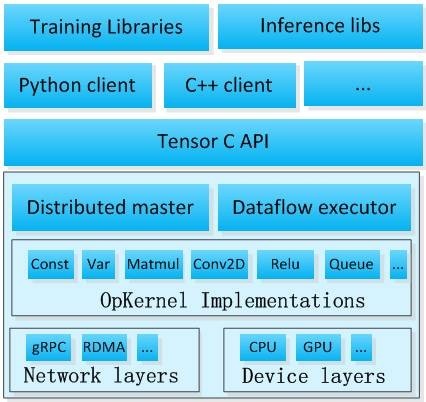
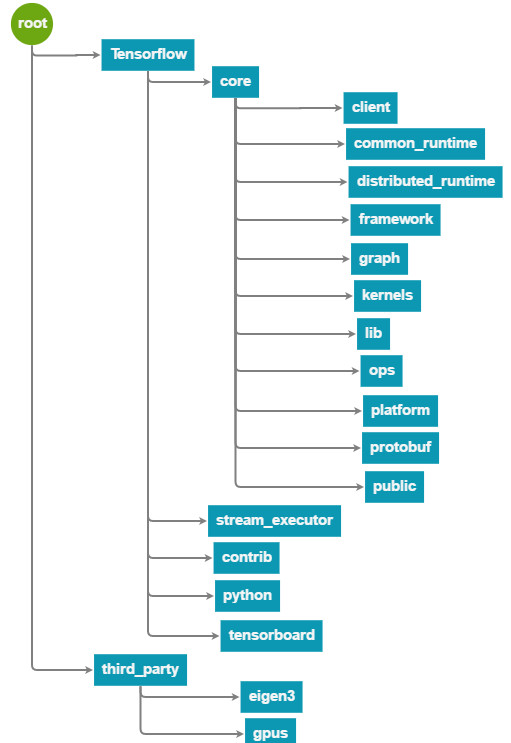
## TF系统架构

图2是TF的系统架构，从底向上分为设备管理和通信层、数据操作层、图计算层、API接口层、应用层。其中设备管理和通信层、数据操作层、图计算层是TF的核心层。



## TF代码结构

图3是TF的代码结构视图，下面将简单介绍TF的目录组织结构。



## TF目录解释

Tensorflow/core目录包含了TF核心模块代码。

* public: API接口头文件目录，用于外部接口调用的API定义，主要是session.h 和tensor\_c\_api.h。
* client: API接口实现文件目录。
* platform: OS系统相关接口文件，如file system, env等。
* protobuf: 均为.proto文件，用于数据传输时的结构序列化.
* common\_runtime: 公共运行库，包含session, executor, threadpool, rendezvous, memory管理, 设备分配算法等。
* distributed\_runtime: 分布式执行模块，如rpc session, rpc master, rpc worker, graph manager。
* framework: 包含基础功能模块，如log, memory, tensor
* graph: 计算流图相关操作，如construct, partition, optimize, execute等
* kernels: 核心Op，如matmul, conv2d, argmax, batch\_norm等
* lib: 公共基础库，如gif、gtl(google模板库)、hash、histogram等。
* ops: 基本ops运算，ops梯度运算，io相关的ops，控制流和数据流操作
* Tensorflow/stream\_executor目录是并行计算框架，由google stream executor团队开发。
* Tensorflow/contrib目录是contributor开发目录。
* Tensroflow/python目录是python API客户端脚本。
* Tensorflow/tensorboard目录是可视化分析工具，不仅可以模型可视化，还可以监控模型参数变化。
* third\_party目录是TF第三方依赖库。
* eigen3: eigen矩阵运算库，TF基础ops调用
* gpus: 封装了cuda/cudnn编程库

## TF代码阅读

1、BUILD使用&编译运行tensorflow自带test代码

掌握如何添加、修改源码，并在源码中编写test代码，编译运行test代码，掌握做源码上做实验的技巧

2、tfgdb使用

3、tensorboard使用

如何添加常用可视化数据显示，各种可视化数据内容作用讲解，知道有哪些，如何用

4、common\_runtime: 公共运行库，包含session, executor, threadpool, rendezvous, memory管理, 设备分配算法等。

session.run具体做了哪些工作

内存和显存如何分配，数据如何传递

scope作用

5、graph: 计算流图相关操作，如construct, partition, optimize, execute等

图如何表示，优化，执行

6-0、kernels: 核心Op，如matmul, conv2d, argmax, batch\_norm等

分析一个最基本的nn\_op（conv2d）、match\_op（matmul）的实现细节，了解如何调用底层

device layer

6-1、eigen3: eigen矩阵运算库，TF基础ops调用

6-2、gpus: 封装了cuda/cudnn编程库

上面两部分的提供了哪些接口

7、contrib/TPU

阅读这部分代码，了解一个graph要在TPU上面运行需要做哪些修改

尝试从这部分代码推断TPU如何做到兼容常规的graph

8、添加一个OP

添加一个OP需要改动的文件list，相对路径，各个改动的作用（声明，实现，注册，编译...）

9、python wrap实现

C++到python的接口通过什么技术实现，了解这部分内容并写个简单的小demo讲解，并从tensorflow代码中扣一个简单的例子出来讲解，例如tf.conv2d如何实现

10、自定义数据集，熟悉tf.data功能

图像方面训练数据保存格式都哪些，如何将自己采集的数据处理成数据集

11、并行计算

//Tensorflow/stream\_executor目录是并行计算框架，由google stream executor团队开发。

使用多线程需要哪些步骤，代码实现简单demo，CPU、GPU如何协同工作

这部分并行计算实现的是哪个层级的并行

12、序列化

//protobuf: 均为.proto文件，用于数据传输时的结构序列化.

数据如何序列化传输，哪些地方用到了序列化传输, 数据如何在PC和显卡之间传输

梯度，feature，weight等在序列化传输的时候如何降低精度，例如，根据很多资料可知梯度信息并不需要很高的位宽，如何降低位宽加速训练

13、多机多卡并行训练

//distributed\_runtime: 分布式执行模块，如rpc session, rpc master, rpc worker, graph manager。

docker安装及使用，机器如何配置，net是否需要修改,estimator API分析

分布式训练的时，参数服务器和worker服务器都需要做哪些工作，BP如何进行

14、解析tflite文件格式

tflite格式存储的时候放弃了protobuf，使用flatbuffers，好处是反序列化的时候不需要解码

了解flatbuffers的好处，如何利用

学习lite里面解析weight、bias、graph的代码，将对应的解析代码抠出来用于ndm

15、尝试fake quant 低比特训练

fakequant代码可以实现2-16bit的训练，尝试2/4bit时的训练效果

16、optimizer

常用optimizer有哪些，各个optimizer实现公式，优点缺点分析

17、损失函数&梯度

常用损失函数有哪些，各个损失函数公式，作用以及和激活函数存在的配合等

如何添加梯度计算代码，如何中断梯度回传