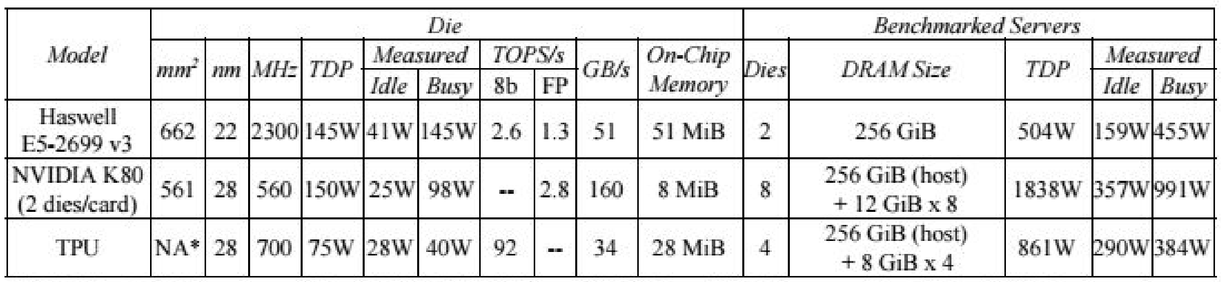
Michael Laucella
Multicore Programming

<u>Analysis of TPU report</u>

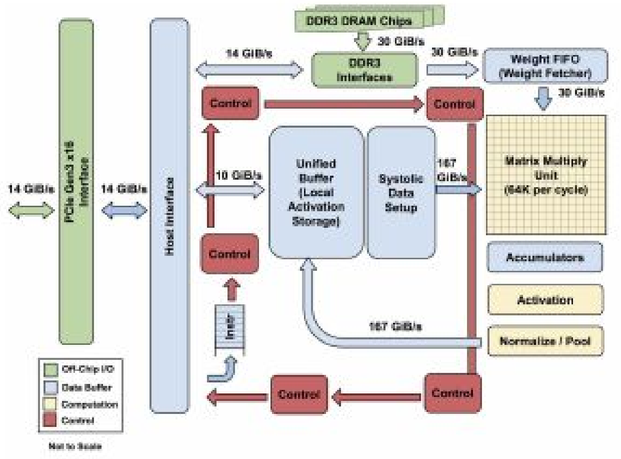
Google’s Tensor Processing Unit (TPU) is an Application Specific Integrated Chip (ASIC) developed internally by Google’s engineering teams with one specific task in mind. Given future projections for increased artificial intelligence (AI) needs doubling, Google saw a need for a faster more efficient approach to meet this future demand, this is when work on TPU’s began. The T (Tensor) comes from the programming language TensorFlow used to write programs on TPU’s. Google’s claim to fame with the TPU is that given the very specific problem of Neural Networks (NN) and using the type of NN algorithms and datasets that Google's machines typically face as a benchmark, the TPU drastically outperforms both central processing units (CPU) and graphics processing units (GPU). The design for the TPU itself came from tinkering and studying commercial NVIDIA GPU’s and using what they found to develop designs and improvements. The testing shows bias in the fact that it was directly tested to Google’s means, which is fair as it is an ASIC. However the GPU test sparked remarks from NVIDIA for using a 2012 K80 and not with the 2016 P40 (in Google’s defense the TPU’s were deployed in 2015 for testing), also Google ran the K80’s in an unoptimized mode (boost mode disabled) because of cooling concerns as the test was literally conducted to meet their specific environment, but this would have increased the frequency. NVIDIA claims that the P40 has a 26x better at inference than the K80 due to memory bandwidth. But even so the TPU still heralds a significantly higher performance per watt to GPU’s which are typically known to be power hungry machines. Taking a closer look at the data gathered by Google the analysis is purely on the inference phase in which the K80 is highly underutilized. Google stated that they received a 15-30x speedup over the K80 and the Intel Haswell CPU, interestingly enough they stated a 30-80x performance per watt over what they called “contemporary products” (does this imply the K80?), a very vague phrase in a mostly precise paper. The TPU has 25x the number of MAC’s as compared to the K80 GPU’s, these MAC’s can each perform an 8-bit multiply and add on integers.

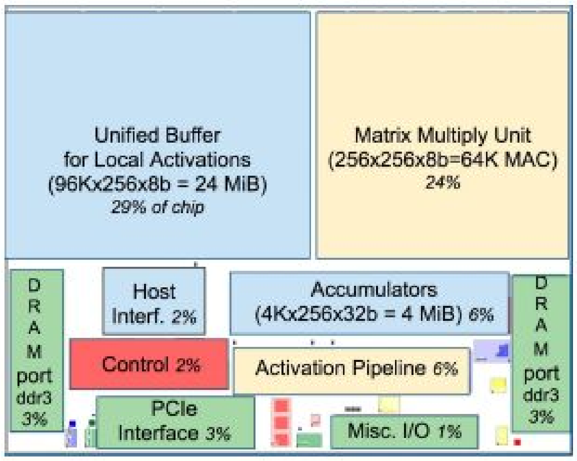


Model	*Die*										*Benchmarked Servers*				
	mm²	*nm*	*MHz*	*TDP*	*Measured*		*TOPS/s*		*GB/s*	*On-Chip Memory*	*Dies*	*DRAM Size*	*TDP*	*Measured*	
					Idle	*Busy*	8b	FP						*Idle*	*Busy*
Haswell E5-2699 v3	662	22	2300	145W	41W	145W	2.6	1.3	51	51 MiB	2	256 GiB	504W	159W	455W
NVIDIA K80 (2 dies/card)	561	28	560	150W	25W	98W	--	2.8	160	8 MiB	8	256 GiB (host) + 12 GiB x 8	1838W	357W	991W
TPU	NA*	28	700	75W	28W	40W	92	--	34	28 MiB	4	256 GiB (host) + 8 GiB x 4	861W	290W	384W

Another factor for the explained speedup as seen in the table above is due to the GPU performing inference using floating points (FP) whereas the TPU uses 8-bit integers (importance of the MAC’s) which are originally converted by a process called quantization. Using 8-bit integers reduces not only energy costs but are also much quicker to modify and are more compact as compared to FP's. The testing also suggest that 4 K80 GPU's were used overall and together had almost 3x the wattage when busy even though it only had 2x the number of dies, and exhibited far less performance in terms of tera operations per second (TOPS) with the TPU at 92 and the GPU lagging far behind at 2.8.

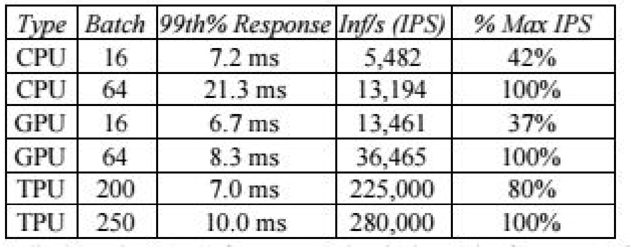
The architecture for the TPU itself is designed as coprocessor, meaning it is not a standalone device in the sense that it requires a CPU present on the machine to function, similar to a GPU it receives all of its instructions from the host into a buffer (in NVIDIA's case a stream). The upside to this is that a TPU can be plugged into a peripheral component interconnect express (PCIe) a common place component in any motherboard. Within the TPU is a large matrix multiplication unit (MMU) housing several MAC's (256^2). These MAC's perform 8-bit operations as mentioned earlier, and the products of these are placed into a 32-bit Accumulator. The MMU loads in 64KiB sized tiles of weights. This demonstrates its strict design as the problems using TPU's require a set of weights to use on the matrix which is common in NN. The TPU also contains special storage for these weights known as the weight memory which uses a first in first out (FIFO) structure. It can store enough weights for 4 tiles of the matrix (matrices are commonly broken down into tiles however for GPUs this tiling is explicit, here it is handled by the hardware). Outputs are stored in a 24MiB on-chip buffer called the unified buffer stores intermediate results. This means partial calculations can be stored in the unified buffer but can also be reinserted into the MMU from the unified buffer. Not only does it cycle data from the MMU but can move data to or from the host memory by use of a programmable direct memory access controller (DMA). DMA's are used to move and manage memory without blocking the CPU allowing it to continue working until an interrupt signal is received by the DMA controller. This allows the TPU to continue working while memory is being moved to and from the device. The TPU contains about 12 machine instructions of which 5 are most prevalent. These mainly deal with reading/writing data as well as matrix operations.




Unified Buffer for Local Activations (96Kx256x8b = 24 MiB) 29% of chip
Matrix Multiply Unit (256x256x8b=64K MAC) 24%
DRAM port ddr3 3%
Host Interf. 2%
Accumulators (4Kx256x32b = 4 MiB) 6%
DRAM port ddr3 3%
Control 2%
Activation Pipeline 6%
PCIe Interface 3%
Misc. I/O 1%

While spending all this time considering performance as the most important metric a very good marketing point was made Google. As a business incentive what matters most of the time is less about performance and more about cost, electricity bills for a datacenter are a huge concern for a company, and when running high powered processes using GPU's far more power is being expended as compared to a TPU making the TPU the more economic choice. However only if the TPU is being properly utilized. The TPU has problems in a low work state while the Haswell used 56% of its energy at a 10% workload, and the GPU 66%, the TPU used 88%. These numbers fluctuate based on the workload percentage and algorithm but at low workloads the TPU still remains the largest power consumer per die.

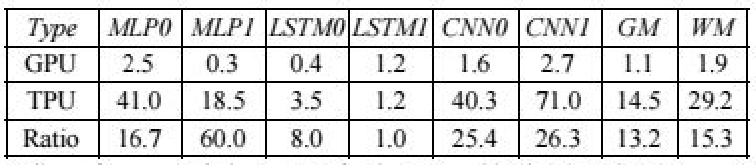
In user facing systems latency is more important than the throughput as a user is likely to leave if a result takes longer than expected. In the case of NN algorithms this is a reference to the inference phase (exactly what the TPU specializes in).



Type	*Batch*	*99th% Response*	*Inf/s (IPS)*	*% Max IPS*
CPU	16	7.2 ms	5,482	42%
CPU	64	21.3 ms	13,194	100%
GPU	16	6.7 ms	13,461	37%
GPU	64	8.3 ms	36,465	100%
TPU	200	7.0 ms	225,000	80%
TPU	250	10.0 ms	280,000	100%

Here we see that for the 99% response time the GPU is .3ms faster than the TPU for MLP0 but utilizing only 37% of its max throughput, whereas the TPU is using 80%. This argument is only valid while the application limit was set for 7ms (a mandatory setting by the application developers), a relaxing of this time limit would favor the GPU in both latency and throughput as can be seen in the table itself. But this comes at the cost of batch sizes the TPU can handle a size of 200 at 7ms and 80% throughput where the GPU can only handle 64 at max throughput so the TPU can generate more results within the 7ms, meaning it will take longer to serve but you can effectively serve 184 more results on only a .3ms difference.

When it comes to the six algorithms the TPU was tested on, a geometric mean (GM) and a weighted mean (WM) were taken comparing the TPU and GPU. From it we get a nice picture of how much the TPU outperformed the GPU overall and we can see on a per algorithm bases besides LSTM1 the TPU performed much faster as Google put it 13.4 (14.5-1.1) times faster than the GPU.



Type	MLP0	MLP1	LSTM0	LSTM1	CNN0	CNN1	GM	WM
GPU	2.5	0.3	0.4	1.2	1.6	2.7	1.1	1.9
TPU	41.0	18.5	3.5	1.2	40.3	71.0	14.5	29.2
Ratio	16.7	60.0	8.0	1.0	25.4	26.3	13.2	15.3

One of the most interesting things about the TPU is that it manages to exploit so much performance while being a minimized lightweight chip. By lightweight I'm referring to the great amount of features that were chosen not to be a part of the chip. There is no out of order execution features, multithreading, context switching, branch prediction, caches, instruction prefetching which can normally be found in CPU's and some features in GPU's (caching for sure). Subtracting all these components give greater space on the chip for more processing features which contributes to it's speed up but also hurts its ability to be more flexible, however it was never designed with that intent in mind.

One might ask the relevance this has to our study of exploiting multicore parallelism. And the reality is that these systems both the TPU and GPU are PCIe connected IO devices. While not the traditional multicore processor that we have learned so much about these SIMD devices exploit extremely high levels of parallelism when faced with certain challenges, for instance NN or graphics processing in which we see from the table above that the CPU isn't even worth mentioning. These devices can also be used within the confines of a concurrent applications and can also spare a programmer from writing highly inefficient algorithms on a multicore processor which can gain massive speedup by using an SIMD type device. Now before assuming TPU's are the next big thing it's also important to draw some logical conclusions. The TPU was designed for and only been tested to run efficiently for a specific problem, and its testing was highly specific to only Google's current workloads and there datacenter setups, whereas GPU's are a much more flexible and conventional device used for a number of applications. Most companies will not likely make drastic switches to using TPU's as it will involve rewriting applications for the TPU. But what has NVIDIA frightened is the threat to the very specific industry in which the TPU's were developed. Up till this point in time GPU's were the most popular device for NN which worked well enough but now TPU's threaten this sector of profit, which could cost NVIDIA a great deal if the TPU were to become successful.

Sources:

In-Datacenter Performance Analysis of a Tensor Processing Unit
https://drive.google.com/file/d/0Bx4hafXDDq2EMzRNcy1vSUxtcEk/view

Nvidia claims Pascal GPUs would challenge Google's TensorFlow TPU in updated benchmarks

https://www.extremetech.com/computing/247403-nvidia-claims-pascal-gpus-challenge-googles-tensorflow-tpu-updated-benchmarks