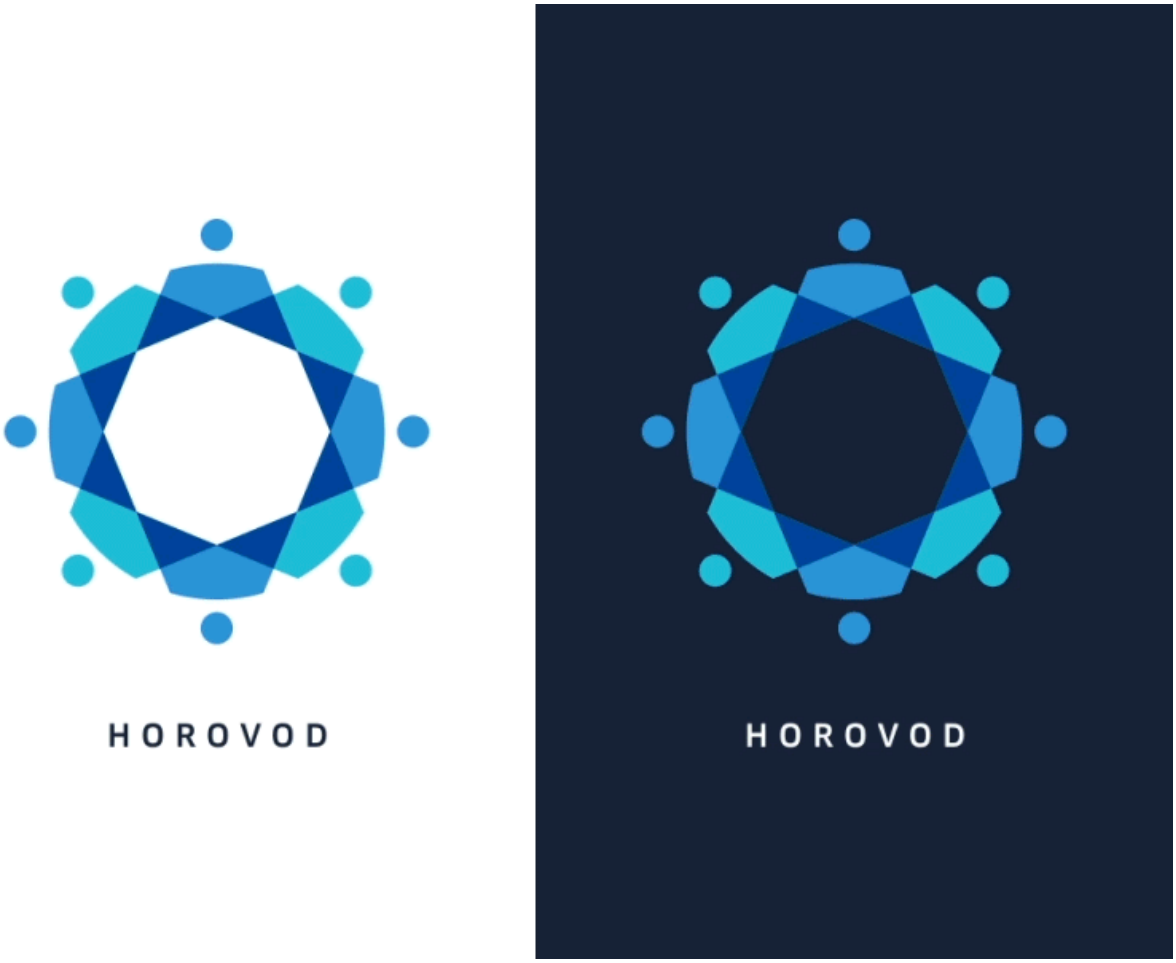
深度学习分布式训练相关介绍 - Part 2 详解分布式训练架构 PS-Worker与Horovod

zhuanlan.zhihu.com/p/70603273

Zhang Bin 心有猛虎 细嗅蔷薇



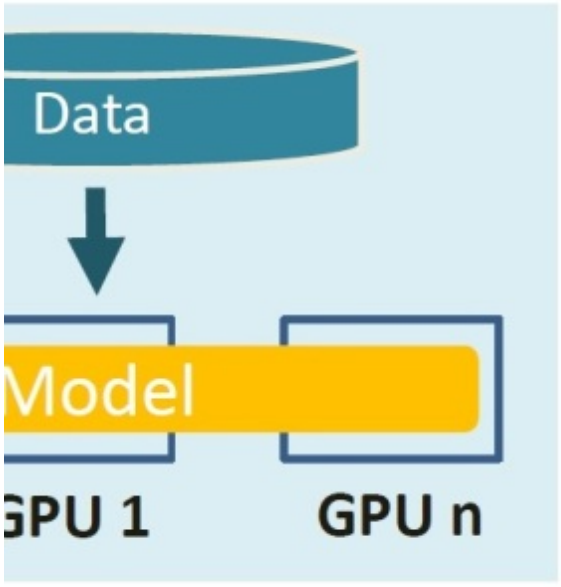
之前写的一篇文章：

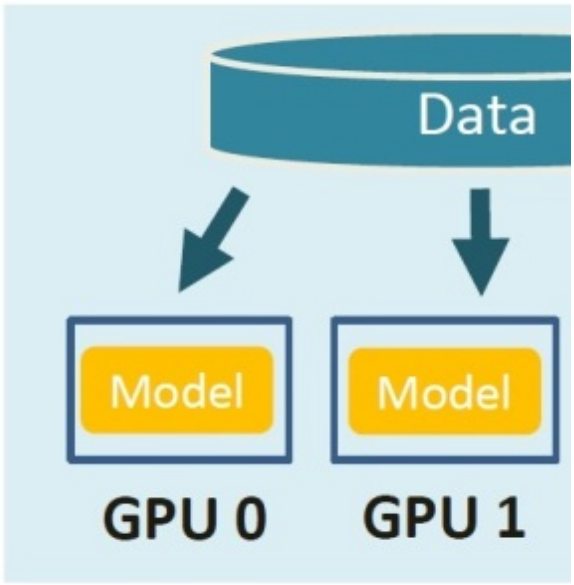
Zhang Bin：深度学习分布式训练相关介绍 - Part 1 多GPU训练 zhuanlan.zhihu.com

我们介绍了多GPU机器的训练方式，可以利用一台机器上的所有GPU设备进行模型的训练，但是一般我们的服务器最多只支持装下8张GPU卡，怎样进一步利用更多的将几十甚至几百台服务器的GPU资源来提升我们的训练效果呢？

在这篇文章中，我们将探讨：

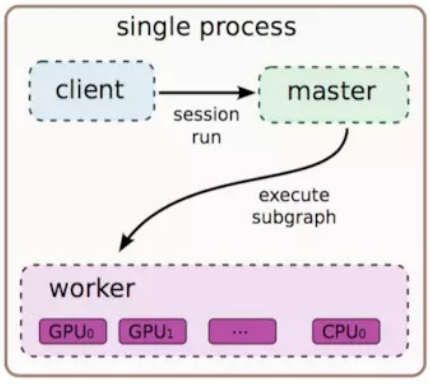
- TensorFlow原生的PS-Worker架构是如何进行分布式训练的？
- 为什么Uber提出的Horovod框架相比于TensorFlow原生 PS-Worker架构 有非常高的GPU效率优势？
- 如何通过Horovod进行分布式训练？

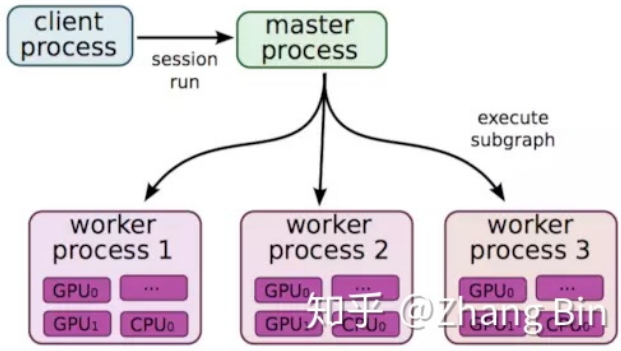




多机多卡 分布式训练

但相比于单机多卡，多机多卡分布式训练方式的配置更复杂一些，不仅要保证多台机器之间是可以互相通信的，还需要配置不同机器之间的角色以及不同机器之间梯度传递。



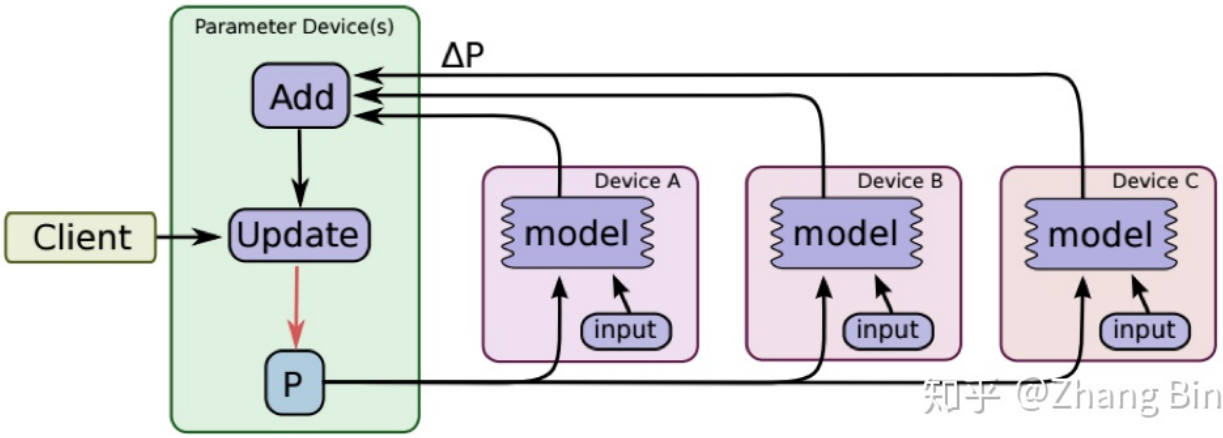


1. TensorFlow 原生 PS架构

在Parameter server架构（PS架构）中，集群中的节点被分为两类：参数服务器（parameter server）和工作服务器（worker）。其中参数服务器存放模型的参数，而工作服务器负责计算参数的梯度。在每个迭代过程，工作服务器从参数服务器中获得参数，然后将计算的梯度返回给参数服务器，参数服务器聚合从工作服务器传回的梯度，然后更新参数，并将新的参数广播给工作服务器。

PS-Worker 架构的梯度更新有着 **同步更新** 和 **异步更新** 两种方式：

同步更新



在同步训练中， 所有的Worker设备采用同一个Batch的不同小批(mini-batch)数据来训练，等待所有设备该批次的梯度计算完成后，模型才会根据所有的梯度进行一次参数更新，然后PS将更新后的模型下发到各个设备。

虽然同步训练可以在一定程度上加快我们的训练，但同步模式受网络环境影响较大，且存在着**木桶效应**，计算能力强的设备需要等待计算能力差的设备，**一个拖油瓶会严重拖慢训练的进度**。

异步更新

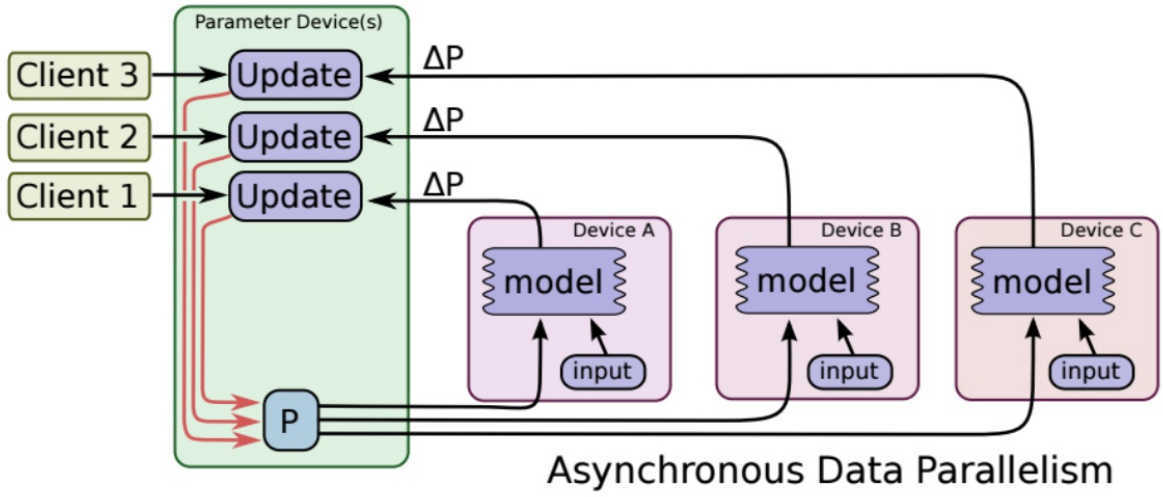


Figure 7: Synchronous and asynchronous data parallel training

异步训练中，没有设备需要去等待其他设备的梯度计算和参数更新，所有设备独立算并与将梯度结果更新到中心节点（PS）。异步训练总体会**训练速度会快很多**，但是异步训练的一个很严重的问题是**梯度失效问题（stale gradients）**，刚开始所有设备采用相同的参数来训练，但是异步情况下，某个设备完成一步训练后，可能发现模型参数已经被其它设备更新过了，此时这个设备计算出的梯度就过期了。

通过TensorFlow原生的PS-Worker架构可以采用分布式训练进而提升我们的训练效果，但是实际应用起来并不轻松：

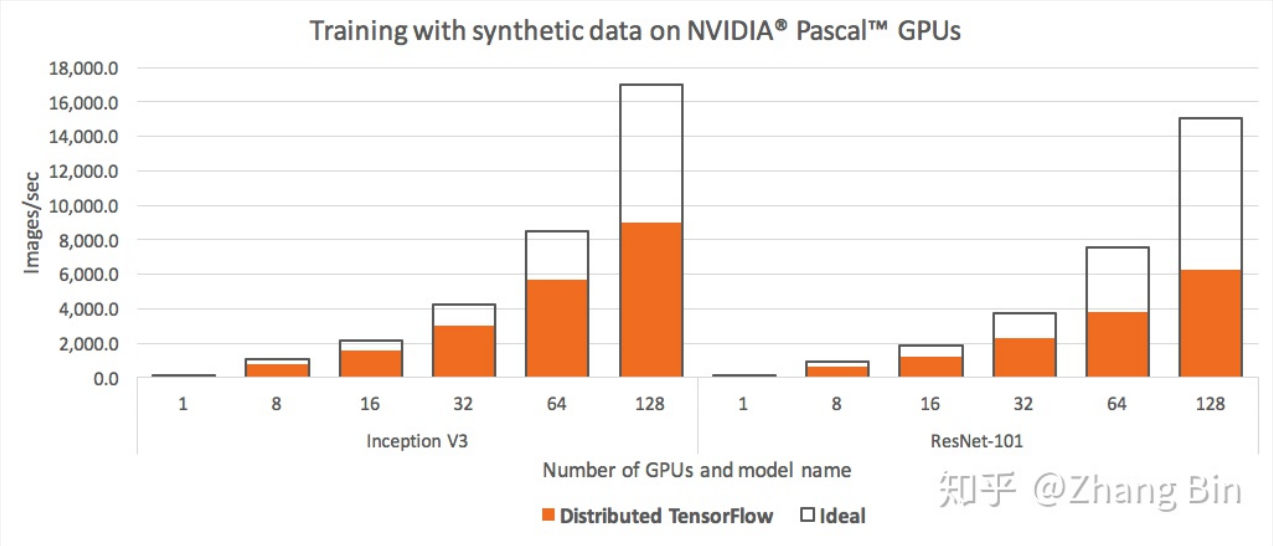
- **难以确定工作服务器与参数服务器的正确比例**：如果只使用一个参数服务器，它可能会成为网络或计算瓶颈。 如果使用多个参数服务器，则通信模式变为“All-to-All”，这可能使网络互连饱和。
- **处理增加的TensorFlow程序复杂性:** 每个使用分布式 TensorFlow 的案例都需要指定初始工作线程和参数服务器，传递服务发现信息，如所有工作线程和参数服务器的主机和端口，并使用合适的 tf.ClusterSpec() 构建 tf.Server()，进而调整训练程序。此外，用户必须保证所有的操作都正确地使用 tf.train.device_replica_setter()，并使用 towers 让代码符合服务器中多 GPU 的设置。这通常导致**陡峭的学习曲线**和**大量的代码重构**，压缩了实际建模的时间。

并且中心化的PS-Worker架构的性能并不理想：

- 假设模型的总参数量为 X
- PS的数量为1， Worker的数量为 N
- 设备间数据最大传输速度为 B

由于 PS 要接收所有Worker的梯度，然后再将更新后模型的参数发送给所有Worker， 因此数据通信的耗时为： $2(X \cdot N)/B$ ， 随着N的线性增加，每一个iteration中数据通信所占用的时间也成线性增加。因此当GPU设备增加到一定规模的时候每一个Batch的训练将耗费大量的时间在不同设备间的通信上。

实际性能测试可以看下图：



可以看到当GPU规模达到64个以上的时候，其运行性能达到不到理想性能的一半。

2. Horovod架构

Horovod 是一套支持TensorFlow, Keras, PyTorch, and Apache MXNet 的分布式训练框架，由 Uber 构建并开源，Horovod 的主要主要有两个优点：

1. 采用Ring-Allreduce算法，提高分布式设备的效率；
2. 代码改动少，能够简化分布式深度学习项目的启动与运行。

下面将分别介绍以上两点

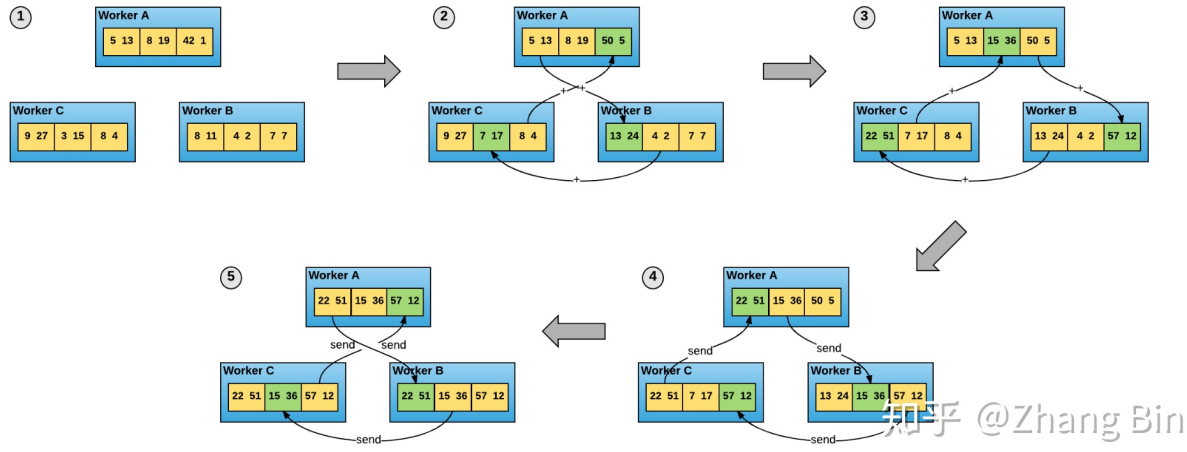
Ring-Allreduce 架构

Ring-Allreduce 算法最早由 Baidu Silicon Valley AI Lab (SVAIL) 在2017年2月提出，并应用在其PaddlePaddle平台上，同年8月，SVAIL将其提交到tensorflow的contrib package中。

也是在同年8月，Uber发布了基于Ring-Allreduce算法和OpenMPI通信的Horovod架构。

Ring-Allreduce 的命名中Ring意味着设备之间的拓扑结构为环形，Allreduce则代表着没有中心节点，架构中的每个节点都是梯度的汇总计算节点。

此种算法各个节点之间只与相邻的两个节点通信，并不需要参数服务器。因此，所有节点都参与计算也参与存储。



使用 Ring-Allreduce 算法进行分布式训练基本过程如下：

1. 每个设备根据各自的训练数据分别进行梯度的计算，得到梯度
2. 将每个设备上的梯度向量切分成长度大致相等的 N 个分片（其中分片数N与设备数量相等）
3. ScatterReduce 阶段：通过 N-1 轮梯度传输和梯度相加，在每个设备上的梯度向量都有一小部分为所有设备中该分片梯度之和 （图③)
4. AllGather 阶段：通过 N-1 轮梯度传输，将上个阶段计算出的每个梯度向量分片之和广播到其他设备（图④)
5. 在每个设备上合并分片梯度，并根据梯度更新每个设备上的模型

大致计算一下Ring-Allreduce的通信消耗时间， 同样假设：

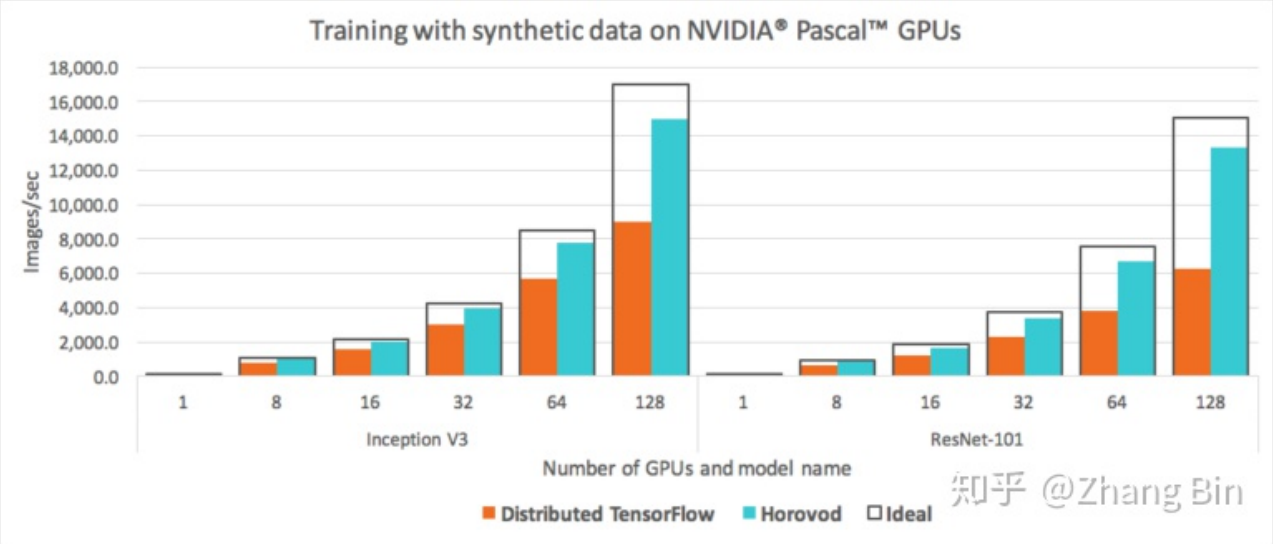
- 假设模型的总参数量为 X
- PS的数量为1， Worker的数量为 N
- 设备间数据最大传输速度为 B

其中每个设备需要传输 2(N-1) 次信息，并且每次发送 X/N 数据，因此完成一个batch iteration数据传输所需要的耗时为 2(N-1)*X/NB = 2X(N-1)/NB，可以看到当N逐渐增加时，数据传输所消耗的时间趋近于常数 2X/B。相比PS架构，Ring Allreduce架构是带宽优化的，因为集群中每个节点的带宽都被充分利用。对比于PS-Worker架构的耗时2(X·N)/B， **Ring-Allreduce的参数传递耗时并不会随着设备的增加而线性增长**，这是该架构性能优于PS-Worker的最大原因。

此外，在深度学习训练过程中，计算梯度采用BP算法，其特点是后面层的梯度先被计算，而前面层的梯度慢于前面层，Ring-allreduce架构可以充分利用这个特点，在前面层梯度计算的同时进行后面层梯度的传递，从而进一步减少训练时间。

后来，TensorFlow官方也在1.11版本中支持了allreduce的分布式训练策略CollectiveAllReduceStrategy，可以与tf.estimator配合使用。

Horovod的实际性能对比测试可以看下图：



可以看出Horovod在分布式训练的GPU设备的拓展能力对比TensorFlow有着很大的提高，在GPU设备达到一定规模的后其效率将是TensorFlow原生PS-Worker架构的**两倍以上**。

Horovod 代码实践

采用Horovod架构的另一个优点则是不需要大量的代码修改和重构就可以实现多GPU训练和分布式训练的功能。 TensorFlow 的原生PS-Worker架构通常需要引入额外上百行代码，并且更改部分逻辑使得在代码在不同的设备上运行不同的功能。

Horovod的参考代码如下, 其他框架的参考代码可以参考Horovod官方例程：

```
import tensorflow as tf
import horovod.tensorflow as hvd


# Initialize Horovod
hvd.init()

# Pin GPU to be used to process local rank (one GPU per process)
config = tf.ConfigProto()
config.gpu_options.visible_device_list = str(hvd.local_rank())

# Build model...
loss = ...
opt = tf.train.AdagradOptimizer(0.01 * hvd.size())

# Add Horovod Distributed Optimizer
opt = hvd.DistributedOptimizer(opt)

# Add hook to broadcast variables from rank 0 to all other processes during
# initialization.
hooks = [hvd.BroadcastGlobalVariablesHook(0)]

# Make training operation
train_op = opt.minimize(loss)

# Save checkpoints only on worker 0 to prevent other workers from corrupting them.
checkpoint_dir = '/tmp/train_logs' if hvd.rank() == 0 else None

# The MonitoredTrainingSession takes care of session initialization,
# restoring from a checkpoint, saving to a checkpoint, and closing when done
# or an error occurs.
with tf.train.MonitoredTrainingSession(checkpoint_dir=checkpoint_dir,
                                       config=config,
                                       hooks=hooks) as mon_sess:
  while not mon_sess.should_stop():
    # Perform synchronous training.
    mon_sess.run(train_op)
```

其中：

1. `hvd.init()` **初始化** Horovod，**启动相关线程和**MPI**线程。**
2. `config.gpu_options.visible_device_list = str(hvd.local_rank())` **为不同的进程分配不同的**GPU
3. `opt = tf.train.AdagradOptimizer(0.01 * hvd.size())` **根据**Worker**的数量增加学习率的大小**
4. `opt=hvd.DistributedOptimizer(opt)` **把常规**TensorFlow Optimizer**通过**Horovod**包起来，进而使用** ring-allreduce **来得到平均梯度。**
5. `hvd.BroadcastGlobalVariablesHook(0)` **将模型的参数从第一个设备传向其他设备，以保证初始化模型参数的一致性。**
6. `tf.train.MonitoredTrainingSession` if `hvd.rank() != 0` **设置只有设备**0**需要保存模型参数**

之后，用户可以使用 `horovodrun` **命令在多个服务器中运行：**

在一个拥有4**个**GPUs**的机器上:**

```
horovodrun -np 4 -H localhost:4 python train.py
```

在每台拥有4**个**GPU**的**4**个机器上:**

```
horovodrun -np 16 -H server1:4,server2:4,server3:4,server4:4 python train.py
```

Horovod 问题

checkpoint 的保存与恢复

只需要将 Worker 0 的checkpoint保存即可， 重启恢复的时候 Worker 0 的参数会通过 `hvd.BroadcastGlobalVariablesHook(0)` 传播到其他 Worker

horovod Failures due to SSH issues

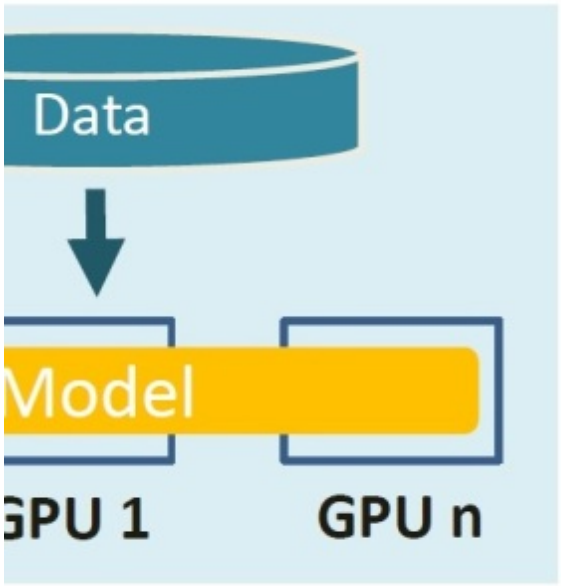
执行horovodrun命令的主机必须能够免密通过SSH登陆到其他主机，

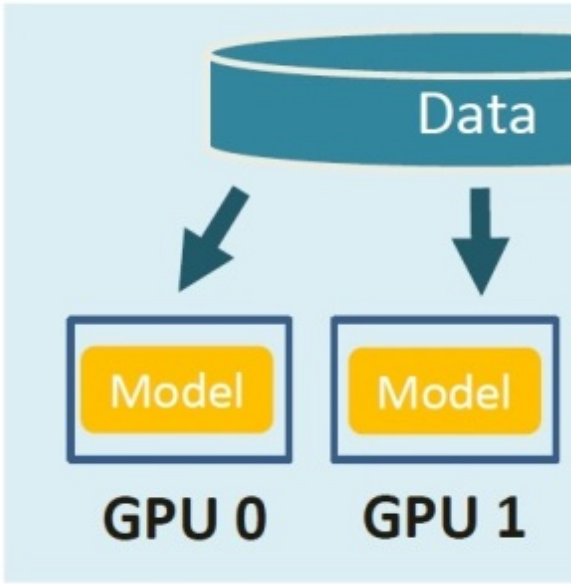
参考 SSH login without password

更多问题请见Troubleshooting

相关参考链接

Zhang Bin：深度学习分布式训练相关介绍 - Part 1 多GPU训练 zhuanlan.zhihu.com





Meet Horovod: Uber’s Open Source Distributed Deep Learning Framework for TensorFlow

Distributed TensorFlow

一文说清楚分布式训练必备知识

Effectively Scaling Deep Learning Frameworks

Horovod: fast and easy distributed deep learning in TensorFlow

Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour

Bringing HPC Techniques to Deep Learning

是时候放弃 TensorFlow 集群，拥抱 Horovod 了