第五章 编程框架机理

TensorFlow设计原则

1.高性能

1. 算子优化
2. 计算图优化
3. 调度器可以根据网络特点，并发运行没有依赖的数据节点

2.易开发

3.可移植

TensorFlow计算图机制

1.计算图的自动求导

常用求导方法：

1.手动求导

传统的反向传播算法

2.数值求导

利用导数的原始定义求导（一开始直接代入数值近似求解）

3.符号求导

利用求导规则对表达式自动操作（直接对代数表达式求解，最后才带入问题数字）

4.自动求导

介于数值求导和符号求导的方法

计算图结构天然适用自动求导

2.检查点机制

使用tf.trian.Saver()保存模型

saver.restore恢复变量

3.TensorFlow控制流

使用控制流算子实现不同复杂控制流场景，TensorFlow中，每一个操作都会在一个执行帧中被执行，控制流操作负责创建和管理这些执行帧

Switch

Merge

Enter(name) 输入推向执行帧

Exit　将一个张量从一个子执行帧推向它的父执行帧

NextIteration

```
cond(pred, true_fn, false_fn)
```

循环操作

while_loop(pred,body,loop_vars)

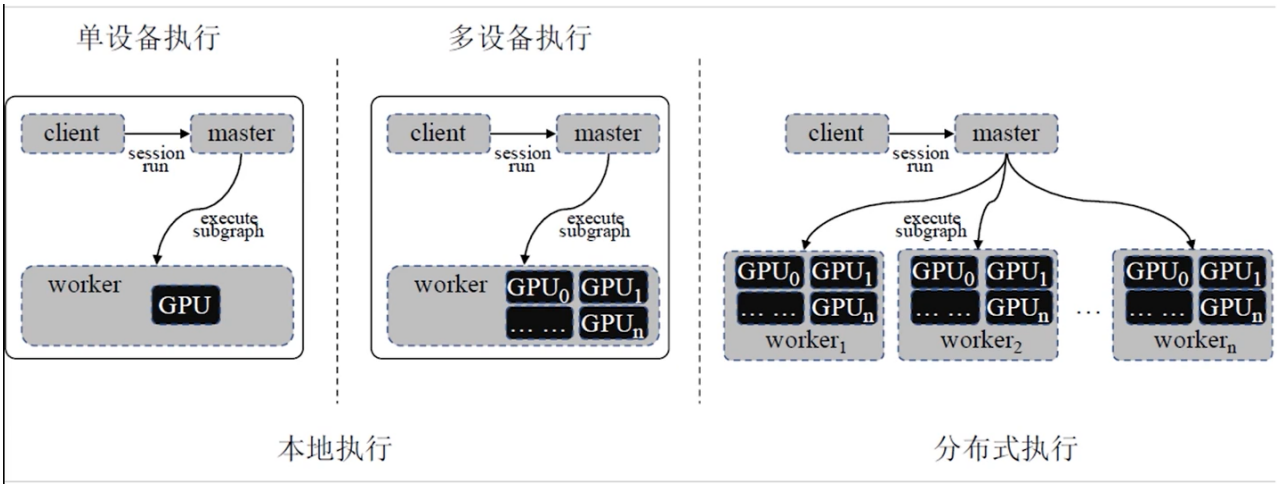
4.计算图的执行模式

client：通过session接口与master和worker接口通信

master：控制所有的worker按照计算图执行

worker：每一个worker负责一个或多个计算设备仲裁访问，并根据master指令执行计算设备中的计算图节点

设备



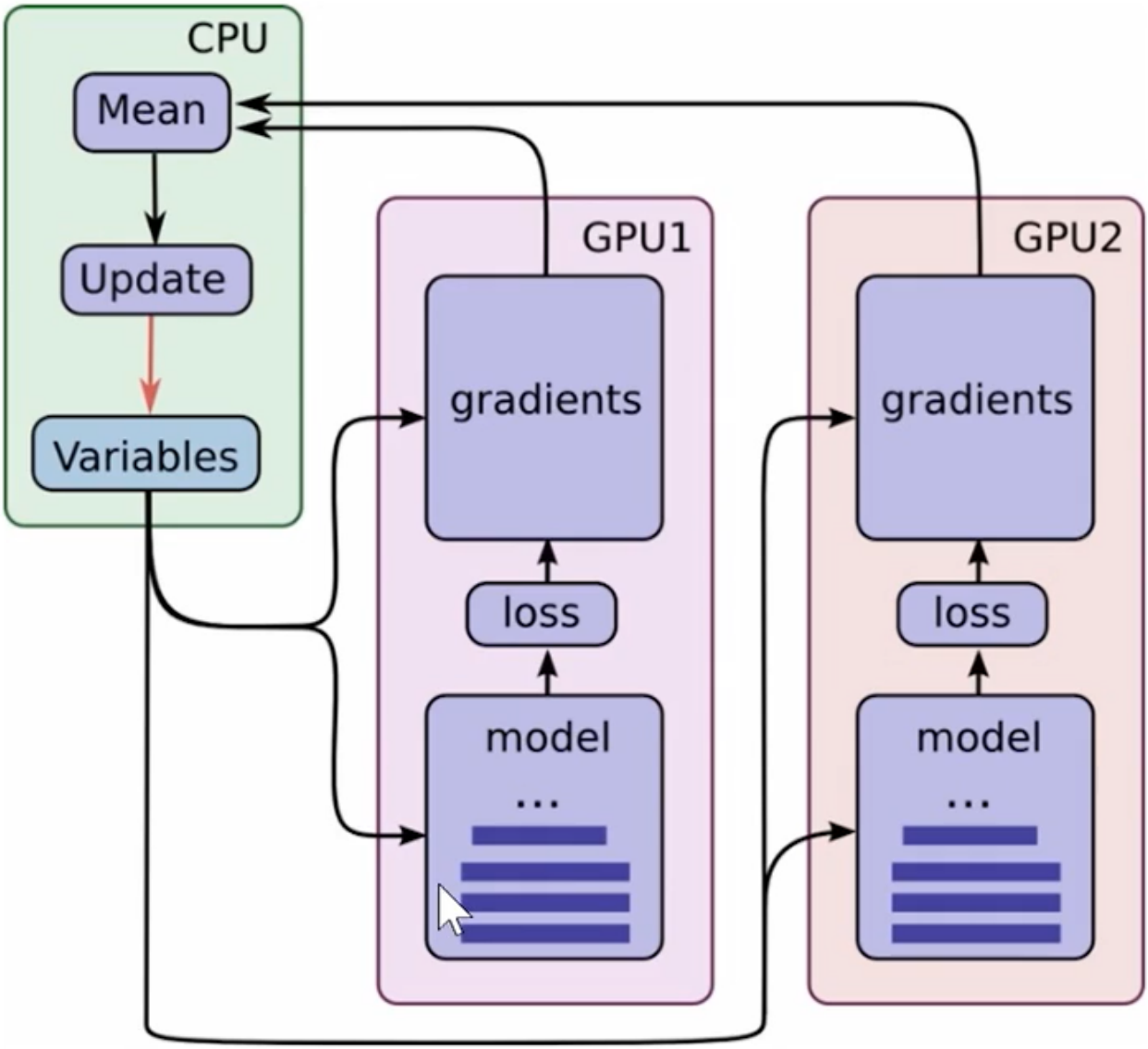
本地单设备执行

一个worker进程中仅包含一个设备情况

本地多设备执行

CPU作为参数服务器，用于保存参数和变量，计算梯度平均

GPU作为worker，用于模型训练



分布式执行

client,master,worker可以工作在不同机器上的不同进程中

5.计算图本地执行

1.计算图剪枝

目的：得到本地运行最小子图

包括：

- 为输入输出建立与外界的交互

FunctionCallFrame函数调用帧来解决输入输出值传递问题

在每个输入节点前插入Arg节点，所有的输入节点连接到Source节点上，并通过控制依赖边相连

在每个输出节点后面加入RetVal节点，所有的输出节点连接到Sink节点上，也通过控制依赖边相连

- 去除与最终输出节点无关的节点和边

从输出节点开始进行宽度搜索遍历，删除没有接触到的节点和边

将每个连接图中入度为0的节点通过控制依赖边与source节点相连，出度为0的节点通过控制依赖边和sink节点连

2.计算图分配

问题：多设备运行环境中，对计算图每个节点如何分配计算设备

3.计算图优化

TensorFlow的图优化由Grappler模块实现

图优化，可以根据硬件结构调整计算调度策略

能减少推断过程所需的峰值内存

ConstFold 常量折叠

有的常熟节点可以被提前计算，用得到的结果生成新的节点来代替原来的常数节点

- MaterializeShapes 处理与Shape相关的节点
- FoldGraph 对每个节点输入进行检测，如果为Const,提前计算其中的值
- SimplifyGraph 简化节点中的常量运算

Arithmetic 算数简化

俩部分：公共子表达式消除，算数简化

Layout 布局优化

TensorFlow中默认采用NHWC

GPU中采用NCHW

Remapper 算子融合

将出现频率高的子图用一个单独算子来替代，提高计算效率

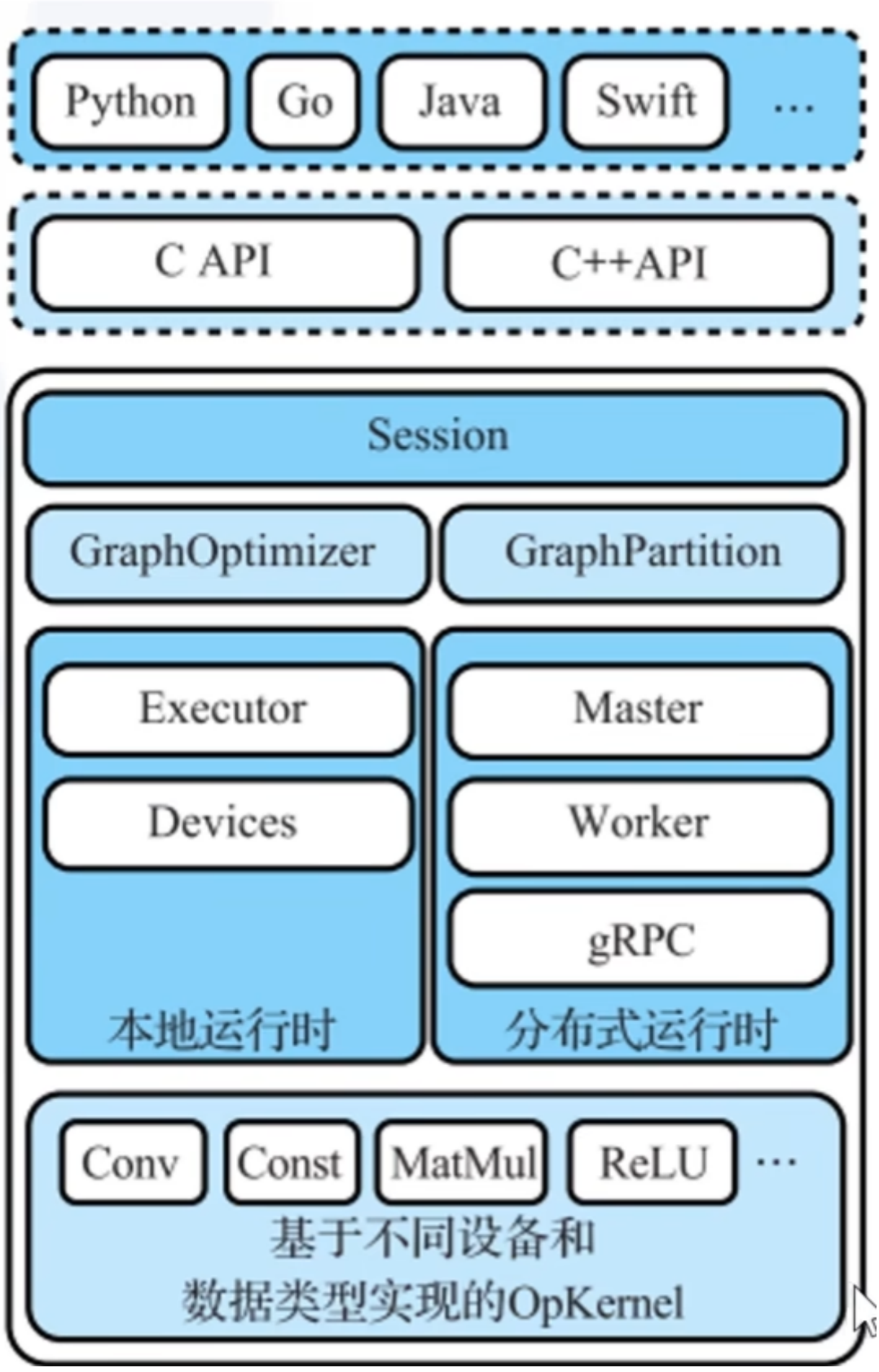
好处：

- 消除子图调度开销
- 计算Conv2D +BiasAdd , Conv2D的数据处理是分块进行，融合后的BiasAdd也可以在片上存储里进行

4.计算图切分

TensorFlow系统实现

1.整体架构



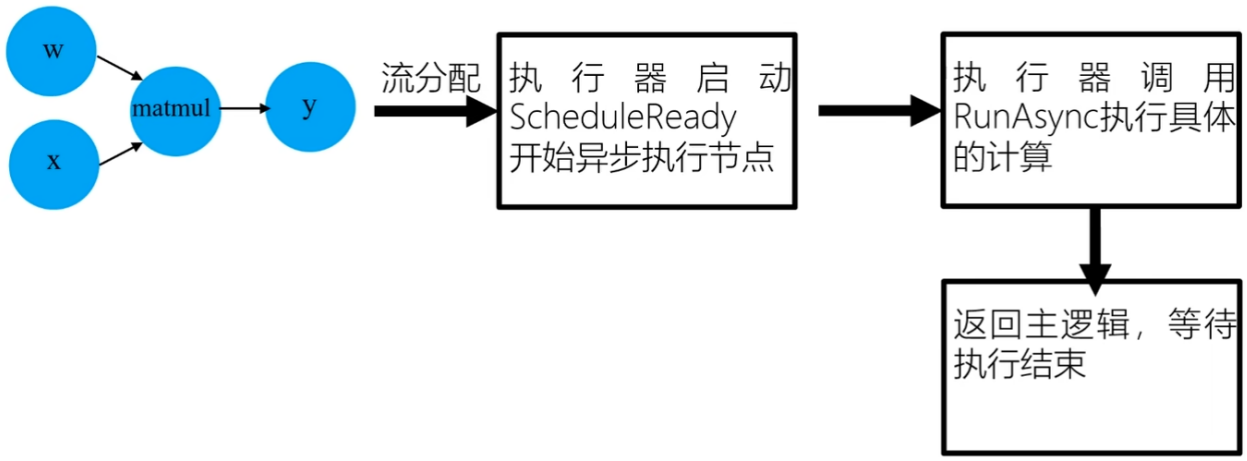
2.计算图执行模块

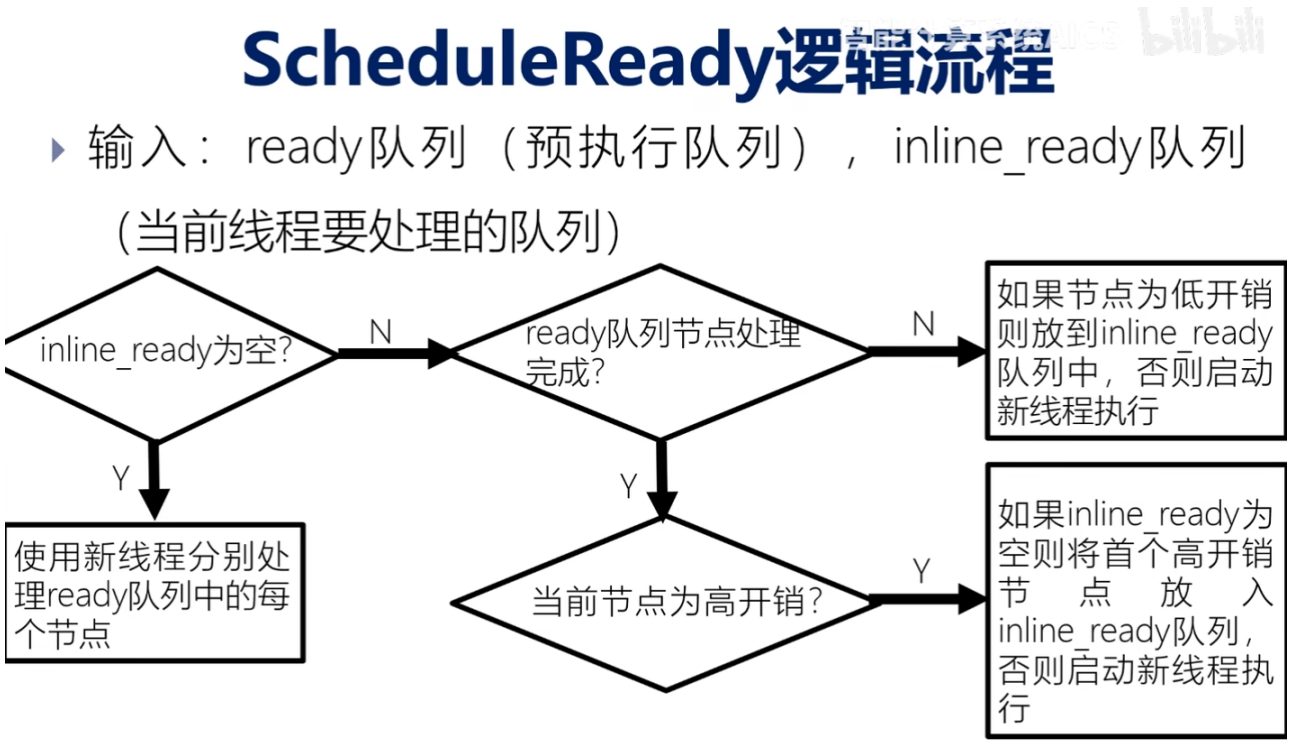
Session是用户与TensorFlow运行时的接口。在Session接受到输入数据时，便可执行

执行器逻辑

执行流（stream）：一个能够存储计算任务的队列

流间任务可以并行执行，流内任务串行执行





3.设备抽象和管理

TF将设备分为本地和远程俩类

TF通过注册机制管理设备

4.网络和通信

TF设备间通信由Send和Receive节点进行，使用Rendezvous机制完成数据交互

Rendezvous 提供了最基本的**Send**、**Recv**和RecvAsync接口和实现

TF提供了LocalRendezvous实现类

5.算子实现

算子是TF基本单元，OpKernel是算子特定执行，依赖于底层硬件

TF通过注册机制来支持不同算子和相应的OpKernel函数

驱动范例

编程框架对比