GOOGLE CLOUD TPUS FOR ML ACCELERATION



We already wrote how machine learning frameworks are using NVIDIA GPUs (graphical processing units) to speed machine learning. Now Google is taking that idea and using it to speed machine learning using their own ASIC hardware, called **TPUs**, Tensor Processing Units. What Google has really done is take technology invented by NVIDIA (GPUs) and pushed it to the cloud.

A **Tensor** is an n-dimensional matrix. This is the basic unit of operation in with TensorFlow, the open source machine learning framework launched by Google Brain.

A Tensor is analogous to a numpy array and in fact uses Numpy. According to their documentation it is "NumPy is the fundamental package for scientific computing with Python. It contains among other things a powerful N-dimensional array object … "

Arrays are the fundamental data structures used by machine learning algorithms. Multiplying and taking slices from arrays takes a lot of CPU clock cycles and memory. So Numpy was written to make writing code to do that easier. GPUs now make those operations run faster.

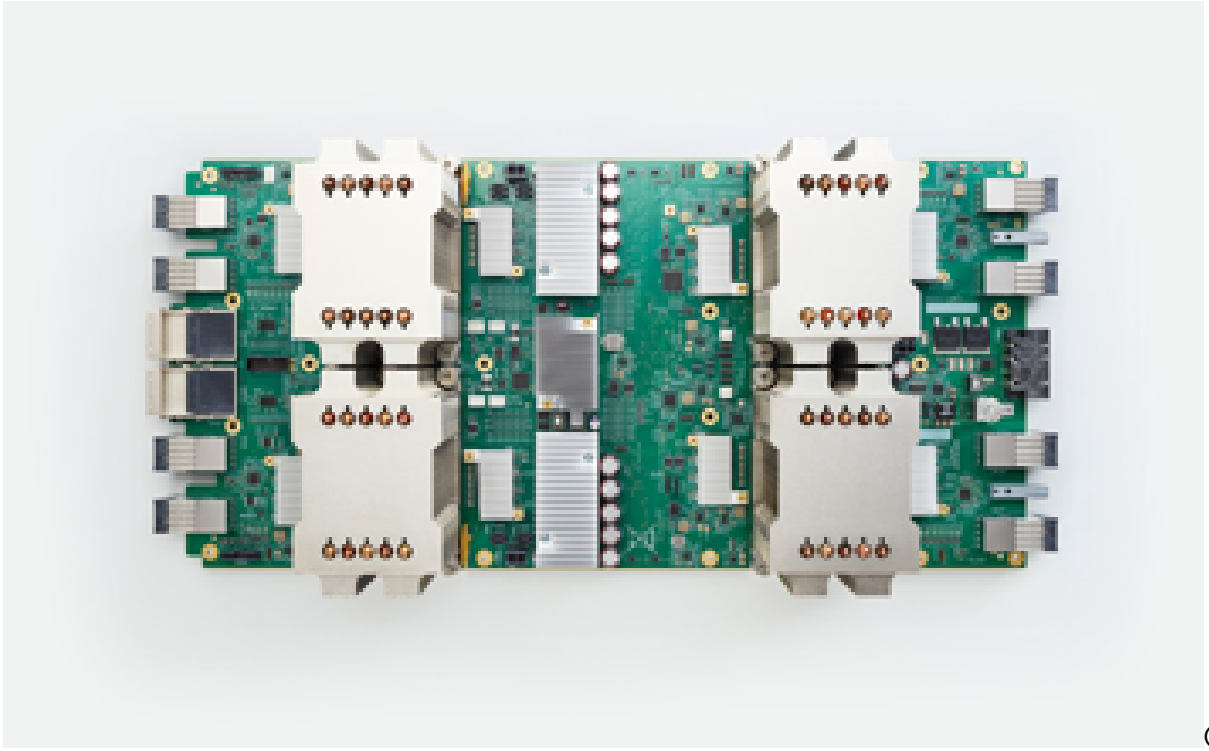
In particular, the math involved in doing ML includes adding and multiplying these objects:

- scalar
- vector
- matrix

GPUs were originally built to offload the intensive math calculations need to rotate graphics on a screen and otherwise speed up any kind of graphical operation, like painting screens in gaming applications. The goal was to not overburden the CPU. But then NVIDIA wrote the **CUDA SDK** letting programmers who write things like Tensorflow use GPUs for any kind of scalar, vector, or matrix

addition or multiplication.

A CPU has 1 to 8 cores or more. A GPU has hundreds. The GPU and TPU are the same technology. The only difference is now selling it as a cloud service using proprietary GPU chips that they sell to no one else.

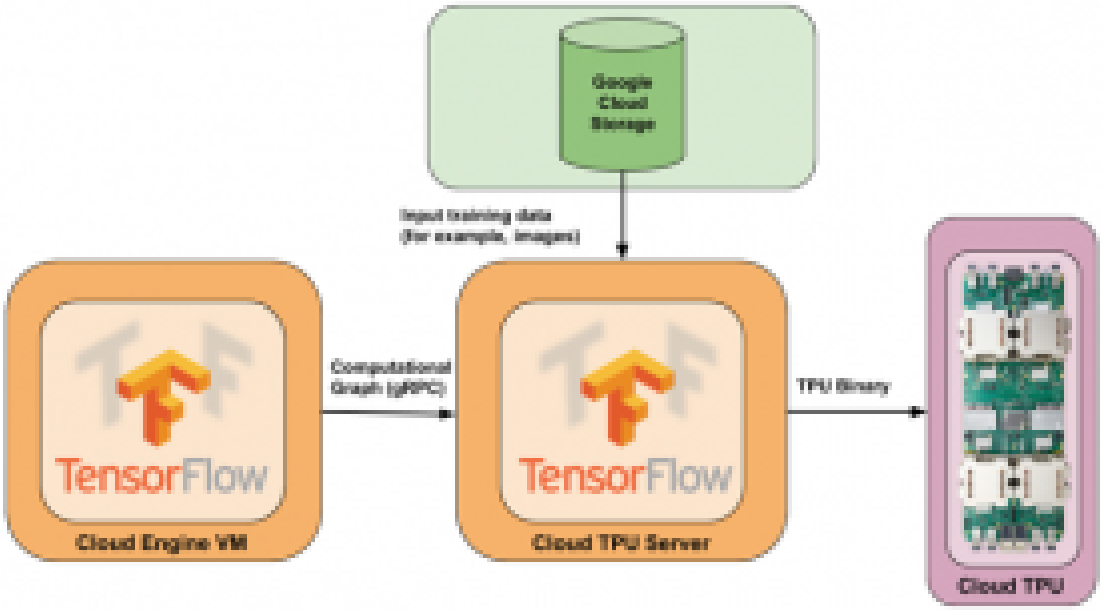


Google's approach to provisioning a TPU is different than Amazon's. At Amazon you pick a GPU-enabled template and spin up a virtual machine with that. Those templates all start with the letters **P3** and are listed here.

With Google you use their command line tool cptu to provide machines with TPUs. (And you can continue to use NVIDIA GPUs as well.)

According to Google's pricing information, each TPU cost $4.50 hour. Apparently they do not charge different rates for different TPU models even though they show three models on their website. That seems confusing as TPUs have different memory sizes and clock speeds. So one should be more expensive than another.

The TPU workload is distributed to what they call their **TPU Cloud Server**, as shown below.



(This tutorial is part of our Guide to Machine Learning with TensorFlow & Keras. Use the right-hand menu to navigate.)

Hardware

According to their architecture docs, their TPUs are connected to their cloud machines through a PCI interface. That is the same way that NVIDIA let gamers add graphical expansion cards to boost the performance of the graphics on the computer. So they are not just onboard chips, but expansion cards.

Google says, "Each chip consists of two compute cores called Tensor Cores. A Tensor Core consists of scalar, vector and matrix units (MXU). In addition, 16 GB of on-chip memory (HBM) is associated with each Tensor Core."

Software

An estimator is the tf.estimator.Estimator class. These are the implementation of neural networks, linear regression, and other objects with Python code that makes creating those kinds of objects simpler, since they leverage Numpy, Pandas, and other Python data structures and utilities.

Now Google says they have a TPU Estimator. You cannot download and use the GPU-enabled version of Tensorflow, which is different than regular TensorFlow in that it uses the CUDA SDK for that part of that code that is written in C and C++. There is no separate TPU-enabled version of TensorFlow. And unlike GPU, there appears to be no way to explicitly tell the code to use the TPU device, like in this code snippet that multiplies two matrices using GPU device **/device:GPU:n.** (To use the CPU you would write **/device:CPU:n,** where n can be any of the n CPUs on the computer.)

```
with tf.device('/device:GPU:0'):
   c = tf.matmul(x, y)
```

Scale Advantage?

One advantage of the TPU design would be that it lets you scale operations across different machines with their TPU servers. The user, of course, does not need to write any code to do that. This researcher has not yet studied how to do that with GPUs. In other words what do you do when

your calculation runs out of memory? You can add PCI expansion cards '/device:GPU:1, 2, ..., n or effectively do the same thing by paying Amazon for a larger template. But how do you implement something like a Mesos equivalent that would let you scale across a cluster of servers without having to hard-code device and server names? We will look at that and write you back.