TensorFlow技术内幕（一）：导论

本篇中我将介绍tensorflow的设计。

简单历史回顾

官网地址：www.tensorflow.org
github地址：github.com/tensorflow/tensorflow
模型仓库地址：github.com/tensorflow/models

Tensorflow基础架构

图1:TensorFlow基础架构

Tensorflow前端支持多种开发语言，包括Python,C++，Go,Java等，出于性能的考虑，后端实现采用了C++和CUDA。采用Tensorflow实现的算法可以在许多不同的系统上运行，包括Android，IPhone，普通CPU服务器，以及GPU集群等。

如图1所示，Tensorflow通过分层是设计，将上层的算法实现相与不同的硬件和操作系统平台隔离，使得TensorFlow可以方便的部署到各种平台，实现了一处编码，处处执行。

Tensorflow的设计中，抽象出了张量、计算图、节点、运算、运算核等概念，下面我们将一一介绍。

核心概念

我们通过一个具体的例子来解释TensorFlow中的核心概念。

```
import tensor flow as tf

#生成一个1维度向量，长度为100，初始化为0
```

```
b=tf.Variable(tf.zeros([100]))

#生成一个二维数组，大小为784x100,随机初始化
W=tf.Variable(tf.random_uniform([784,100],-1,1))

#生成输入的Placeholder，计算的时候填入输入值
x=tf.palceholder(name="x")

#计算最终输出
s=tf.matmul(W,x) + b
out=tf.nn.relu(s)

#开始计算
with tf.Session() as sess:
    r = sess.run(out, feed_dict={x:input})
    print(r)
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20

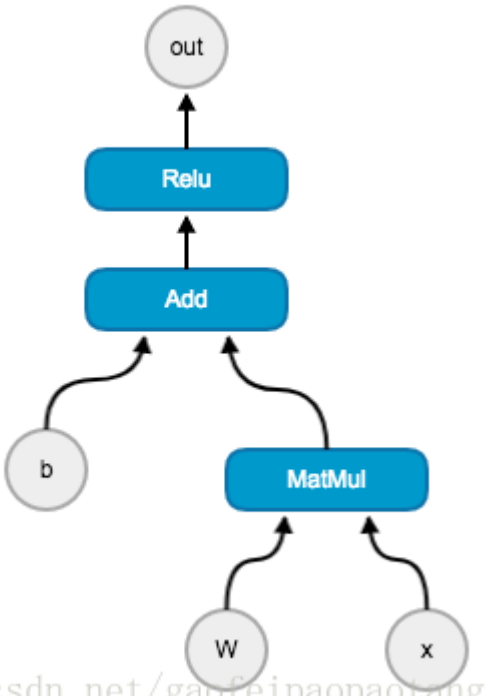

图2：计算图（Graph）例子

这个例子中，我们算法输入是 x x x,输出是 o u t = R e l u (w x + b) out=Relu(wx+b) out=Relu(wx+b).

计算图(graph): TensorFlow中的计算都可以表示为一个有向图(directed graph),图2就是一个计算图(Graph)。

节点(node): 计算图中的每个运算操作，比如Add,Relu,Matmul就将作为节点，b,w,x,out等也是节点

运算(operation)： 例如Add,Relu,Matmul，运算代表一种类型的抽象运算。运算可以有自己的属性，但是所有属性必须被预先设置，或则能在计算的时候被推断出来。

运算核： 是一个运算在一个具体硬件(GPU,CPU,ARM等)上的实现。

张量（Tensor）： 张量是对Tensorflow中所有数据的抽象，张量有自己的维度和size。有自己是数据类型，比如浮点，整形等。张量沿着计算图的边流动，这也是平台名称tensorflow名字的来源。

会话（session）： 会话包含了计算图运行的所有上述信息。

TensorFlow的运行方式

客户端通过Session接口，与master和多个worker相连。其中每个worker管理着一个或则多个硬件设备(device),可以是CPU,GPU等等；master负责规划和协调所有worker按流程执行计算图。

TensorFlow运行方式有单机模式和分布式模式两种，区别在于客户端、master、worker分布的位置不同；具体来说，单机模式中，客户端、master、worker在同一个进程中，当然这就意味着肯定是在同一台机器上；分布式模式中，客户端、master、worker在不同的进程中，可以在同一台机器上，也可以在不同的机器上。

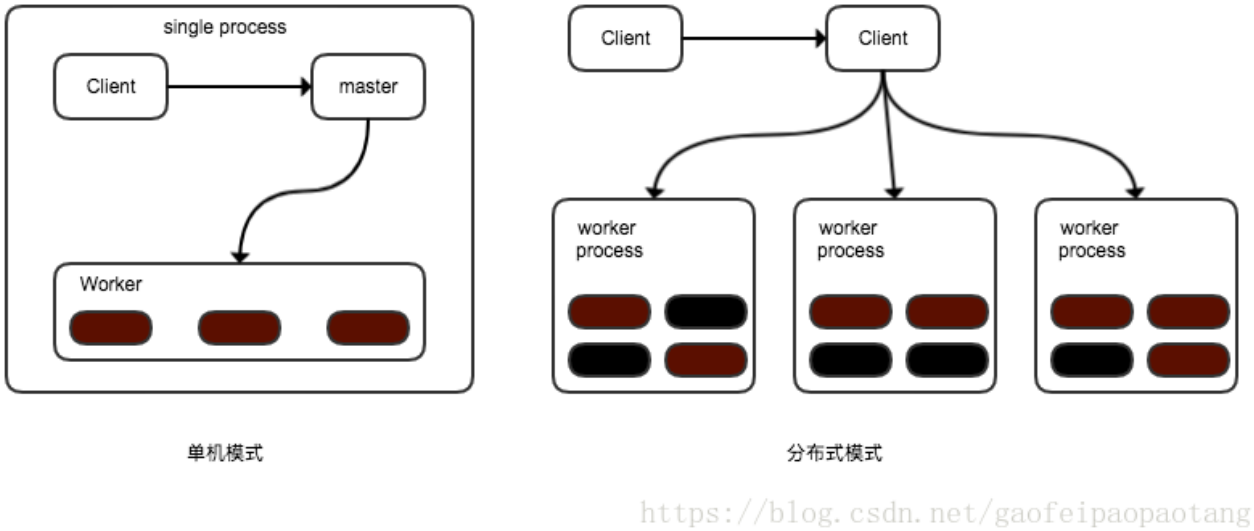

图3：单机模式（左）与分布式模式（右）

两个难题

当只存在一个设备的时候，情况比较简单，计算图的中的节点按照输入输出的依赖关系顺序执行就可以。当任意一个节点的所有依赖的上游节点执行完成后，就可以执行当前节点了。这实际上就是按照计算图的拓扑排序执行。

当存在多个设备，甚至多个异构的设备的时候，问题就出现了，我们介绍以下两点：

(1)、计算图的每个节点应该怎么在哪个硬件上执行呢?
tensorflow设计了一套为节点分配设备的策略。这个策略中我们
1，首先需要计算一个代价模型，估算每一个节点的输入、输出张量的大小和所需要的计算时间。这个代价模型一部分是由人工经验制定的启发式规则，另一部分是对一小部分数据进行实际运算而得到的。

2，然后，分配策略会模拟执行整个计算图，首先会从起点开始，按拓扑顺序执行。在模拟执行一个节点时，会把每个能执行这个节点的设备都测试一遍，这个测试会考虑代价模型对这个节点的计算事件的估计，加上数据传到这个设备上所需要的通信时间。

3，最后选择一个综合时间最短的设备作为这个节点的运算设备。

看得出这个策略是一个简单的贪婪策略，不能保证找到全局的最优解，但是可以用最快的速度找到一个不错的解。

除了运行时间，内存的最高使用峰值也会被考虑进来。

另外，Tensorflow还允许用户对节点的分配设置一些限制条件。例如只分配GPU类型的设备，只分配/job:worker/task:3上的设备，等等。

(2)、在不同硬件上的节点如果通信呢?

我们知道，在存在多设备的情况下，整个计算图的节点会由分配策略分配到不同的设备上执行，那么张量是怎么跨越设备在计算图中流动的呢？也就是一个设备中的节点的输出，是怎么传输到另一个设备中的下游节点的呢？答案如下图所示：

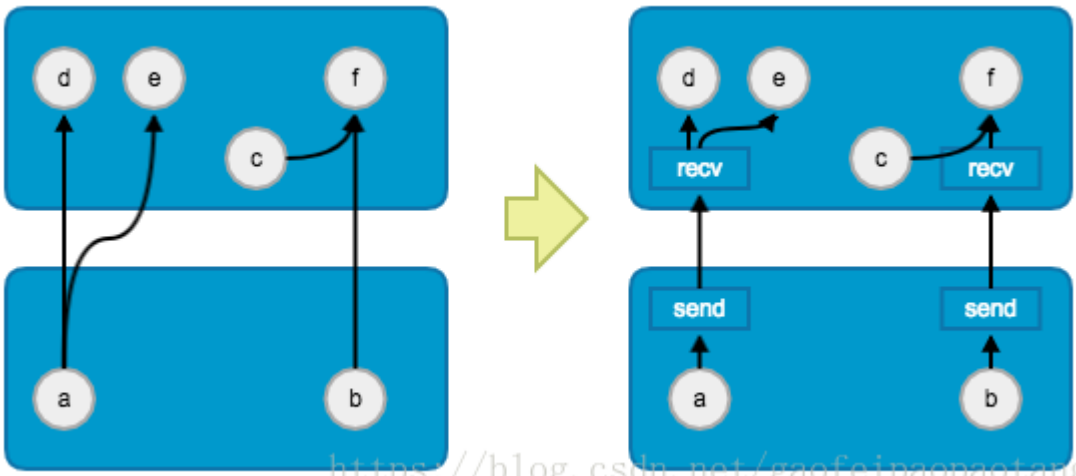

图4：节点间跨设备通信方式

tensorflow采用的设计是在需要夸设备通信的节点之间插入Send和Recv节点，通信节点的插入和底层是通信方式，都是用户透明的，用户无需关心。

性能优势

TensorFlow在大规模分布式系统上的并行效率相当高,如下图所示：

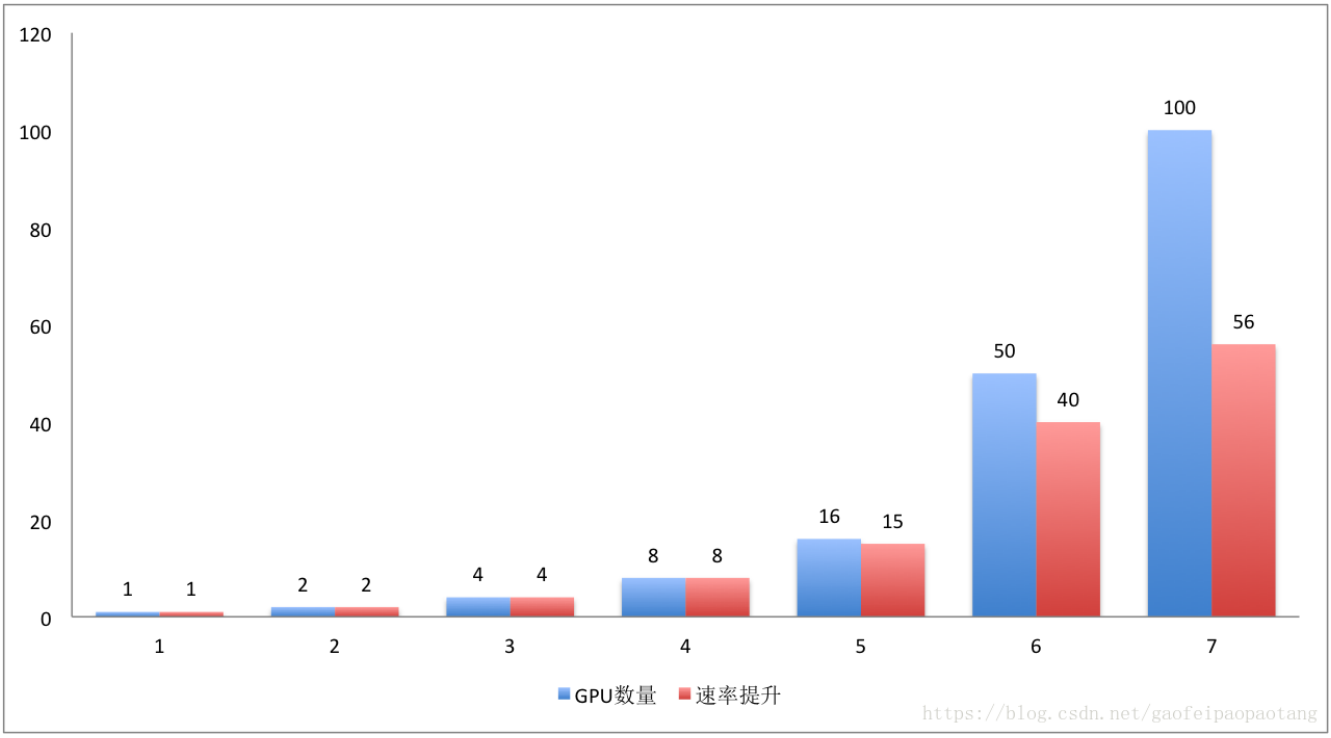

图5：TensorFlow并发效率

在GPU数量小于16时，基本没有性能损耗，在50块的时候，可以获得80%的效率，也就是40倍的单GPU提速。100块的时候，获得56块的提速。

为了达到这种高效并发性能，tensorflow做了很多优化，包括单不限于以下几点：

子图消重： 在Tensorflow中有很多高层的运算操作，这些运算操作可能是有很多复杂的底层计算组合而成的，当有很多个高层运算存在时，它们的前几层的运算可能是重复计算的(输入何运算内容都一样)。tf会自动识别出这些重复的计算，然后通过改写计算图，共享计算结果，消除重复计算量。

计算顺序优化： 通过调整节点的执行顺序，改善数据传输何内存占用问题。例如错开某些节点的计算时机，避免某些数据同时存在于内存，这对于先显存有限的GPU设备来说至关重要。

复用高效的第三方计算库： 包括线性代数计算库Eigen, 矩阵计算库BLAS，cuBLAS，深度学习计算库cuda-convnet,cuDNN

节点分配设备策略的持续优化： 持续优化节点执行设备的分配策略，未来计划用一个强化学习的网络辅助分配策略。

XLA编译优化： 通过编译优化加速Graph的计算。

多重并行计算模式： TensorFlow提供三种不同的加速神经网络训练的并行计算模式，分别是数据并行，模型并行，流水线并行：

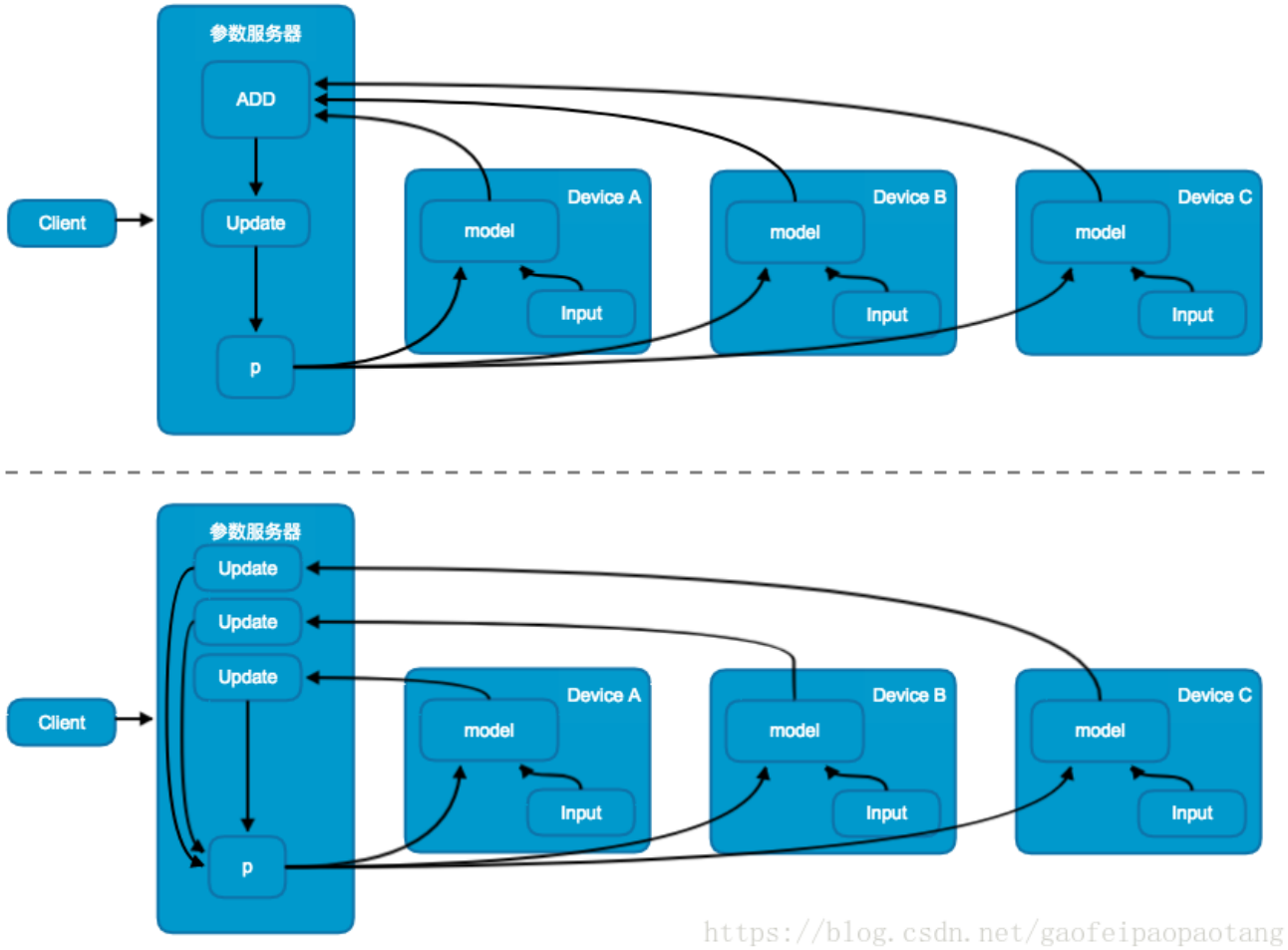

图6：数据并行模式中的同步更新模式（上）与异步模式更新（下）

数据并行模式中，模型在不同的设备上存在相同多份拷贝，共享相同的参数，采用不同的训练数据并行训练。

根据共享参数的更新方式，又分为同步更新模式与异步更新模式；同步更新模式中（图6上），参数的更新值需要进行汇总，然后更新共享参数，这就意味着，需要等待所有的设备当前训练批次训练完成后才能更新共享参数；而异步更新模式中(图6下)，不用进行更新值的汇总，每台设备当前批次训练完成后分别更新共享参数，避免了等待的问题。

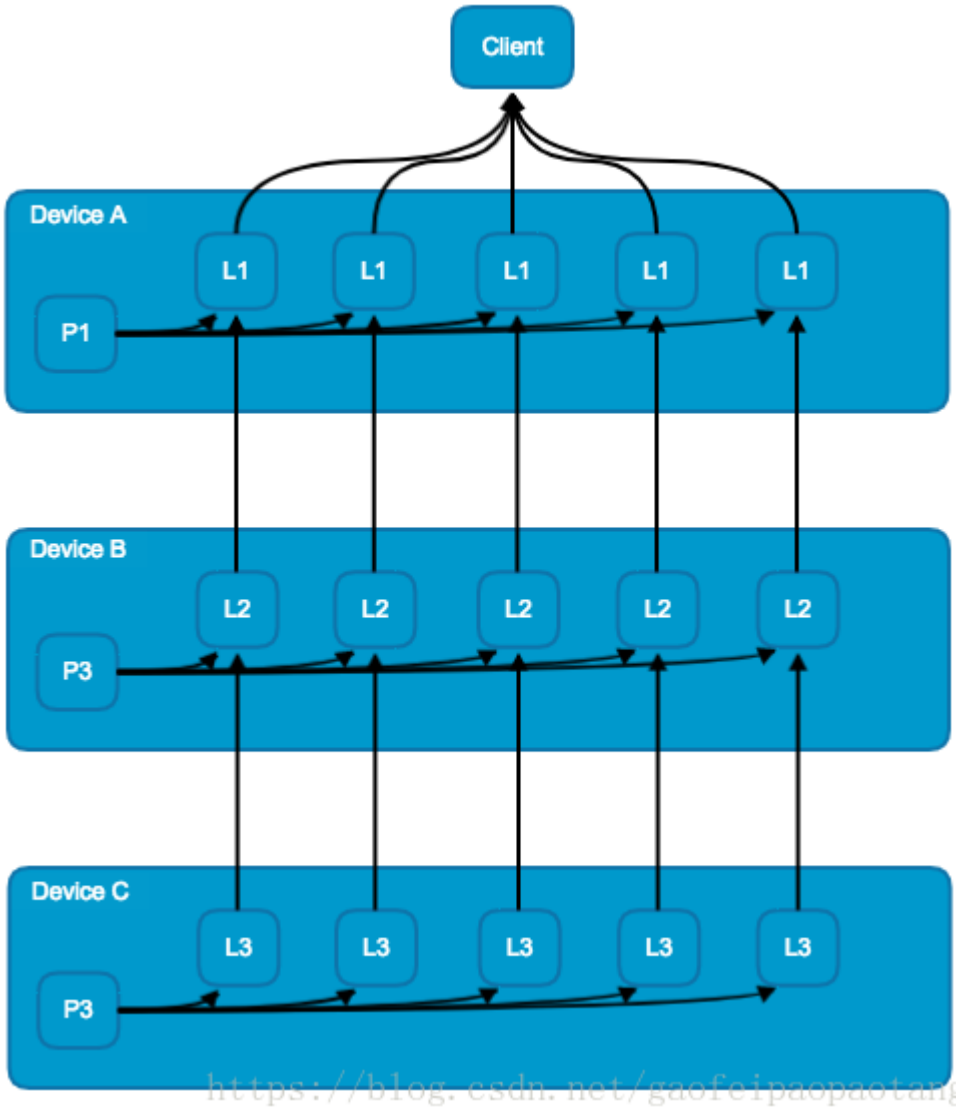

图7：模型并行模式

模型并行模式中，是将模型的不同部分别放在不同的机器上进行训练。模型并行需要模型本身有大量的可以并行的，互相不依赖的或则依赖程度不高的子图。

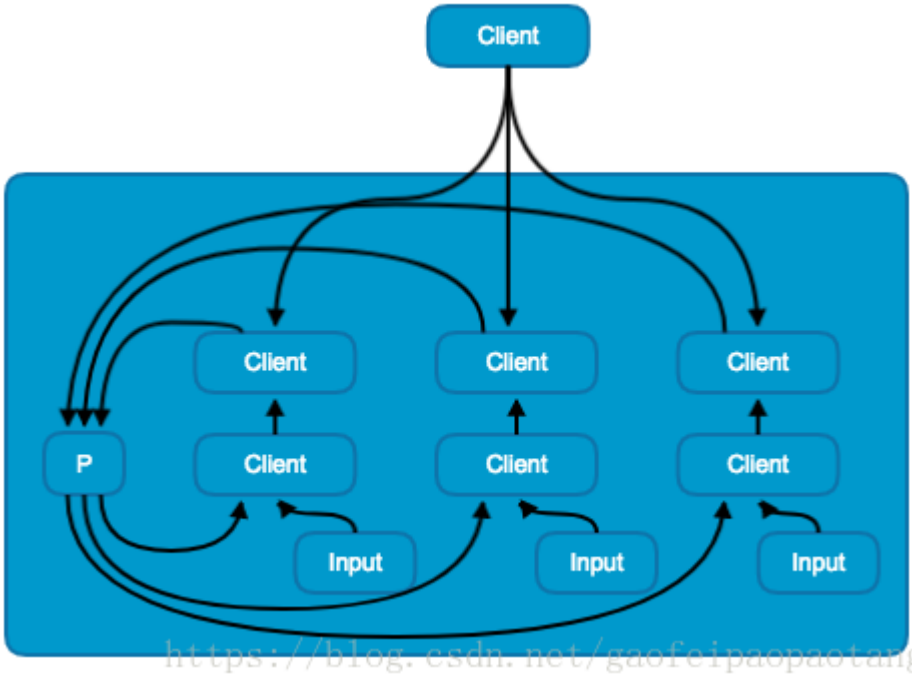

图8：流水线并行模式

流水线并行模式与数据并行模式类似，区别在于流水线并行模式是在单机上训练的。