Review Article- Google TPU

Submitted by:

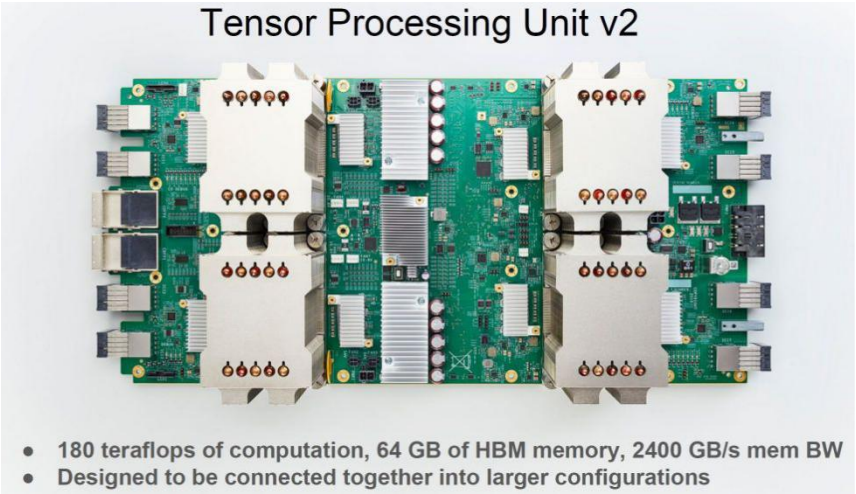
Ripunjay Narula(19BCE0470) and

Samvit Swaminathan(19BCE0629)

What is a TPU?

TPU stands for Tensor Processing Unit. It is an AI accelerator application-specific integrated circuit (ASIC). TPUs have been developed by Google in 2016 at Google I/O. However, TPUs have already been in Google data centers since 2015.

The chip is specifically designed for TensorFlow framework for neural network machine learning. Current TPU versions are already 3rd generation TPUs, launched in May 2018. Edge TPUs have also been launched in July 2018 for ML models for edge computing.

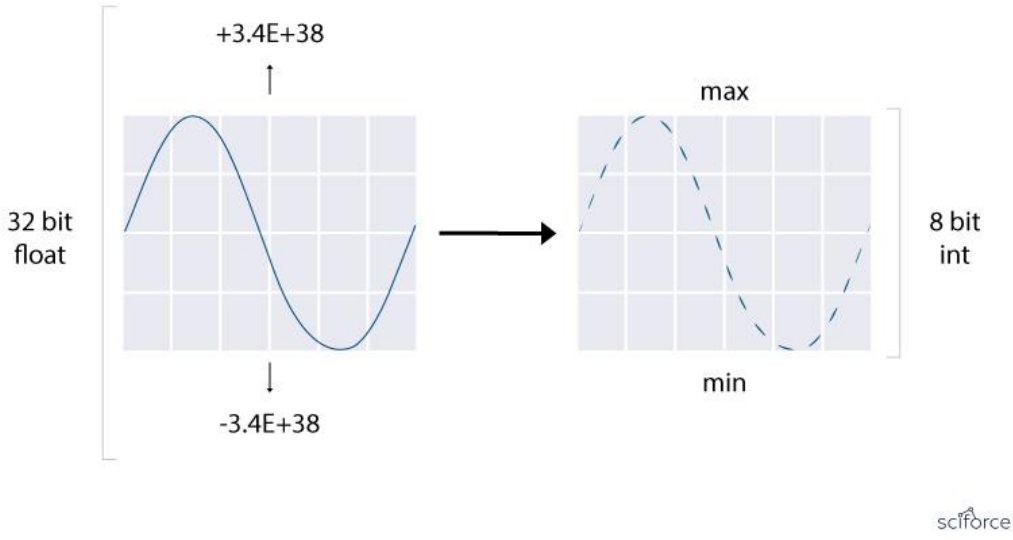


The TPUs have been designed, verified, built and deployed in just under 15 months, whereas typical ASIC development takes years.

How do TPU work?

1. Quantization

In line with the quantization technique, the process of approximation of an arbitrary value between a preset minimum and a maximum value with an 8-bit integer, TPUs contain 65,536 8-bit integer multipliers. In essence, this technique is compression of floating point calculations with 32-bit or even 16-bit numbers to 8-it integers. As you can see, the continuous large set of values (such as the real numbers) is converted to a discrete set (of integers) with maintaining the curve:



Quantization is the first powerful tool TPUs use to reduce the cost of neural network predictions without significant losses in accuracy.

2. Focus on inference maths

Secondly, the TPU design itself encapsulates the essence of neural network calculation. A TPU includes the following computational resources:

Matrix Multiplier Unit (MXU): 65,536 8-bit multiply-and-add units for matrix operations;

Unified Buffer (UB): 24MB of SRAM that work as registers;

Activation Unit (AU): Hardwired activation functions.

They are controlled with a dozen high-level instructions that focus on the major mathematical operations required for neural network inference. A special compiler and software stack translate API calls from TensorFlow graphs into TPU instructions.

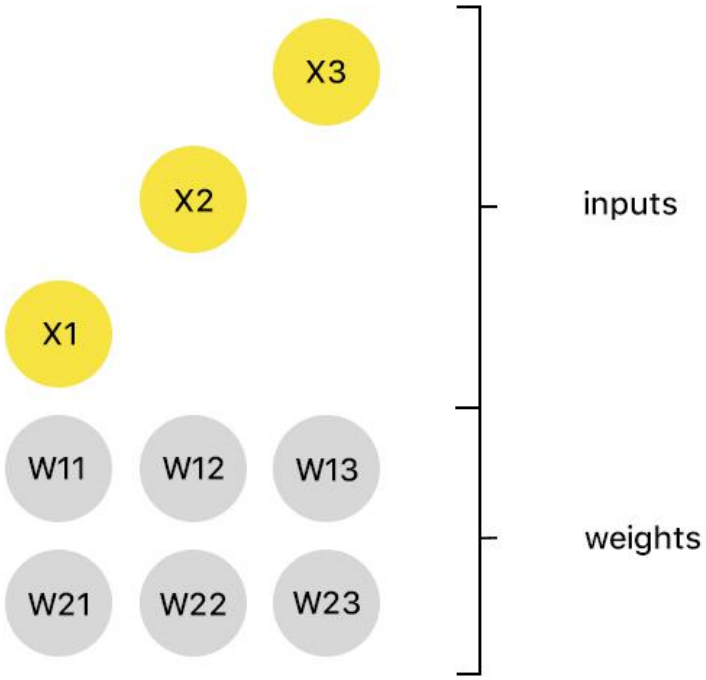
3. Parallel Processing

Typical RISC processors provide instructions for simple calculations such as multiplying by processing a single, or scalar, operation with each instruction. As you remember, a TPU contains a Matrix Multiplier Unit. It is designed as a **matrix, rather than scalar, processor** and processes hundreds of thousands of operations (= matrix operation) in a single clock cycle. Using such a matrix processor is like printing documents a whole page at a time rather than character-by-character or line-by-line.

4. A systolic array

The heart of the TPU is the new architecture of the MXU called a systolic array. In traditional architectures (such as CPUs or GPUs), values are stored in registers, and a program tells the Arithmetic Logic Units (ALUs) which registers to

read, the operation to perform (such as an addition, multiplication or logical AND) and the register into which to put the result. A program consists of a sequence of these operations. In an MXU, matrix multiplication reuses inputs many times to procduce the final output. A value is read once but used for many different operations without storing it back to a register. The ALUs perform only multiplications and additions in fixed patterns, and wires connect adjacent ALUs, which makes them short and energy-efficient.



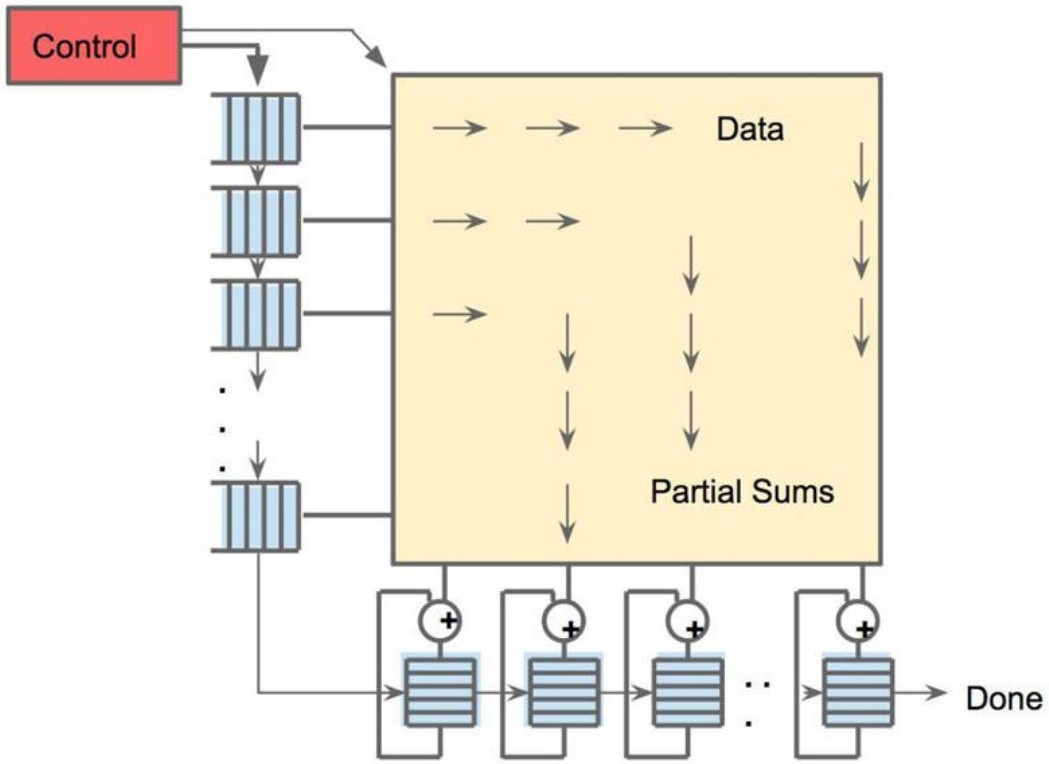
To understand this design, think of the heart pumping blood — like the data flowing through the chip in waves.

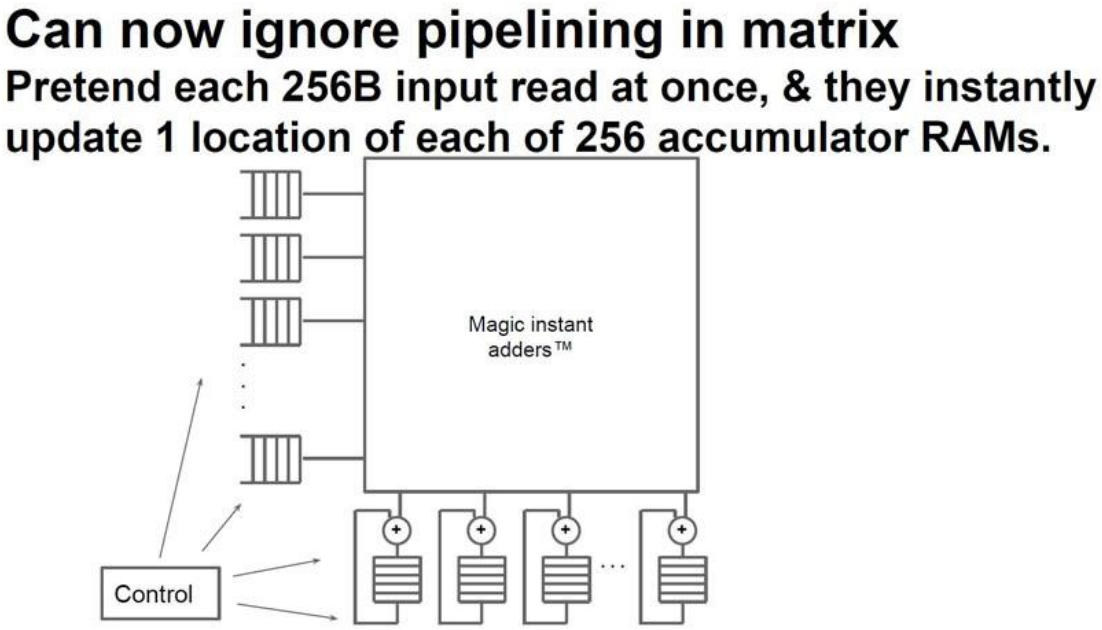
Google TPU Systolic Execution Diagram:

Memory bandwidth is extremely important in the architecture so the TPU is designed to efficiently move data around.

Matrix Multiplication Unit

- Contains 256 x 256 = 65,536 ALUs
- TPU runs at 700 MHz
- Able to compute 46 x 1012 multiply-and-add operations per second
- Equivalent to 92 Teraops per second in matrix unit





Google TPU Ignore Pipelining In Matrix

TPU instruction set

TPU is designed to be flexible enough to accelerate computation times of many kinds of neural networks model.

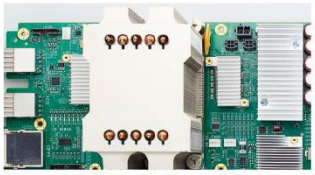
Modern CPUs are influenced by the Reduced Instruction Set Computer (RISC) design style. The idea of RISC is to define simple instructions (load, store, add, multiply) and execute them as fast as possible.

TPUs use Complex Instruction Set Computer (CISC) style as an instruction set. CISC focus on implementing high-level instructions that run complex tasks such as multiply-and-add many times with each instruction.

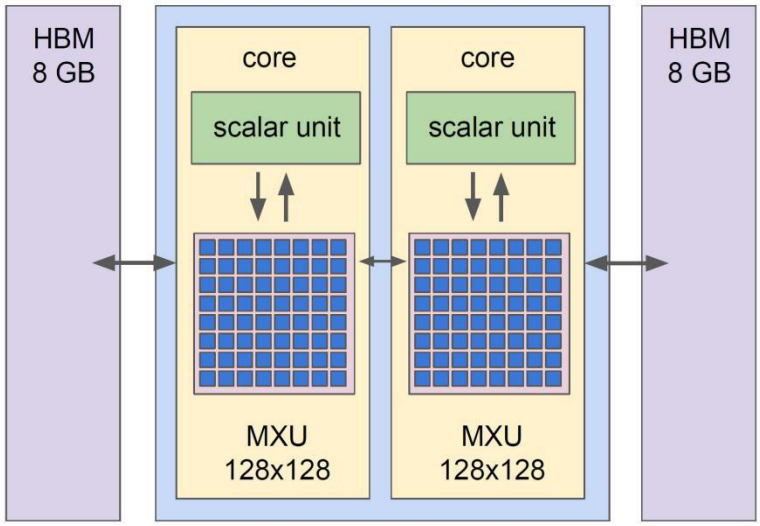
Cloud TPU v2 Architecture

How Google actually followed up the TPU is with the TPUv2. Instead of updating to higher bandwidth GDDR5, Google is using even faster HBM in 16GB capacity. That gives them 600GB/s of memory bandwidth. For comparison, that is about four sockets worth of AMD EPYC.

TPUv2 Chip



- 16 GB of HBM
- 600 GB/s mem BW
- Scalar unit: 32b float
- MXU: 32b float accumulation but reduced precision for multipliers
- 45 TFLOPS



What is a TPU made of?

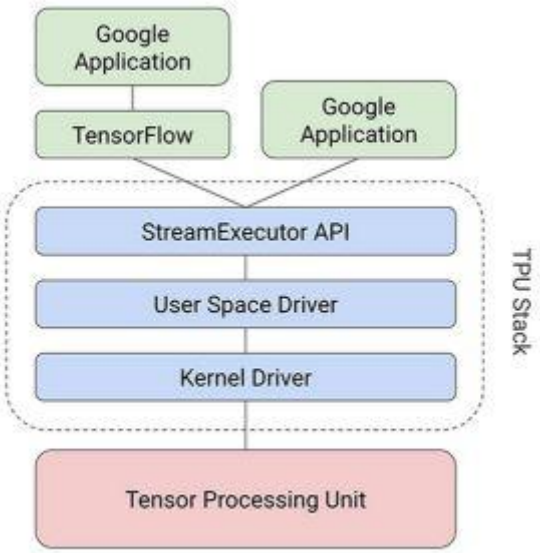
TPUs are made of several computational resources :

- Matric Multiplier Unit (MXU): 65,536 8-bit multiply-and-add units for matrix operations
- Unified Buffer (UB): 24MB of SRAM that works as registers
- Activation Unit (AU): Hardwired activation functions

Here's an example of some high-levels instructions specifically designed for neural network inference that control how the MXU, UB, and AU work :

- Read_Host_Memory : Read data from memory
- Read_Weights : Read weights from memory
- MatrixMultiply/Convolve: Multiply or convolve with the data and weights, accumulate the results
- Activate : Apply activation functions
- Write_Host_Memory : Write results to memory

Google has created a compiler and software stack that translates API calls from TensorFlow graphs inti TPU instructions following this schema :

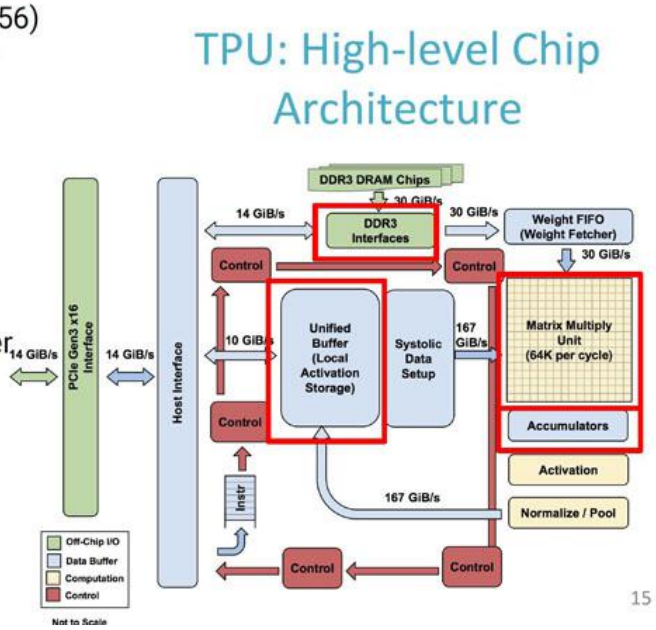


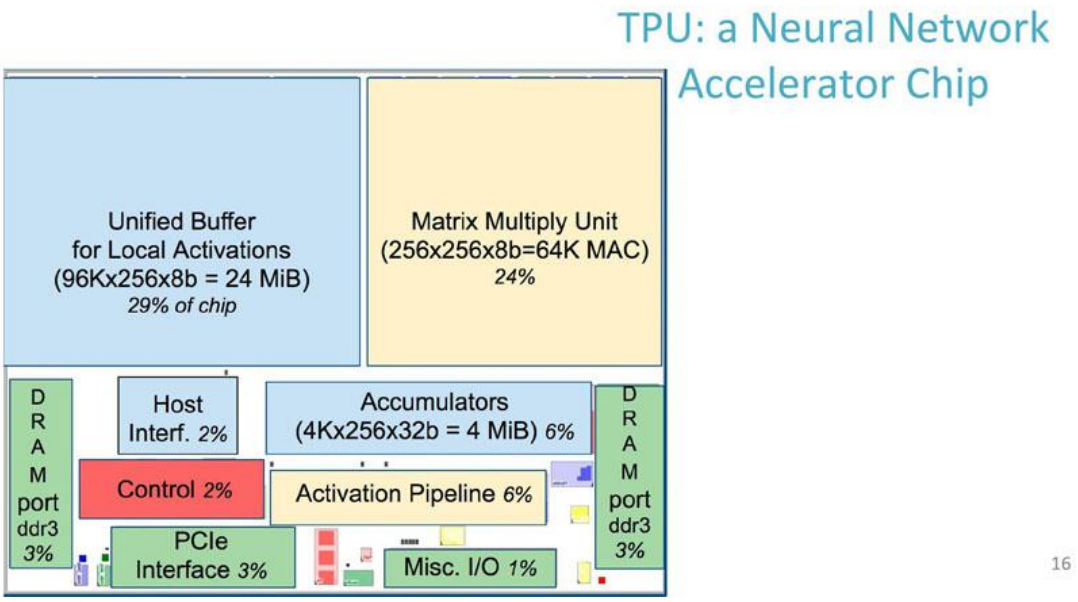
Specifically, it is designed for fast matrix multiply, important in deep learning applications.

Google TPU High-Level Architecture

Here is a view of how much logic is dedicated in the chip to different functions:

- The Matrix Unit: 65,536 (256x256) 8-bit multiply-accumulate units
- 700 MHz clock rate
- Peak: 92T operations/second
  - 65,536 * 2 * 700M
- >25X as many MACs vs GPU
- >100X as many MACs vs CPU
- 4 MiB of on-chip Accumulator memory
- 24 MiB of on-chip Unified Buffer (activation memory)
- 3.5X as much on-chip memory vs GPU
- Two 2133MHz DDR3 DRAM channels
- 8 GiB of off-chip weight DRAM memory



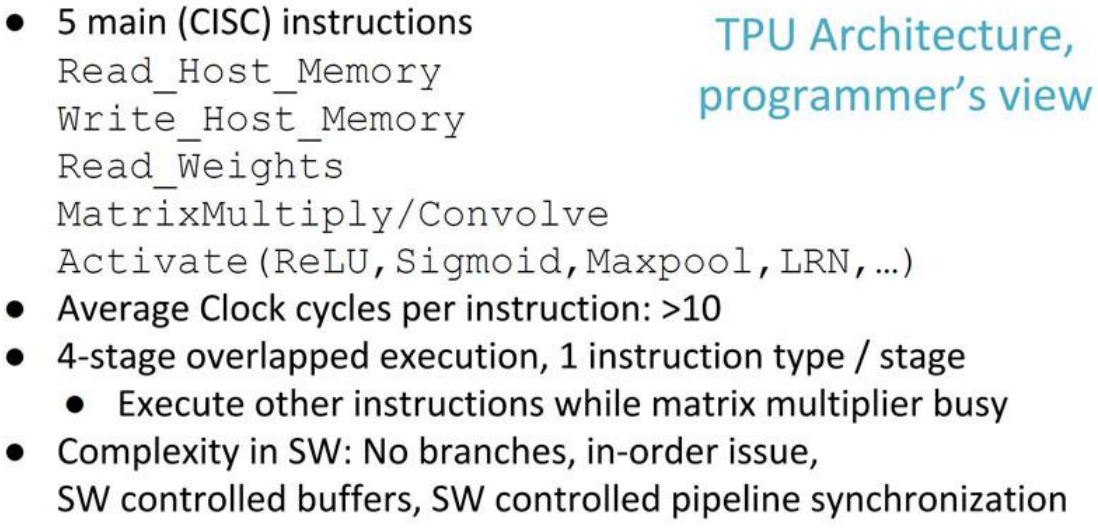


Google TPU Neural Network Accelerator Chip

To the programmer, Google is providing a view in terms of what using the TPU looks like.

Google TPU Architecture Programmer View

The TPU has only 11 instructions, five are commonly used. It is an in order design that relies on software to handle complexity.



What if there was a Google TPU (v1.1) Update GDDR5 for Bandwidth?

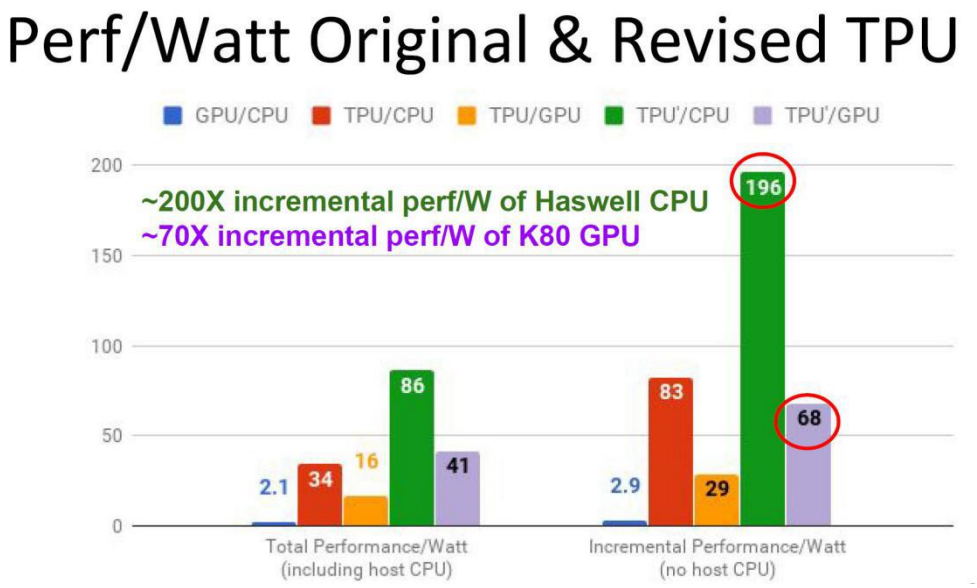
Google used DDR3 in its original design. One option was to move to DDR4 for higher bandwidth operation. Another potential design decision was to move to GDDR5 which is what Google showed at Hot Chips:

- Current DRAM
  - 2 DDR3 2133 ⇒ 34 GB/s
- Replace with GDDR5 like in K80 ⇒ 180 GB/s
  - Move Ridge Point from 1400 to 256

Improving TPU: Move "Ridge Point" to the left

Updated Google TPU GDDR5

The company explained that they were not necessarily compute limited in their original design. DDR3 was too slow. Here is what Google is claiming if the TPU had GDDR5 in terms of performance per Watt:



Updated Google Revised TPU GDDR5

The key here is that the GDDR5 version would have had more bandwidth and lower latency boosting performance. Instead of building the v1.1 version using GDDR5, Google leapfrogged GDDR5 and moved to HBM for its TPU v2 which could also be used for training, not just inference.

Cloud TPU v2 Architecture

How Google actually followed up the TPU is with the TPUv2? Instead of updating to higher bandwidth GDDR5, Google is using even faster HBM in 16GB capacity. That gives them 600GB/s of memory bandwidth. For comparison, that is about four sockets worth of AMD EPYC.

These are the much larger compute units with more TDP headroom that Google is making available to developers.

Google Cloud TPUs for ML Acceleration:

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:

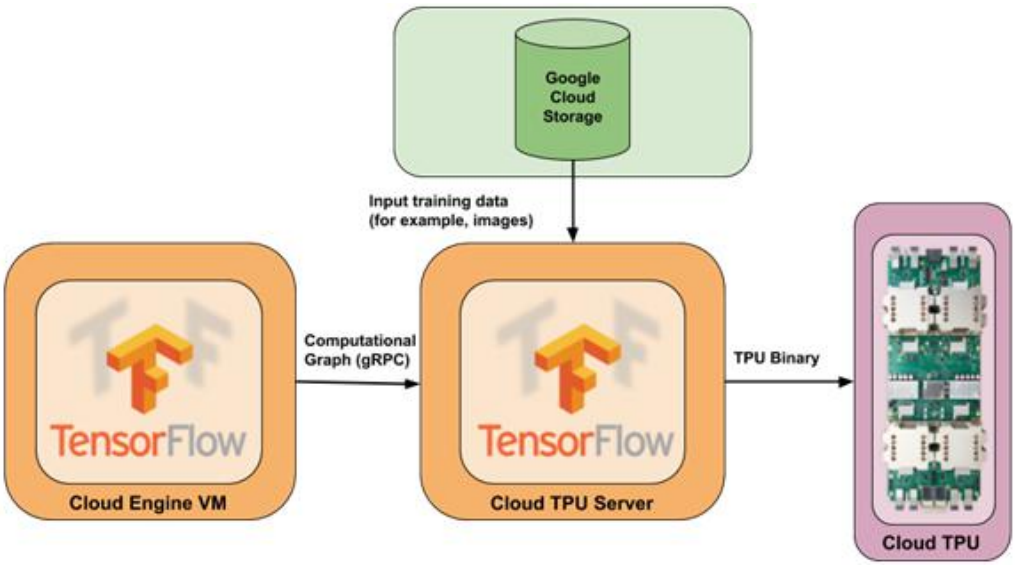
- scalar
- vector
- matrix

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.



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.

USES

- RankBrain algorithm used by Google search
- Google Photos
- Google Translate
- Google Cloud Platform

Future Development

- Uses TPU version 2
- Each TPU include a high-speed network
- Allows to build machine learning supercomputers called “TPU Pods”
- Improvement in training times
- Allows mixing and matching with other hardware which includes Skylake CPUs and NVIDIA GPUs

Sources/Citations:

https://www.bmc.com/blogs/google-cloud-tpu/
https://medium.com/sciforce/understanding-tensor-processing-units-10ff41f50e78
http://meseec.ce.rit.edu/551-projects/fall2017/3-4.pdf
https://cloud.google.com/tpu/docs
https://www.servethehome.com/case-study-google-tpu-gddr5-hot-chips-29/