitmodules back to github	2 months ago
ACKNOWLEDGMENTS	TensorFlow: Improve performance of Alexnet	6 months ago
AUTHORS	TensorFlow: Initial commit of TensorFlow library.	7 months ago
CONTRIBUTING.md	Change contributing.md for new contribution policy.	5 months ago
ISSUE_TEMPLATE.md	Merge changes from github.	a month ago
LICENSE	TensorFlow: Initial commit of TensorFlow library.	7 months ago



tensorflow / tensorflow

Computation using data flow graphs for scalable machine learning http://tensorflow.org

4,188 commits
10 branches
7 rel

kevinrobinson committed with mrry Update master_session.cc (#2436) ys ago

google	Updated protobuf submodule to fb714b3 to bring in updates to gr	hs ago
tensorflow	Update master_session.cc (#2436)	ys ago
third_party	Merge commit for internal changes	2 days ago
tools	Merge commit for internal changes	5 days ago
util/python	Rollforward of "Merge changes from github."	2 months ago
.gitignore	added cuda/extras and cuda/lib to gitignore (#2182)	20 days ago
.gitmodules	Revert protobuf gitmodules back to github	2 months ago
ACKNOWLEDGMENTS	TensorFlow: Improve performance of Alexnet	6 months ago
AUTHORS	TensorFlow: Initial commit of TensorFlow library.	7 months ago
CONTRIBUTING.md	Change contributing.md for new contribution policy.	5 months ago
ISSUE_TEMPLATE.md	Merge changes from github.	a month ago
LICENSE	TensorFlow: Initial commit of TensorFlow library.	7 months ago

TensorFlow:
Large-Scale Machine Learning on Heterogeneous Distributed Systems
(Preliminary White Paper, November 9, 2015)

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro,
Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow,
Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser,
Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray,
Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar,
Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals,
Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng
Google Research*

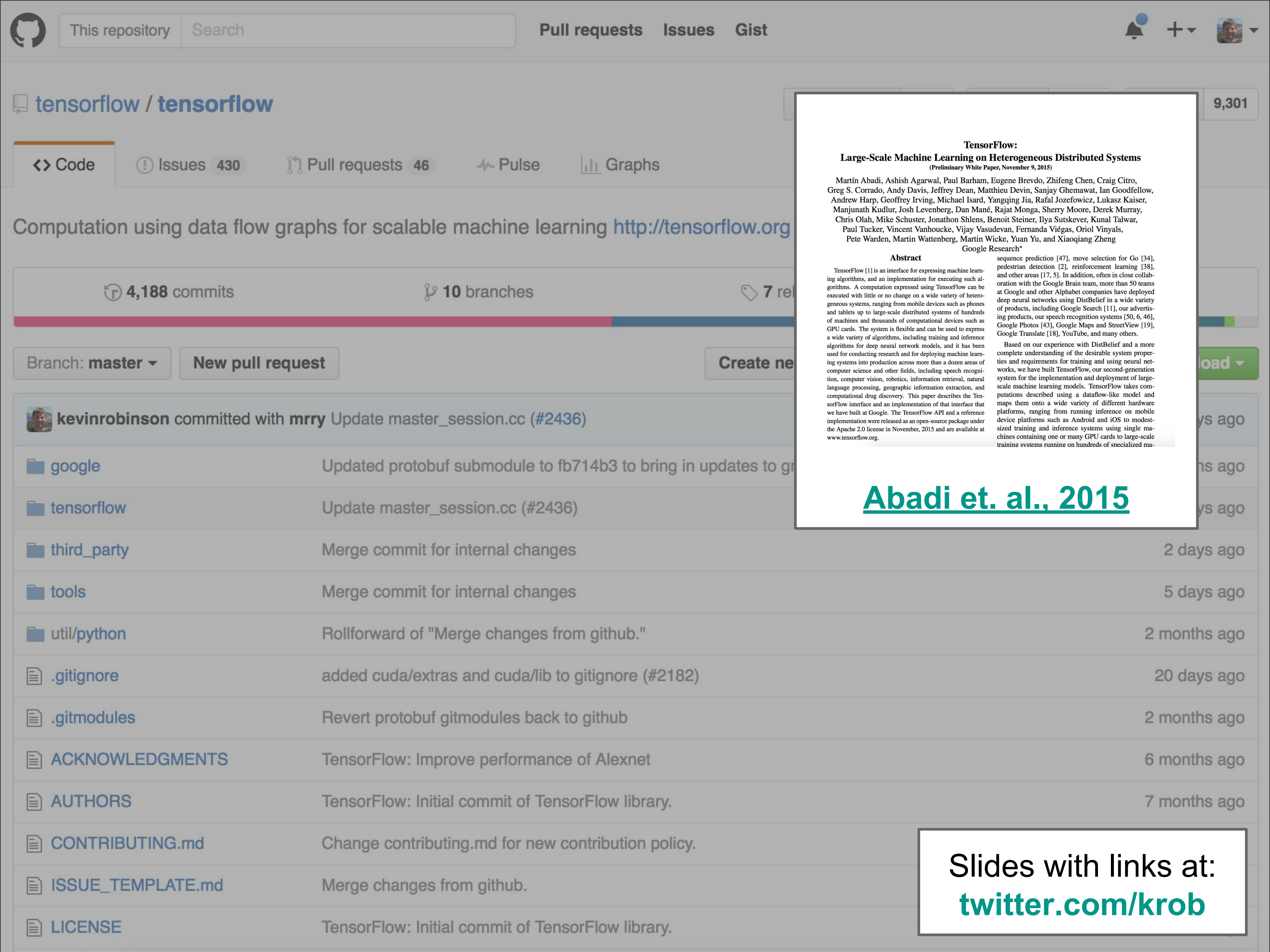
Abstract

TensorFlow [1] is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms. A computation expressed using TensorFlow can be executed with little or no change on a wide variety of heterogeneous systems, ranging from mobile devices such as phones and tablets up to large-scale distributed systems of hundreds of machines and thousands of computational devices such as GPU cards. The system is flexible and can be used to express a wide variety of algorithms, including training and inference algorithms for deep neural network models, and it has been used for conducting research and for deploying machine learning systems into production across more than a dozen areas of computer science and other fields, including speech recognition, computer vision, robotics, information retrieval, natural language processing, geographic information extraction, and computational drug discovery. This paper describes the TensorFlow interface and an implementation of that interface that we have built at Google. The TensorFlow API and a reference implementation were released as an open-source package under the Apache 2.0 license in November, 2015 and are available at www.tensorflow.org.

sequence prediction [47], move selection for Go [34], pedestrian detection [2], reinforcement learning [38], and other areas [17, 5]. In addition, often in close collaboration with the Google Brain team, more than 50 teams at Google and other Alphabet companies have deployed deep neural networks using DistBelief in a wide variety of products, including Google Search [11], our advertising products, our speech recognition systems [50, 6, 46], Google Photos [43], Google Maps and StreetView [19], Google Translate [18], YouTube, and many others.

Based on our experience with DistBelief and a more complete understanding of the desirable system properties and requirements for training and using neural networks, we have built TensorFlow, our second-generation system for the implementation and deployment of large-scale machine learning models. TensorFlow takes computations described using a dataflow-like model and maps them onto a wide variety of different hardware platforms, ranging from running inference on mobile device platforms such as Android and iOS to modest-sized training and inference systems using single machines containing one or many GPU cards to large-scale training systems running on hundreds of specialized ma-

Abadi et. al., 2015



tensorflow / tensorflow

9,301

Computation using data flow graphs for scalable machine learning http://tensorflow.org

4,188 commits
10 branches
7 rel

kevinrobinson committed with mrry Update master_session.cc (#2436) ys ago

google Updated protobuf submodule to fb714b3 to bring in updates to gr hs ago

tensorflow Update master_session.cc (#2436) ys ago

third_party Merge commit for internal changes 2 days ago

tools Merge commit for internal changes 5 days ago

util/python Rollforward of "Merge changes from github." 2 months ago

.gitignore added cuda/extras and cuda/lib to gitignore (#2182) 20 days ago

.gitmodules Revert protobuf gitmodules back to github 2 months ago

ACKNOWLEDGMENTS TensorFlow: Improve performance of Alexnet 6 months ago

AUTHORS TensorFlow: Initial commit of TensorFlow library. 7 months ago

CONTRIBUTING.md Change contributing.md for new contribution policy.

ISSUE_TEMPLATE.md Merge changes from github.

LICENSE TensorFlow: Initial commit of TensorFlow library.

TensorFlow:
Large-Scale Machine Learning on Heterogeneous Distributed Systems
(Preliminary White Paper, November 9, 2015)

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro,
Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow,
Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser,
Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray,
Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar,
Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals,
Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng
Google Research*

Abstract

TensorFlow [1] is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms. A computation expressed using TensorFlow can be executed with little or no change on a wide variety of heterogeneous systems, ranging from mobile devices such as phones and tablets up to large-scale distributed systems of hundreds of machines and thousands of computational devices such as GPU cards. The system is flexible and can be used to express a wide variety of algorithms, including training and inference algorithms for deep neural network models, and it has been used for conducting research and for deploying machine learning systems into production across more than a dozen areas of computer science and other fields, including speech recognition, computer vision, robotics, information retrieval, natural language processing, geographic information extraction, and computational drug discovery. This paper describes the TensorFlow interface and an implementation of that interface that we have built at Google. The TensorFlow API and a reference implementation were released as an open-source package under the Apache 2.0 license in November, 2015 and are available at www.tensorflow.org.

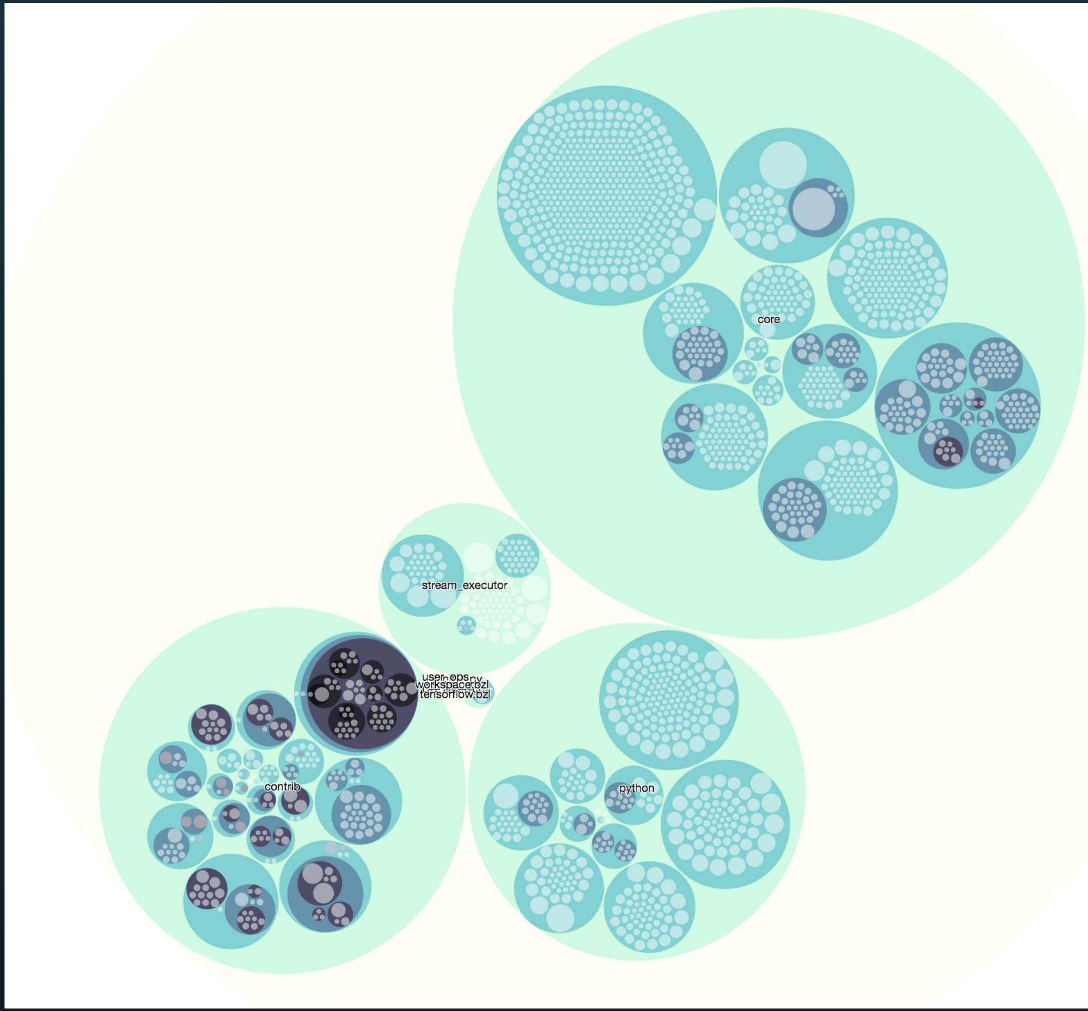
sequence prediction [47], move selection for Go [34], pedestrian detection [2], reinforcement learning [38], and other areas [17, 5]. In addition, often in close collaboration with the Google Brain team, more than 50 teams at Google and other Alphabet companies have deployed deep neural networks using DistBelief in a wide variety of products, including Google Search [11], our advertising products, our speech recognition systems [50, 6, 46], Google Photos [43], Google Maps and StreetView [19], Google Translate [18], YouTube, and many others.

Based on our experience with DistBelief and a more complete understanding of the desirable system properties and requirements for training and using neural networks, we have built TensorFlow, our second-generation system for the implementation and deployment of large-scale machine learning models. TensorFlow takes computations described using a dataflow-like model and maps them onto a wide variety of different hardware platforms, ranging from running inference on mobile device platforms such as Android and iOS to modest-sized training and inference systems using single machines containing one or many GPU cards to large-scale training systems running on hundreds of specialized ma-

Abadi et. al., 2015

Slides with links at:
twitter.com/krob

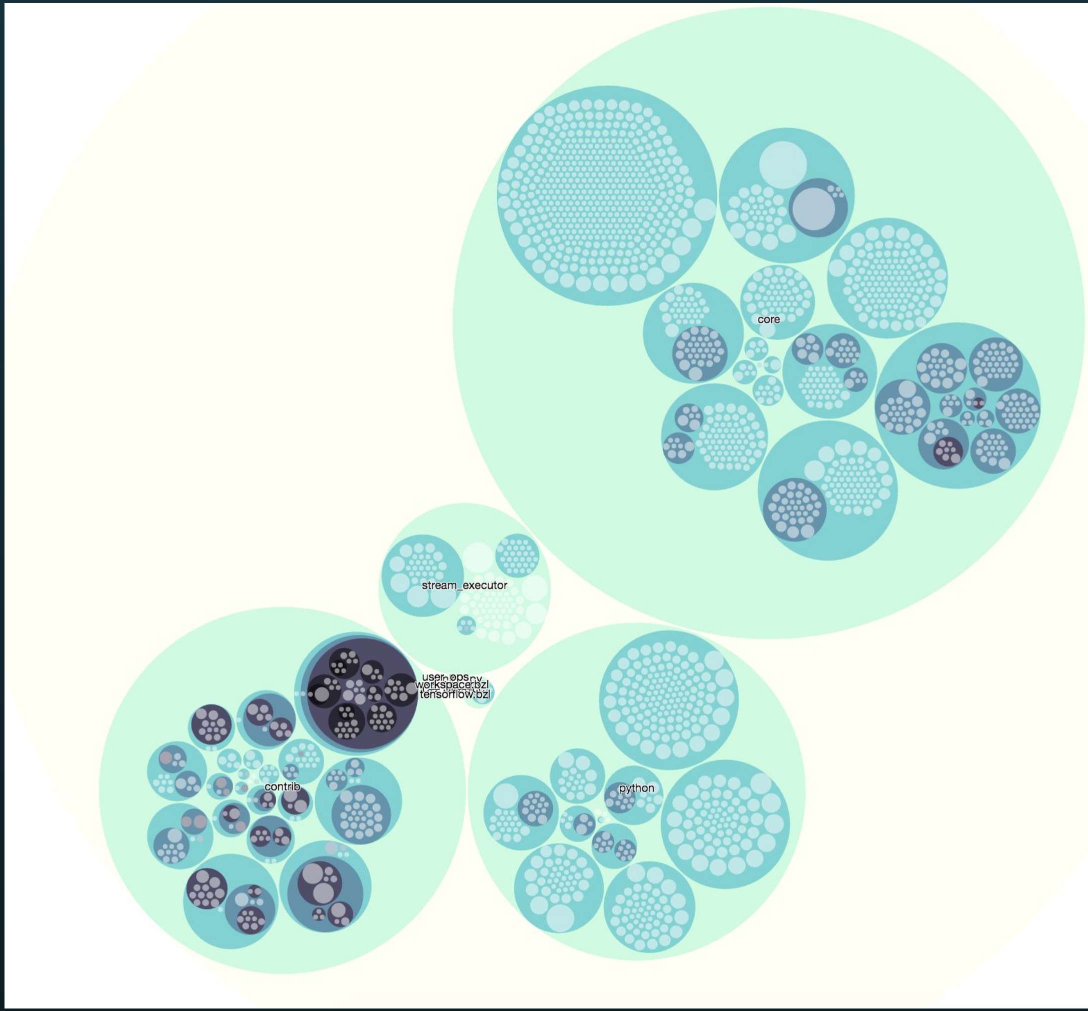
A tour of the TensorFlow codebase



http://public.kevinrobinsonblog.com/tensorflow-codebase/

A tour of the TensorFlow codebase

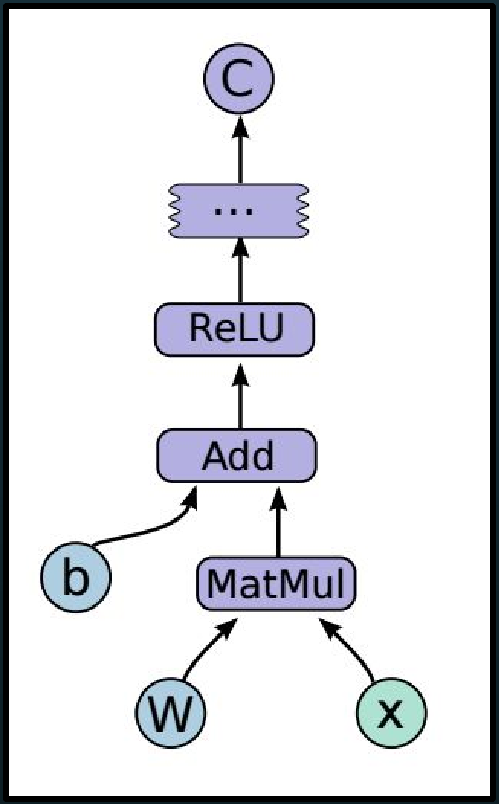
1. **Expressing** computation
2. **Distributing** computation
3. **Executing** computation

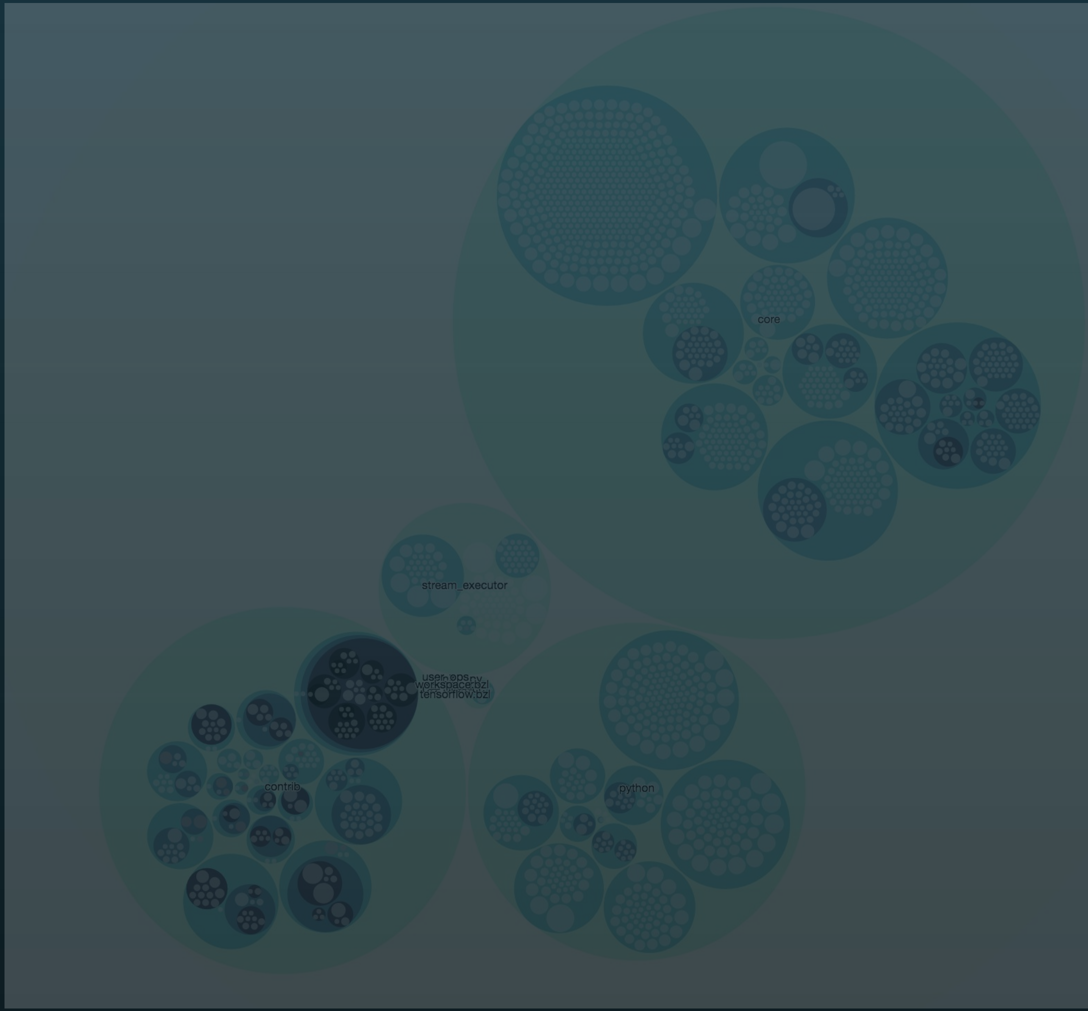


http://public.kevinrobinsonblog.com/tensorflow-codebase/

A tour through the TensorFlow codebase

1. **Expressing** graphs

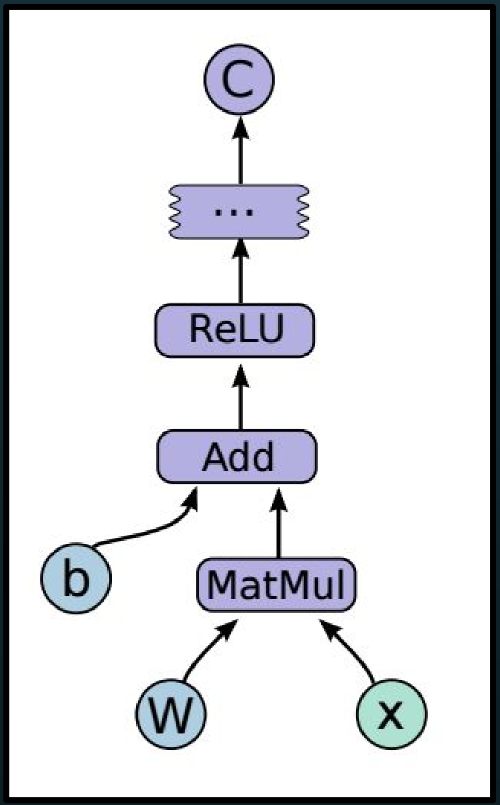


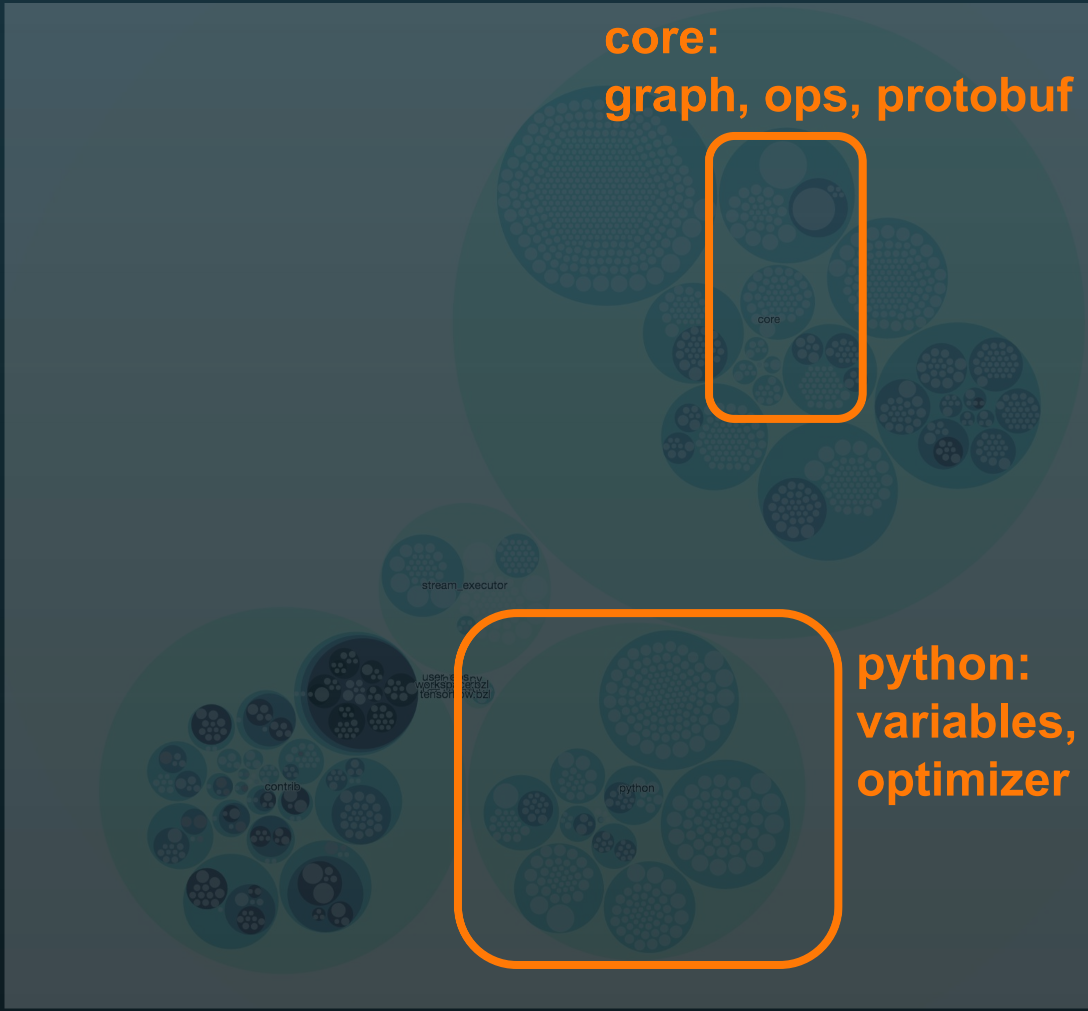


http://public.kevinrobinsonblog.com/tensorflow-codebase/

A tour through the TensorFlow codebase

1. **Expressing** graphs





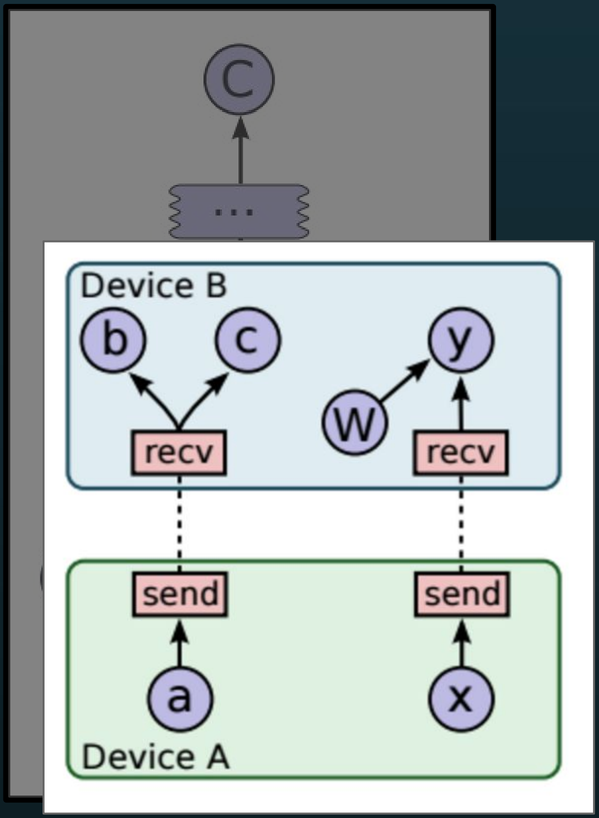
core:
graph, ops, protobuf

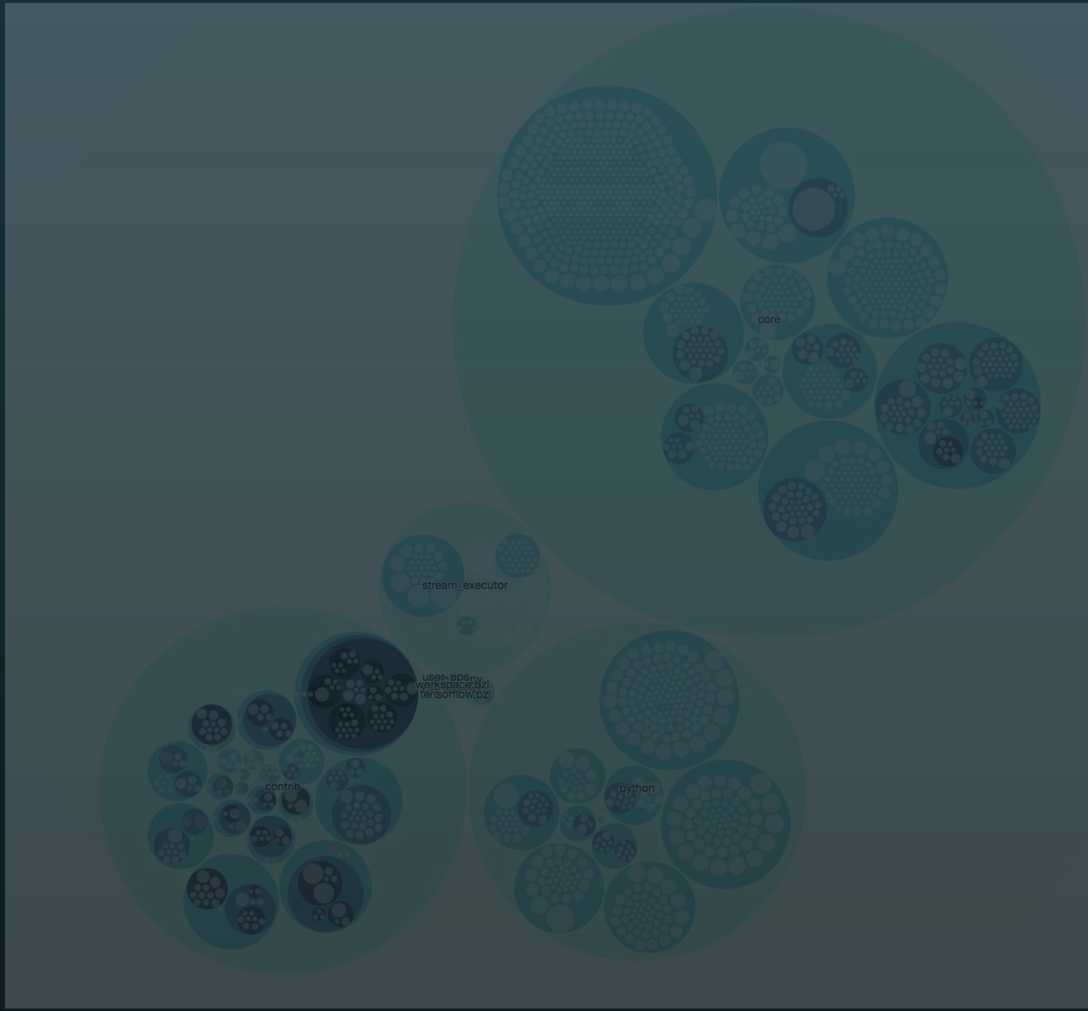
python:
variables,
optimizer

http://public.kevinrobinsonblog.com/tensorflow-codebase/

A tour through the TensorFlow codebase

2. **Distributing** graphs

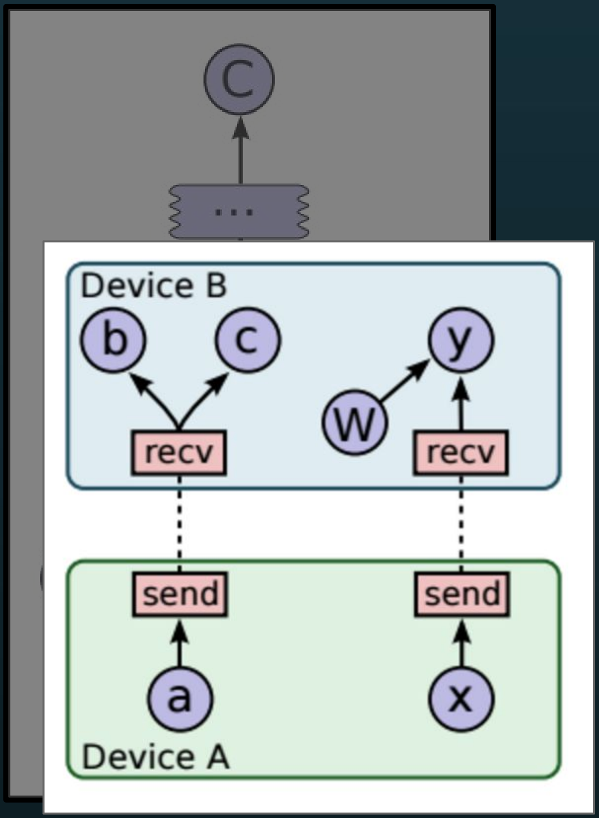


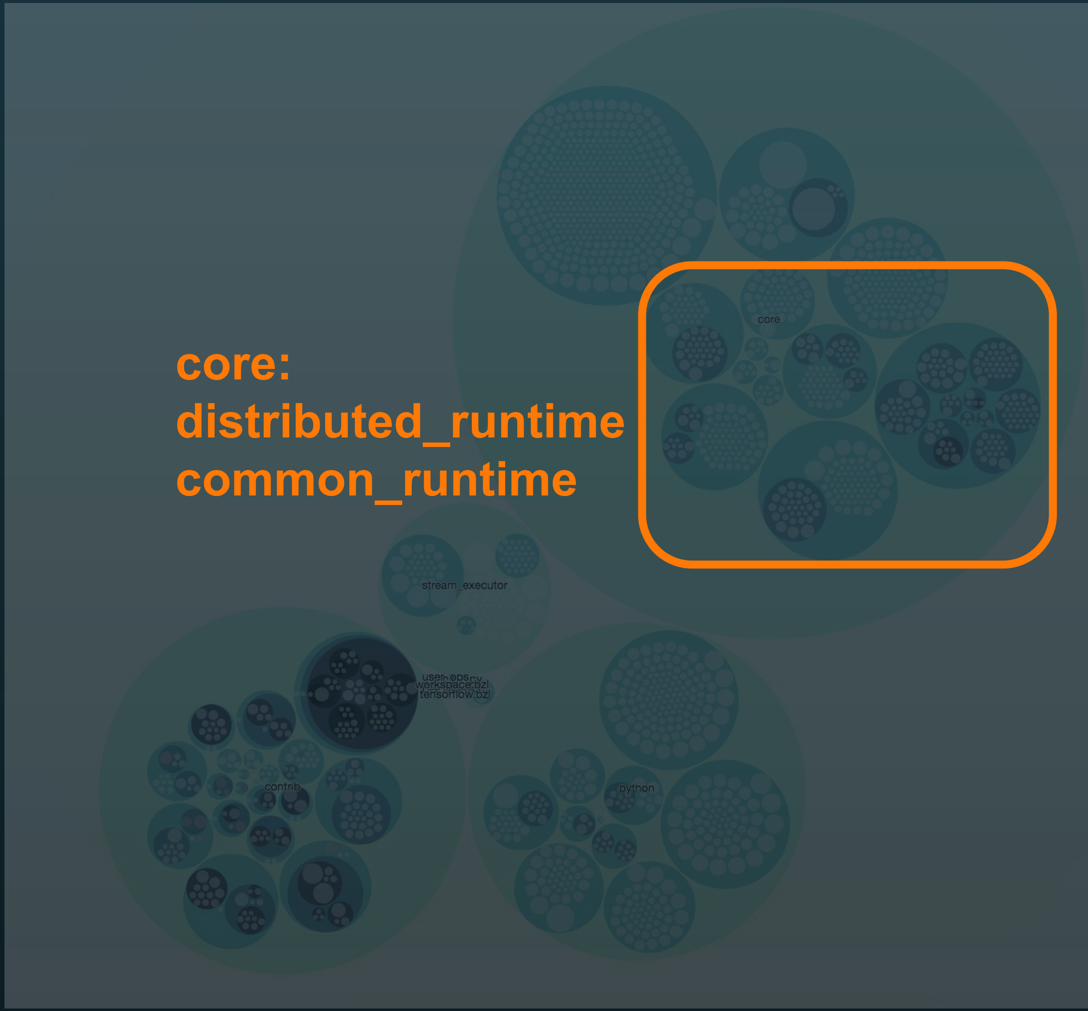


http://public.kevinrobinsonblog.com/tensorflow-codebase/

A tour through the TensorFlow codebase

2. **Distributing** graphs



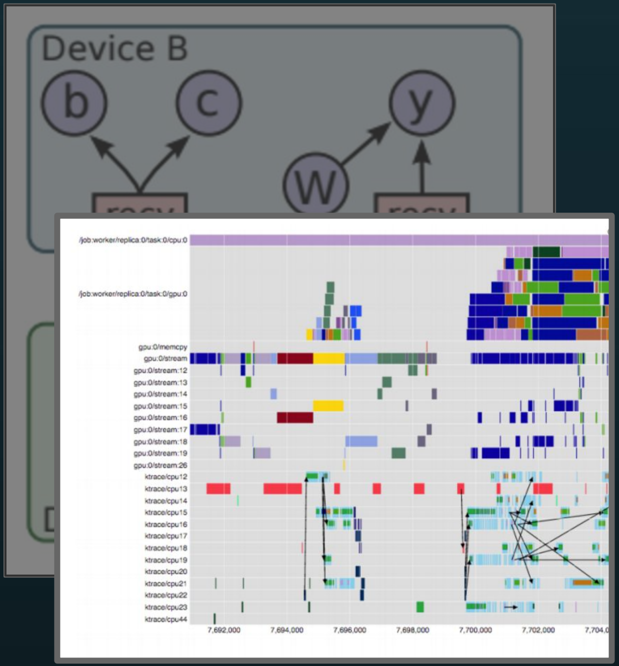


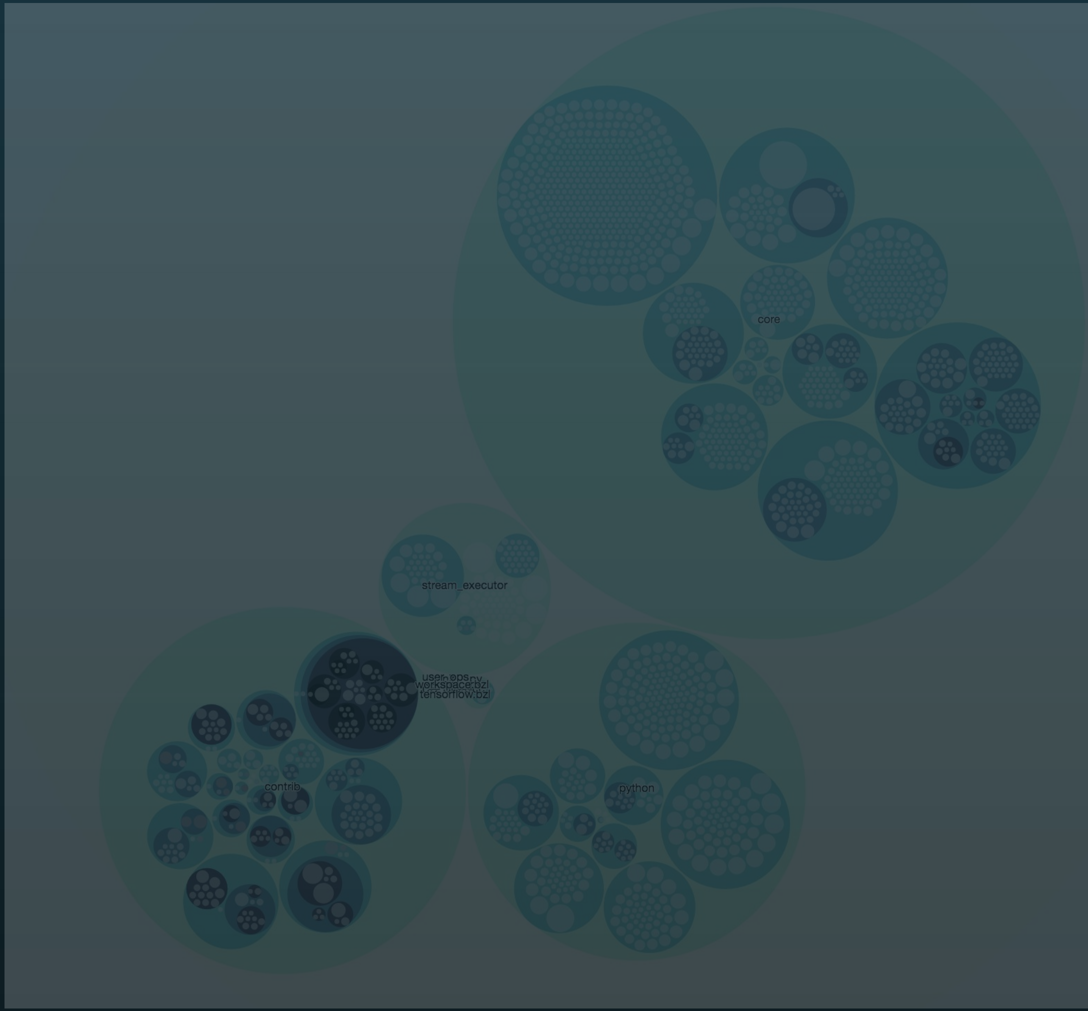
core:
distributed_runtime
common_runtime

http://public.kevinrobinsonblog.com/tensorflow-codebase/

A tour through the TensorFlow codebase

3. **Executing** graphs

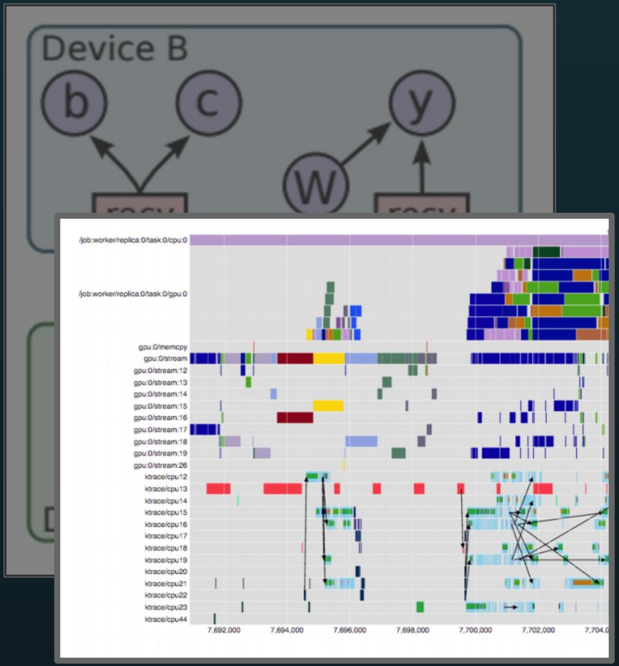


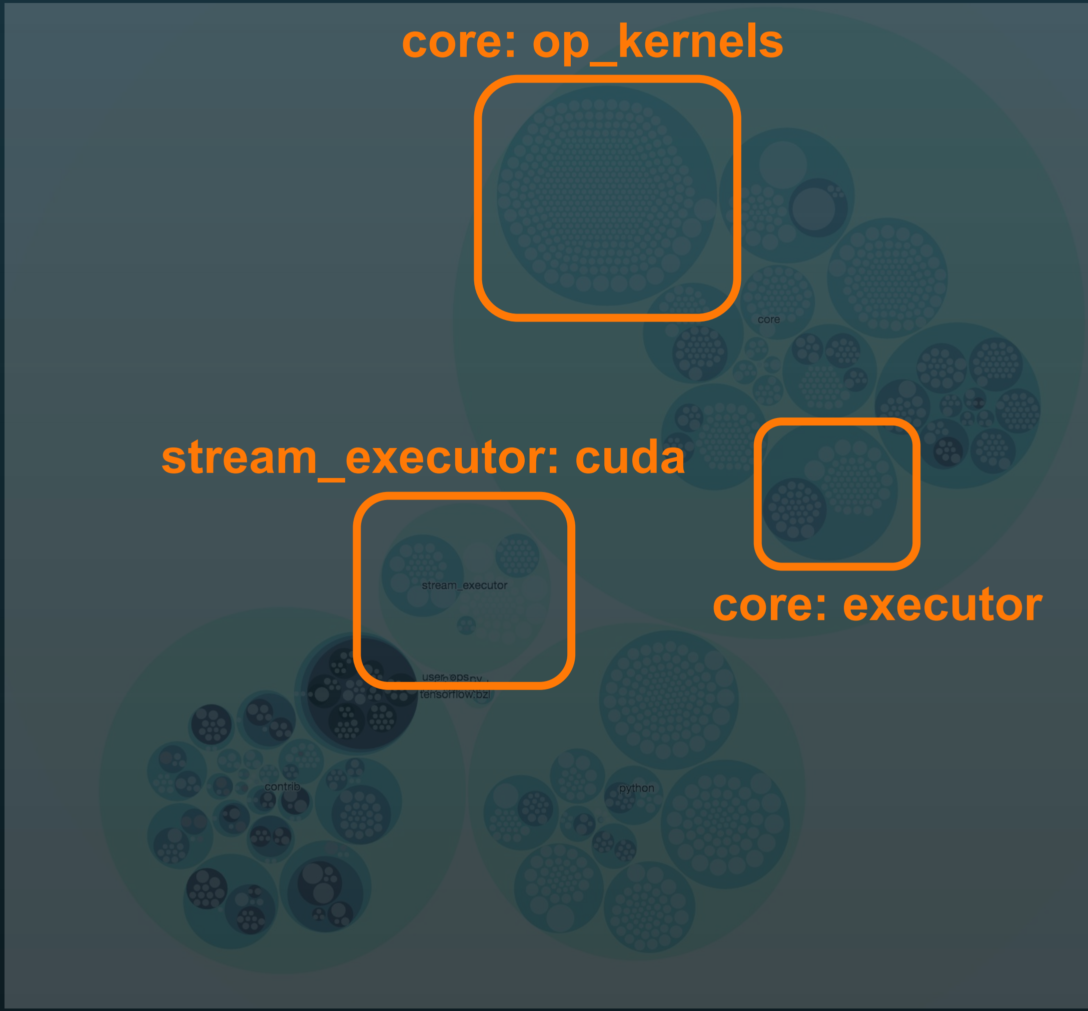


http://public.kevinrobinsonblog.com/tensorflow-codebase/

A tour through the TensorFlow codebase

3. **Executing** graphs




core: op_kernels

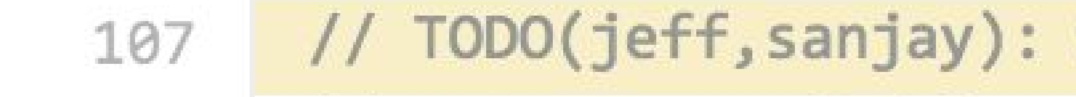
stream_executor: cuda

core: executor

http://public.kevinrobinsonblog.com/tensorflow-codebase/

A tour through the TensorFlow codebase

4. And my favorite **TODO**

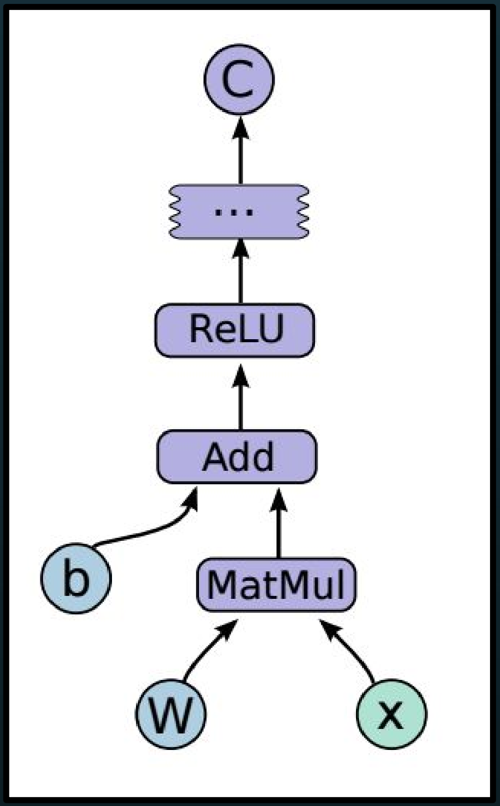


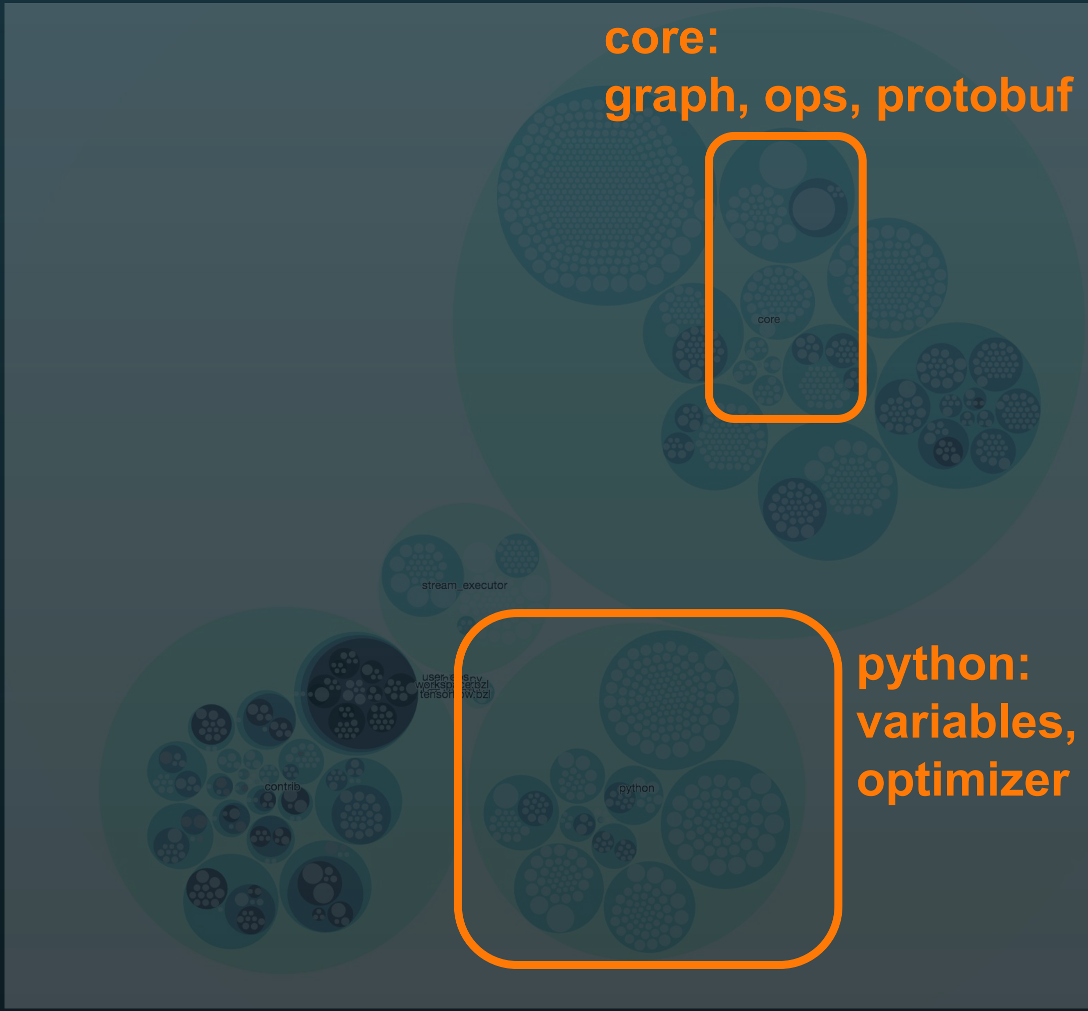
```
// TODO(jeff,sanjay):
```

?

A tour through the TensorFlow codebase

1. **Expressing** graphs





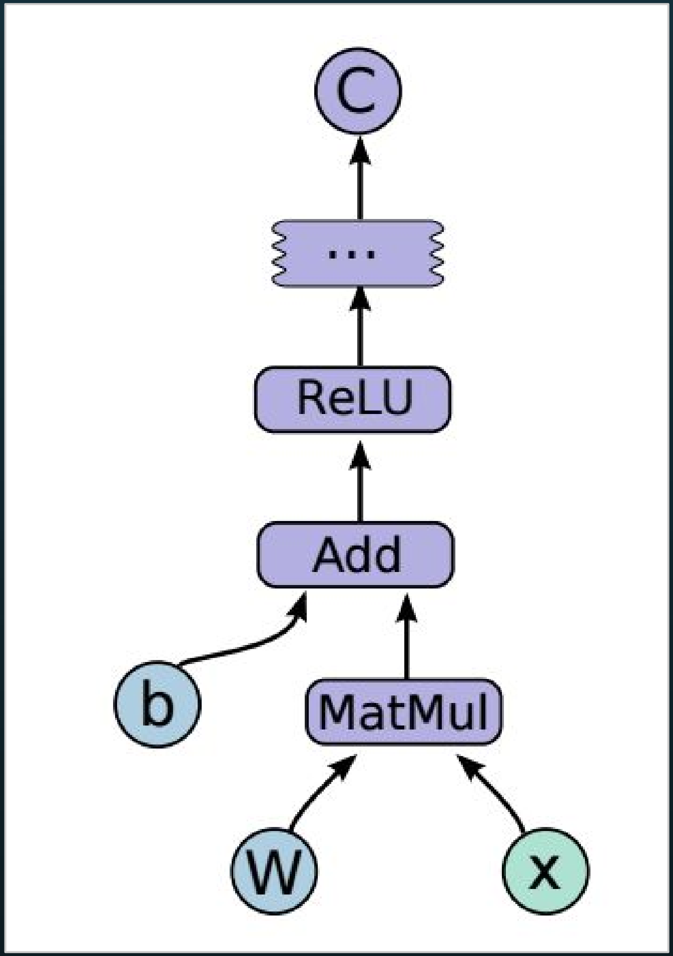
core:
graph, ops, protobuf

python:
variables,
optimizer

http://public.kevinrobinsonblog.com/tensorflow-codebase/

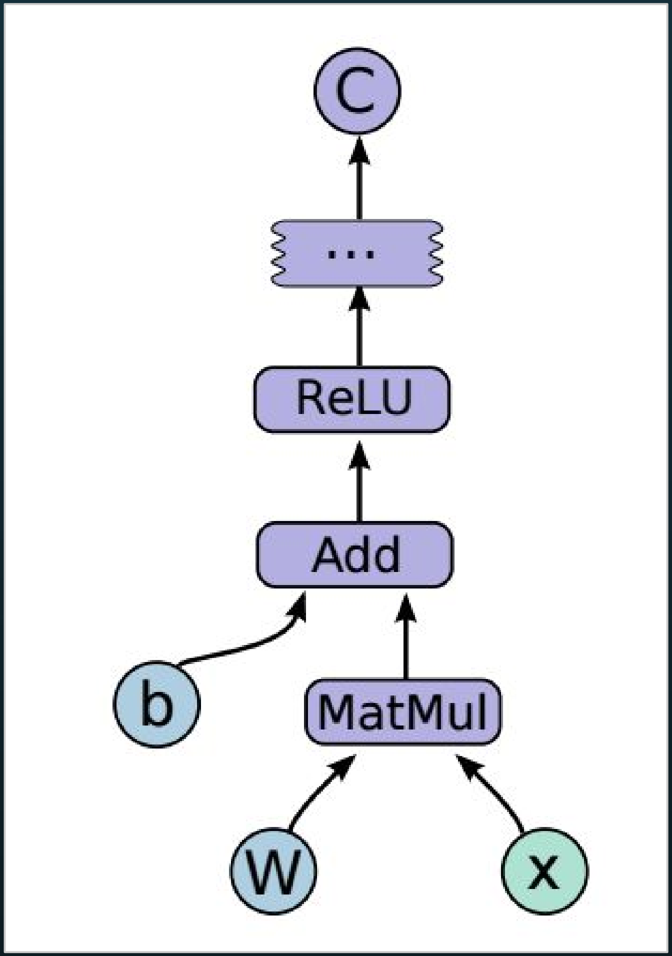
Expressing: Graphs and Ops

Graph

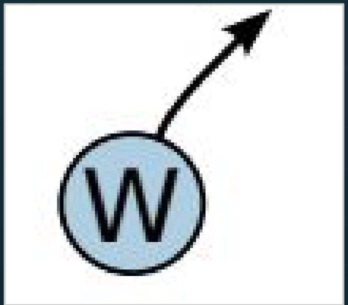


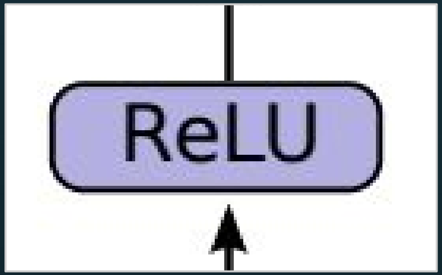
Expressing: Graphs and Ops

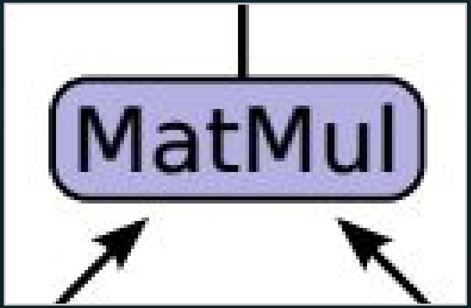
Graph



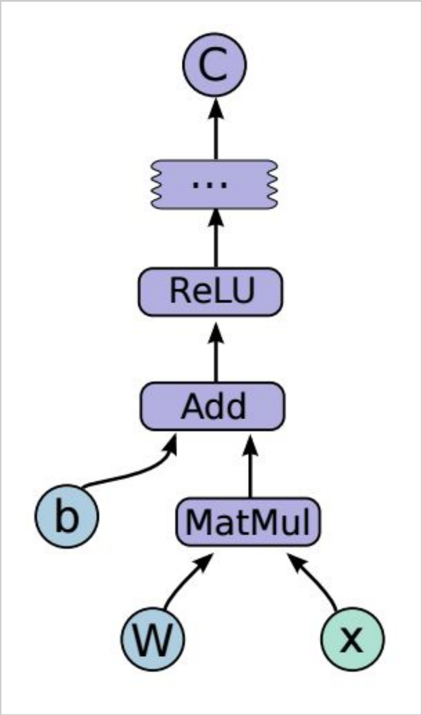
Ops

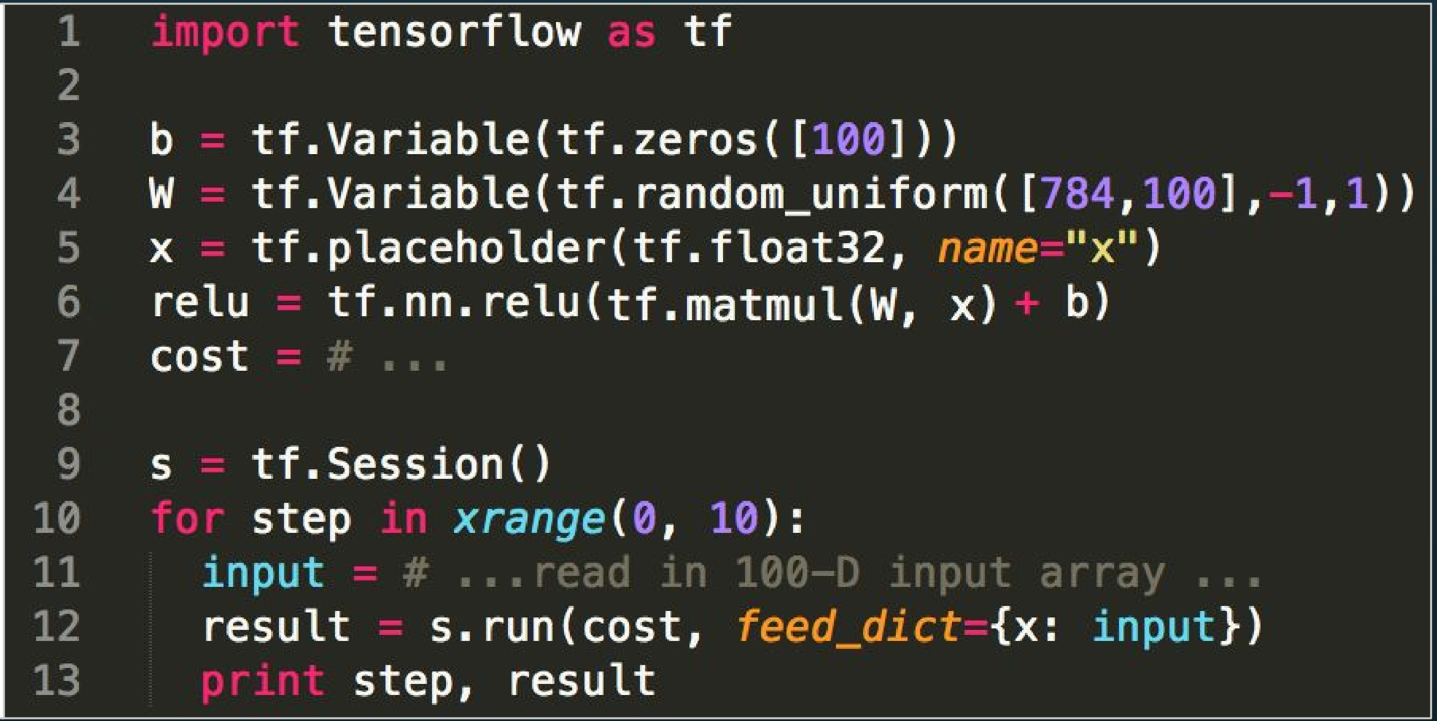






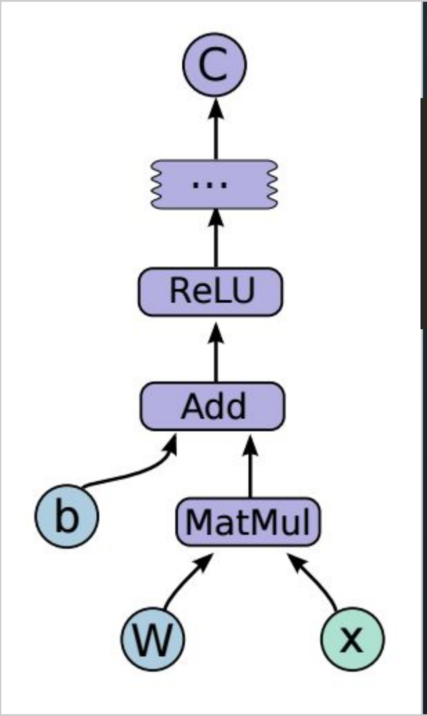
Expressing: Graphs and Ops

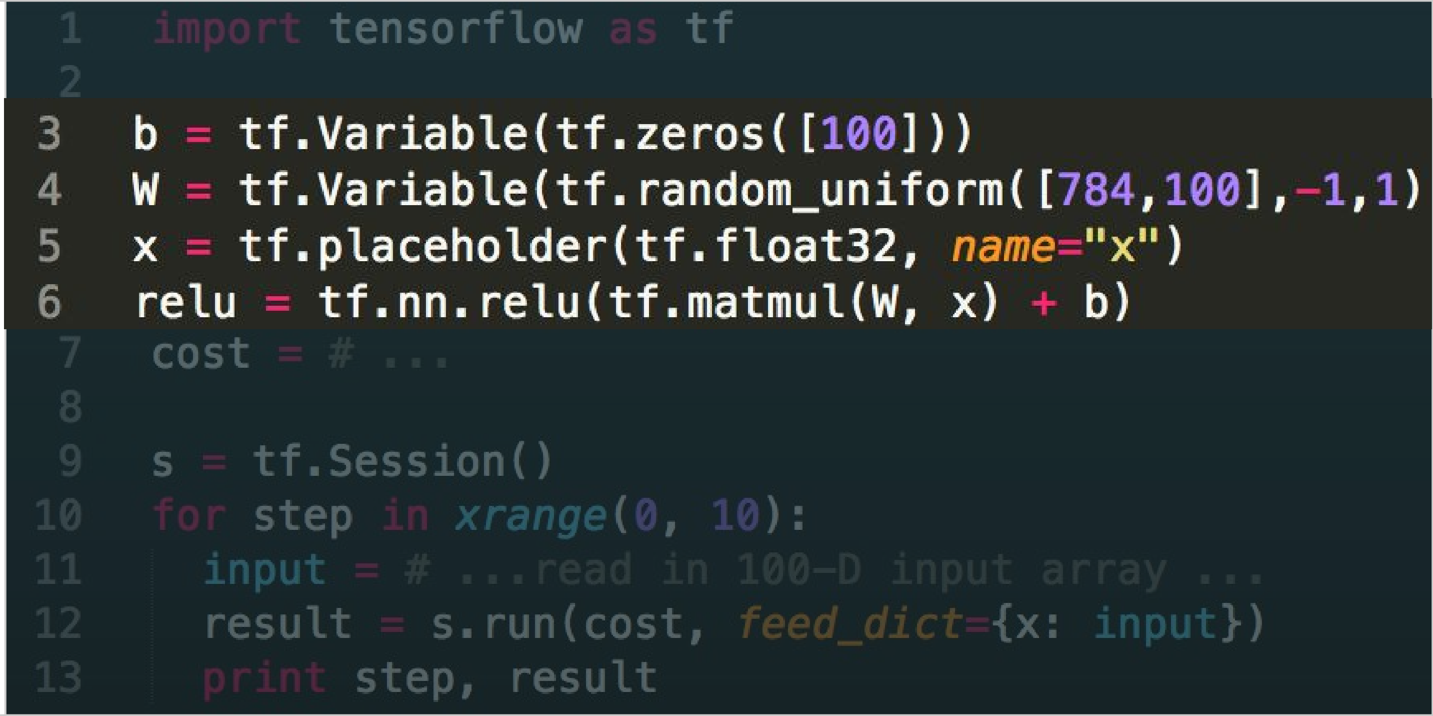




```python
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784,100],-1,1))
x = tf.placeholder(tf.float32, name="x")
relu = tf.nn.relu(tf.matmul(W, x) + b)
cost = # ...

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

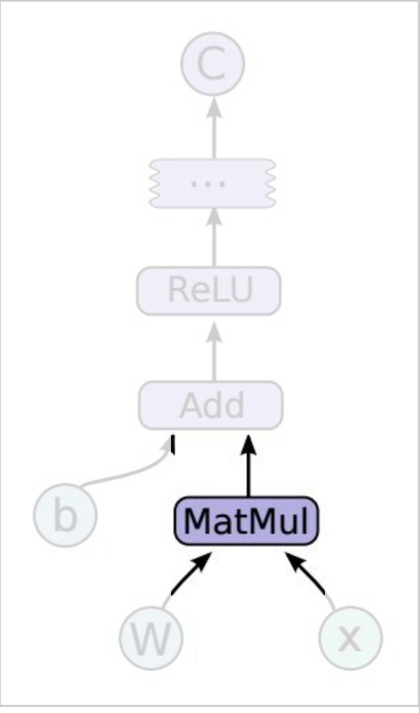
Expressing: Ops

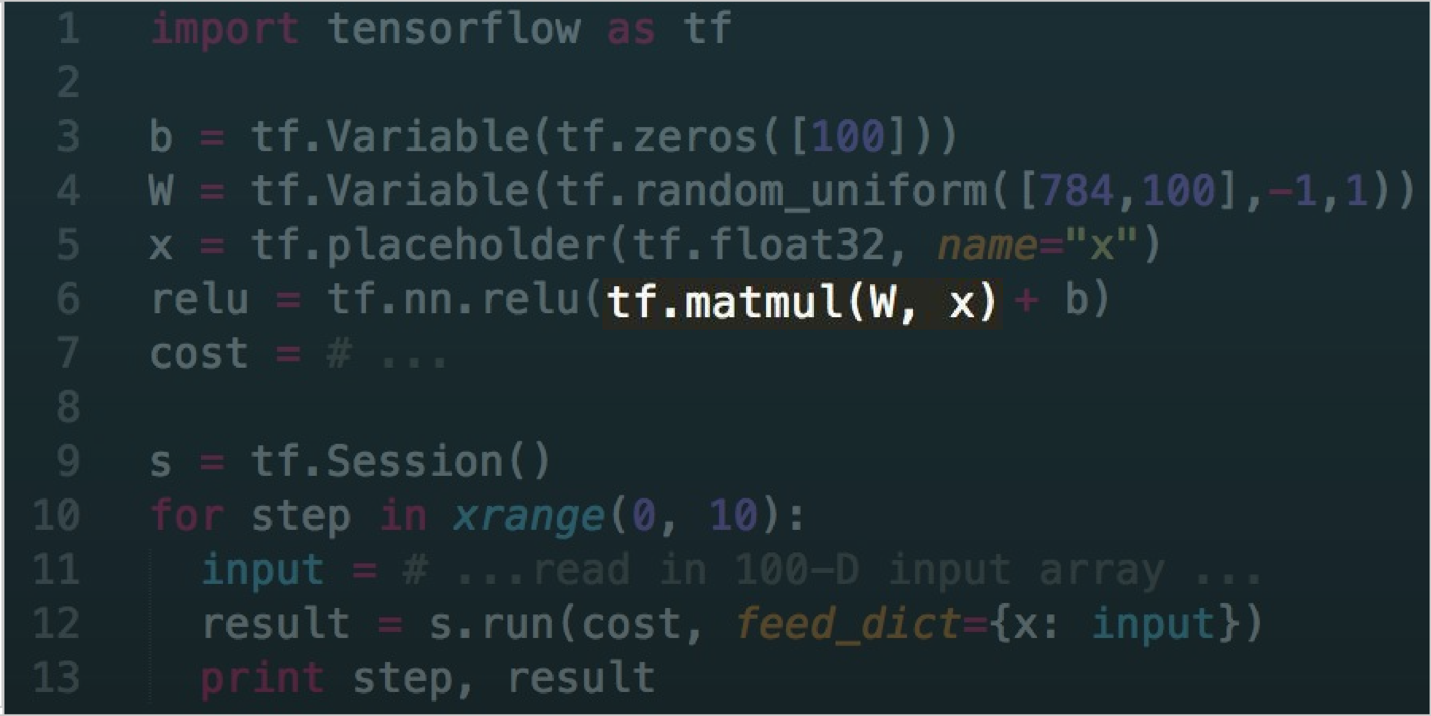




```python
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784,100],-1,1)
x = tf.placeholder(tf.float32, name="x")
relu = tf.nn.relu(tf.matmul(W, x) + b)
cost = # ...

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

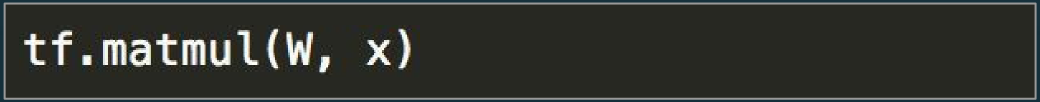
Expressing: Ops





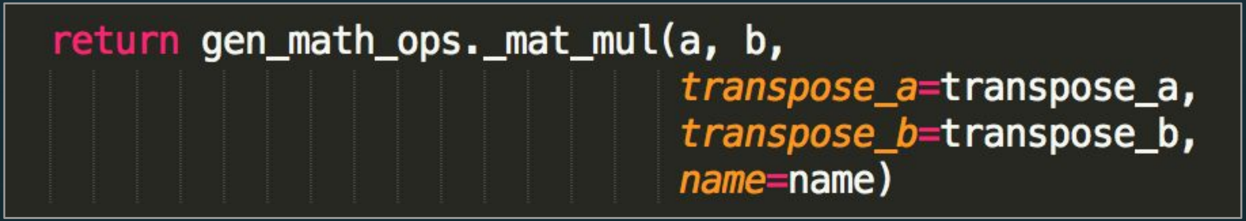
```
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784,100],-1,1))
x = tf.placeholder(tf.float32, name="x")
relu = tf.nn.relu(tf.matmul(W, x) + b)
cost = # ...

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

Expressing: Ops



```
tf.matmul(W, x)
```

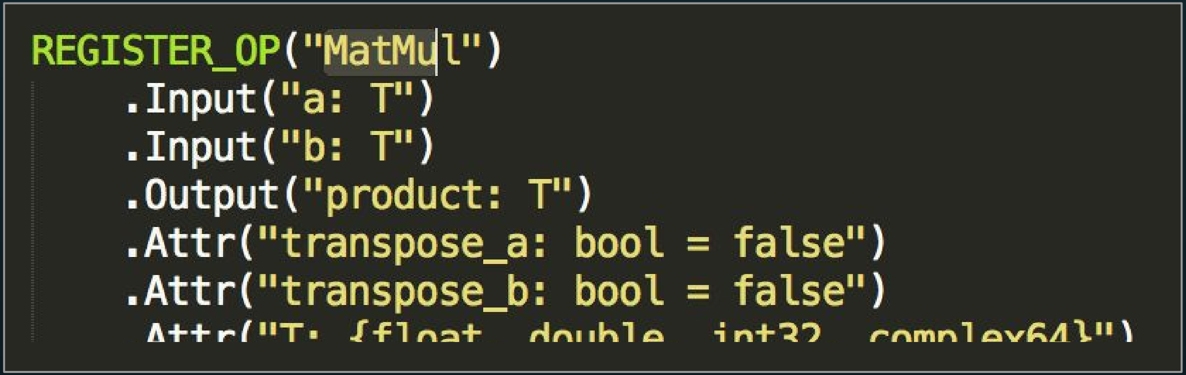
in math_ops.py#L1137



```
  return gen_math_ops._mat_mul(a, b,
                               transpose_a=transpose_a,
                               transpose_b=transpose_b,
                               name=name)
```

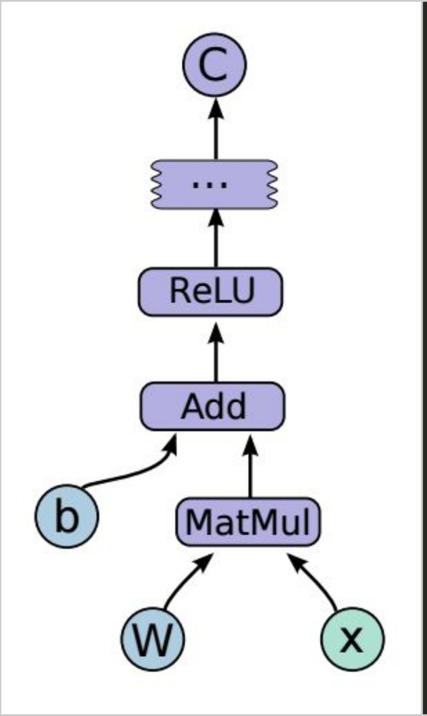
calls C++ wrappers generated by cc/BUILD#L27

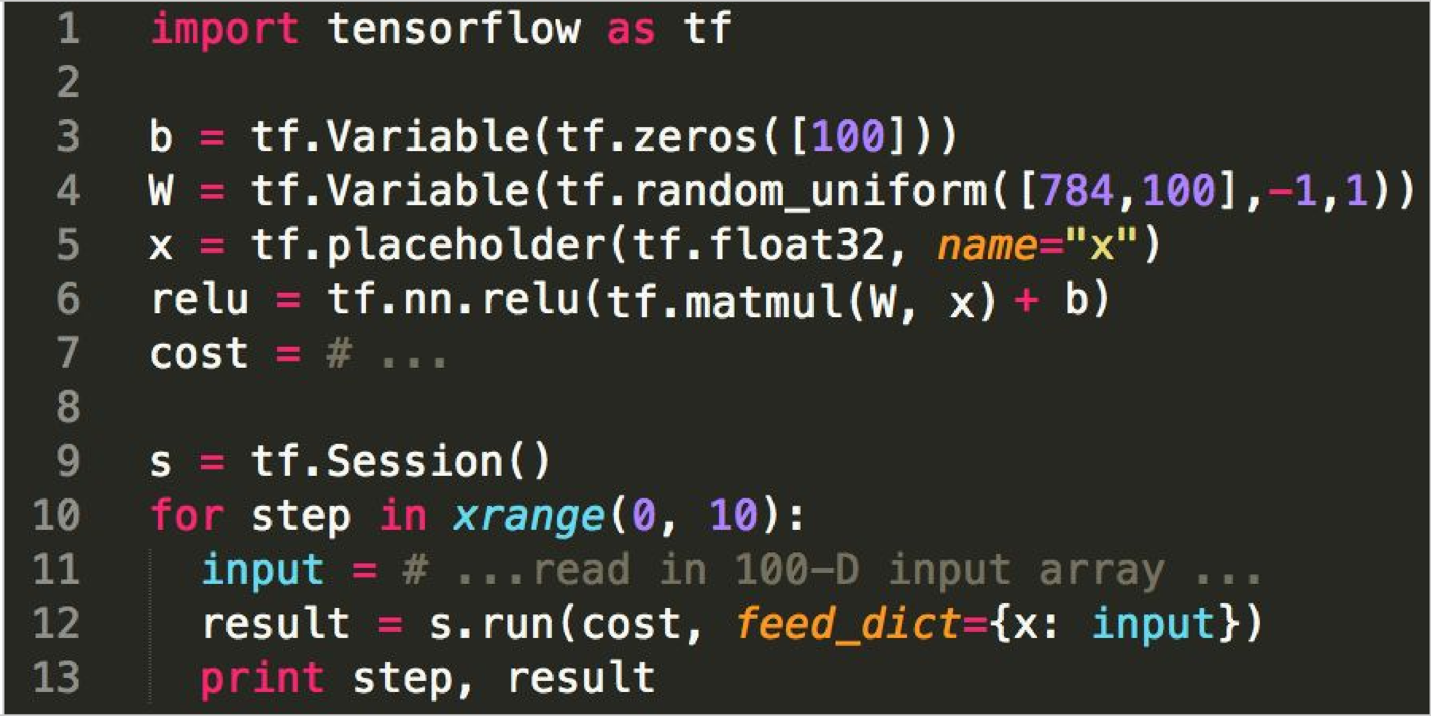
OpDef interface defined in math_ops.cc#L607



```
REGISTER_OP("MatMul")
    .Input("a: T")
    .Input("b: T")
    .Output("product: T")
    .Attr("transpose_a: bool = false")
    .Attr("transpose_b: bool = false")
    .Attr("T: {float, double, int32, complex64}")
```

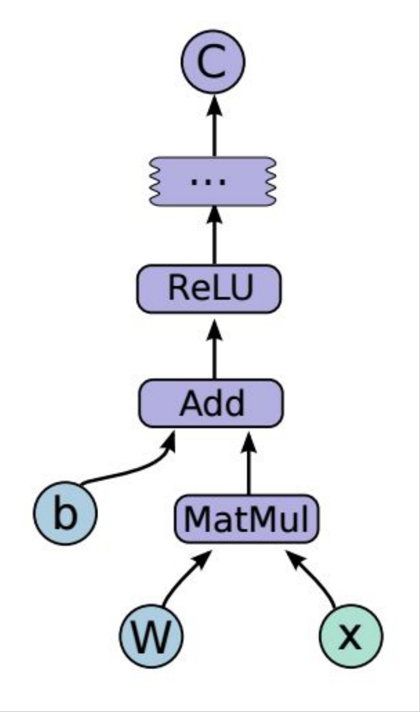
Expressing: Graph

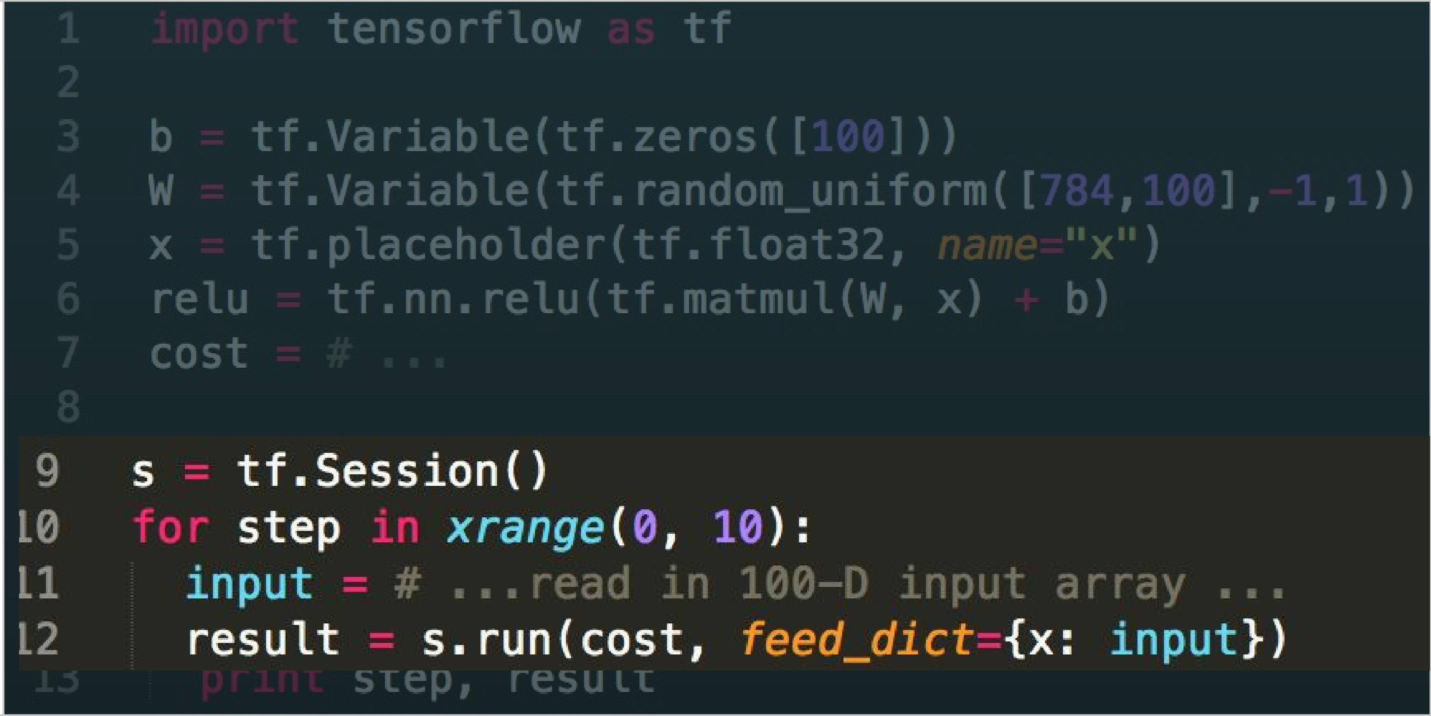




```python
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784,100],-1,1))
x = tf.placeholder(tf.float32, name="x")
relu = tf.nn.relu(tf.matmul(W, x) + b)
cost = # ...

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

Expressing: Graph



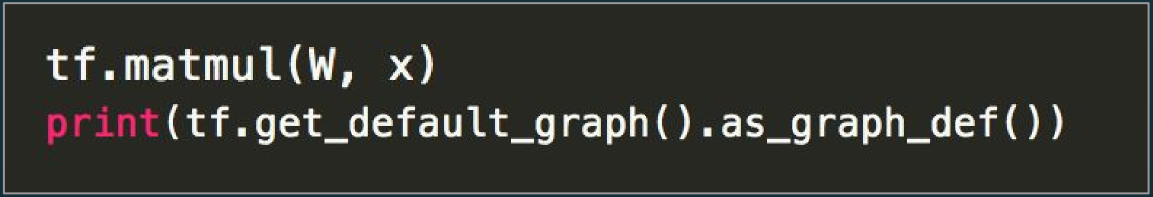


```python
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784,100],-1,1))
x = tf.placeholder(tf.float32, name="x")
relu = tf.nn.relu(tf.matmul(W, x) + b)
cost = # ...

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

Expressing: Graph

Graph is built implicitly

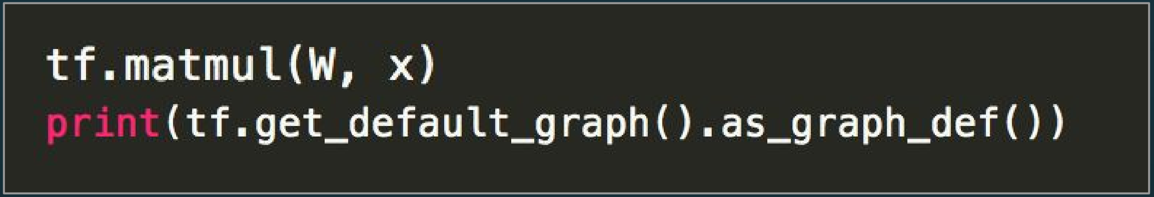
session.py#L896



```
tf.matmul(W, x)
print(tf.get_default_graph().as_graph_def())
```

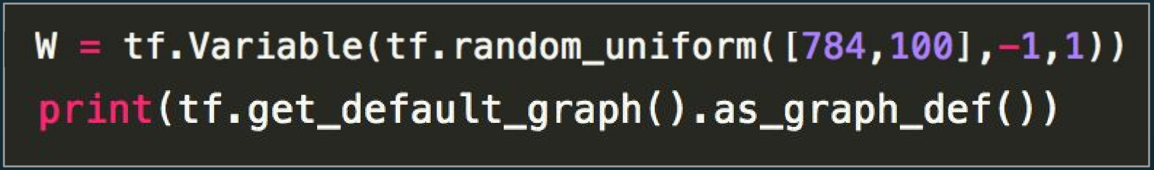
Expressing: Graph

Graph is built implicitly
[session.py#L896]



```
tf.matmul(W, x)
print(tf.get_default_graph().as_graph_def())
```

Variables add implicit ops
[variables.py#L146]



```
W = tf.Variable(tf.random_uniform([784,100],-1,1))
print(tf.get_default_graph().as_graph_def())
```

Expressing: Graph

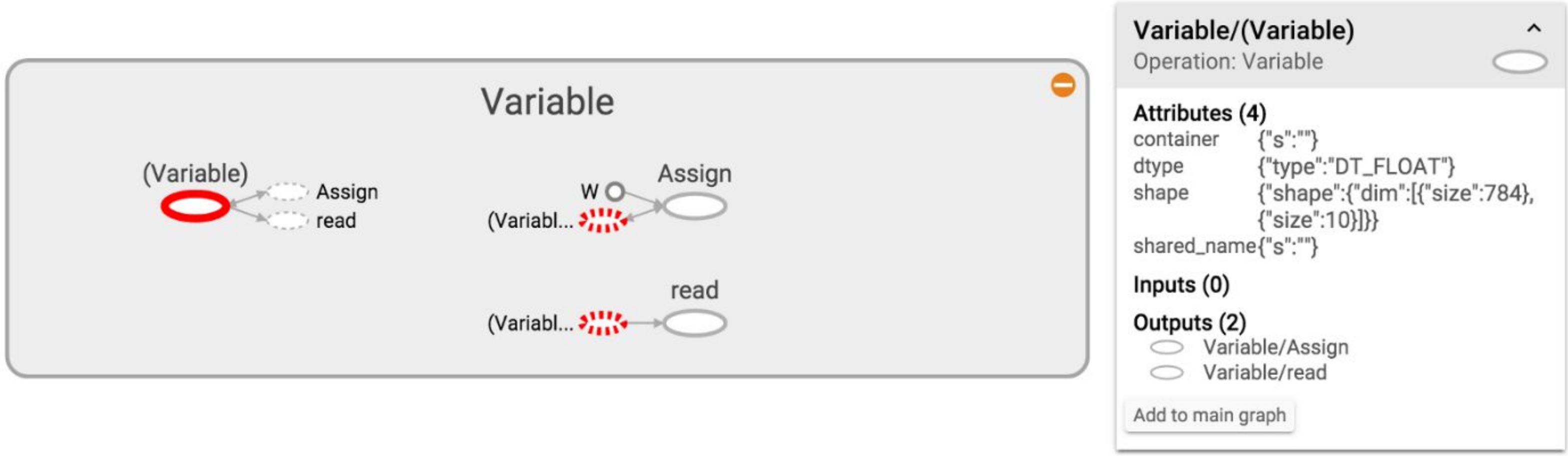
Graph is built implicitly
session.py#L896

```
tf.matmul(W, x)
print(tf.get_default_graph().as_graph_def())
```

Variables add implicit ops
variables.py#L146

```
W = tf.Variable(tf.random_uniform([784,100],-1,1))
print(tf.get_default_graph().as_graph_def())
```

In TensorBoard:



Variable/(Variable)
Operation: Variable

Attributes (4)
container {"s":""}
dtype {"type":"DT_FLOAT"}
shape {"shape":{"dim":[{"size":784},{"size":10}]}}
shared_name{"s":""}

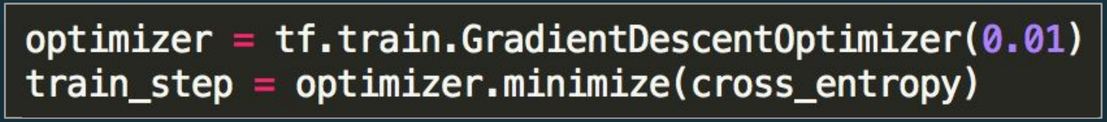
Inputs (0)

Outputs (2)
Variable/Assign
Variable/read

Expressing: Optimizers

Optimizer fns extend the graph

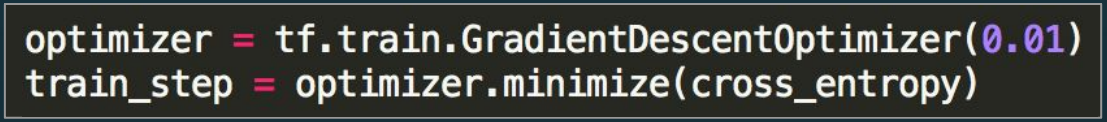
[optimizer.py:minimize#L155]



```
optimizer = tf.train.GradientDescentOptimizer(0.01)
train_step = optimizer.minimize(cross_entropy)
```

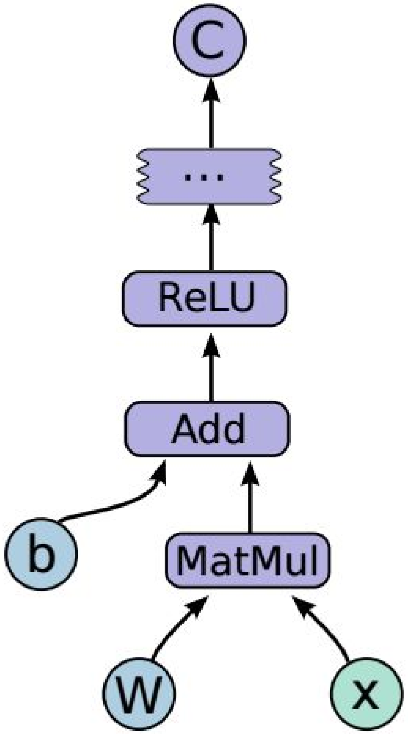
Expressing: Optimizers

Optimizer fns extend the graph
optimizer.py:minimize#L155



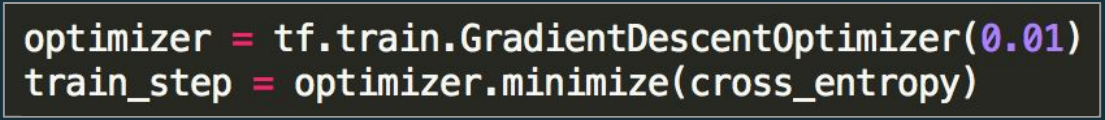
```
optimizer = tf.train.GradientDescentOptimizer(0.01)
train_step = optimizer.minimize(cross_entropy)
```

Trainable variables collected
variables.py#L258



Expressing: Optimizers

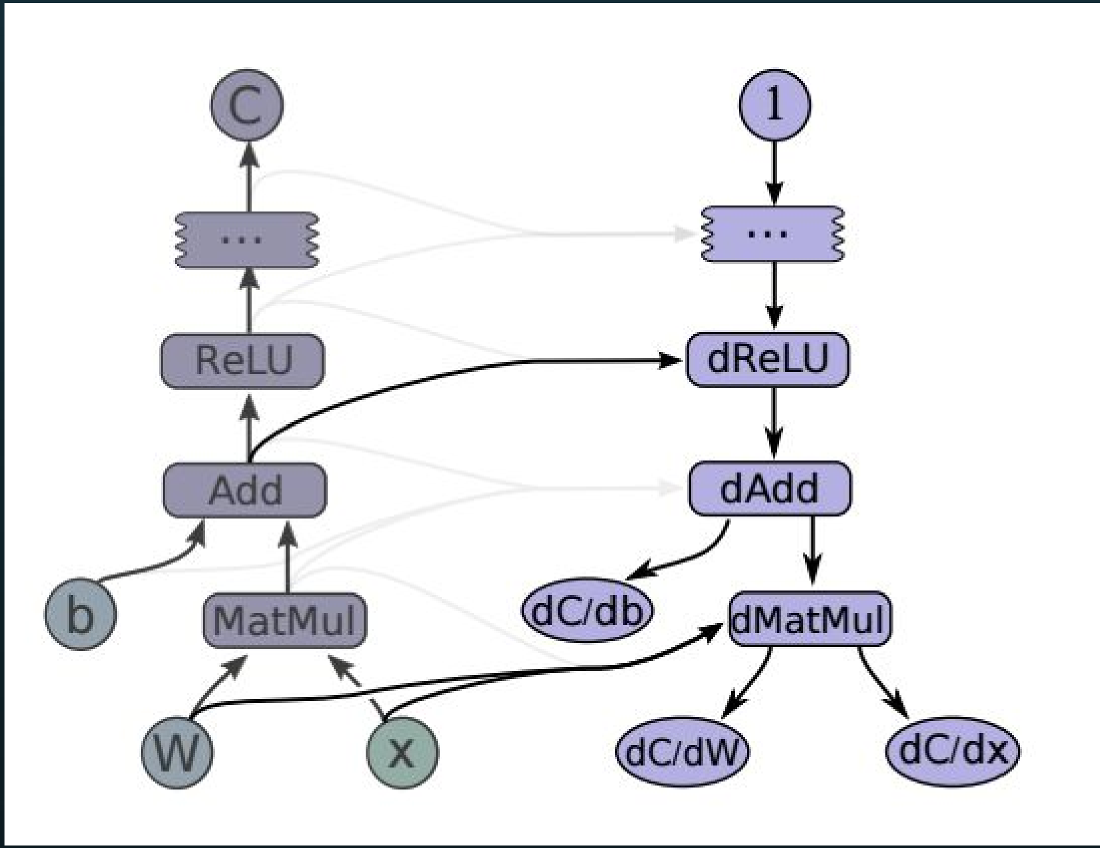
Optimizer fns extend the graph
optimizer.py:minimize#L155



```
optimizer = tf.train.GradientDescentOptimizer(0.01)
train_step = optimizer.minimize(cross_entropy)
```

Trainable variables collected
variables.py#L258

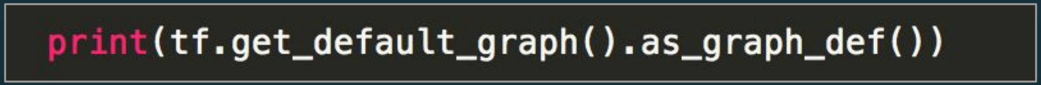
Graph is extended with gradients
gradients.py#L307



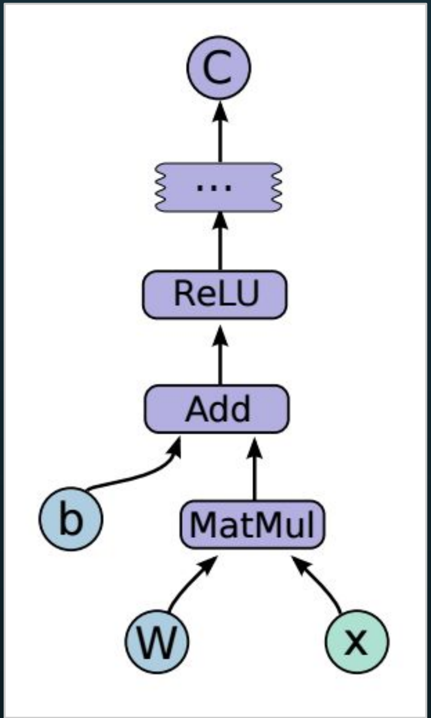
Expressing: Graph

Serialized as GraphDef

graph.proto



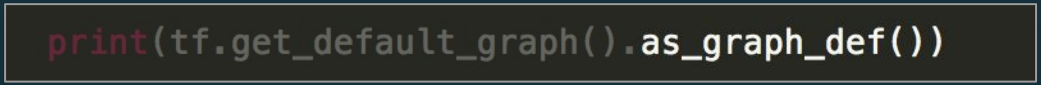
```
print(tf.get_default_graph().as_graph_def())
```



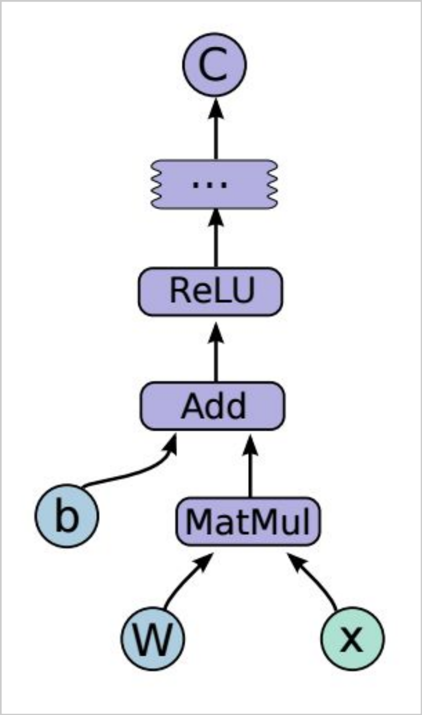
Expressing: Graph

Serialized as GraphDef

graph.proto



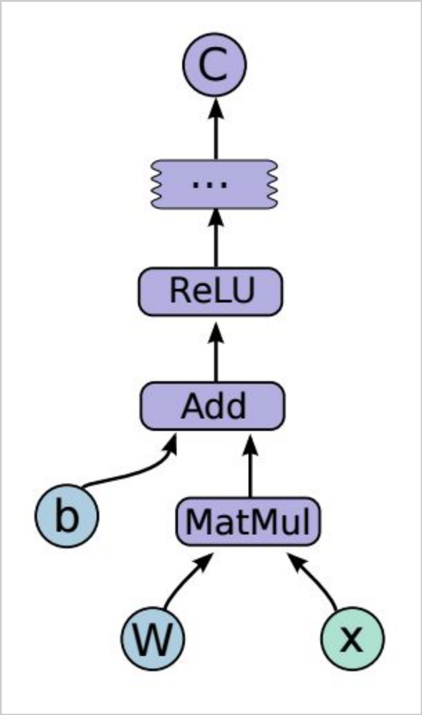
```
print(tf.get_default_graph().as_graph_def())
```



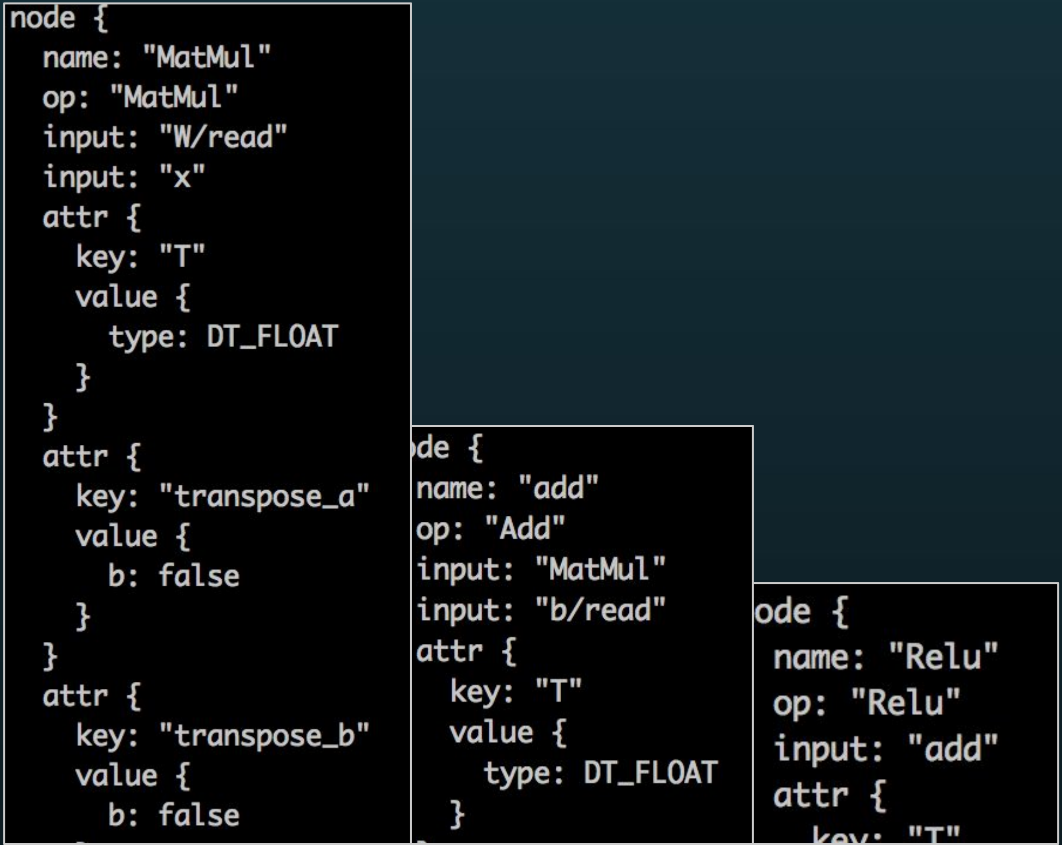
Expressing: Graph

Serialized as GraphDef

graph.proto



```
print(tf.get_default_graph().as_graph_def())
```



```
node {
  name: "MatMul"
  op: "MatMul"
  input: "W/read"
  input: "x"
  attr {
    key: "T"
    value {
      type: DT_FLOAT
    }
  }
  attr {
    key: "transpose_a"
    value {
      b: false
    }
  }
  attr {
    key: "transpose_b"
    value {
      b: false
```

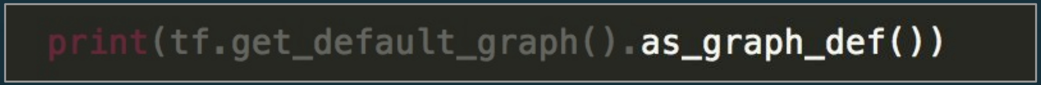
```
de {
name: "add"
op: "Add"
input: "MatMul"
input: "b/read"
attr {
  key: "T"
  value {
    type: DT_FLOAT
  }
```

```
ode {
 name: "Relu"
 op: "Relu"
 input: "add"
 attr {
   key: "T"
```

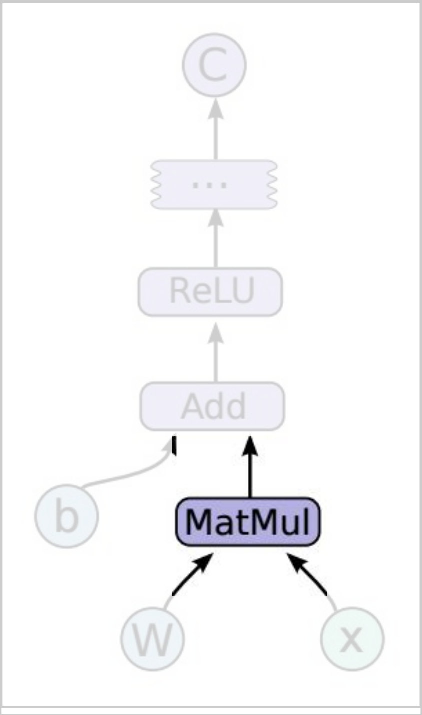
Expressing: Graph

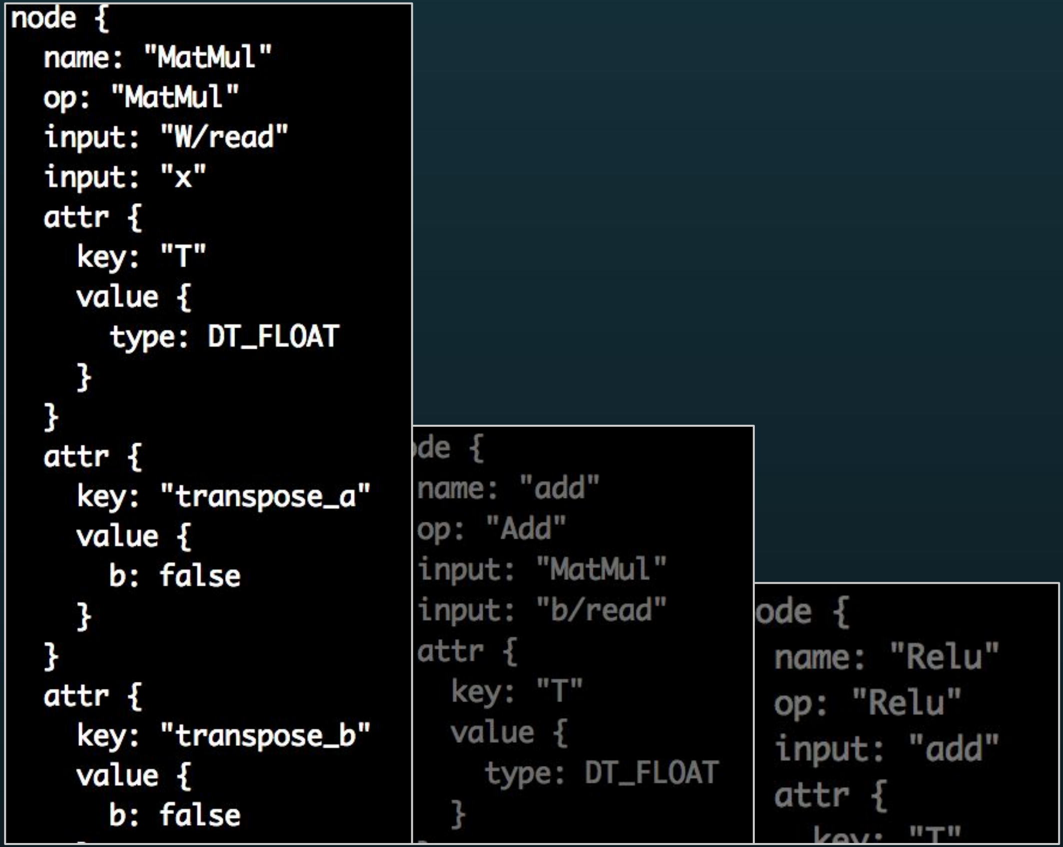
Serialized as GraphDef

graph.proto



```
print(tf.get_default_graph().as_graph_def())
```





```
node {
  name: "MatMul"
  op: "MatMul"
  input: "W/read"
  input: "x"
  attr {
    key: "T"
    value {
      type: DT_FLOAT
    }
  }
  attr {
    key: "transpose_a"
    value {
      b: false
    }
  }
  attr {
    key: "transpose_b"
    value {
      b: false
```

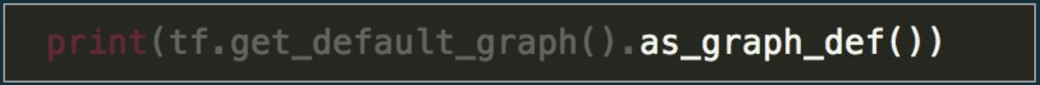
```
de {
name: "add"
op: "Add"
input: "MatMul"
input: "b/read"
attr {
  key: "T"
  value {
    type: DT_FLOAT
  }
```

```
ode {
 name: "Relu"
 op: "Relu"
 input: "add"
 attr {
   key: "T"
```

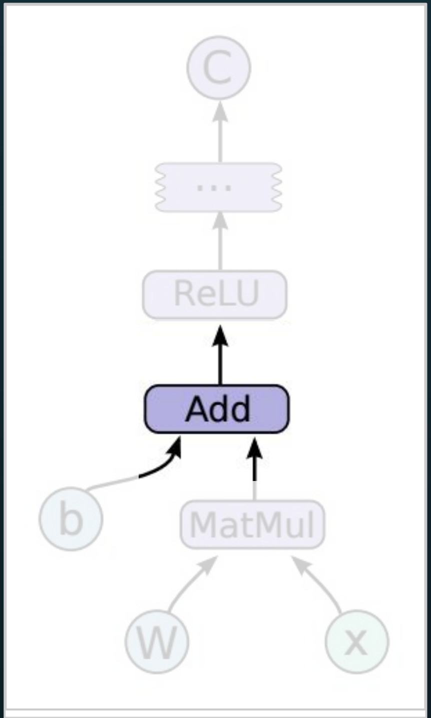
Expressing: Graph

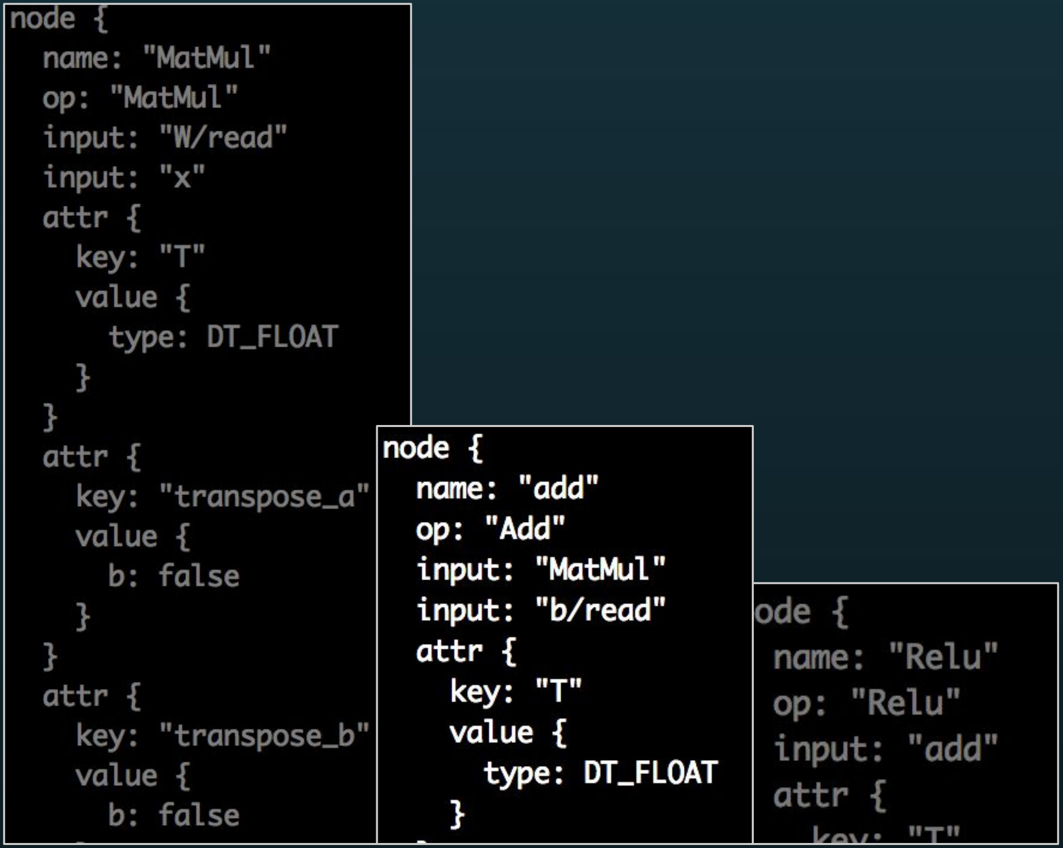
Serialized as GraphDef

graph.proto



```
print(tf.get_default_graph().as_graph_def())
```





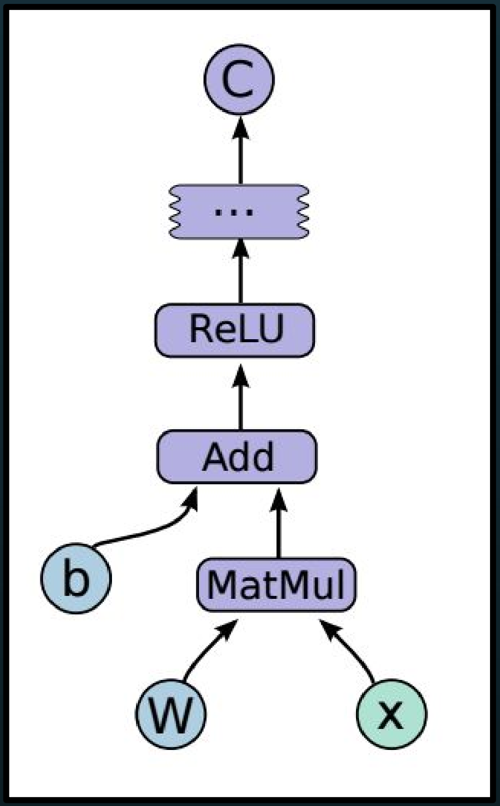
```
node {
  name: "MatMul"
  op: "MatMul"
  input: "W/read"
  input: "x"
  attr {
    key: "T"
    value {
      type: DT_FLOAT
    }
  }
  attr {
    key: "transpose_a"
    value {
      b: false
    }
  }
  attr {
    key: "transpose_b"
    value {
      b: false
```

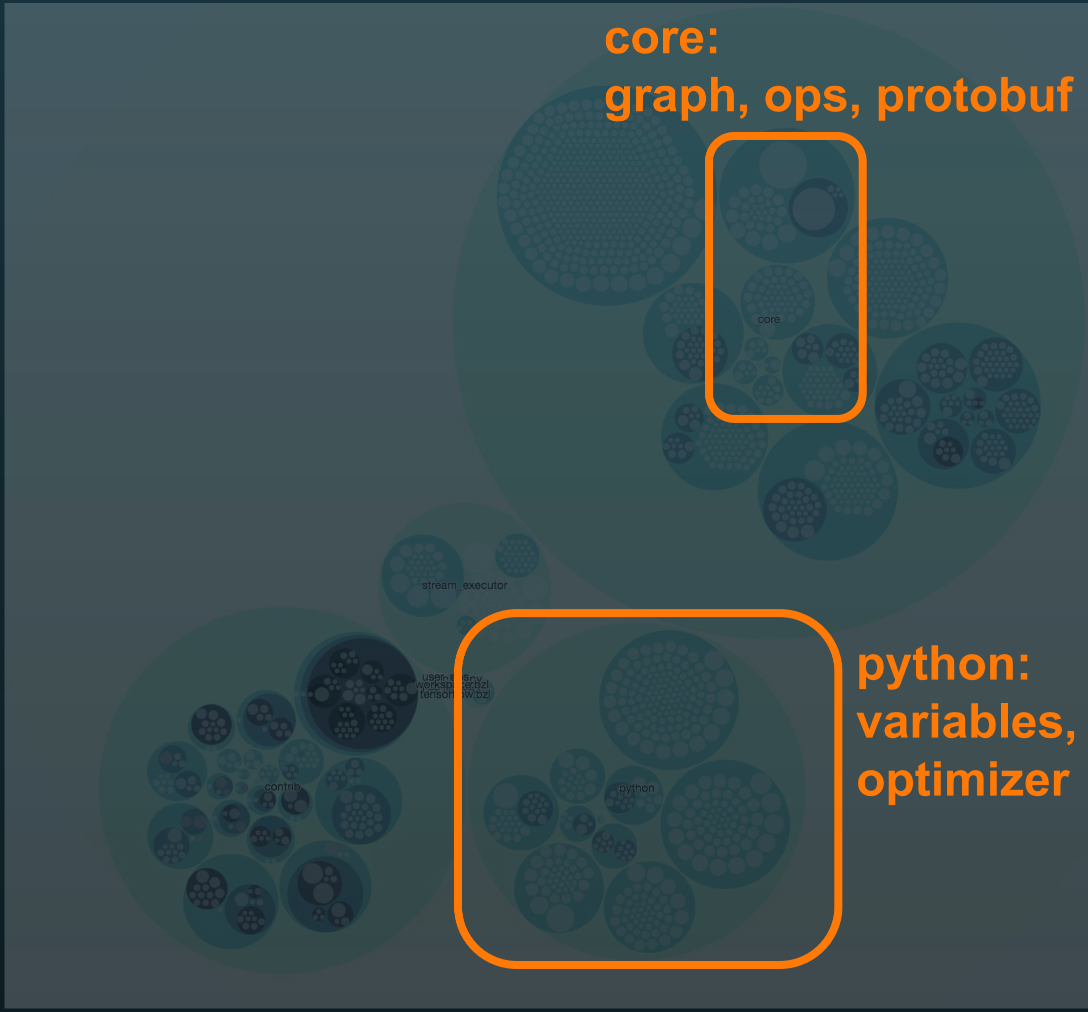
```
node {
  name: "add"
  op: "Add"
  input: "MatMul"
  input: "b/read"
  attr {
    key: "T"
    value {
      type: DT_FLOAT
    }
```

```
ode {
 name: "Relu"
 op: "Relu"
 input: "add"
 attr {
   key: "T"
```

A tour through the TensorFlow codebase

1. **Expressing** graphs





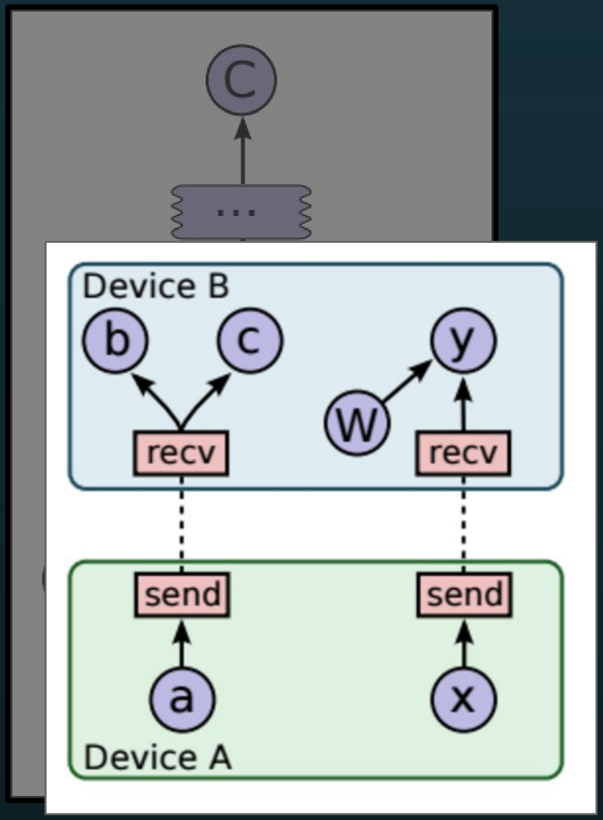
**core:
graph, ops, protobuf**

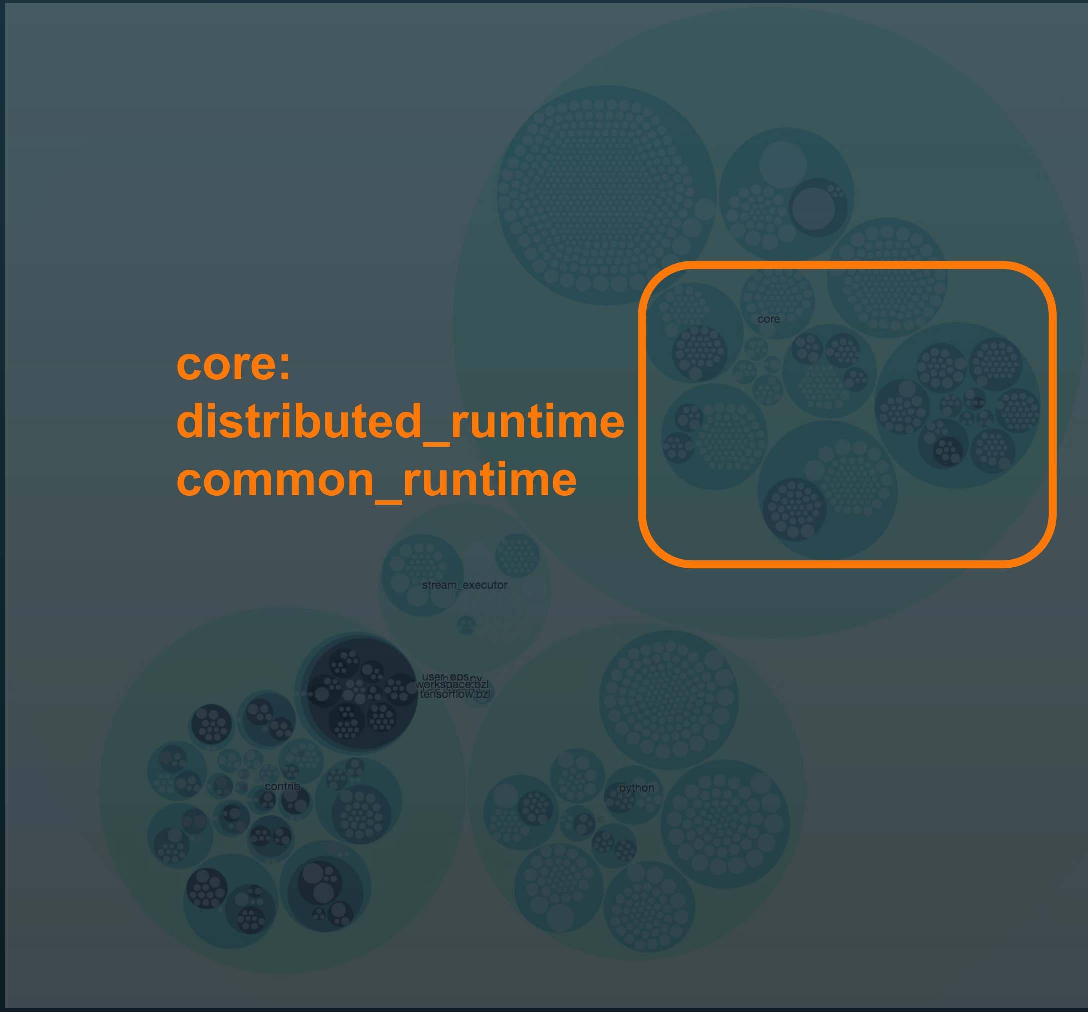
**python:
variables,
optimizer**

http://public.kevinrobinsonblog.com/tensorflow-codebase/

A tour through the TensorFlow codebase

2. **Distributing** graphs





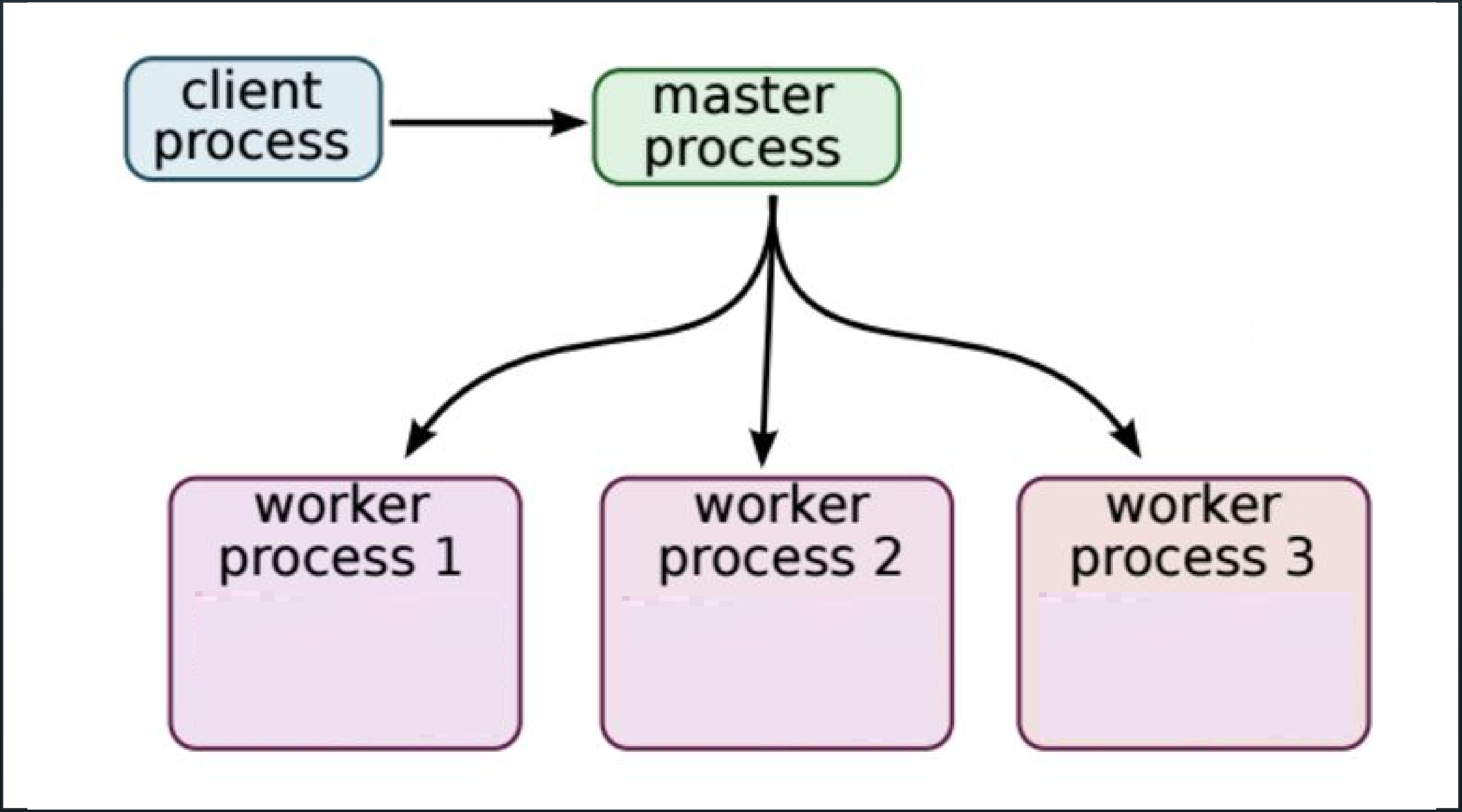
core:
distributed_runtime
common_runtime

http://public.kevinrobinsonblog.com/tensorflow-codebase/

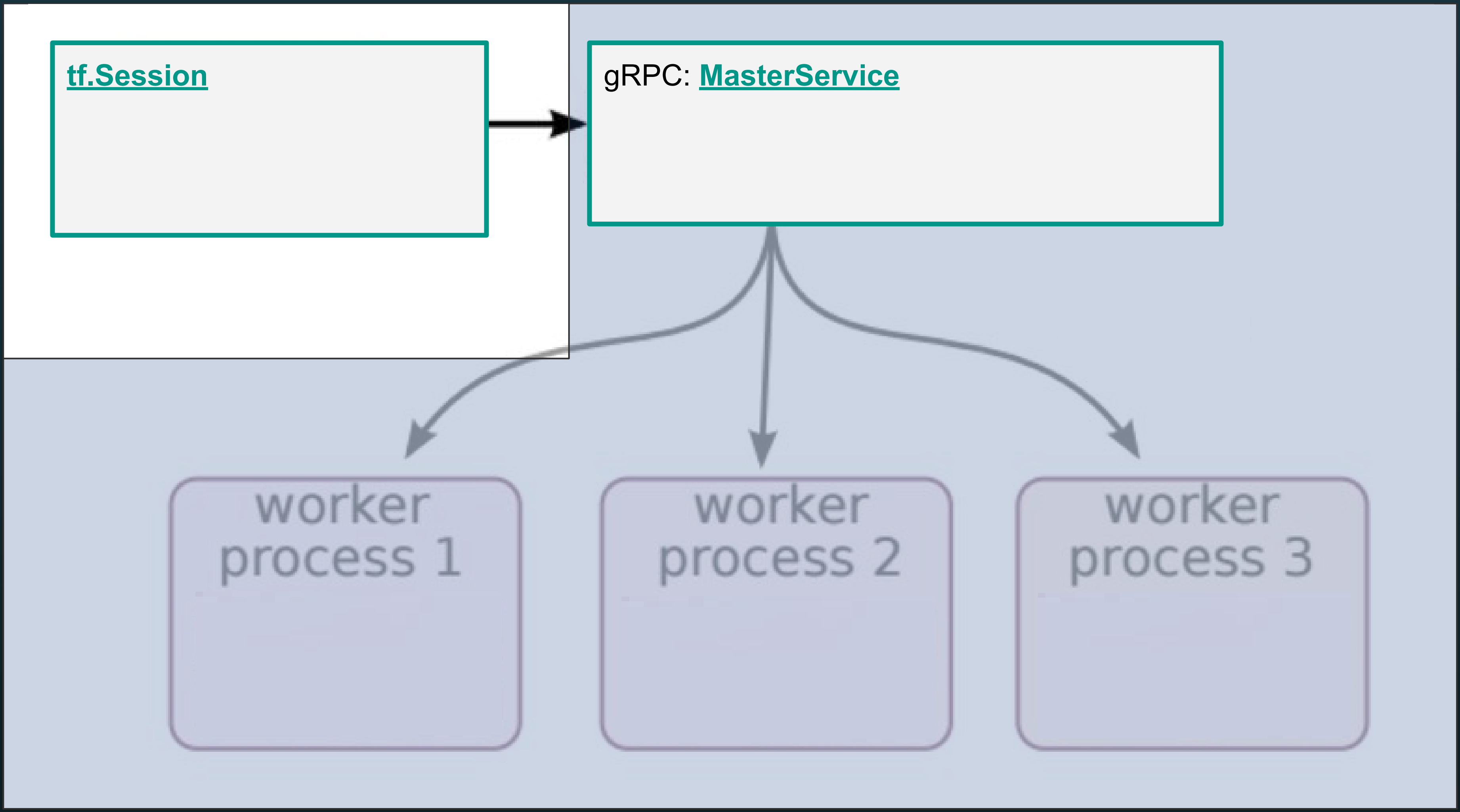
Distributing

- Sessions in distributed runtime
- Pruning
- Placing and Partitioning

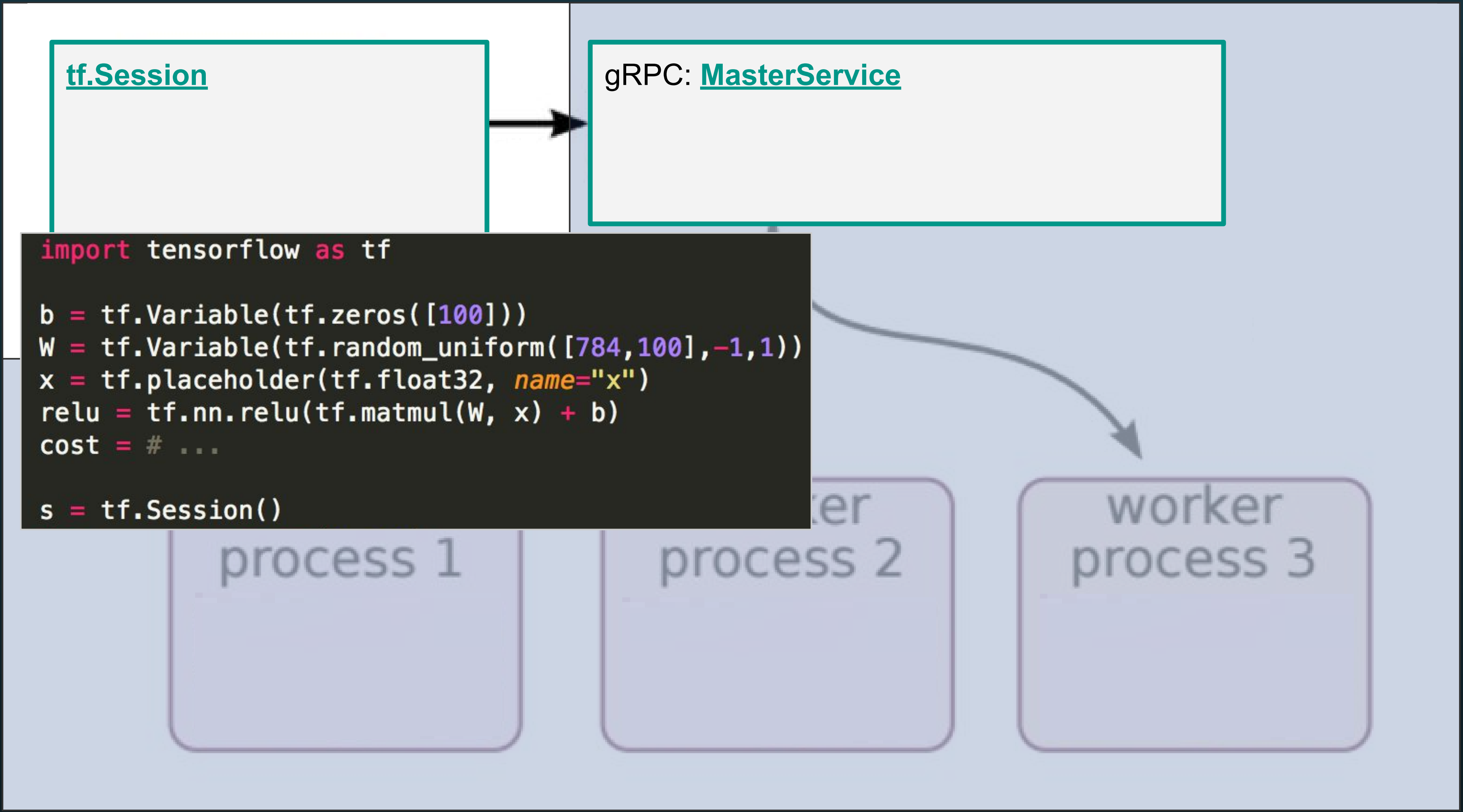
Distributing: Creating a session



Distributing: Creating a session

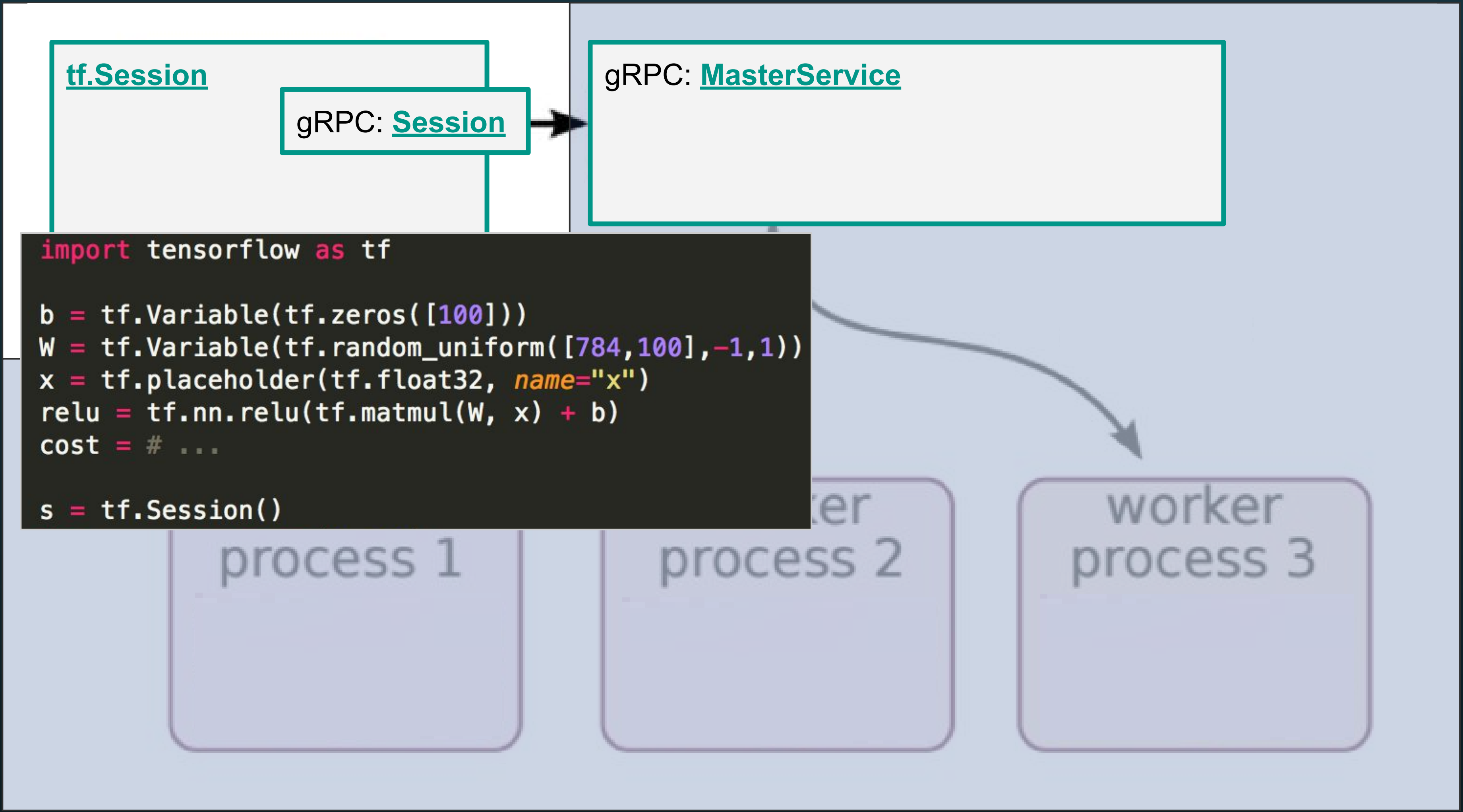


Distributing: Creating a session



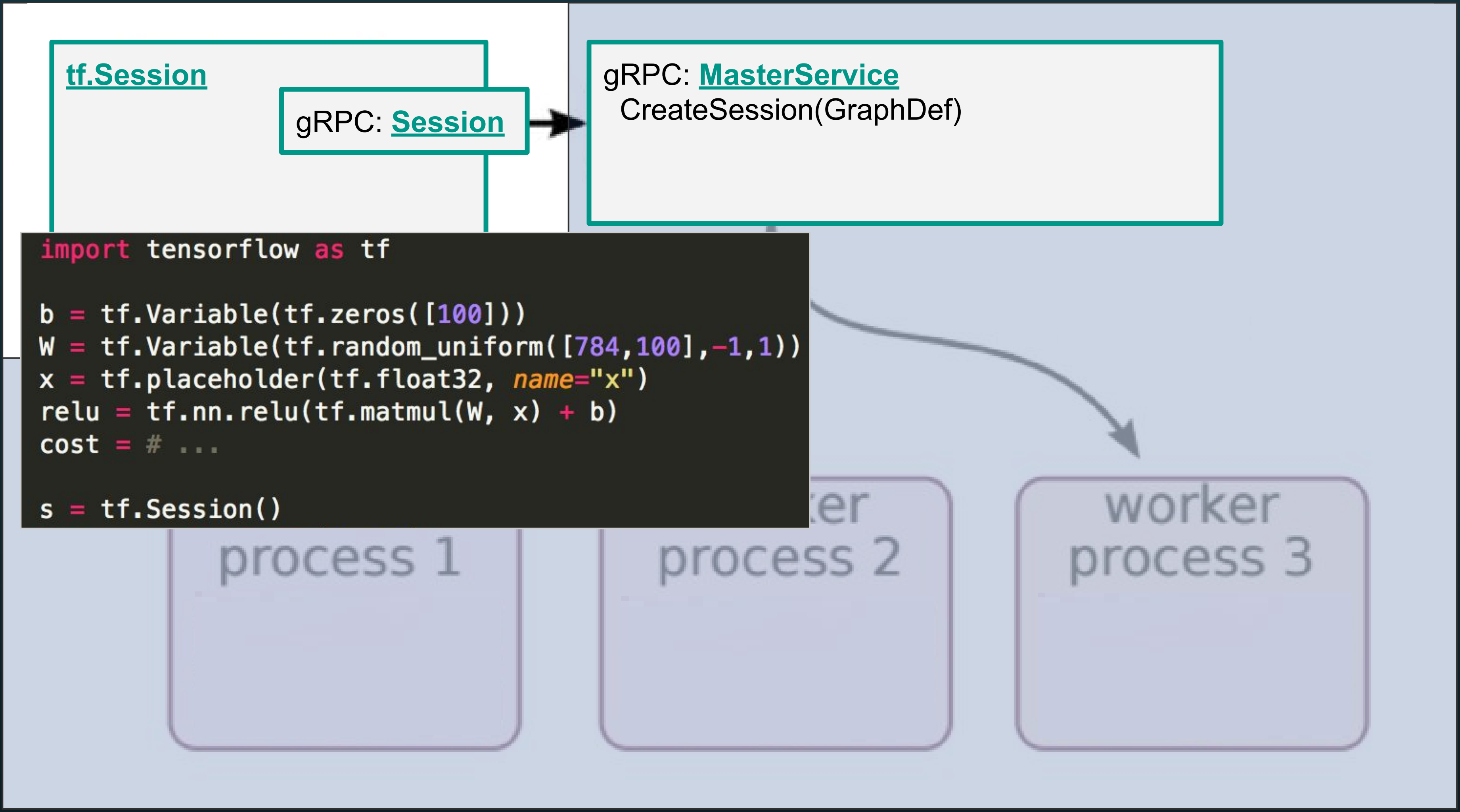
```python
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784,100],-1,1))
x = tf.placeholder(tf.float32, name="x")
relu = tf.nn.relu(tf.matmul(W, x) + b)
cost = # ...

s = tf.Session()
```

Distributing: Creating a session



```python
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, name="x")
relu = tf.nn.relu(tf.matmul(W, x) + b)
cost = # ...

s = tf.Session()
```

Distributing: Creating a session



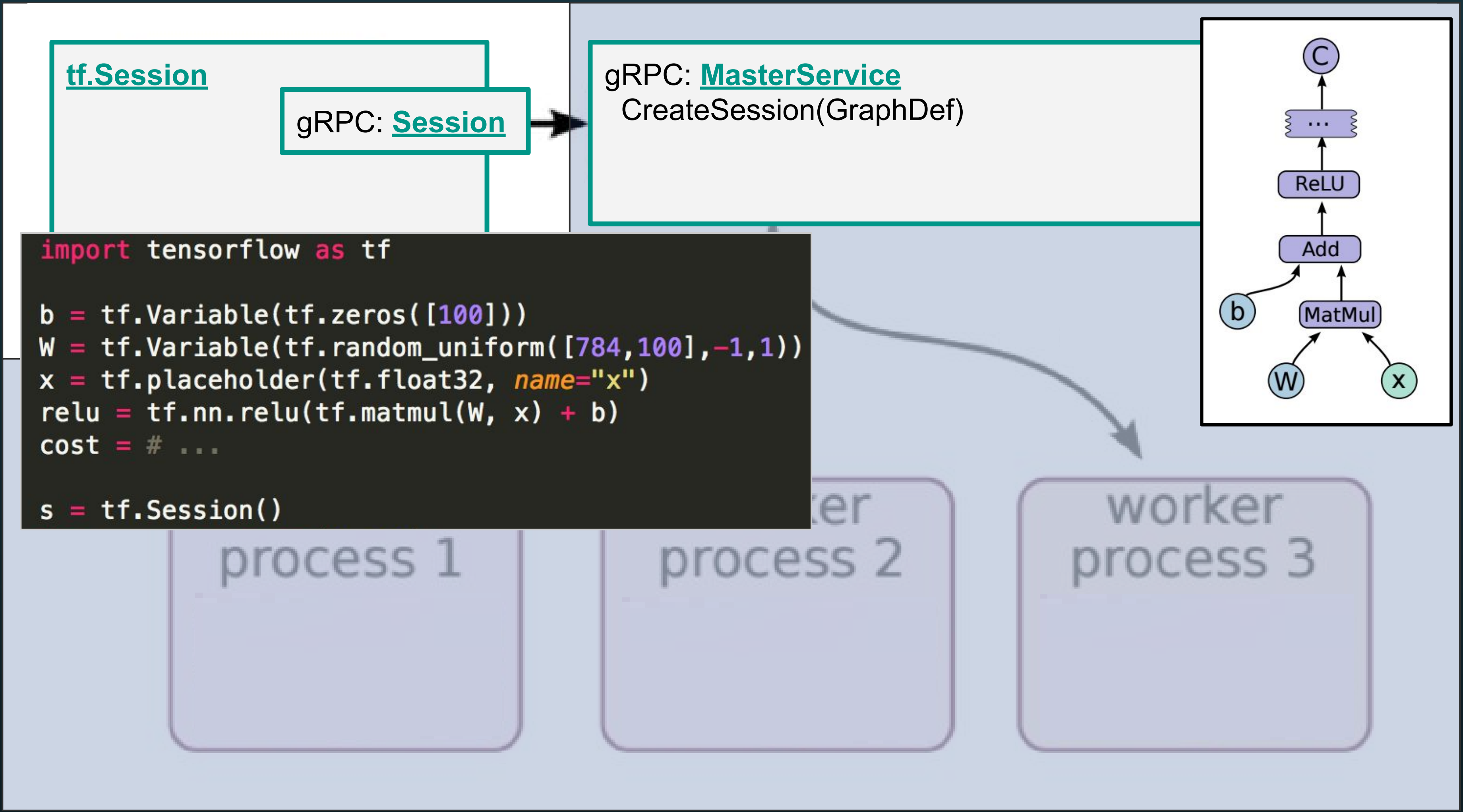
tf.Session

gRPC: Session

gRPC: MasterService
CreateSession(GraphDef)

```python
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, name="x")
relu = tf.nn.relu(tf.matmul(W, x) + b)
cost = # ...

s = tf.Session()
```

Distributing: Creating a session



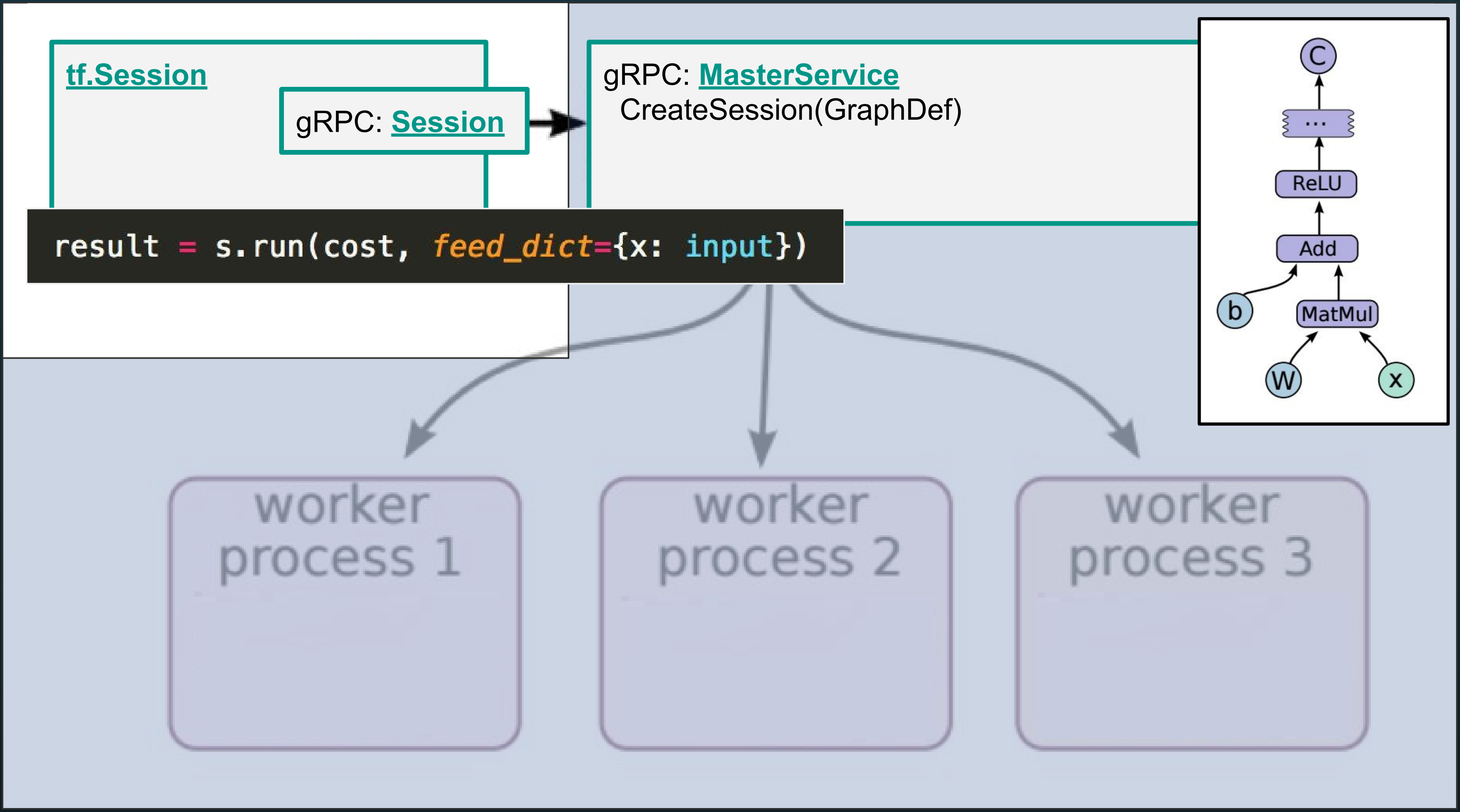
tf.Session

gRPC: Session

gRPC: MasterService
CreateSession(GraphDef)

```
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, name="x")
relu = tf.nn.relu(tf.matmul(W, x) + b)
cost = # ...

s = tf.Session()
```

Distributing: Running a session



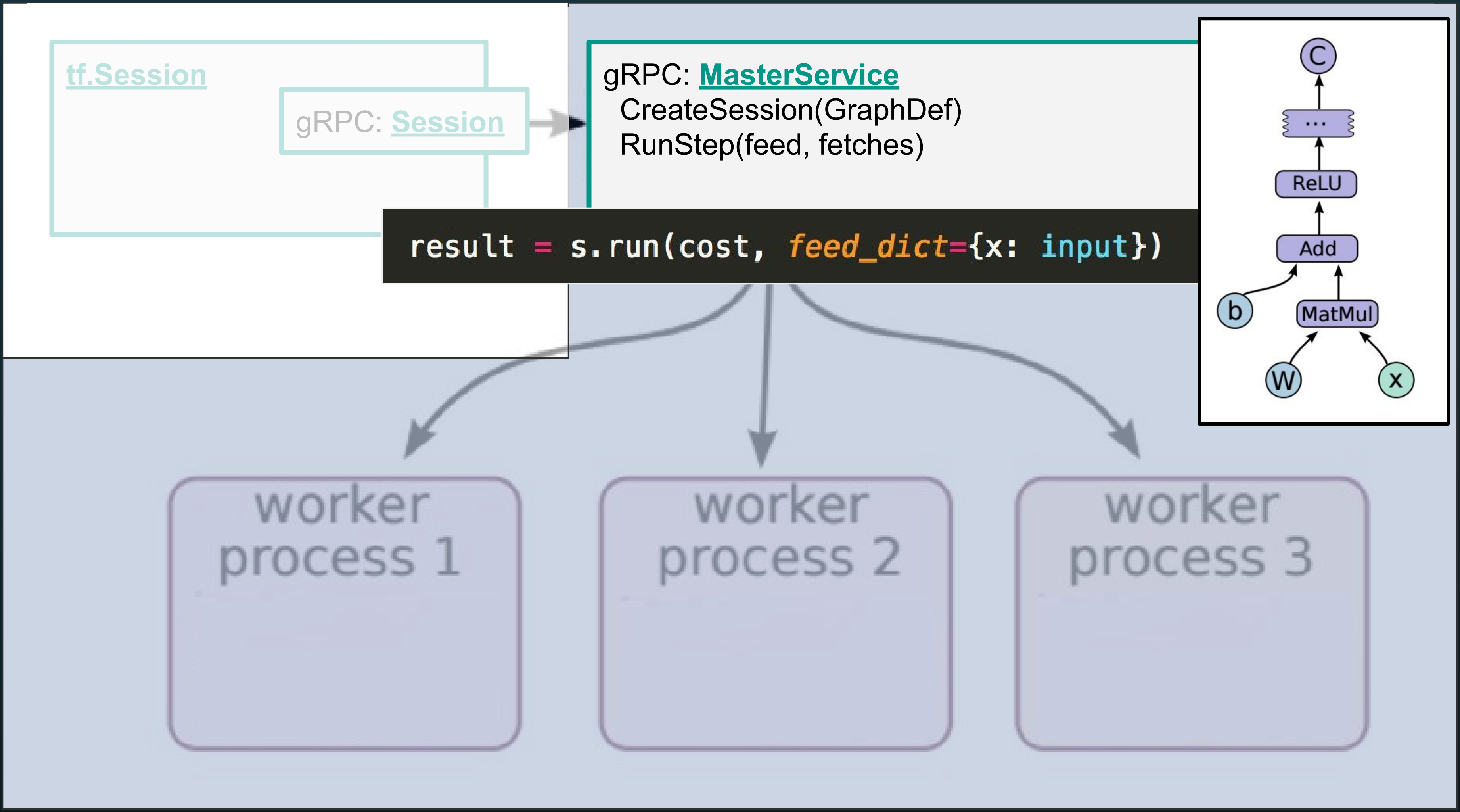
tf.Session

gRPC: Session

gRPC: MasterService
CreateSession(GraphDef)

```
result = s.run(cost, feed_dict={x: input})
```

Distributing: Running a session



tf.Session

gRPC: Session

gRPC: MasterService
  CreateSession(GraphDef)
  RunStep(feed, fetches)

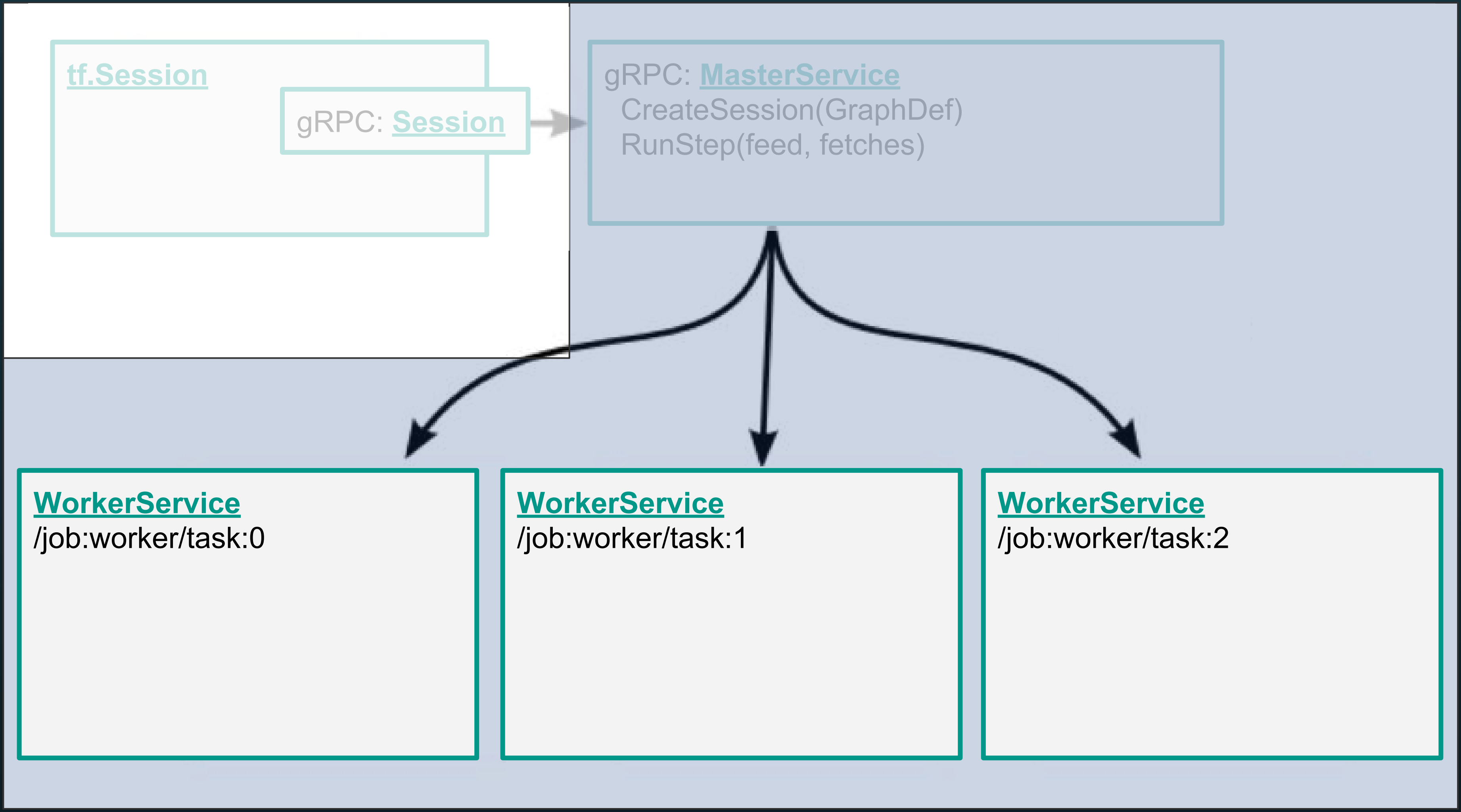
```
result = s.run(cost, feed_dict={x: input})
```

worker process 1

worker process 2

worker process 3

Distributing: Running a session



tf.Session

gRPC: Session

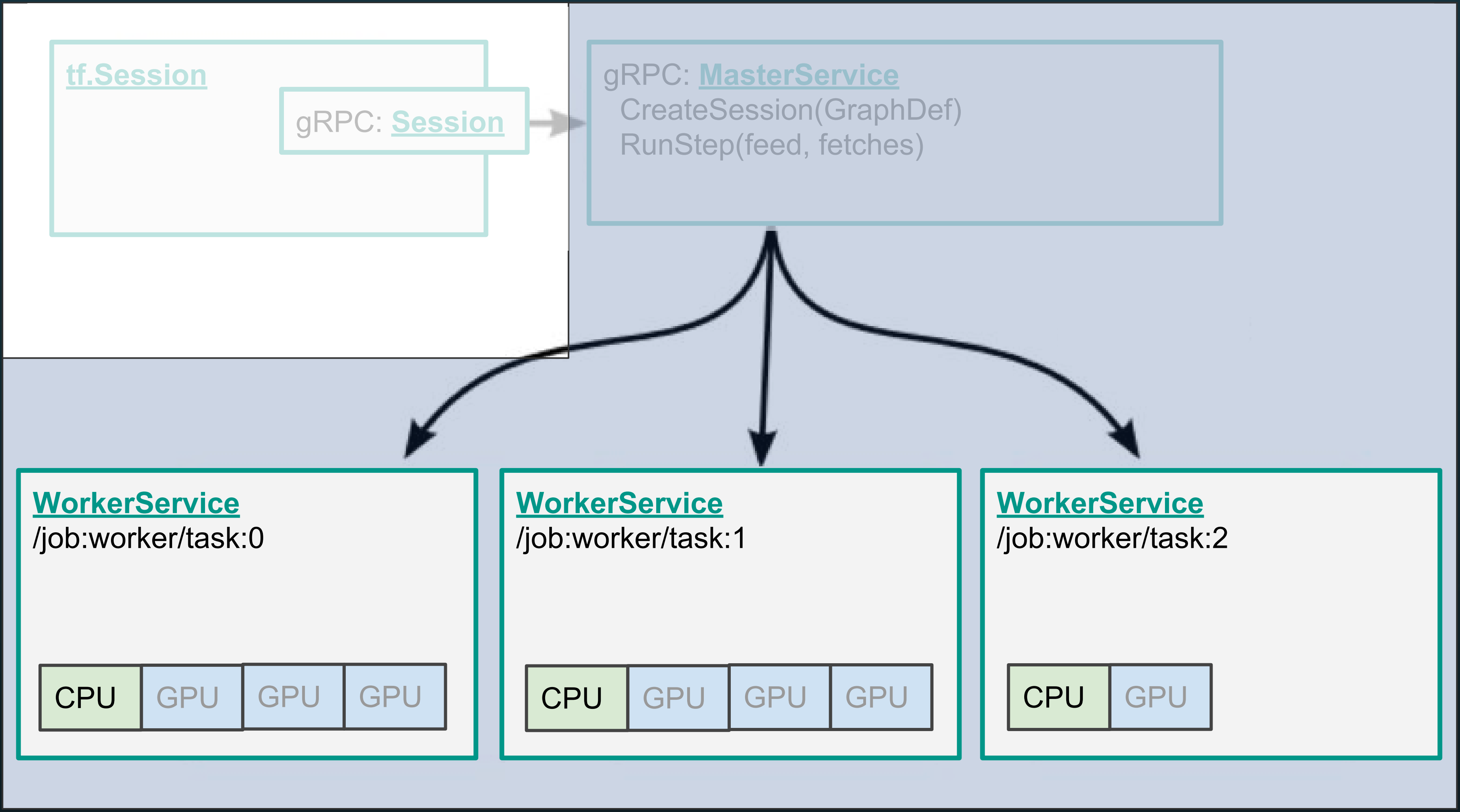
gRPC: MasterService
  CreateSession(GraphDef)
  RunStep(feed, fetches)

WorkerService
/job:worker/task:0

WorkerService
/job:worker/task:1

WorkerService
/job:worker/task:2

Distributing: Running a session



tf.Session

gRPC: Session

gRPC: MasterService
CreateSession(GraphDef)
RunStep(feed, fetches)

WorkerService
/job:worker/task:0

CPU	GPU	GPU	GPU

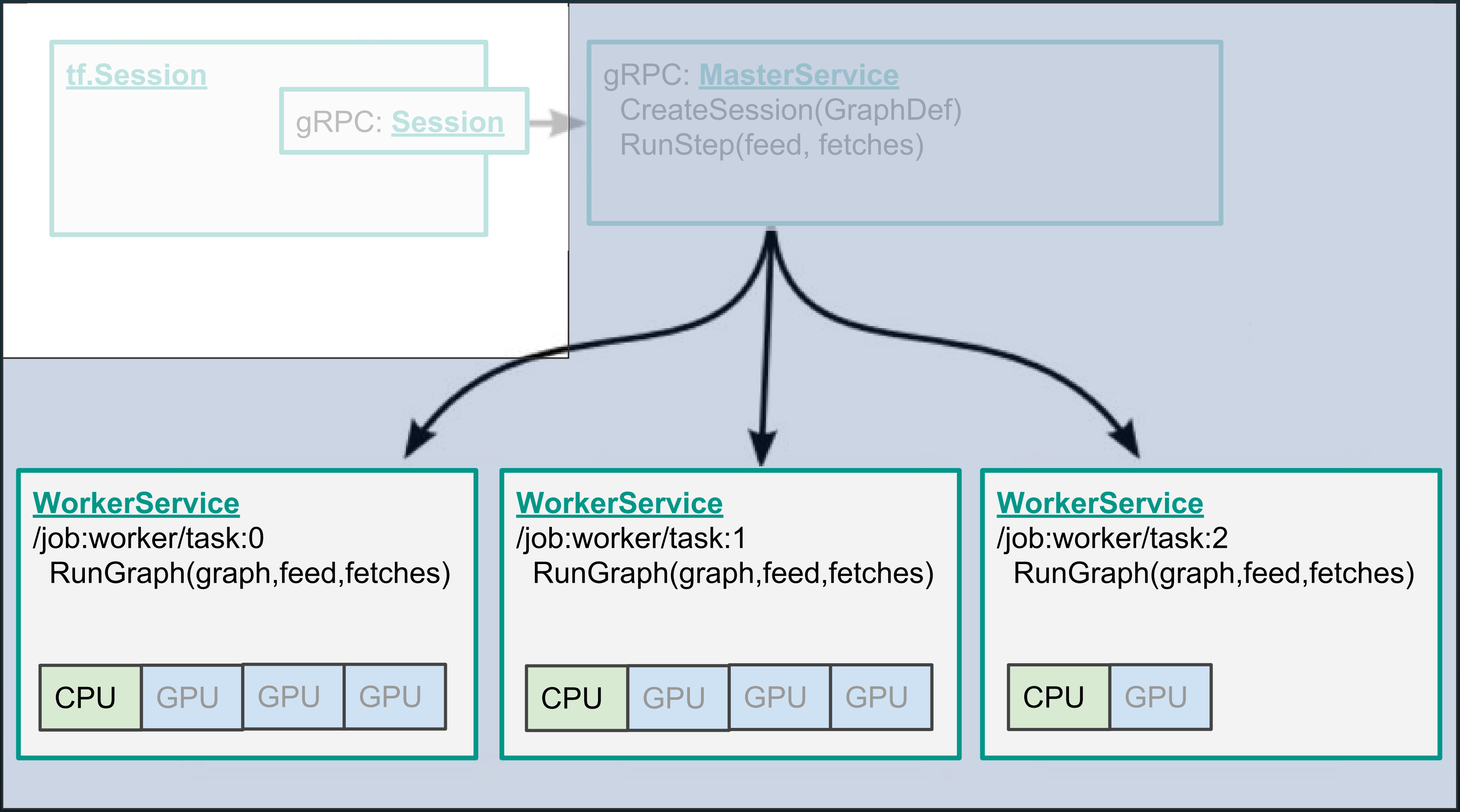
WorkerService
/job:worker/task:1

CPU	GPU	GPU	GPU

WorkerService
/job:worker/task:2

CPU	GPU

Distributing: Running a session



tf.Session

gRPC: Session

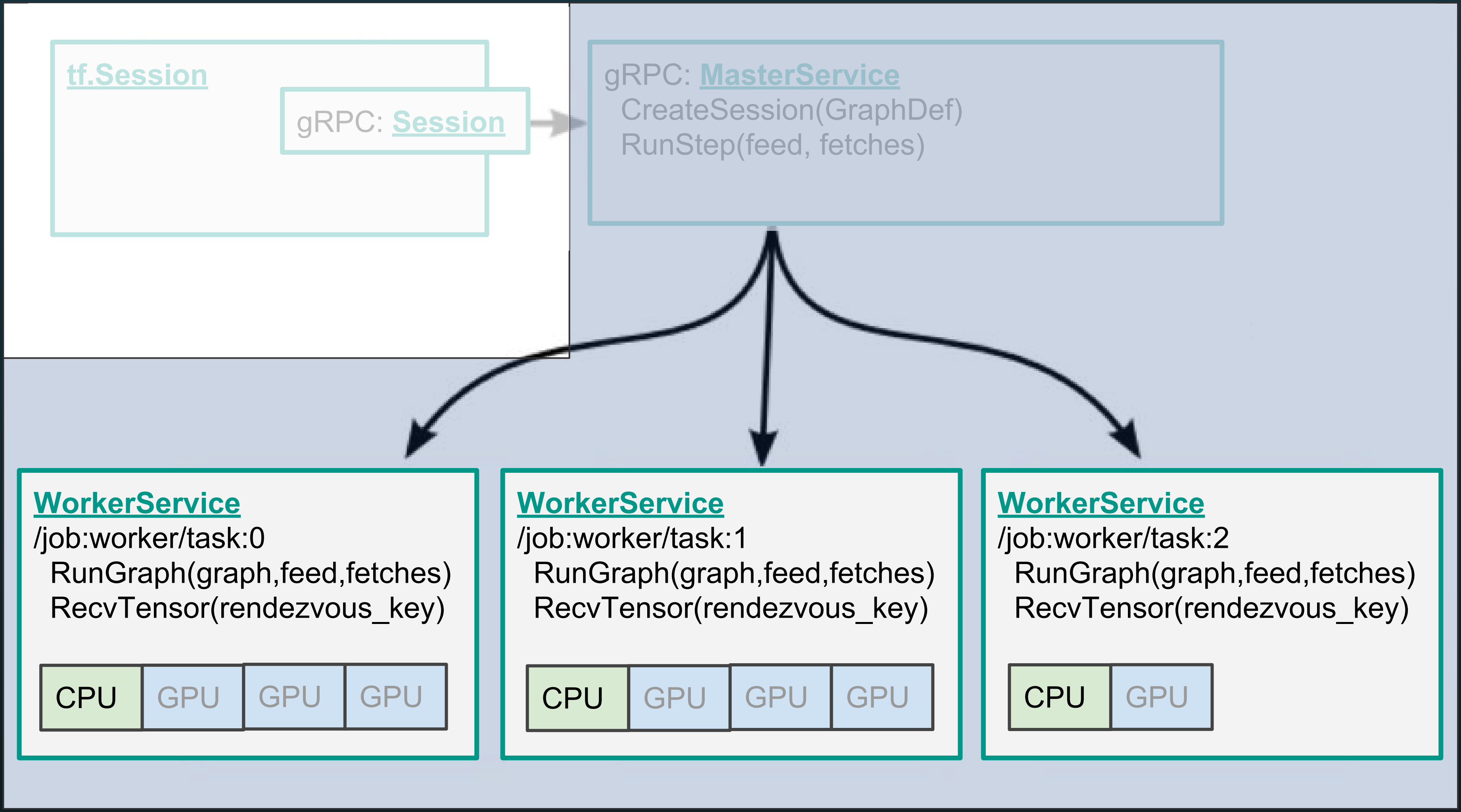
gRPC: MasterService
  CreateSession(GraphDef)
  RunStep(feed, fetches)

WorkerService
/job:worker/task:0
  RunGraph(graph,feed,fetches)

WorkerService
/job:worker/task:1
  RunGraph(graph,feed,fetches)

WorkerService
/job:worker/task:2
  RunGraph(graph,feed,fetches)

Distributing: Running a session



tf.Session

gRPC: Session

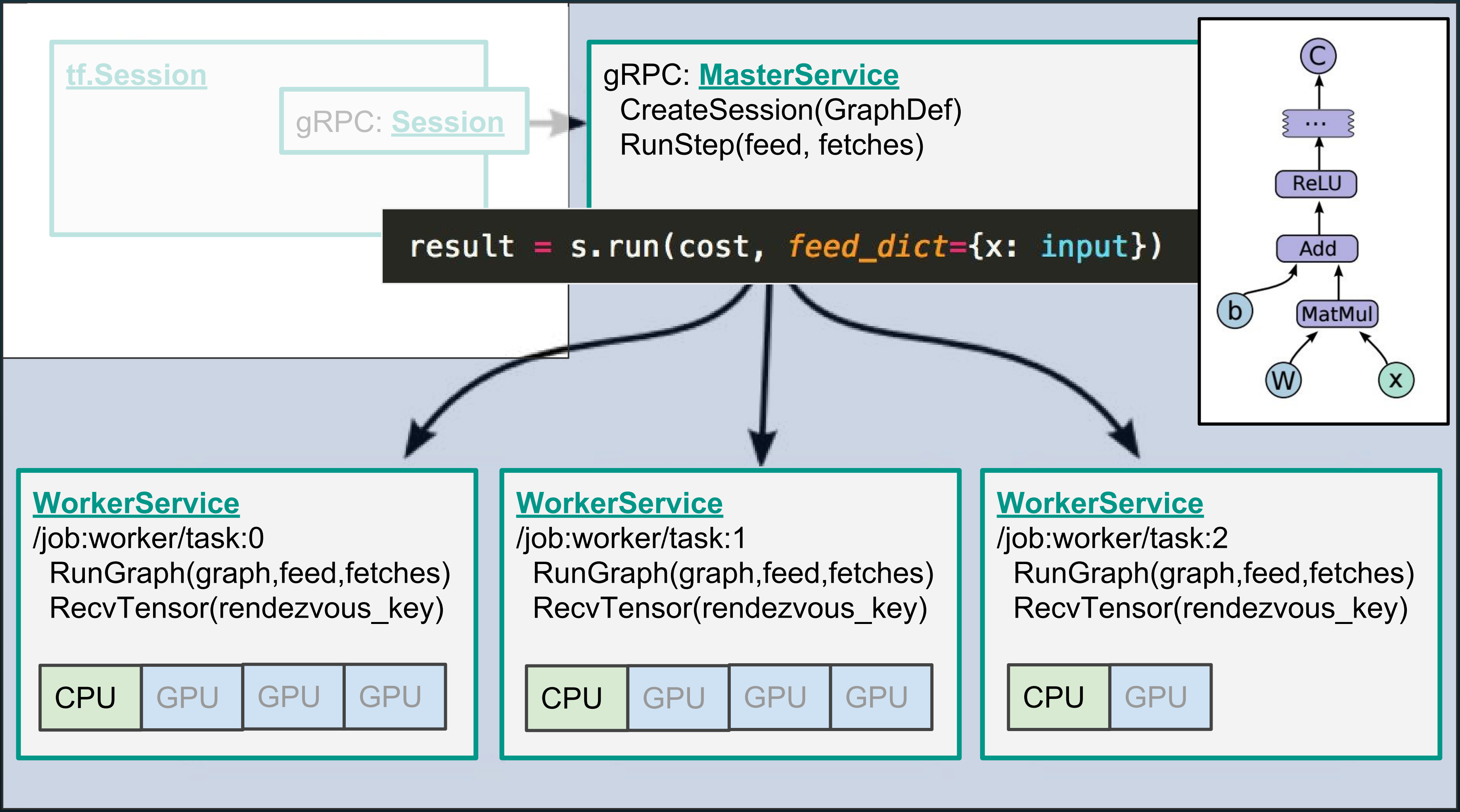
gRPC: MasterService
  CreateSession(GraphDef)
  RunStep(feed, fetches)

WorkerService
/job:worker/task:0
  RunGraph(graph,feed,fetches)
  RecvTensor(rendezvous_key)

WorkerService
/job:worker/task:1
  RunGraph(graph,feed,fetches)
  RecvTensor(rendezvous_key)

WorkerService
/job:worker/task:2
  RunGraph(graph,feed,fetches)
  RecvTensor(rendezvous_key)

Distributing: Running a session



tf.Session

gRPC: Session

gRPC: MasterService
  CreateSession(GraphDef)
  RunStep(feed, fetches)

```
result = s.run(cost, feed_dict={x: input})
```

WorkerService
/job:worker/task:0
  RunGraph(graph,feed,fetches)
  RecvTensor(rendezvous_key)

CPU GPU GPU GPU

WorkerService
/job:worker/task:1
  RunGraph(graph,feed,fetches)
  RecvTensor(rendezvous_key)

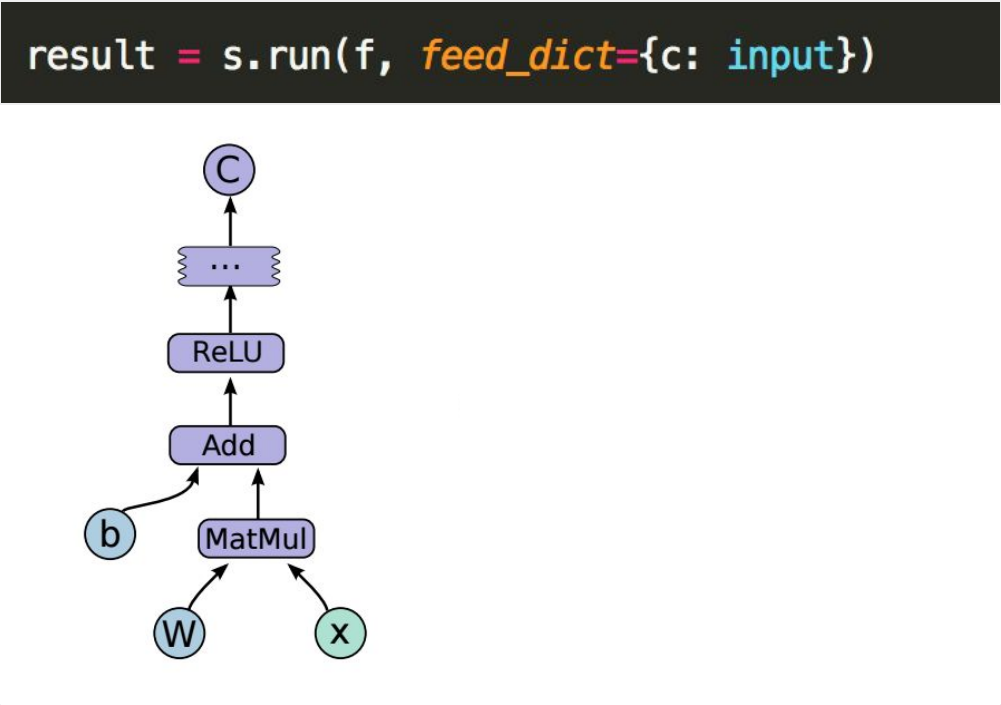
CPU GPU GPU GPU

WorkerService
/job:worker/task:2
  RunGraph(graph,feed,fetches)
  RecvTensor(rendezvous_key)

CPU GPU

Distributing: Pruning

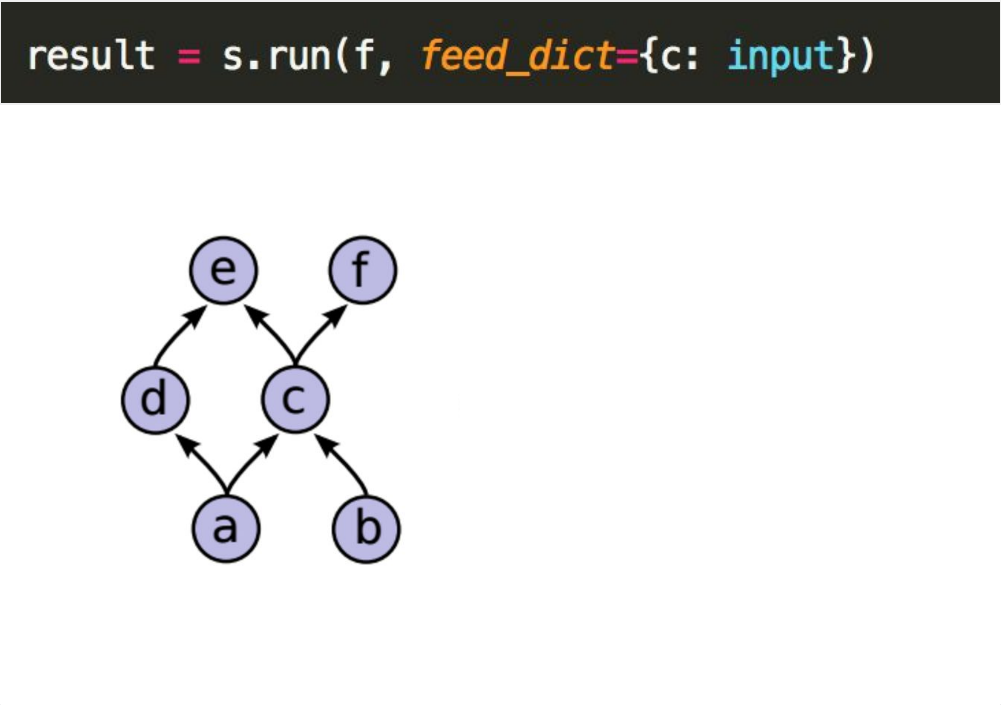
gRPC call to **Session::Run**
in master_session.cc#L835



```
result = s.run(f, feed_dict={c: input})
```

Distributing: Pruning

gRPC call to **Session::Run**
in [master_session.cc#L835]

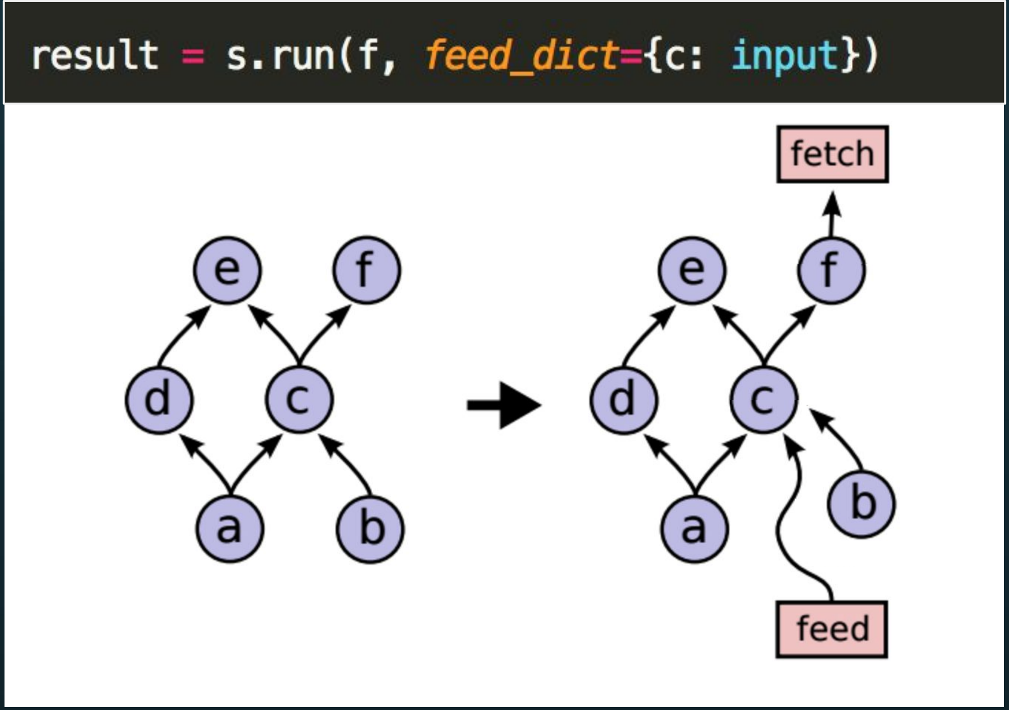


```
result = s.run(f, feed_dict={c: input})
```

Distributing: Pruning

gRPC call to **Session::Run**
in master_session.cc#L835

Rewrite with feed and fetch
RewriteGraphForExecution
in graph/subgraph.cc#L225



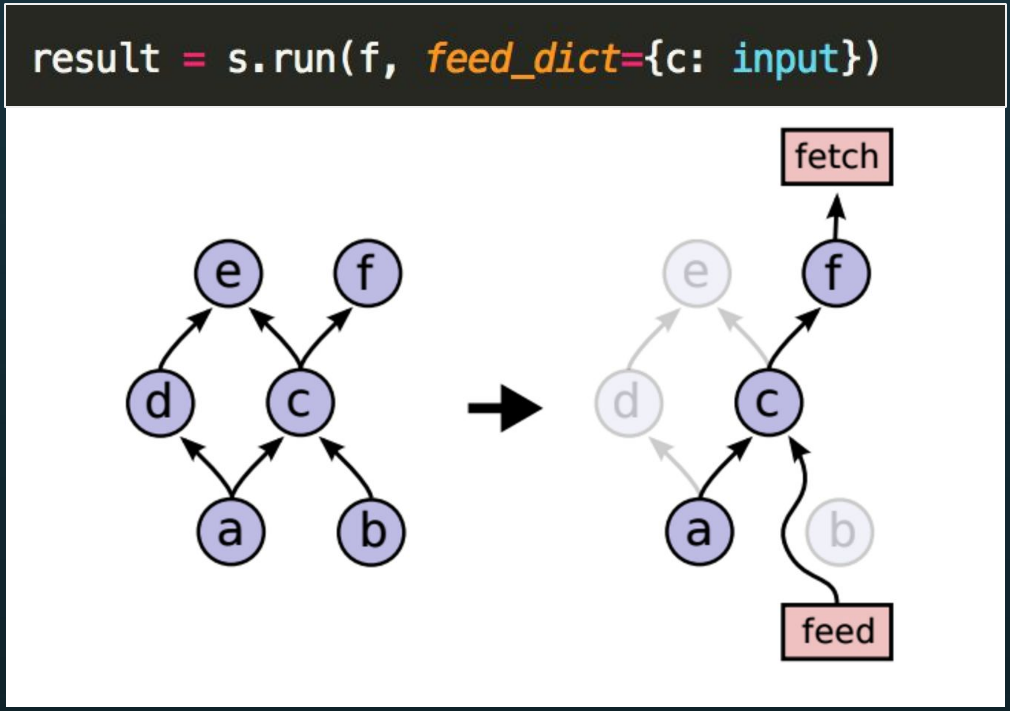
```
result = s.run(f, feed_dict={c: input})
```

Distributing: Pruning

gRPC call to **Session::Run**
in master_session.cc#L835

Rewrite with feed and fetch
RewriteGraphForExecution
in graph/subgraph.cc#L225

Prune subgraph
PruneForReverseReachability
in graph/algorithm.cc#L122
tests in subgraph_test.cc#142

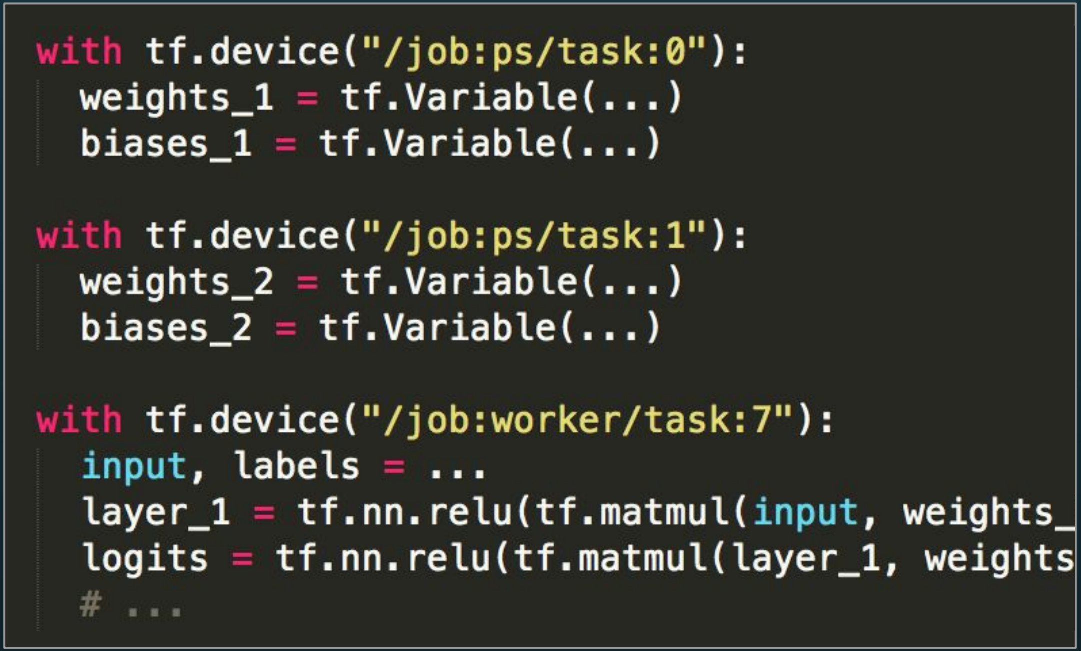


```
result = s.run(f, feed_dict={c: input})
```

Distributing: Placing

Constraints from model

[DeviceSpec in device.py#L24]()

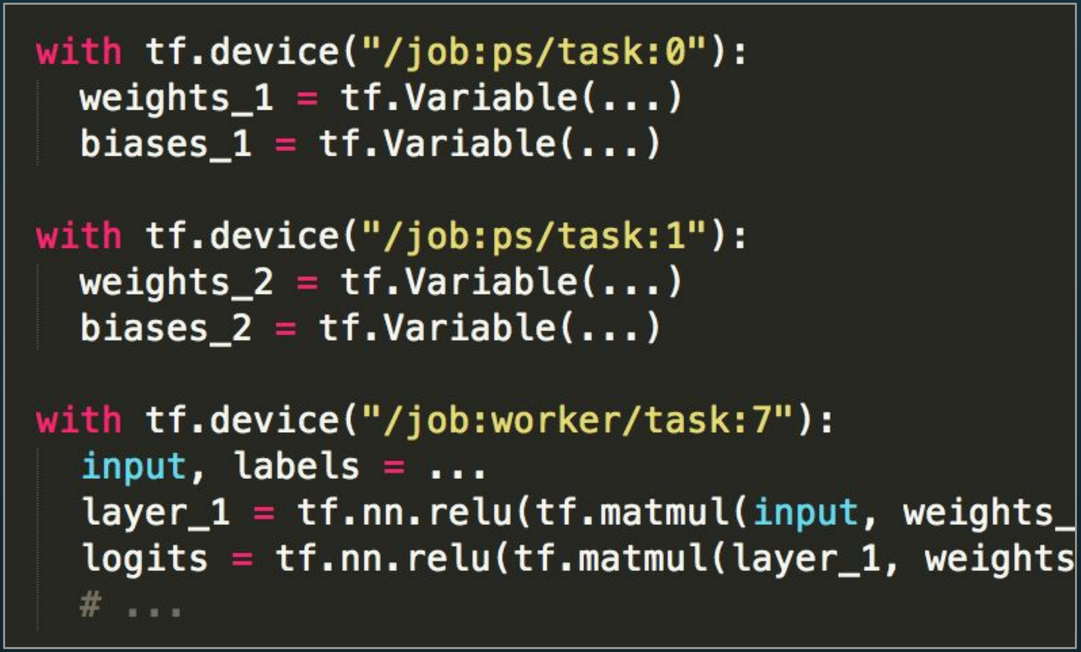


```
with tf.device("/job:ps/task:0"):
  weights_1 = tf.Variable(...)
  biases_1 = tf.Variable(...)

with tf.device("/job:ps/task:1"):
  weights_2 = tf.Variable(...)
  biases_2 = tf.Variable(...)

with tf.device("/job:worker/task:7"):
  input, labels = ...
  layer_1 = tf.nn.relu(tf.matmul(input, weights_
  logits = tf.nn.relu(tf.matmul(layer_1, weights
  # ...
```

Distributing: Placing

Constraints from model

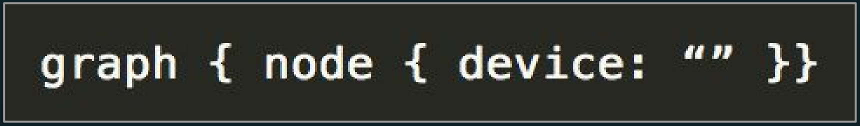
DeviceSpec in device.py#L24



```
with tf.device("/job:ps/task:0"):
  weights_1 = tf.Variable(...)
  biases_1 = tf.Variable(...)

with tf.device("/job:ps/task:1"):
  weights_2 = tf.Variable(...)
  biases_2 = tf.Variable(...)

with tf.device("/job:worker/task:7"):
  input, labels = ...
  layer_1 = tf.nn.relu(tf.matmul(input, weights_
  logits = tf.nn.relu(tf.matmul(layer_1, weights
  # ...
```

By device or colocation

NodeDef in graph.proto



```
graph { node { device: “” }}
```

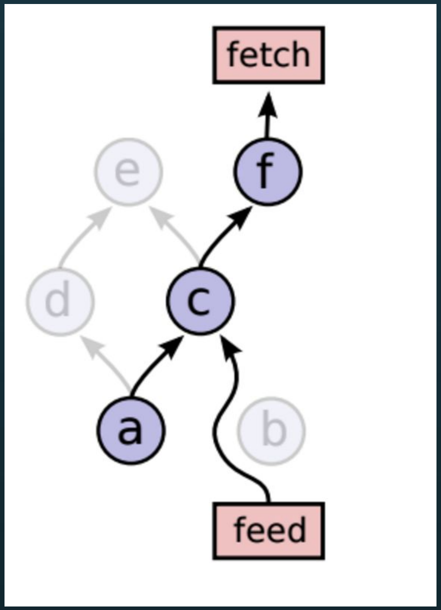
Distributing: Placing

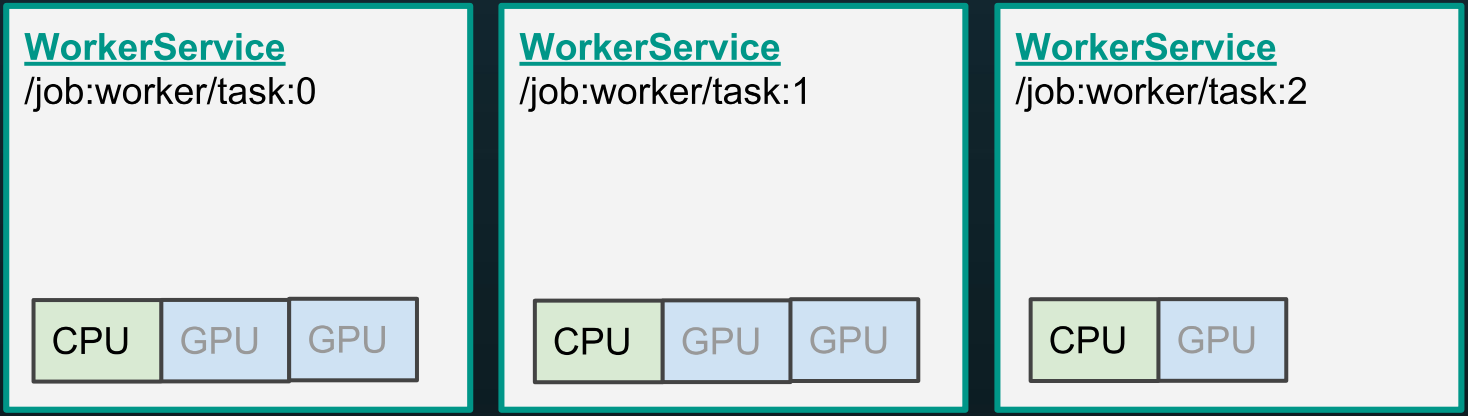
Placing based on **constraints**

SimplePlacer::Run

in **simple_placer.cc#L558**

described in **simple_placer.h#L31**





WorkerService
/job:worker/task:0

CPU GPU GPU

WorkerService
/job:worker/task:1

CPU GPU GPU

WorkerService
/job:worker/task:2

CPU GPU

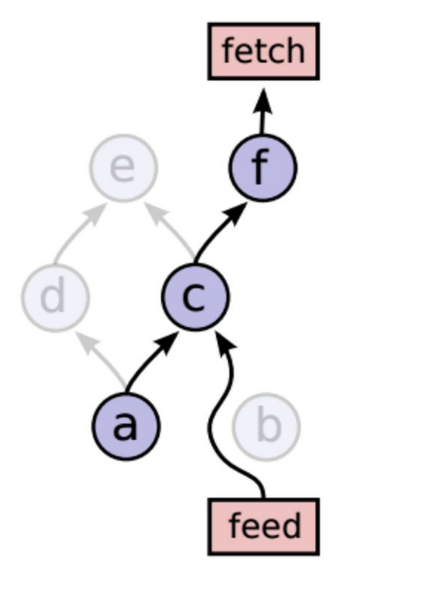
Distributing: Placing

Placing based on **constraints**

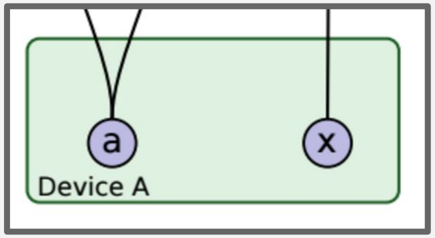
SimplePlacer::Run

in **simple_placer.cc#L558**

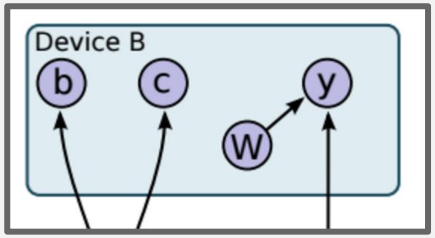
described in **simple_placer.h#L31**



WorkerService
/job:worker/task:0



WorkerService
/job:worker/task:1



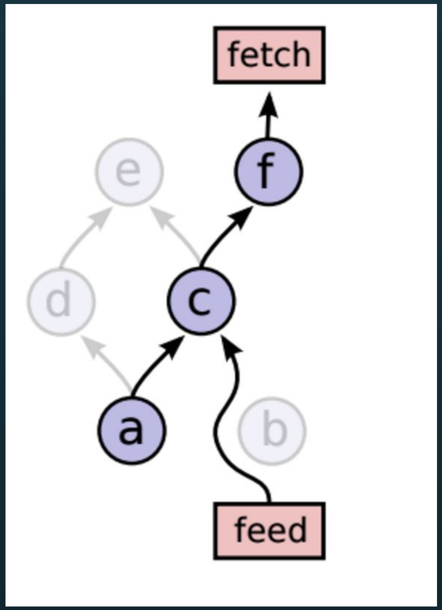
Distributing: Placing

Placing based on **constraints**

SimplePlacer::Run

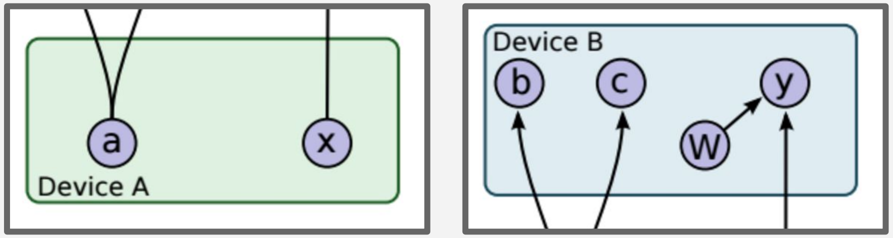
in **simple_placer.cc#L558**

described in **simple_placer.h#L31**



WorkerService

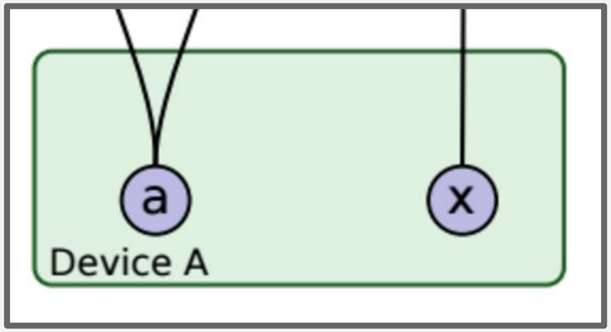
/job:worker/task:0



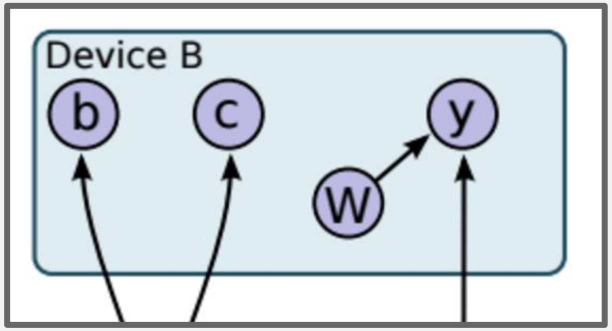
Distributing: Partitioning

Partition into subgraphs
in graph_partition.cc#L883

WorkerService
/job:worker/task:0



WorkerService
/job:worker/task:0

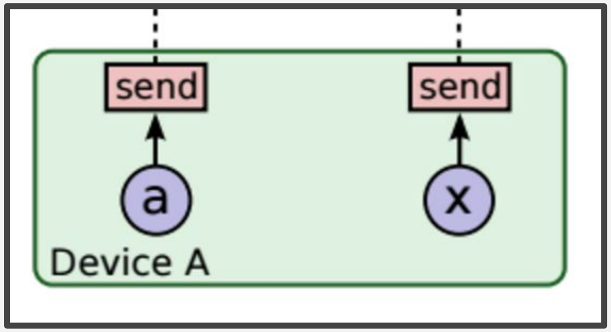


Distributing: Partitioning

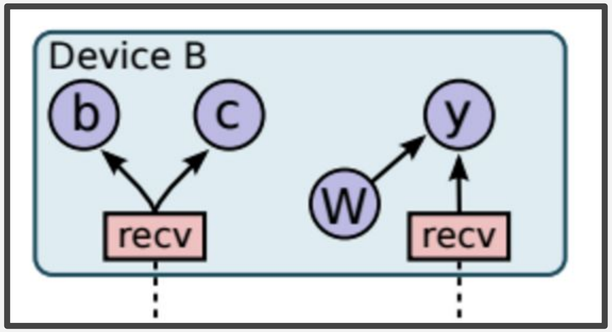
Partition into subgraphs
in graph_partition.cc#L883

Rewrite with **Send** and **Recv**
in sendrecv_ops.cc#L56 **and** #L97

WorkerService
/job:worker/task:0



WorkerService
/job:worker/task:0

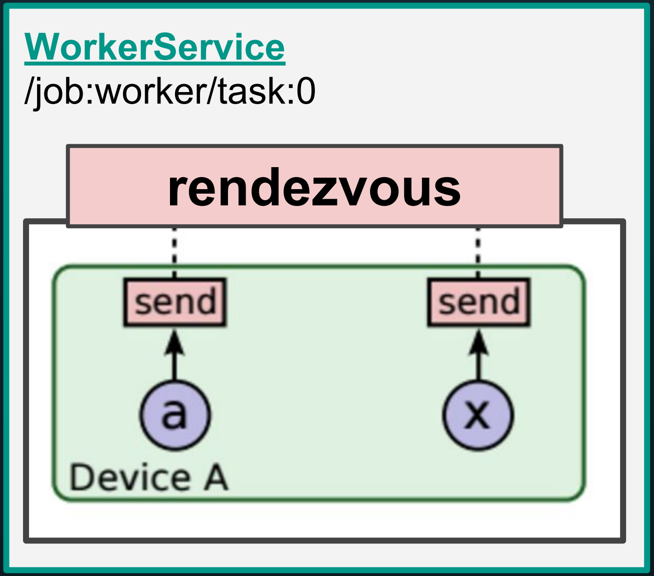


Distributing: Partitioning

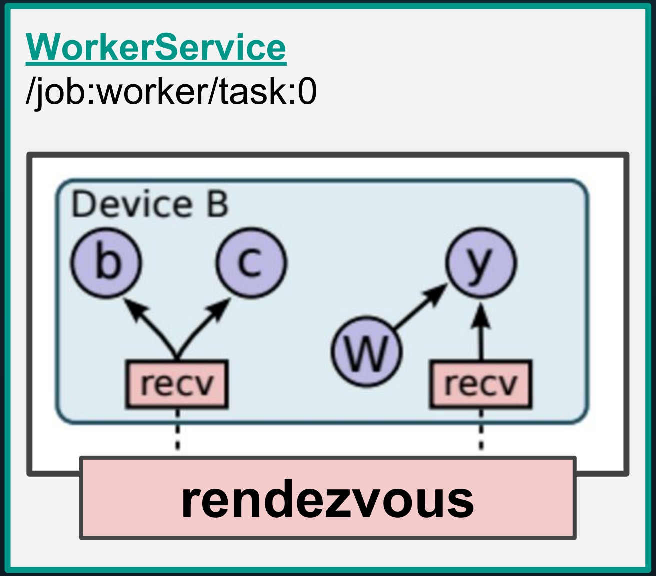
Partition into subgraphs
in graph_partition.cc#L883

Rewrite with **Send** and **Recv**
in sendrecv_ops.cc#L56 **and** #L97

Rendezvous handles coordination
in base_rendezvous_mgr.cc#L236



WorkerService
/job:worker/task:0



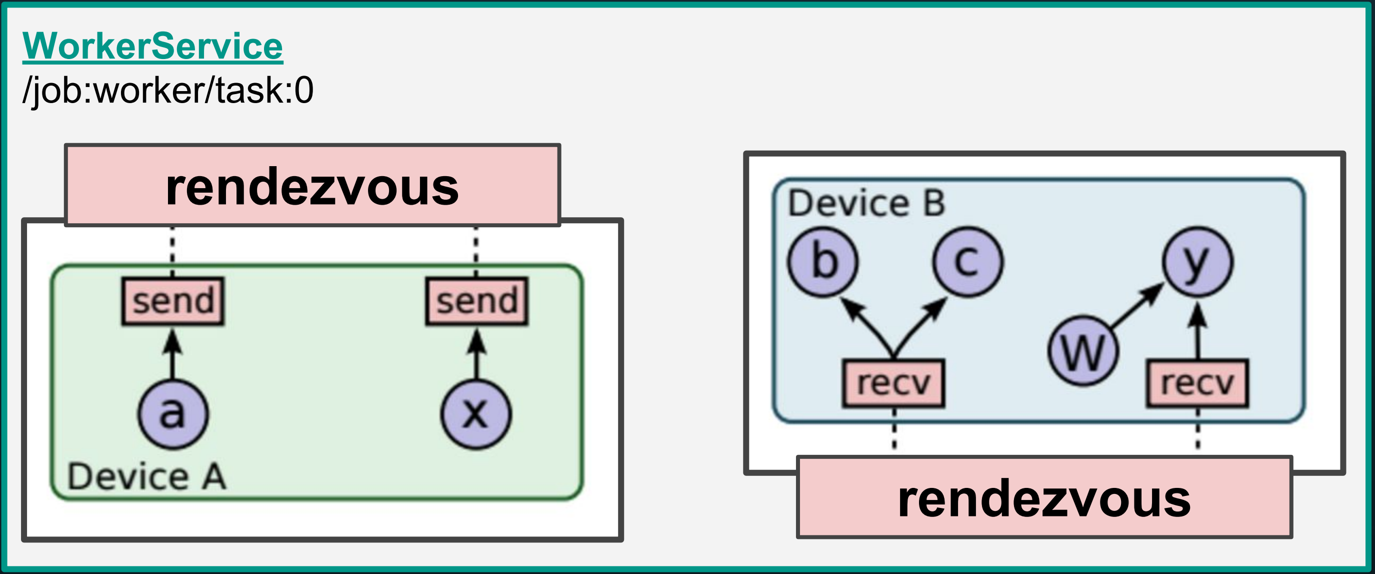
WorkerService
/job:worker/task:0

Distributing: Partitioning

Partition into subgraphs
in graph_partition.cc#L883

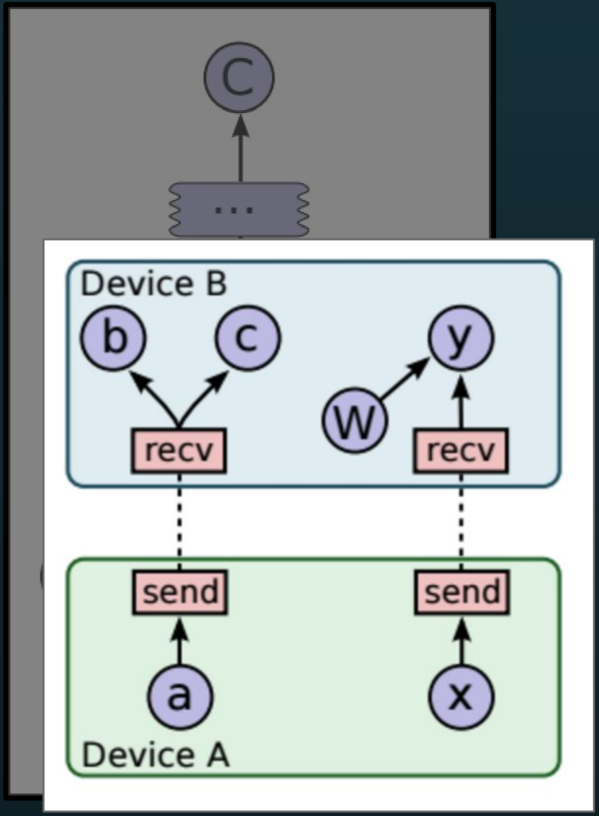
Rewrite with **Send** and **Recv**
in sendrecv_ops.cc#L56 **and** #L97

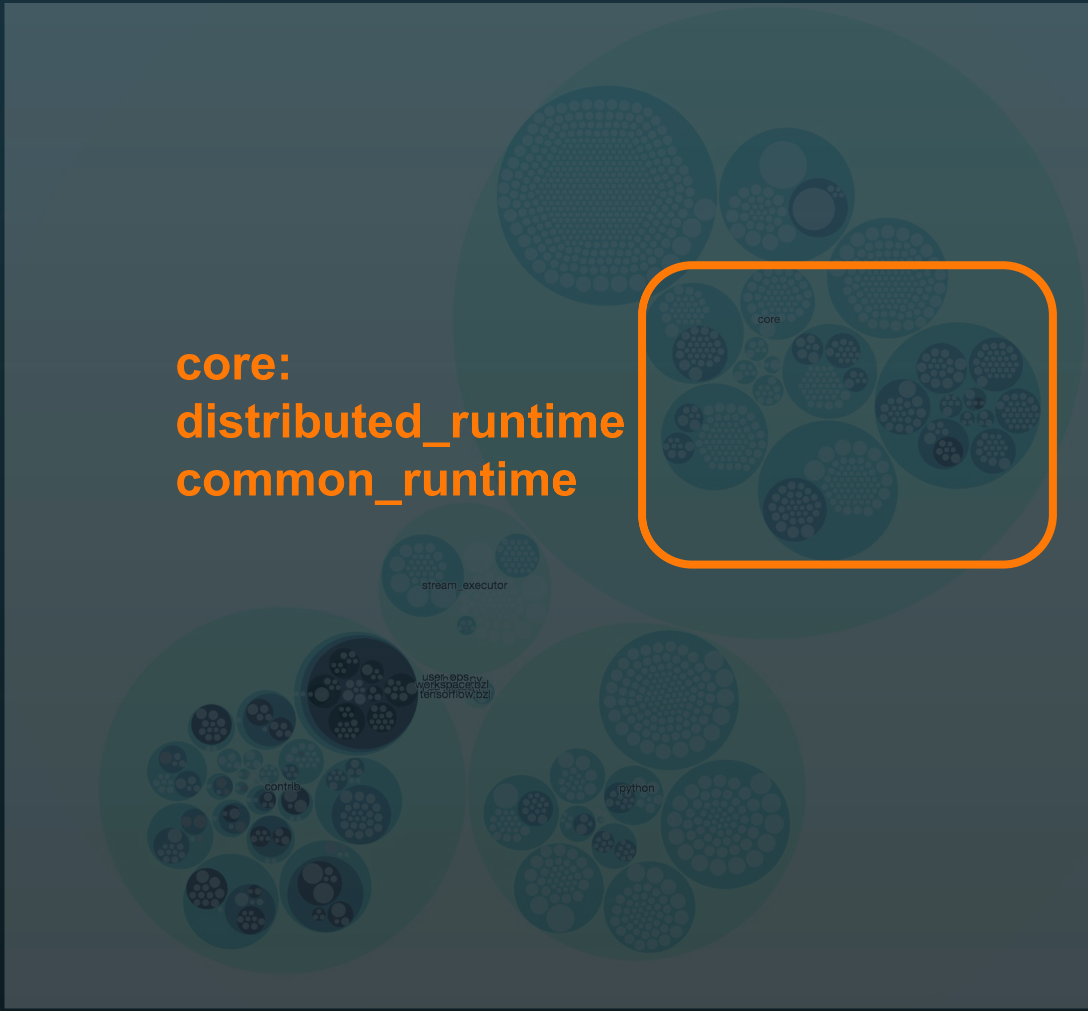
Rendezvous handles coordination
in base_rendezvous_mgr.cc#L236



A tour through the TensorFlow codebase

2. **Distributing** the graph



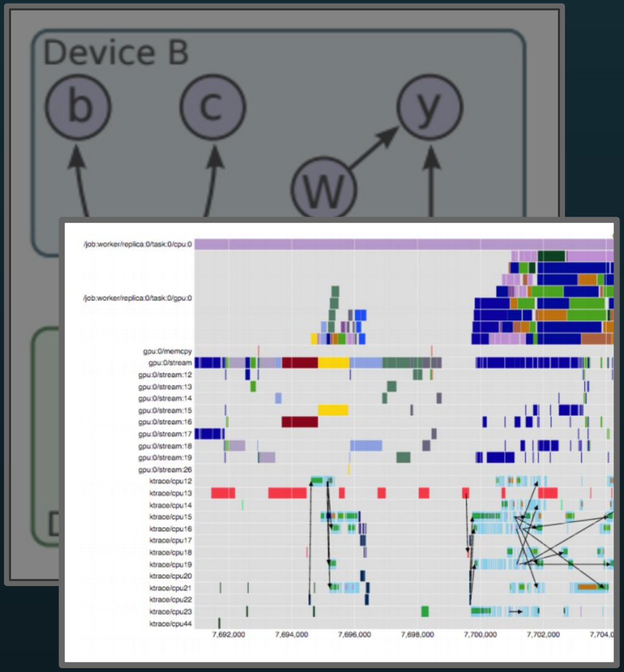


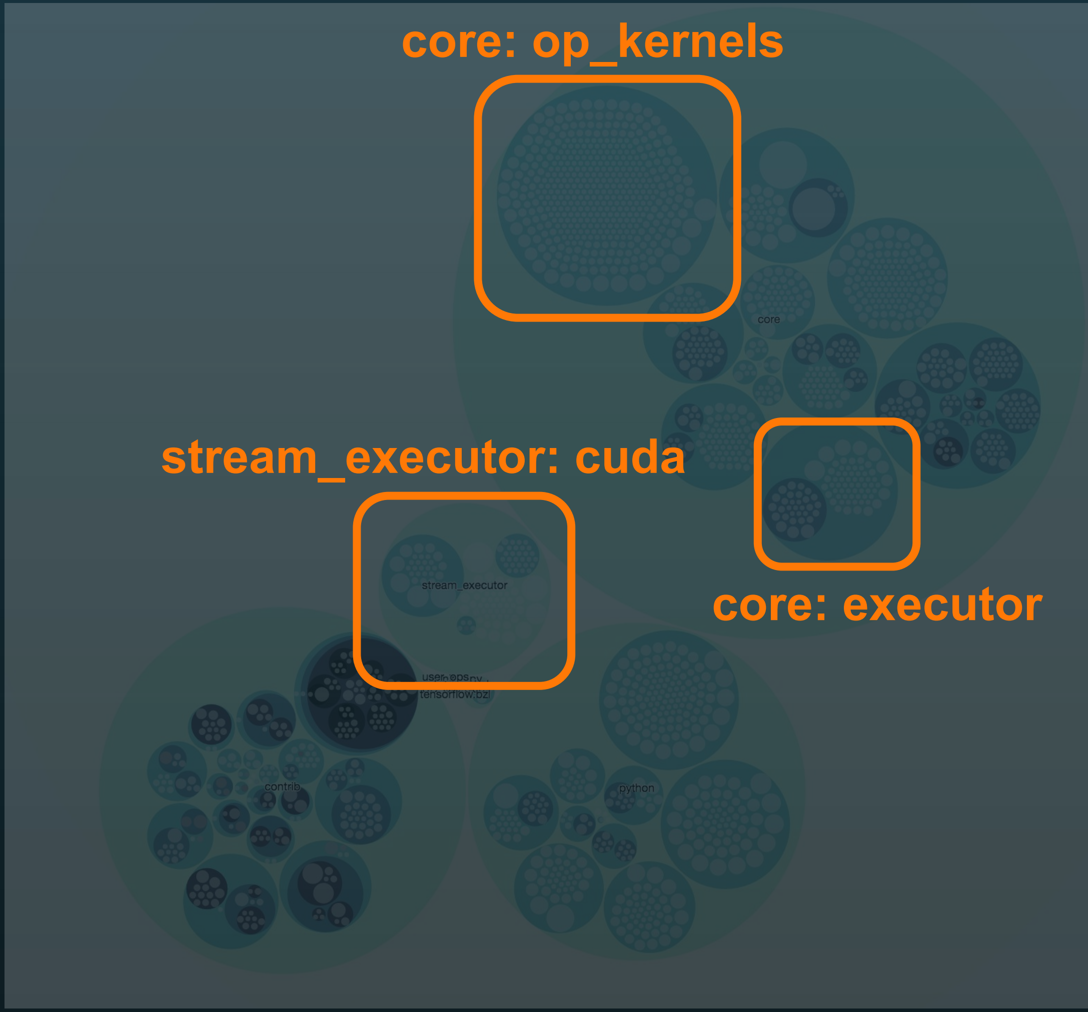
core:
distributed_runtime
common_runtime

http://public.kevinrobinsonblog.com/tensorflow-codebase/

A tour through the TensorFlow codebase

3. **Executing** the graph





core: op_kernels

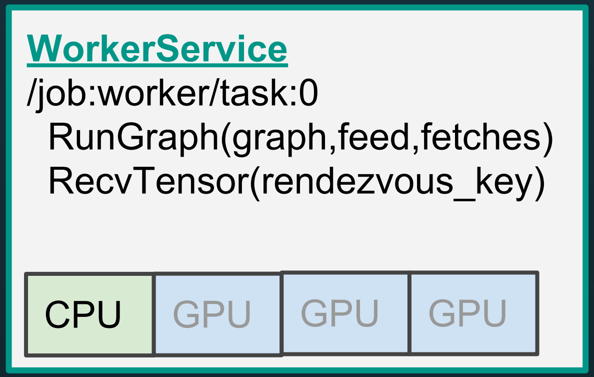
stream_executor: cuda

core: executor

http://public.kevinrobinsonblog.com/tensorflow-codebase/

Executing: Executor

Parallelism on each worker

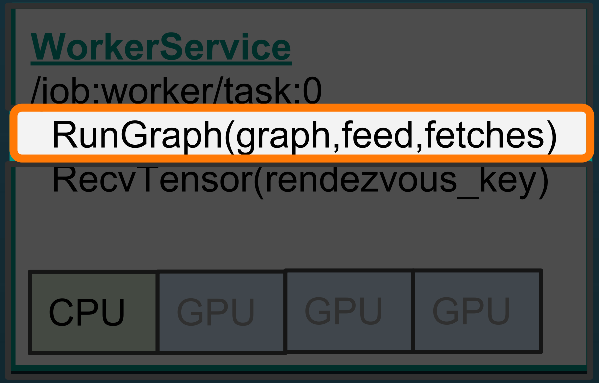


WorkerService
/job:worker/task:0
RunGraph(graph,feed,fetches)
RecvTensor(rendezvous_key)

CPU GPU GPU GPU

Executing: Executor

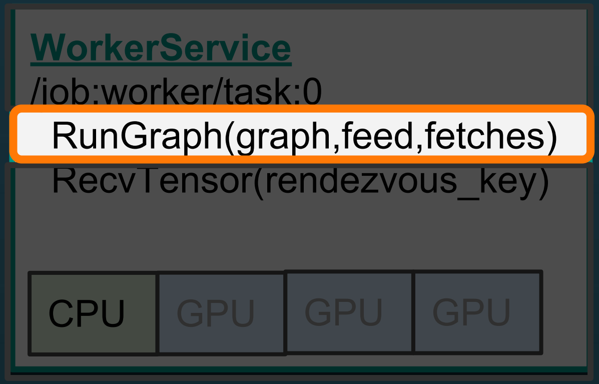
Parallelism on each worker



WorkerService
/job:worker/task:0
RunGraph(graph,feed,fetches)
RecvTensor(rendezvous_key)

Executing: Executor

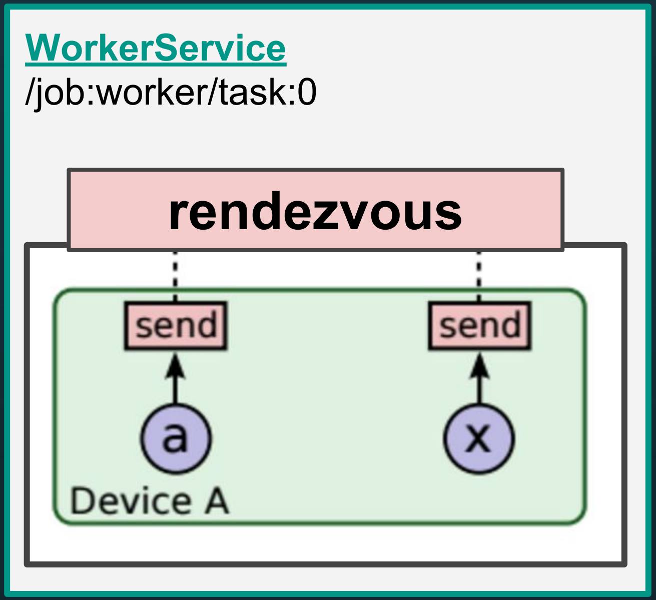
Parallelism on each worker



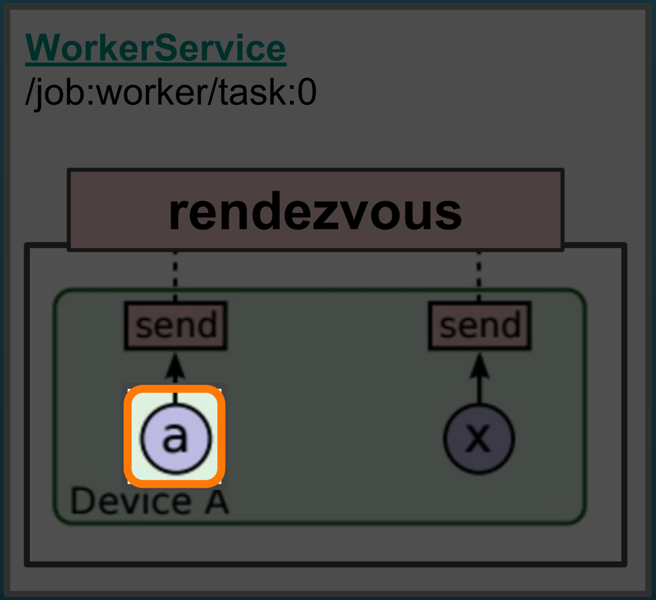
GraphMgr::ExecuteAsync
in **graph_mgr.cc#L283**

ExecutorState::RunAsync
in executor.cc#L867

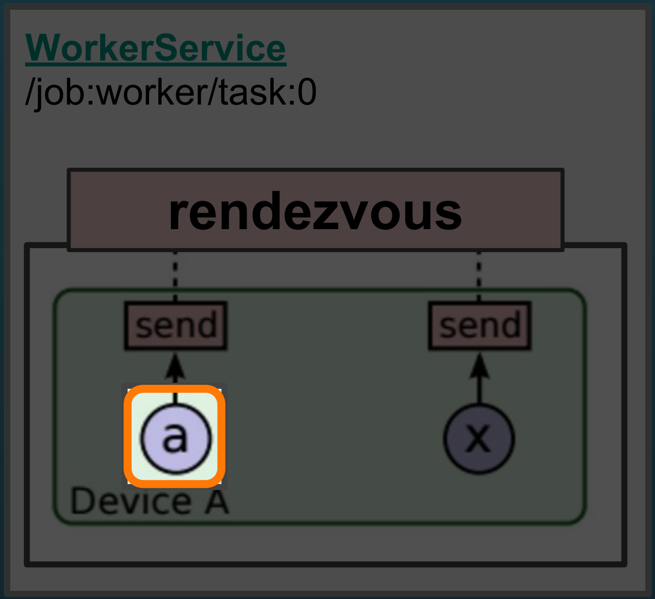
Executing: OpKernels

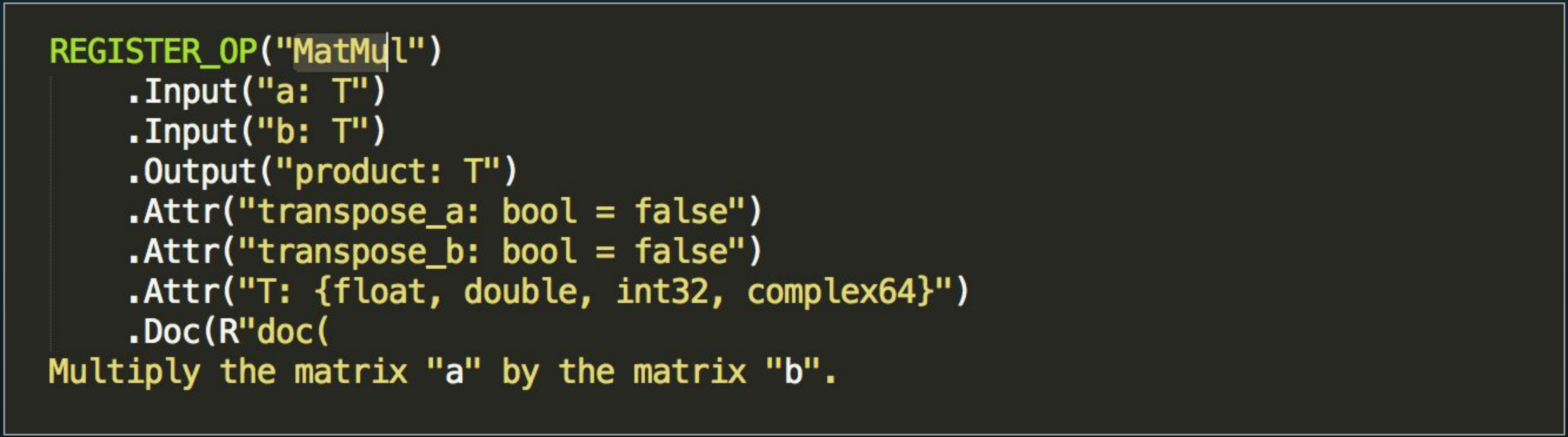


Executing: OpKernels



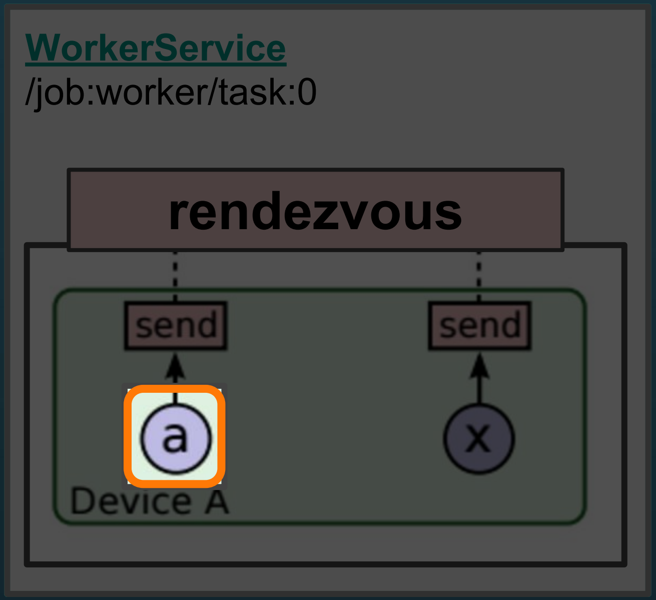
Executing: OpKernels



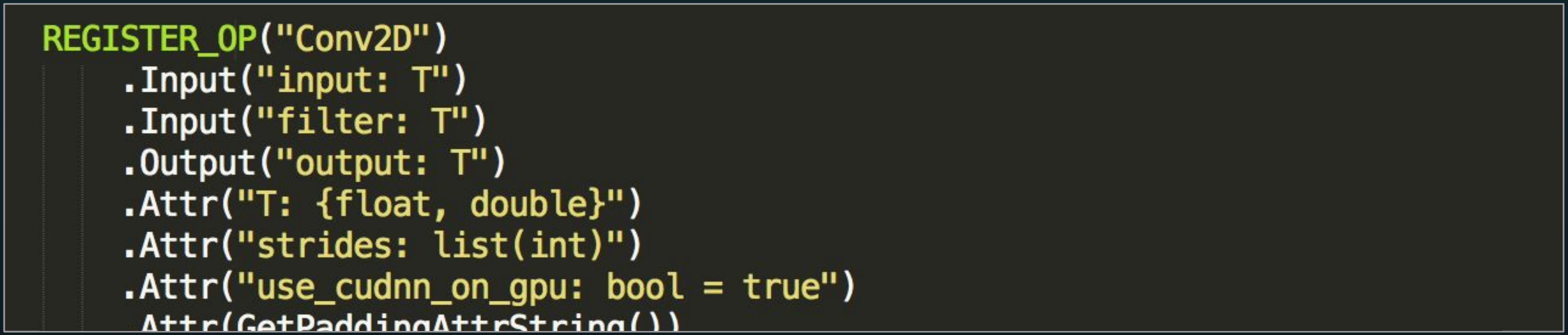


```
REGISTER_OP("MatMul")
    .Input("a: T")
    .Input("b: T")
    .Output("product: T")
    .Attr("transpose_a: bool = false")
    .Attr("transpose_b: bool = false")
    .Attr("T: {float, double, int32, complex64}")
    .Doc(R"doc(
Multiply the matrix "a" by the matrix "b".
```

Executing: OpKernels



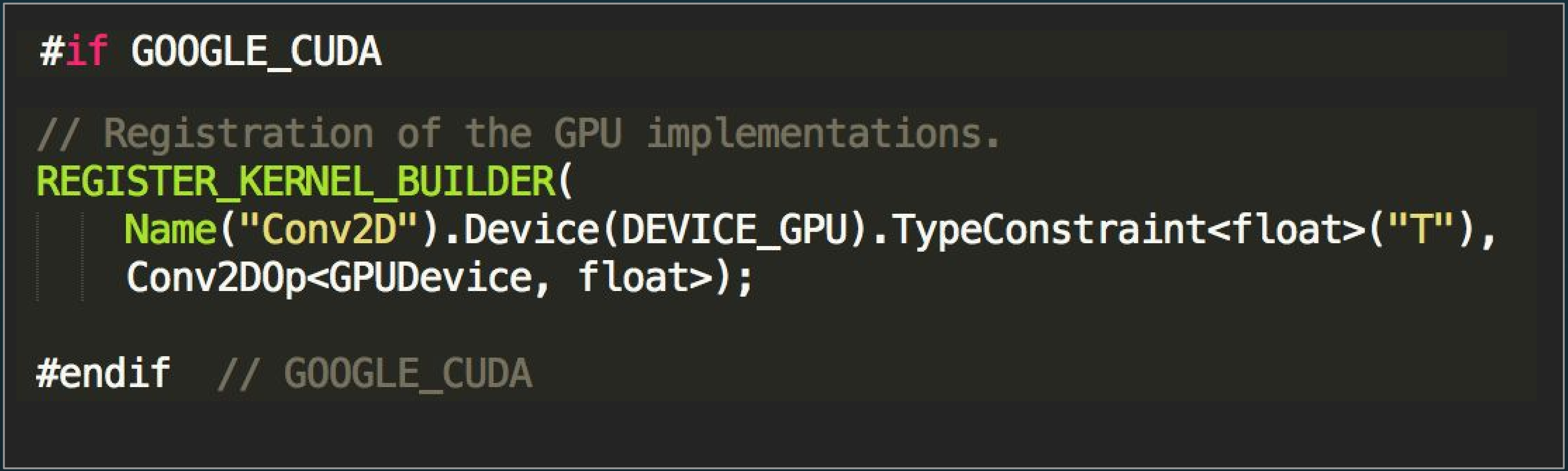
Conv2D OpDef in nn_ops.cc#L221



```
REGISTER_OP("Conv2D")
    .Input("input: T")
    .Input("filter: T")
    .Output("output: T")
    .Attr("T: {float, double}")
    .Attr("strides: list(int)")
    .Attr("use_cudnn_on_gpu: bool = true")
    .Attr(GetPaddingAttrString())
```

Executing: OpKernels

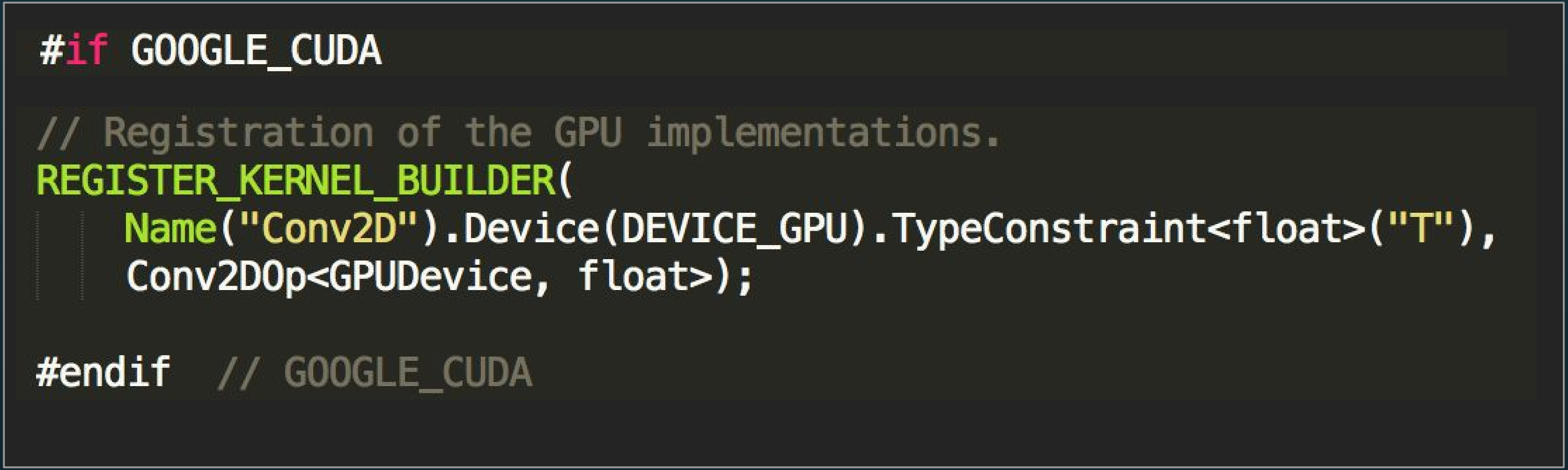
Conditional build for OpKernels



```
#if GOOGLE_CUDA

// Registration of the GPU implementations.
REGISTER_KERNEL_BUILDER(
    Name("Conv2D").Device(DEVICE_GPU).TypeConstraint<float>("T"),
    Conv2DOp<GPUDevice, float>);

#endif  // GOOGLE_CUDA
```

Executing: OpKernels

Conditional build for OpKernels



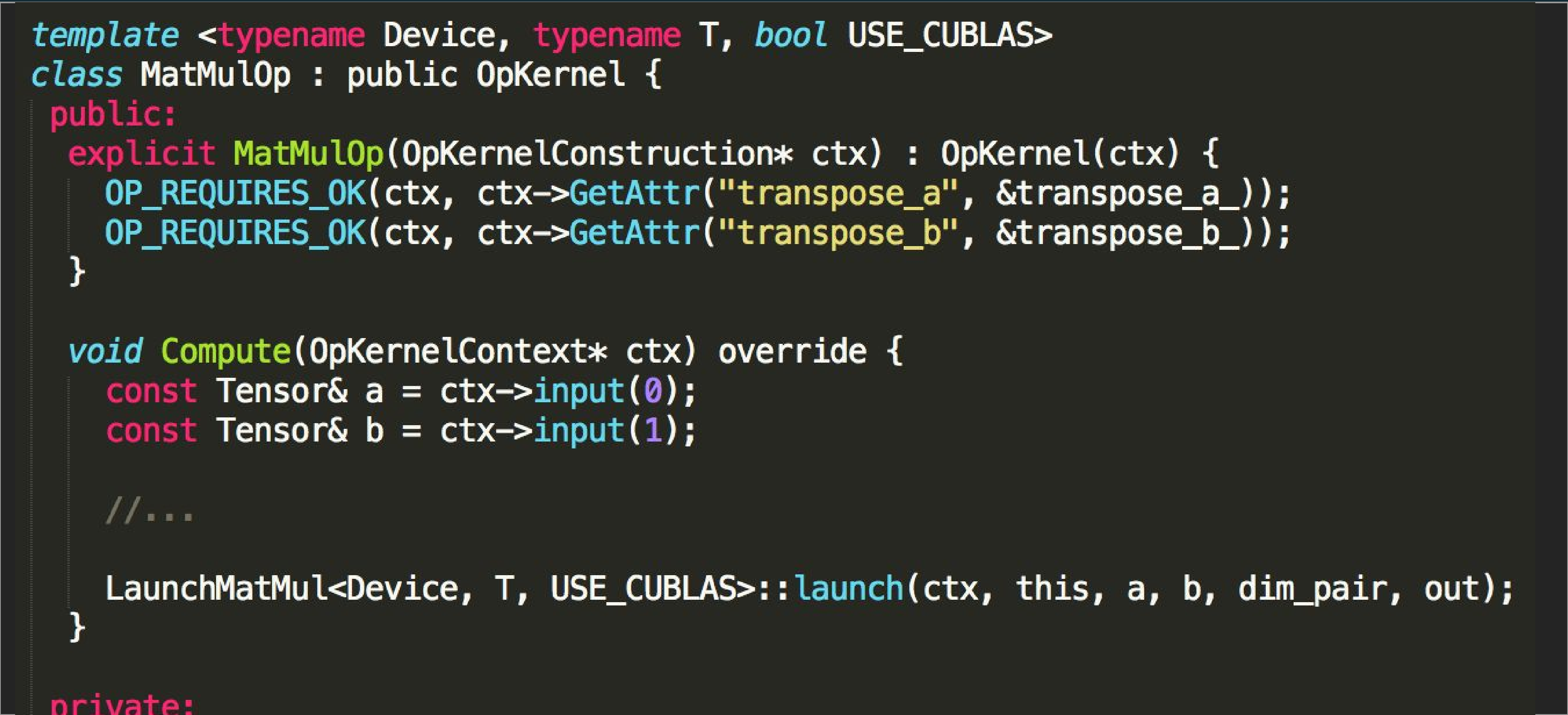
```
#if GOOGLE_CUDA

// Registration of the GPU implementations.
REGISTER_KERNEL_BUILDER(
    Name("Conv2D").Device(DEVICE_GPU).TypeConstraint<float>("T"),
    Conv2DOp<GPUDevice, float>);

#endif  // GOOGLE_CUDA
```

CPU in conv_ops.cc#L91
GPU in conv_ops.cc#L263

Executing: OpKernels

OpKernels are **specialized** by device

adapted from matmul_op.cc#L116

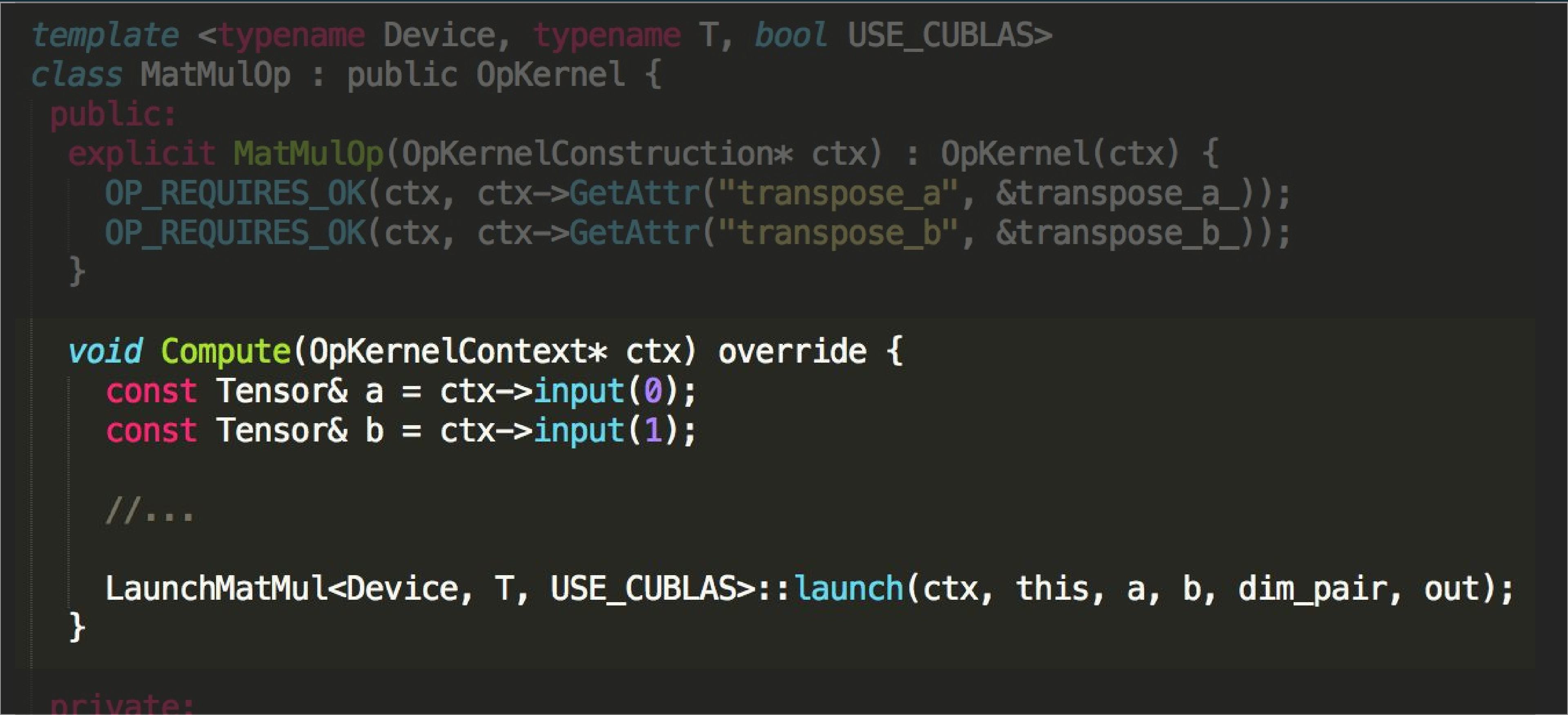


```
template <typename Device, typename T, bool USE_CUBLAS>
class MatMulOp : public OpKernel {
 public:
  explicit MatMulOp(OpKernelConstruction* ctx) : OpKernel(ctx) {
    OP_REQUIRES_OK(ctx, ctx->GetAttr("transpose_a", &transpose_a_));
    OP_REQUIRES_OK(ctx, ctx->GetAttr("transpose_b", &transpose_b_));
  }

  void Compute(OpKernelContext* ctx) override {
    const Tensor& a = ctx->input(0);
    const Tensor& b = ctx->input(1);

    //...

    LaunchMatMul<Device, T, USE_CUBLAS>::launch(ctx, this, a, b, dim_pair, out);
  }

 private:
```

Executing: OpKernels

OpKernels are **specialized** by device

adapted from matmul_op.cc#L116

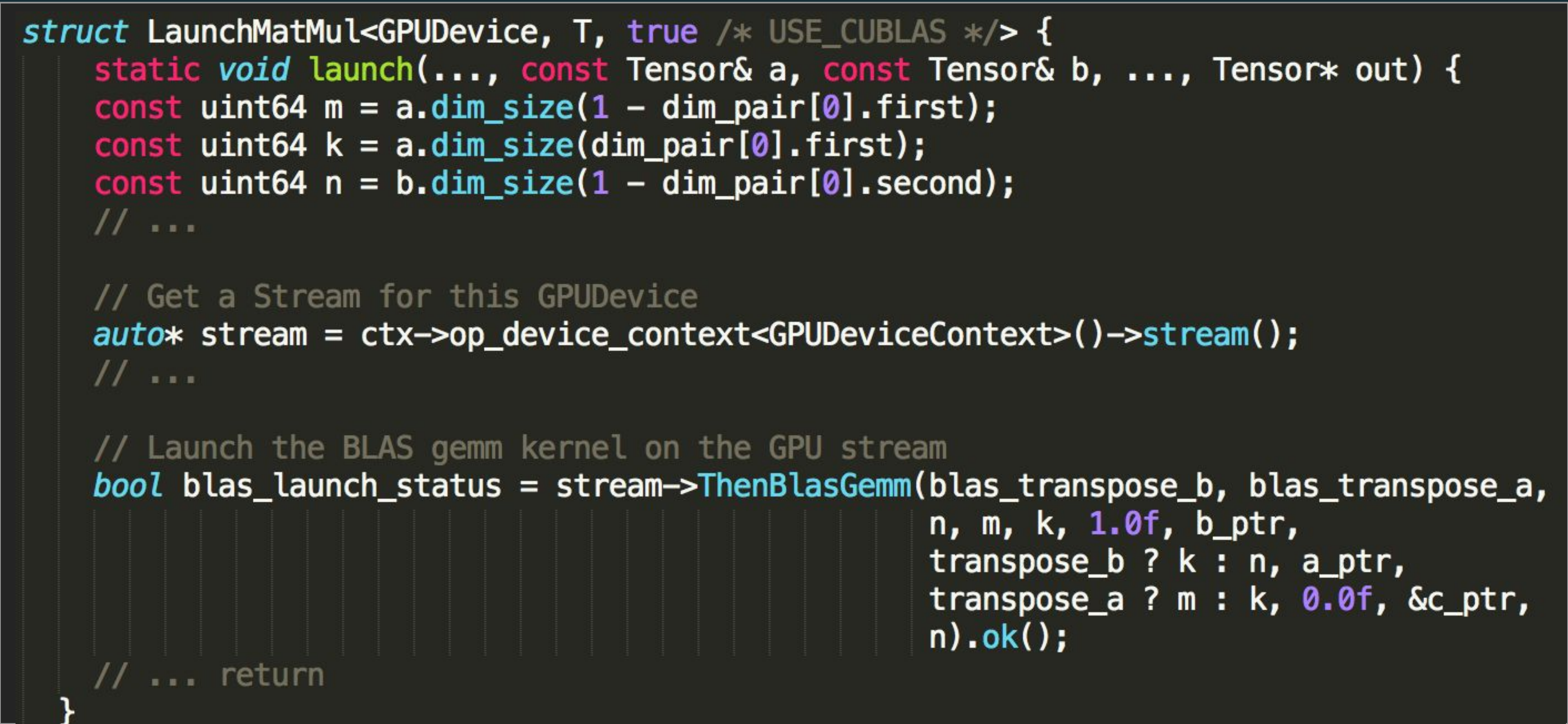


```
template <typename Device, typename T, bool USE_CUBLAS>
class MatMulOp : public OpKernel {
 public:
  explicit MatMulOp(OpKernelConstruction* ctx) : OpKernel(ctx) {
    OP_REQUIRES_OK(ctx, ctx->GetAttr("transpose_a", &transpose_a_));
    OP_REQUIRES_OK(ctx, ctx->GetAttr("transpose_b", &transpose_b_));
  }

  void Compute(OpKernelContext* ctx) override {
    const Tensor& a = ctx->input(0);
    const Tensor& b = ctx->input(1);

    //...

    LaunchMatMul<Device, T, USE_CUBLAS>::launch(ctx, this, a, b, dim_pair, out);
  }

 private:
```

Executing: OpKernels

OpKernels call into **Stream** functions

adapted from matmul_op.cc#L71

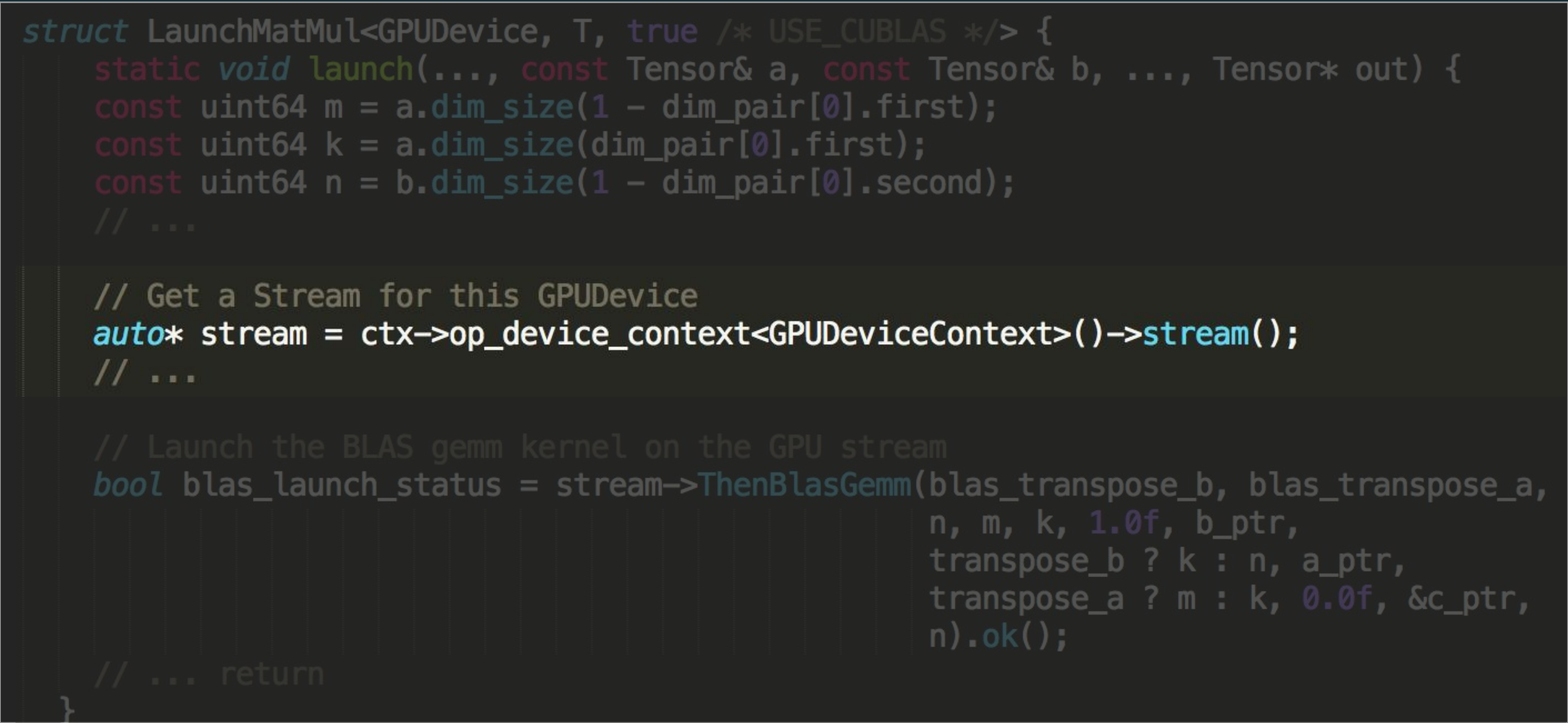


```
struct LaunchMatMul<GPUDevice, T, true /* USE_CUBLAS */> {
    static void launch(..., const Tensor& a, const Tensor& b, ..., Tensor* out) {
    const uint64 m = a.dim_size(1 - dim_pair[0].first);
    const uint64 k = a.dim_size(dim_pair[0].first);
    const uint64 n = b.dim_size(1 - dim_pair[0].second);
    // ...

    // Get a Stream for this GPUDevice
    auto* stream = ctx->op_device_context<GPUDeviceContext>()->stream();
    // ...

    // Launch the BLAS gemm kernel on the GPU stream
    bool blas_launch_status = stream->ThenBlasGemm(blas_transpose_b, blas_transpose_a,
                                                   n, m, k, 1.0f, b_ptr,
                                                   transpose_b ? k : n, a_ptr,
                                                   transpose_a ? m : k, 0.0f, &c_ptr,
                                                   n).ok();
    // ... return
  }
```

Executing: OpKernels

OpKernels call into **Stream** functions

adapted from matmul_op.cc#L71

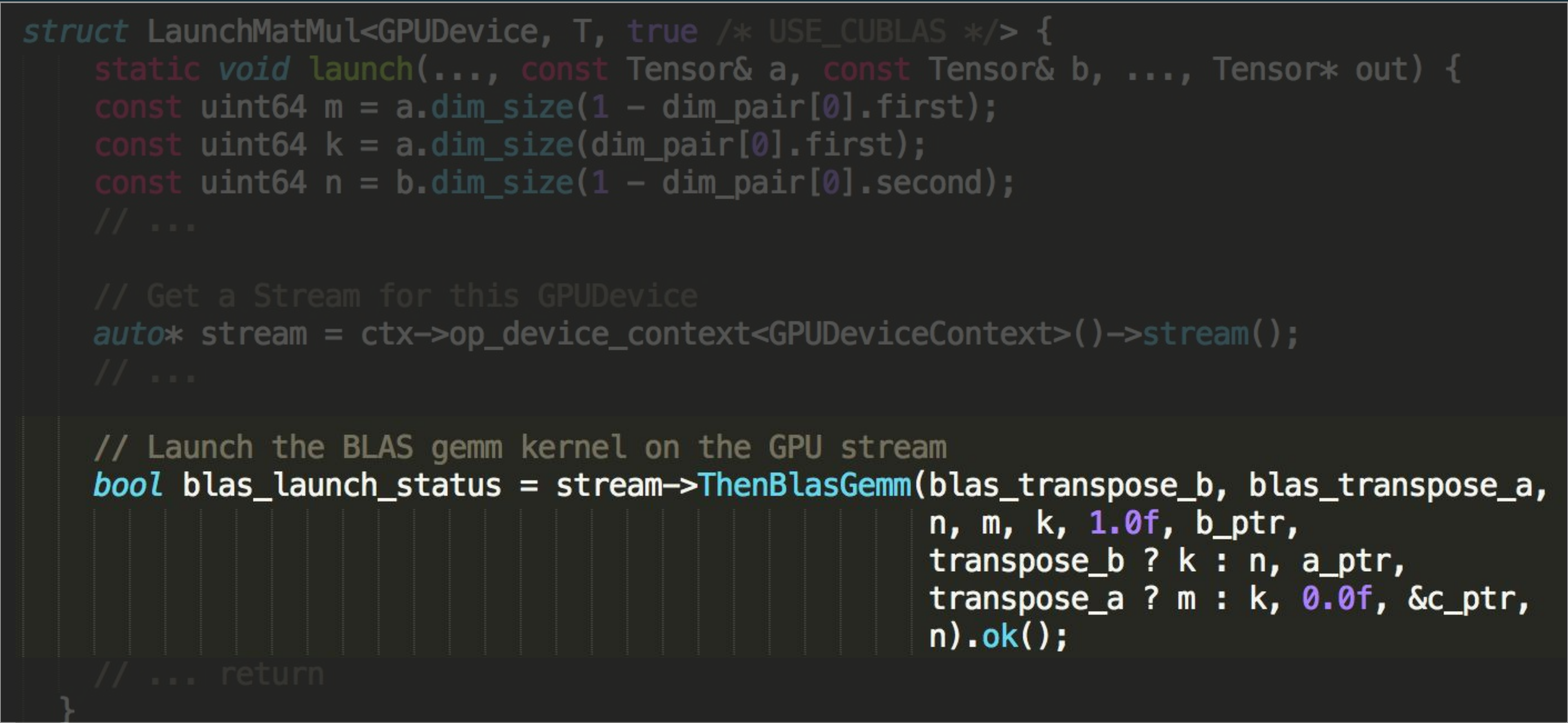


```
struct LaunchMatMul<GPUDevice, T, true /* USE_CUBLAS */> {
    static void launch(..., const Tensor& a, const Tensor& b, ..., Tensor* out) {
    const uint64 m = a.dim_size(1 - dim_pair[0].first);
    const uint64 k = a.dim_size(dim_pair[0].first);
    const uint64 n = b.dim_size(1 - dim_pair[0].second);
    // ...

    // Get a Stream for this GPUDevice
    auto* stream = ctx->op_device_context<GPUDeviceContext>()->stream();
    // ...

    // Launch the BLAS gemm kernel on the GPU stream
    bool blas_launch_status = stream->ThenBlasGemm(blas_transpose_b, blas_transpose_a,
                                                   n, m, k, 1.0f, b_ptr,
                                                   transpose_b ? k : n, a_ptr,
                                                   transpose_a ? m : k, 0.0f, &c_ptr,
                                                   n).ok();
    // ... return
  }
```

Executing: OpKernels

OpKernels call into **Stream** functions

adapted from matmul_op.cc#L71

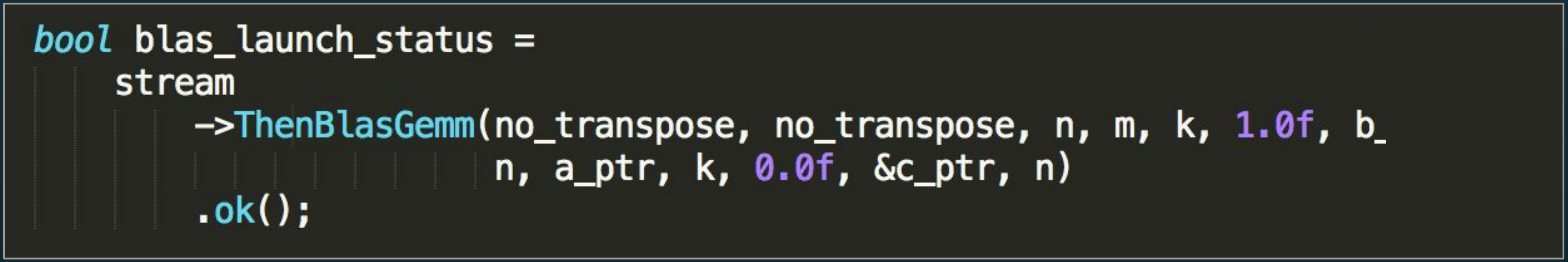


```
struct LaunchMatMul<GPUDevice, T, true /* USE_CUBLAS */> {
    static void launch(..., const Tensor& a, const Tensor& b, ..., Tensor* out) {
    const uint64 m = a.dim_size(1 - dim_pair[0].first);
    const uint64 k = a.dim_size(dim_pair[0].first);
    const uint64 n = b.dim_size(1 - dim_pair[0].second);
    // ...

    // Get a Stream for this GPUDevice
    auto* stream = ctx->op_device_context<GPUDeviceContext>()->stream();
    // ...

    // Launch the BLAS gemm kernel on the GPU stream
    bool blas_launch_status = stream->ThenBlasGemm(blas_transpose_b, blas_transpose_a,
                                                   n, m, k, 1.0f, b_ptr,
                                                   transpose_b ? k : n, a_ptr,
                                                   transpose_a ? m : k, 0.0f, &c_ptr,
                                                   n).ok();
    // ... return
  }
```

Executing: Stream functions

OpKernels call into **Stream** functions

in conv_ops.cc#L292

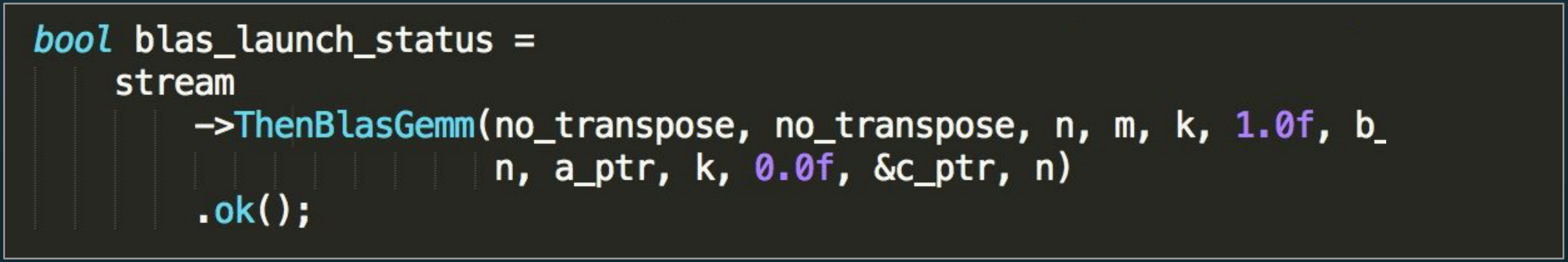


```
bool blas_launch_status =
    stream
        ->ThenBlasGemm(no_transpose, no_transpose, n, m, k, 1.0f, b_
                             n, a_ptr, k, 0.0f, &c_ptr, n)
        .ok();
```

Executing: Stream functions

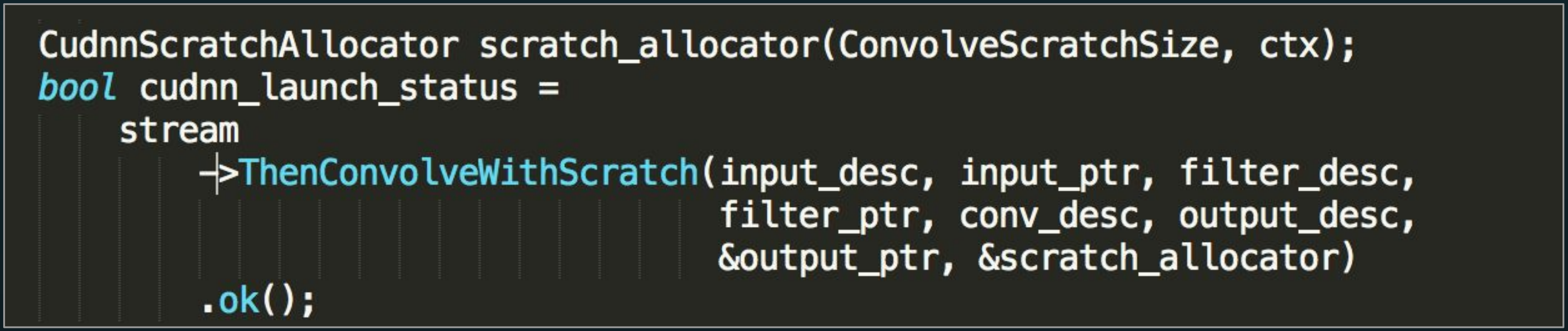
OpKernels call into **Stream** functions

in conv_ops.cc#L292



```
bool blas_launch_status =
    stream
        ->ThenBlasGemm(no_transpose, no_transpose, n, m, k, 1.0f, b_
                       n, a_ptr, k, 0.0f, &c_ptr, n)
        .ok();
```

in conv_ops.cc#L417



```
CudnnScratchAllocator scratch_allocator(ConvolveScratchSize, ctx);
bool cudnn_launch_status =
    stream
        ->ThenConvolveWithScratch(input_desc, input_ptr, filter_desc,
                                  filter_ptr, conv_desc, output_desc,
                                  &output_ptr, &scratch_allocator)
        .ok();
```

Executing: Stream functions

Platforms provide GPU-specific implementations

cuBLAS

BlasSupport in **stream_executor/blas.h#L88**

DoBlasInternal in **cuda_blas.cc#L429**

Executing: Stream functions

Platforms provide GPU-specific implementations

cuBLAS

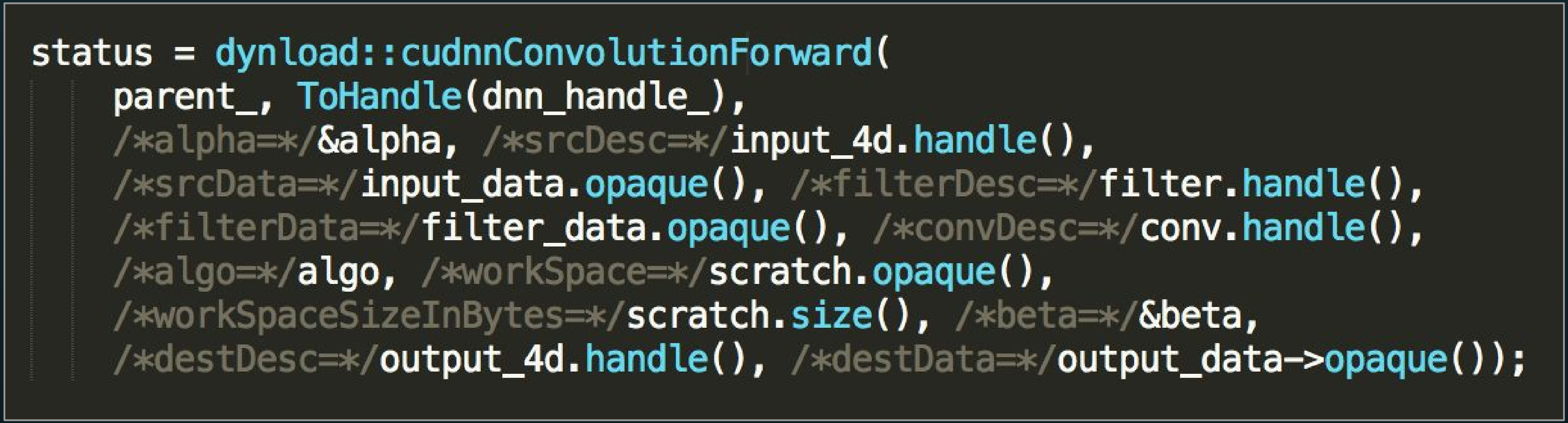
BlasSupport in stream_executor/blas.h#L88

DoBlasInternal in cuda_blas.cc#L429

cuDNN

DnnSupport in stream_executor/dnn.h#L544

DoConvolve in cuda_dnn.cc#L629

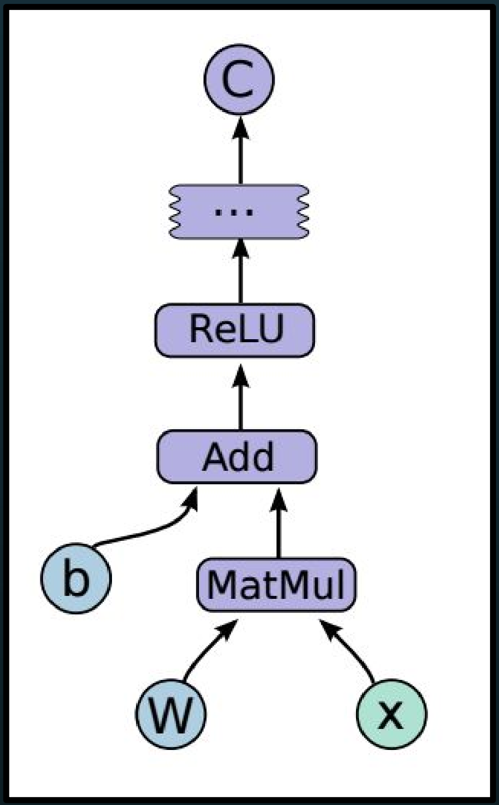


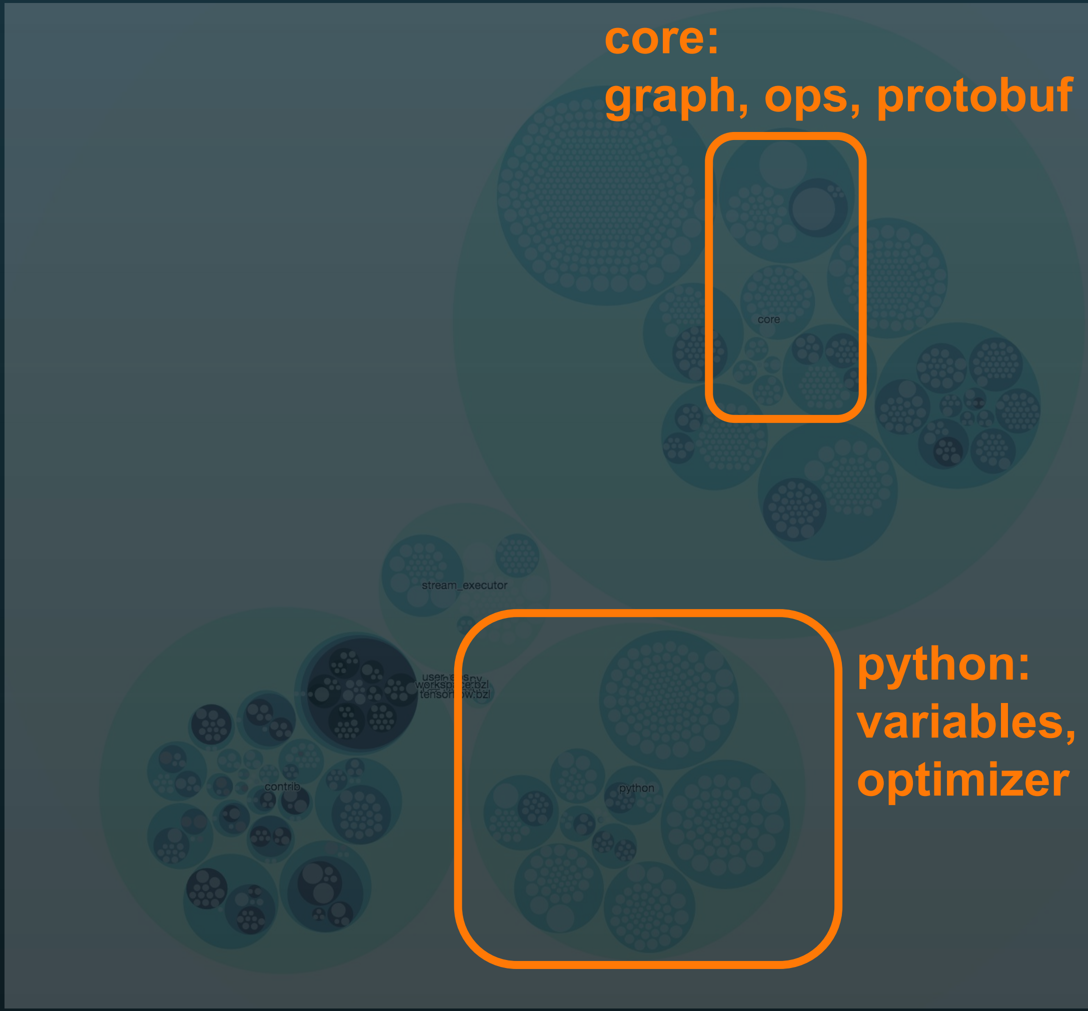
```
status = dynload::cudnnConvolutionForward(
    parent_, ToHandle(dnn_handle_),
    /*alpha=*/&alpha, /*srcDesc=*/input_4d.handle(),
    /*srcData=*/input_data.opaque(), /*filterDesc=*/filter.handle(),
    /*filterData=*/filter_data.opaque(), /*convDesc=*/conv.handle(),
    /*algo=*/algo, /*workSpace=*/scratch.opaque(),
    /*workSpaceSizeInBytes=*/scratch.size(), /*beta=*/&beta,
    /*destDesc=*/output_4d.handle(), /*destData=*/output_data->opaque());
```

A tour through the TensorFlow codebase

A tour through the TensorFlow codebase

1. **Expressing** the graph





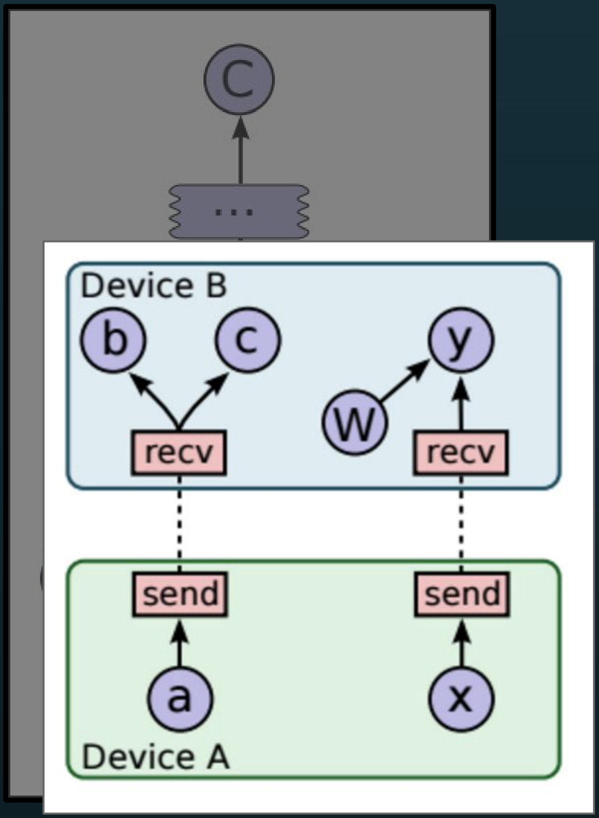
core:
graph, ops, protobuf

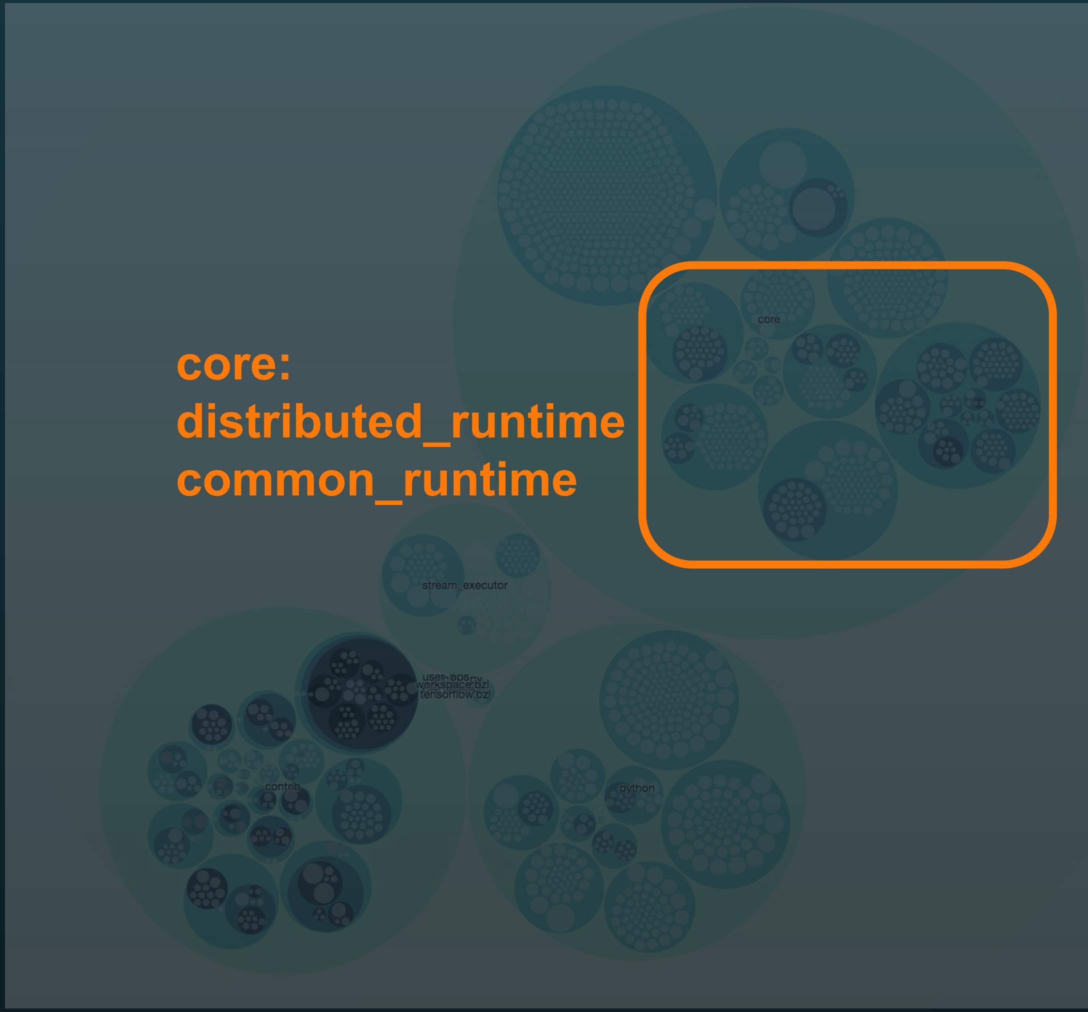
python:
variables,
optimizer

http://public.kevinrobinsonblog.com/tensorflow-codebase/

A tour through the TensorFlow codebase

2. **Distributing** the graph



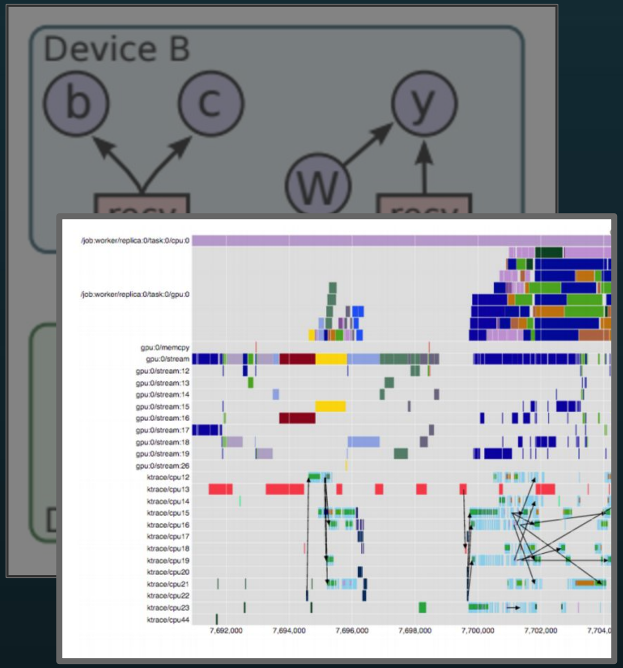


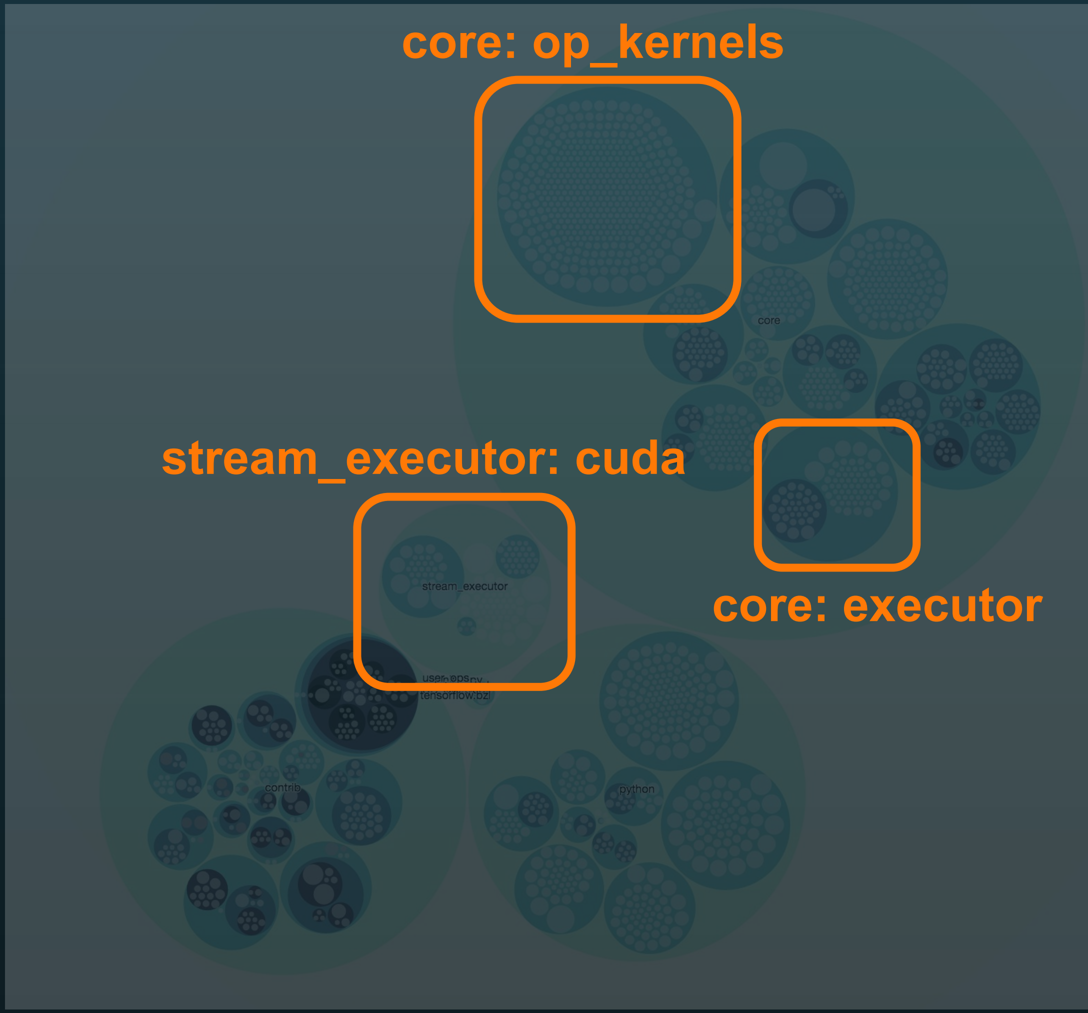
core:
distributed_runtime
common_runtime

http://public.kevinrobinsonblog.com/tensorflow-codebase/

A tour through the TensorFlow codebase

3. **Executing** the graph





core: op_kernels

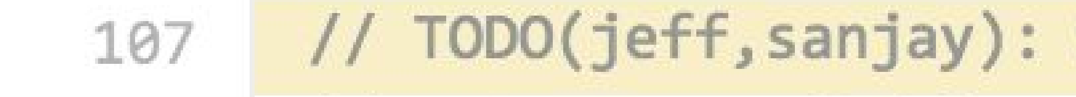
stream_executor: cuda

core: executor

http://public.kevinrobinsonblog.com/tensorflow-codebase/

A tour through the TensorFlow codebase

4. And my favorite **TODO**

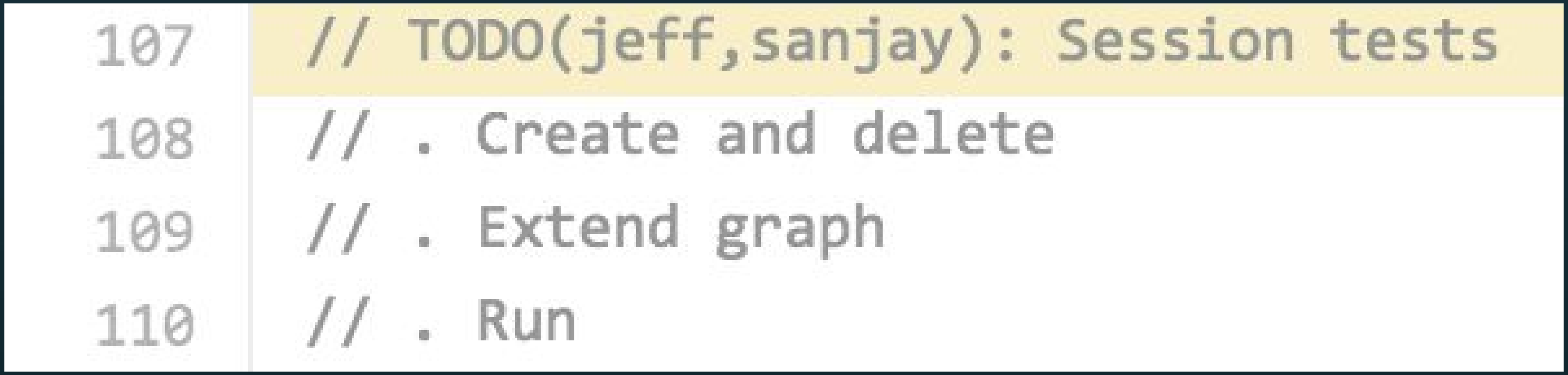


```
// TODO(jeff,sanjay):
```

?

A tour through the TensorFlow codebase

4. And my favorite **TODO**

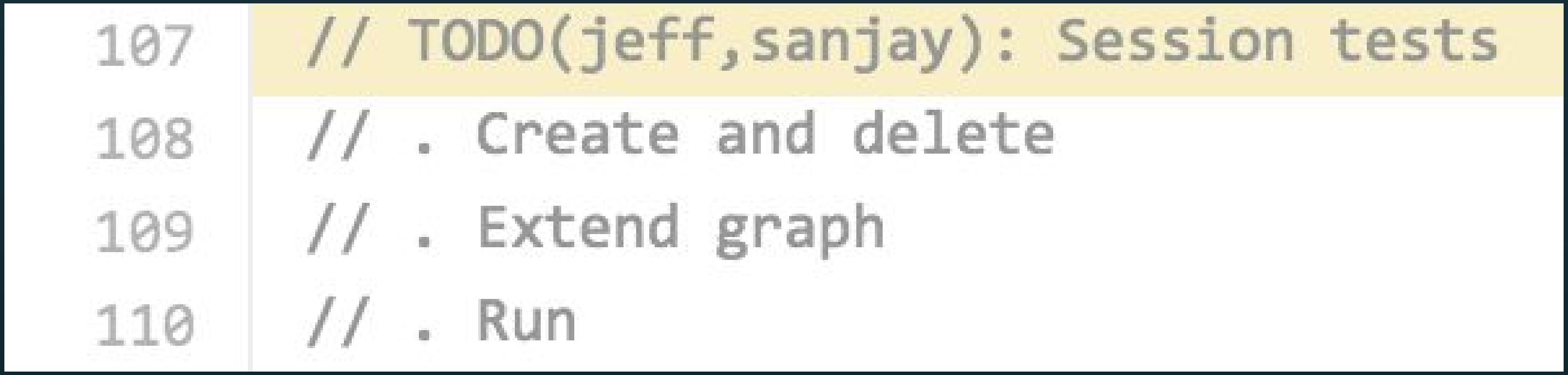


```
// TODO(jeff,sanjay): Session tests
// . Create and delete
// . Extend graph
// . Run
```

in tensor_c_api_test.cc

A tour through the TensorFlow codebase

4. And my favorite **TODO**

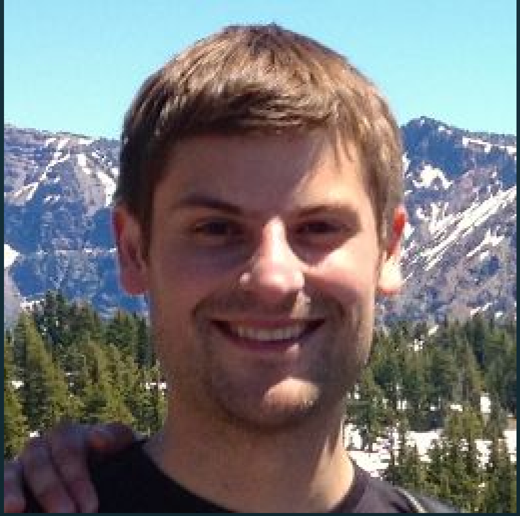


```
// TODO(jeff,sanjay): Session tests
//  . Create and delete
//  . Extend graph
//  . Run
```

in [tensor_c_api_test.cc]()

https://github.com/tensorflow/tensorflow

thanks!



Kevin Robinson

@krob

Teaching Systems Lab, MIT