# TensorFlow CPU和GPU配置

尝试在TensorFlow中加载两个神经网络，充分利用gpu的能量。然而，我的gpu只有8GBs内存，这是相当小的。所以我需要同时使用GPU和CPU。本文主要是针对这一问题进行培训。

首先，TensorFlow使用tf.ConfigProto（）来配置会话。

它还可以通过设置环境变量CUDA\_VISIBLE\_DEVICES来获取运行任务时的参数。-1设置为不使用GPU。id映射到nvidia smi命令中显示的id。

另一种方法是使用导出来控制所有环境

现在，回到tensorflow中的配置。

### 一。默认模式

TensorFlow默认模式是初始化所有可用的GPU。给定以下代码：

输出如下：

为了更好地分析tensorflow如何分配资源，并找出您的操作和tensor分配给了哪些设备，请创建会话，并将log\_device\_placement configuration选项设置为True。

### 2。决定使用GPU或CPU

要不使用GPU，一个好的解决方案是通过设置环境变量CUDA\_VISIBLE\_DEVICES不允许环境看到任何GPU。

输出如下：

使用字典分配可用GPU设备号和CPU设备号的参数设备计数。例如，下面的代码可以使tensorflow不使用任何gpu资源。

以下代码可以分配cpu和gpu。

#### 配置CPU

在一个CPU线程上运行Tensorflow

设备计数限制使用的CPU数量，而不是核心或线程的数量。

### 三。配置GPU

在ConfigProto（）中，gpu\_选项用于配置gpu。

#### 可见设备

首先，您需要使用nvidia smi命令检查计算机的可用gpu。当使用多个GPU时，我们需要使用，来分离它们。

#### GPU增长

第一个是allow\_growth选项，它试图根据运行时分配来分配尽可能多的GPU内存：它开始分配很少的内存，当会话运行并且需要更多的GPU内存时，我们扩展TensorFlow进程所需的GPU内存区域。

### 四。手动装置放置

如果希望某个特定操作在您选择的设备上运行，而不是在自动为您选择的设备上运行，则可以使用tf.device创建设备上下文，以便该上下文中的所有操作都具有相同的设备分配。

现在a和b被分配给cpu:0。

### 5个。在多GPU系统上使用单个GPU

如果系统中有多个GPU，默认情况下将选择ID最低的GPU。如果要在其他GPU上运行，则需要显式指定首选项：

如果您指定的设备不存在，您将得到InvalidArgumentError：

如果您希望TensorFlow自动选择一个现有的和支持的设备来运行操作，以防指定的操作不存在，那么您可以在创建会话时在配置选项中设置StaseUpStFiTyPrimeTuttrueTeo。

### 6。使用多个GPU

如果您想在多个GPU上运行TensorFlow，可以以多塔方式构建模型，其中每个塔被分配到不同的GPU。例如：

您将看到以下输出。

解决我们的问题。

看来我们需要多塔时装。CIFAR-1o就是一个很好的例子。之所以选择CIFAR-10，是因为它足够复杂，可以充分发挥TensorFlow扩展到大型模型的能力。

### 使用多个GPU卡训练模型

现代工作站可能包含多个用于科学计算的gpu。TensorFlow可以利用此环境跨多个卡同时运行培训操作。

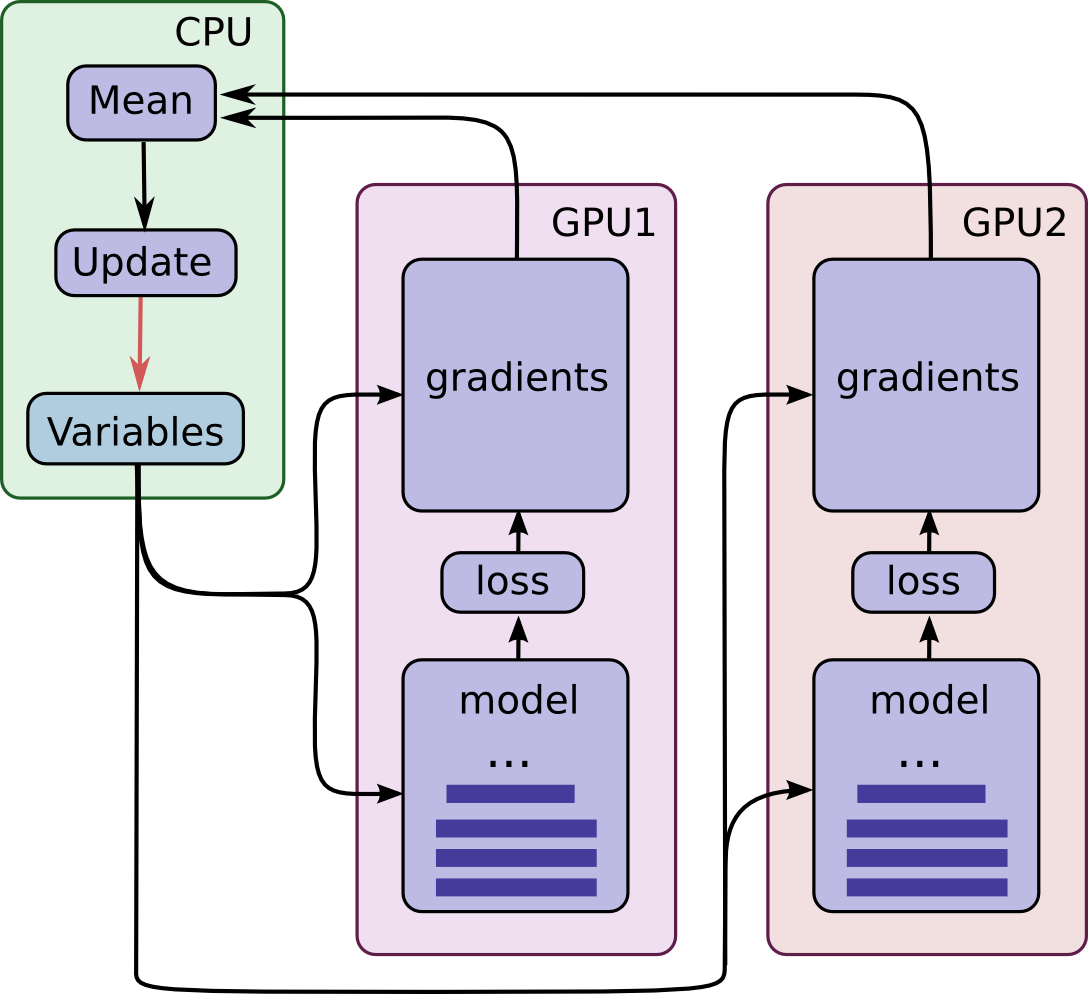
以并行、分布式的方式训练模型需要协调训练过程。接下来我们将模型副本称为数据子集上模型训练的一个副本。

天真地使用模型参数的异步更新会导致次优的训练性能，因为单个模型副本可能会在模型参数的陈旧副本上进行训练。相反，采用完全同步更新的速度将与最慢的模型副本一样慢。

在具有多个GPU卡的工作站中，每个GPU的速度都相似，并且包含足够的内存来运行整个CIFAR-10型号。因此，我们选择以下方式设计我们的培训系统：

* 在每个GPU上放置一个单独的模型副本。
* 通过等待所有GPU完成对一批数据的处理，同步更新模型参数。

下面是此模型的示意图：



请注意，每个GPU都计算一批唯一数据的推断和梯度此设置有效地允许在GPU上分割更大的数据批。

此设置要求所有GPU共享模型参数。一个众所周知的事实是，在gpu之间传输数据的速度相当慢。因此，我们决定在CPU上存储和更新所有模型参数（见绿色框）。当所有GPU处理一批新的数据时，一组新的模型参数被传送到GPU。

GPU在运行中同步。所有梯度都是从GPU累积并取平均值（见绿色框）。使用所有模型副本的平均梯度更新模型参数。

### 在设备上放置变量和操作

在设备上放置操作和变量需要一些特殊的抽象。

我们需要的第一个抽象是计算单个模型副本的推理和梯度的函数。在代码中，我们称这个抽象为“塔”。我们必须为每个塔设置两个属性：

* 塔中所有操作的唯一名称。name\_scope通过在作用域前面加上前缀来提供此唯一名称。例如，第一个塔中的所有操作都以塔0作为准备，例如塔0/conv1/Conv2D。
* A preferred hardware device to run the operation within a tower. tf.device specifies this. For instance, all operations in the first tower reside within device('/gpu:0') scope indicating that they should be run on the first GPU.

**All variables are pinned to the CPU and accessed via**tf.get\_variable**in order to share them in a multi-GPU version.**See how-to on [Sharing Variables](https://www.tensorflow.org/programmers_guide/variable_scope).

A good example is here: <https://github.com/tensorflow/tensorflow/blob/r0.7/tensorflow/models/image/cifar10/cifar10_multi_gpu_train.py>

Can try this out at first, see if it works.