

Case Study for Tensor Flow

Zhibin Huang

hzhbin@bu.edu

1. Technology and platform used for development

TensorFlow is an open source software library for numerical computation using data flow graphs. The graph nodes represent mathematical operations, while the graph edges represent the multidimensional data arrays that flow between them.

Tensor Flow can be used by Python, C++ and CUDA in different platform like Linux, macOS, Windows and Android. TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.

If the project starting today, I do think the same languages would still be the used because Python and C++ are two of the most common and useful languages in the world. Considering of this, it's good to build the project based on them for future usage and development.

2. Testing: describe unit/integration/module tests and the test framework

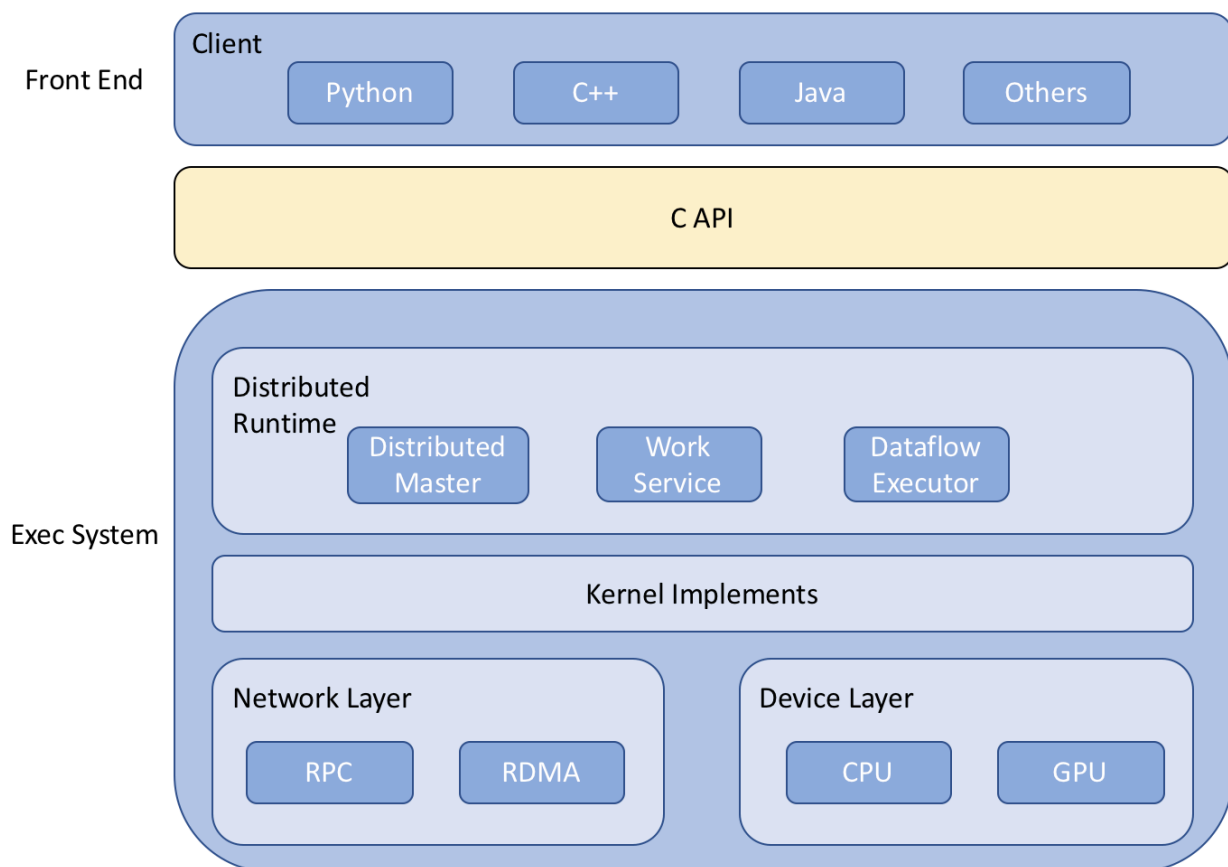
You have two options when running TensorFlow tests locally on your machine. First, using docker, you can run our Continuous Integration (CI) scripts on TensorFlow devel images. The other option is to install all TensorFlow dependencies on your machine and run the scripts natively on your system.

To verify that new changes don't break TensorFlow, we run builds and tests on either Jenkins or a CI system internal to Google. You can trigger builds and tests on updates

to master or on each pull request. Contact one of the repository maintainers to trigger builds on your pull request.

Jenkins is a self-contained, open source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software. Jenkins can be installed through native system packages, Docker, or even run standalone by any machine with a Java Runtime Environment (JRE) installed.

3. Software architecture



TensorFlow System Architecture

The whole system is divided into two-half by C API. Front-end system: provides a programming model responsible for constructing computational graphs; Backend system: Provides a runtime environment responsible for executing calculation graphs.

Client is the main component of the front-end system, it is a multi-language programming environment. It provides a computational graph-based programming model that allows users to construct a variety of complex computational diagrams and implement various forms of model design. The client connects to the "runtime" of the TensorFlow backend through the Session, and starts the execution of the calculation graph.

In a distributed runtime environment, the Distributed Master traverses backwards from the calculation graph based on the Fetching parameter of Session.run to find the "minimum subgraph" that it depends on.

For each task, TensorFlow will start a Worker Service. The Worker Service will perform the OP operation (a typical polymorphic implementation technique) by calling the Kernel of the OP according to the current available hardware environment (GPU/CPU) according to the dependencies between the nodes in the calculation graph.

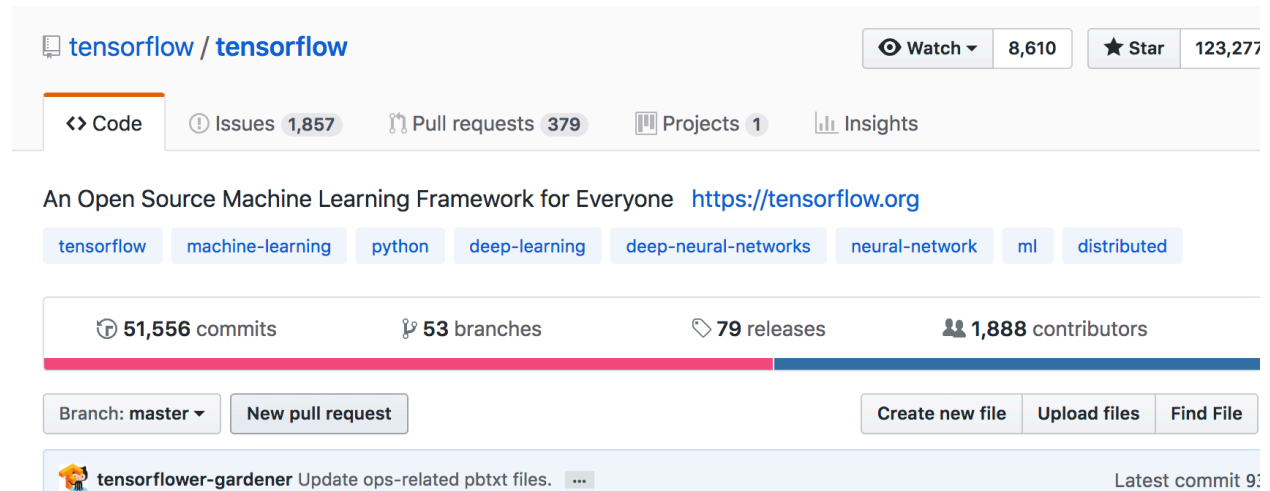
Kernel is a specific implementation of the OP in a hardware device that is responsible for performing OP operations.

When using the TensorFlow, it's so convenient to use the TensorBoard which could realize visual learning. To make it easier to understand, debug, and optimize TensorFlow program, you can use TensorBoard to visualize your TensorFlow graph, plot quantitative metrics about the execution of your graph, and show additional data like images that pass through it.

4. Two defects in TensorFlow

It's true that TensorFlow is the only architecture that support dynamic control flow. But it does do the control flow by themselves in many cases but just record a static graph. If possible, I want to improve this situation by deploy TensorFlow into some core product not just in python, but also in other languages. The other problem

lies in the mass documentation in TensorFlow web, some of the code style and APIs are also in a messy. I could create a new pull request on Github since it's open source. While the problems I found are the root-based, so it's useless even if I create a new pull request on it. Actually, it requires some benefit changes in every pull request.



The screenshot shows the GitHub repository page for TensorFlow. At the top, the repository name 'tensorflow / tensorflow' is displayed. To the right, there are buttons for 'Watch' (8,610), 'Star' (123,277), and 'Fork' (123,277). Below this, there are tabs for 'Code', 'Issues' (1,857), 'Pull requests' (379), 'Projects' (1), and 'Insights'. The main heading reads 'An Open Source Machine Learning Framework for Everyone' with a link to 'https://tensorflow.org'. Below this, there are tags for 'tensorflow', 'machine-learning', 'python', 'deep-learning', 'deep-neural-networks', 'neural-network', 'ml', and 'distributed'. A statistics bar shows '51,556 commits', '53 branches', '79 releases', and '1,888 contributors'. Below the statistics bar, there are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', and 'Find File'. At the bottom, there is a commit list showing a commit by 'tensorflow-gardener' with the message 'Update ops-related pbtxt files.' and a link to the commit.

5. Use TensorFlow in my own computer

You could install it by just type “pip install tensorflow” for python in command window, it's suitable for Ubuntu 16.04 or later, Windows 7 or later, macOS 10.12.6 (Sierra) or later (no GPU support) and Raspbian 9.0 or later. You can also install other versions of TensorFlow by click [here](#).

Here I will deploy TensorFlow in python in macOS environment. As shown below, I import tensorflow as tf and then use some basic function in it. Every time you want to realize a tensor, you should create a session before running it. After done, you should close the session.

```
test.py x
tf_test ~/Docu
venv
test.py
External Librar
Scratches and

1 import tensorflow as tf
2 import os
3 import numpy as np
4 import pickle
5
6 os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
7 a = np.abs([-1,2,-3])
8 print(a)
9 print('generate array')
10 data = [[1,2,3,9],[3,4,7,9],[5,6,8,9]]
11 x = np.array(data)
12 print(x) #print array
13 print(x.ndim) #print shape of array
14 print(x.shape) #print the length of every shape
15 w = tf.Variable([[1,2]])
16 x = tf.Variable([[2],[5]])
17 y = tf.matmul(w,x)
18
19 init_op = tf.global_variables_initializer()
20
21 with tf.Session() as sess:
22     sess.run(init_op)
23     print(y.eval())
24     sess.close()
25
26 def unpickle(file):
27     with open(file, 'rb') as fo:
28         dict = pickle.load(fo, encoding='bytes')
29     return dict

un: test x
/Users/huang/Documents/PycharmProjects/tf_test/venv/bin/python /Users/huang/D
[1 2 3]
generate array
[[1 2 3 9]
 [3 4 7 9]
 [5 6 8 9]]
2
(3, 4)
[[12]]

Process finished with exit code 0
```