In Datacenter Performance Analysis of a TPU

Majd Almudhry, Andrew Liao, Julia Anitescu

Carnegie Mellon University

Author Introduction

Norman P. Jouppi - Stanford PhD, Computer Engineering at Google, IEEE, ACM and AAAS Fellow

Cliff Young - Harvard PhD, D.E. Shaw Research, Google Computer Engineer

Nishant Patil - Stanford University, Intel, Google

David Patterson - UCLA PhD, Turing Award, Berkeley prof

Gaurav Agrawal - UT Austin MS, Google, now at OpenAI

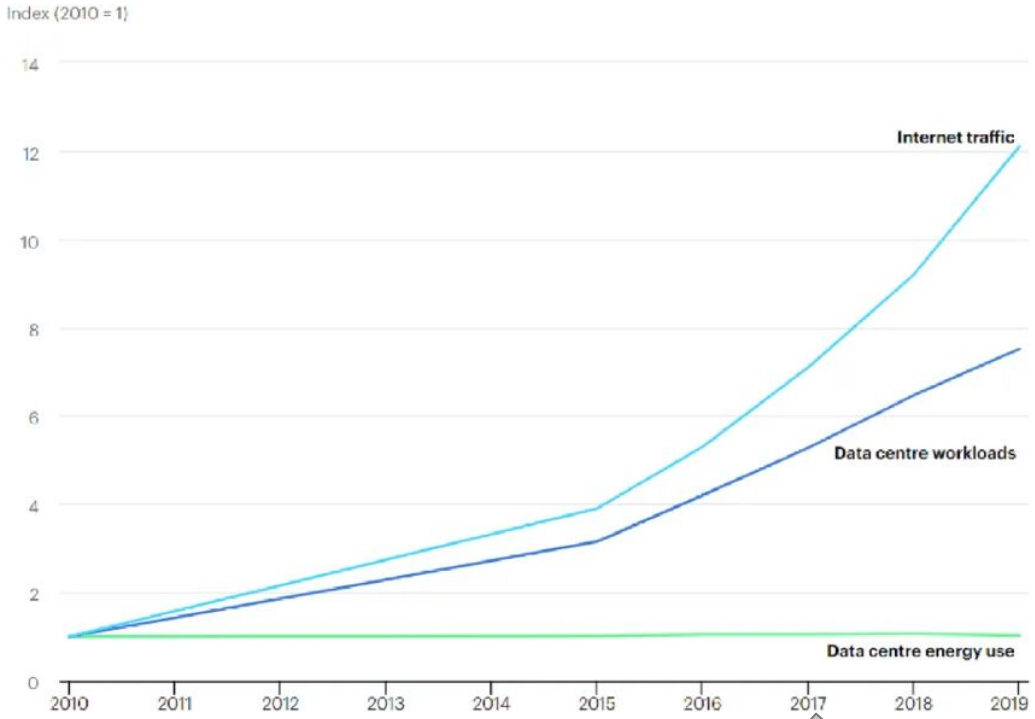
And many more!




Carnegie Mellon University

State of the industry

- Increasing use of datacenters
- Increasing need to improve their performance
- 2013 projection that datacenter speech recognition DNN uses would double the datacenter computation demands



Chukwuka Gideon Monyei, 2019

Carnegie Mellon University

Design Background

Initial assumption was that running applications on specialized hardware could be done for "free" using the datacenter capacity

- Aforementioned projection broke that assumption

TPU was designed, verified, built and deployed in 15 months to match need quickly

- Bought off the shelf GPUs to use for training
- PCIe I/O bus to be able to plug into an existing server(like a GPU)
- Host server sends TPU instructions instead of TPU decoding them, like an FPU

Goal was to run inference in the TPU to minimize interactions with CPU, match NN needs of 2015 and beyond.

Benchmarks (6: 2xMLP 2xLSTM 2xCNN)

3 main kinds of Neural Networks: Multi-Layer Perceptrons(MLP), Convolutional Neural Networks(CNN), Long Short-Term Memory(LSTM) as benchmarks.

- Represent 95% of inference workload in Google Datacenters
- Typically user-facing applications, so would emphasize response time over throughput
- 4/6 are memory bound vs 2/6 are compute bound.
- Using Application popularity as of July 2016

Name	*LOC*	*Layers*					*Nonlinear function*	*Weights*	*TPU Ops / Weight Byte*	*TPU Batch Size*	*% of Deployed TPUs in July 2016*
		FC	*Conv*	*Vector*	*Pool*	*Total*					
MLP0	100	5				5	ReLU	20M	200	200	61%
MLP1	1000	4				4	ReLU	5M	168	168	
LSTM0	1000	24		34		58	sigmoid, tanh	52M	64	64	29%
LSTM1	1500	37		19		56	sigmoid, tanh	34M	96	96	
CNN0	1000		16			16	ReLU	8M	2888	8	5%
CNN1	1000	4	72		13	89	ReLU	100M	1750	32	

Carnegie Mellon University

Software Architecture: CISC

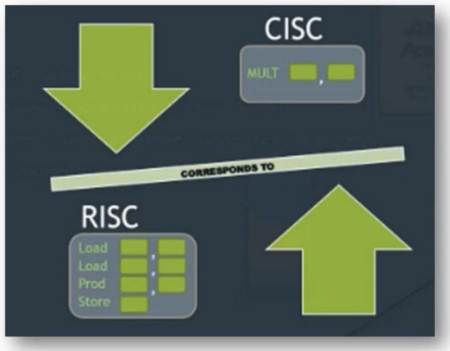
CISC architecture

- Better utilizes the limited bandwidth of the PCIe bus for delivering instructions because they could perform multi-step operations with a single command
- Repeat field helps with that as well
- Long instructions, so decode is negligible

Example: Matrix Multiply

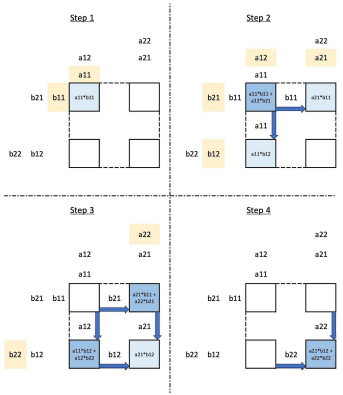
- Instruction is 12 bytes
- 3 Unified Buffer Address
- 2 Accumulator Address
- 4 are length(2D for convolution)
- Opcode and flags

CISC	RISC
MUL 2:2, 3:3 (x,y i.e x=x*y)	LOAD X, 2:2 LOAD Y, 3:3 MUL X,Y STORE 2:2, X



Carnegie Mellon University

Other Relevant Instructions



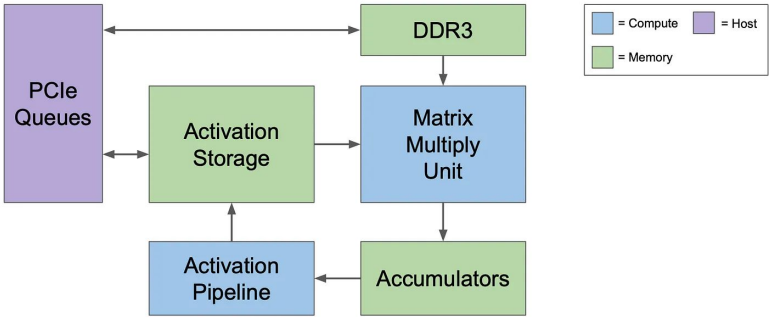
Goal: keep matrix unit busy

- 4 stage pipeline
- Hide the execution of the other instructions with Matrix Multiply

Read_Weights: decoupled access/execute to help latency

- Weights are held in off-chip FIFO called weight memory

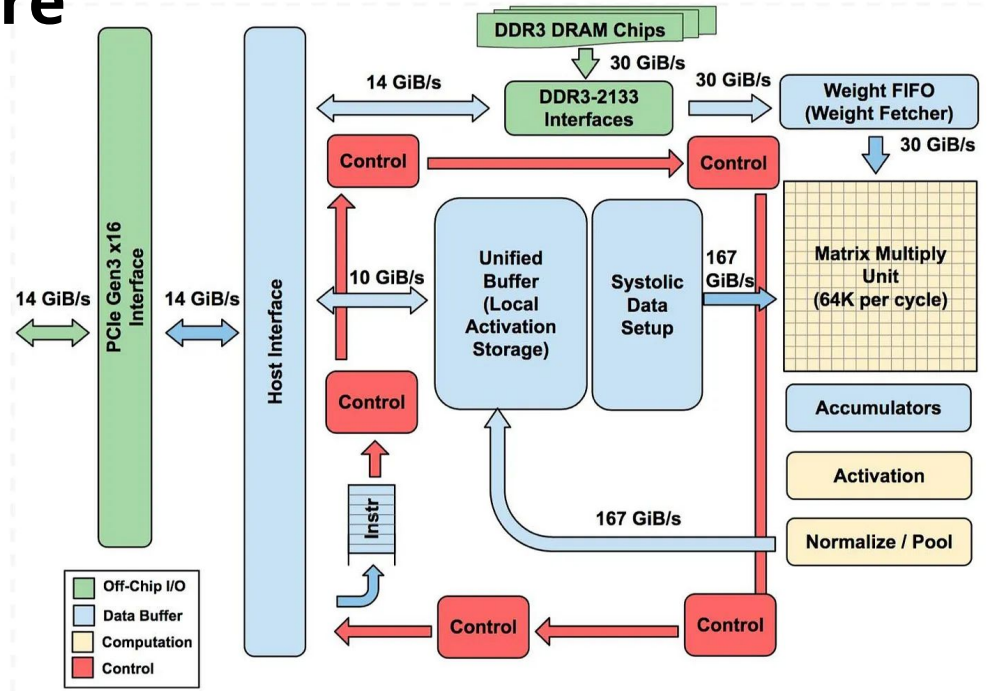
Activate: performs the nonlinear function of the artificial neuron. Inputs are accumulators, output is the unified buffer



Carnegie Mellon University

Hardware Architecture

- 700 MHz
- 24 MiB Unified Buffer
- 8GiB Weight DRAM
- 256x256 8bit Systolic Array
- 4096 32bit 256-element Accumulator
- 4-Stage Pipeline



Carnegie Mellon University

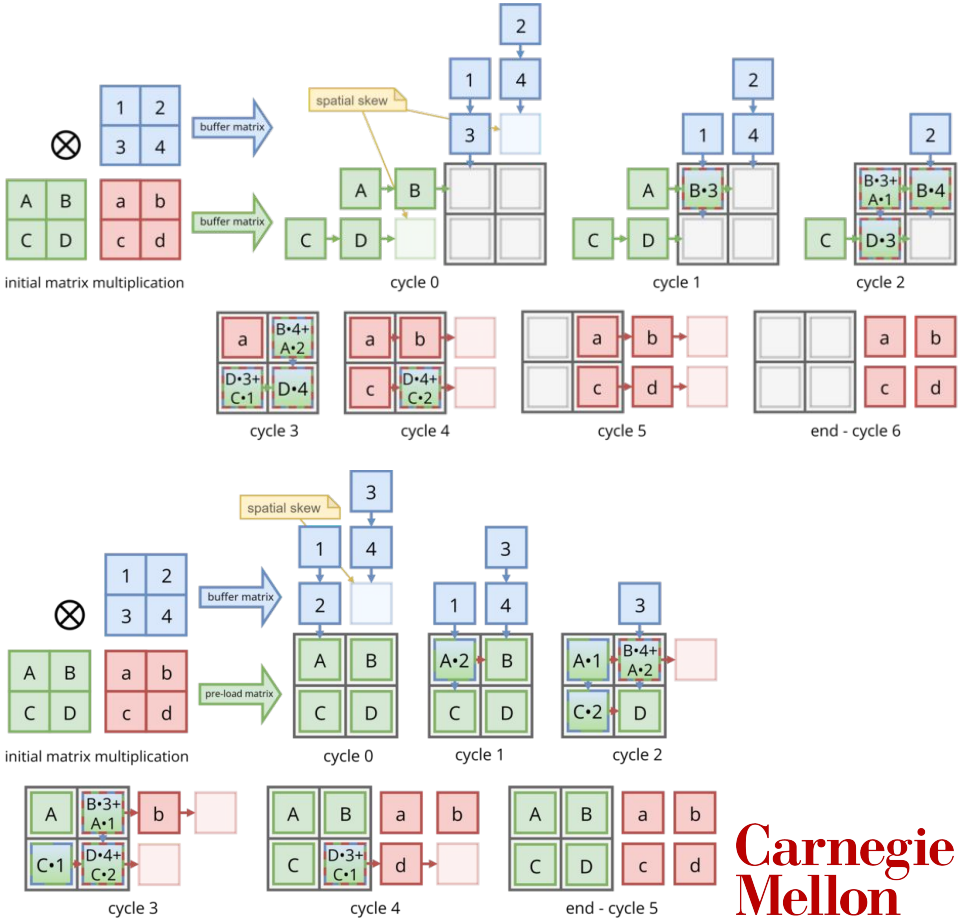
Systolic Array

A bandwidth efficient module for matrix multiplication

- Each matrix is placed on one edge of the array, then transformed and streamed
- Massive data re-use
- Different data-flow orchestrations
  - OS, WS, IS

TPU v1

- Double buffered tiled weights

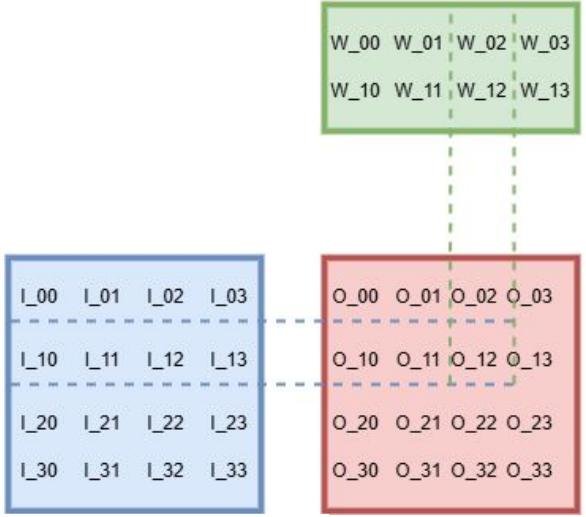


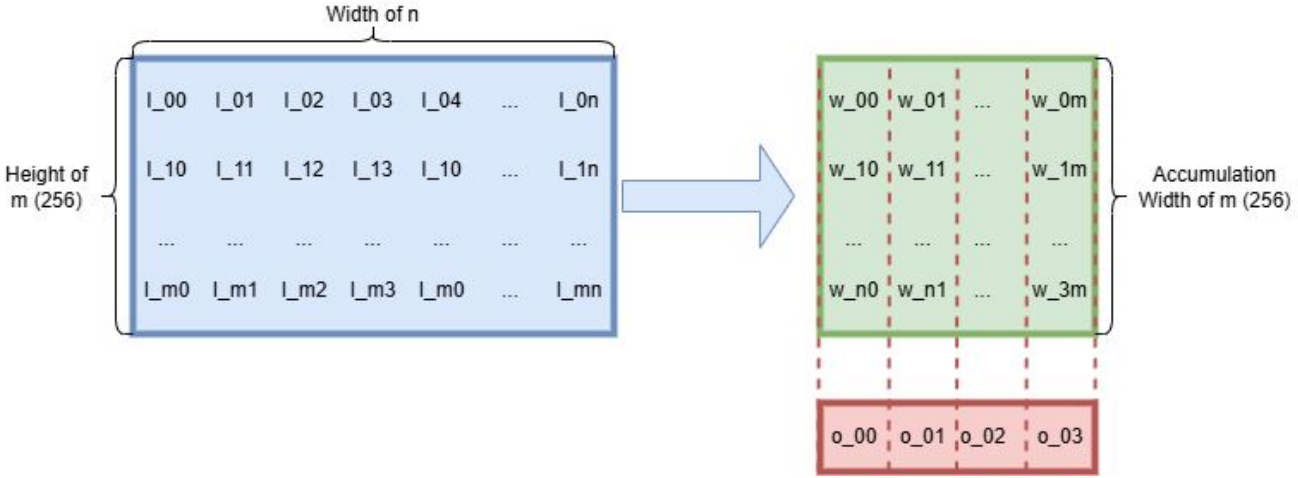
Carnegie Mellon University

Accumulator

Why do we need extra accumulators when systolic array?

- Accumulator is used when matrix are larger than the systolic array
- Different mapping and tiling could require different accumulator organization



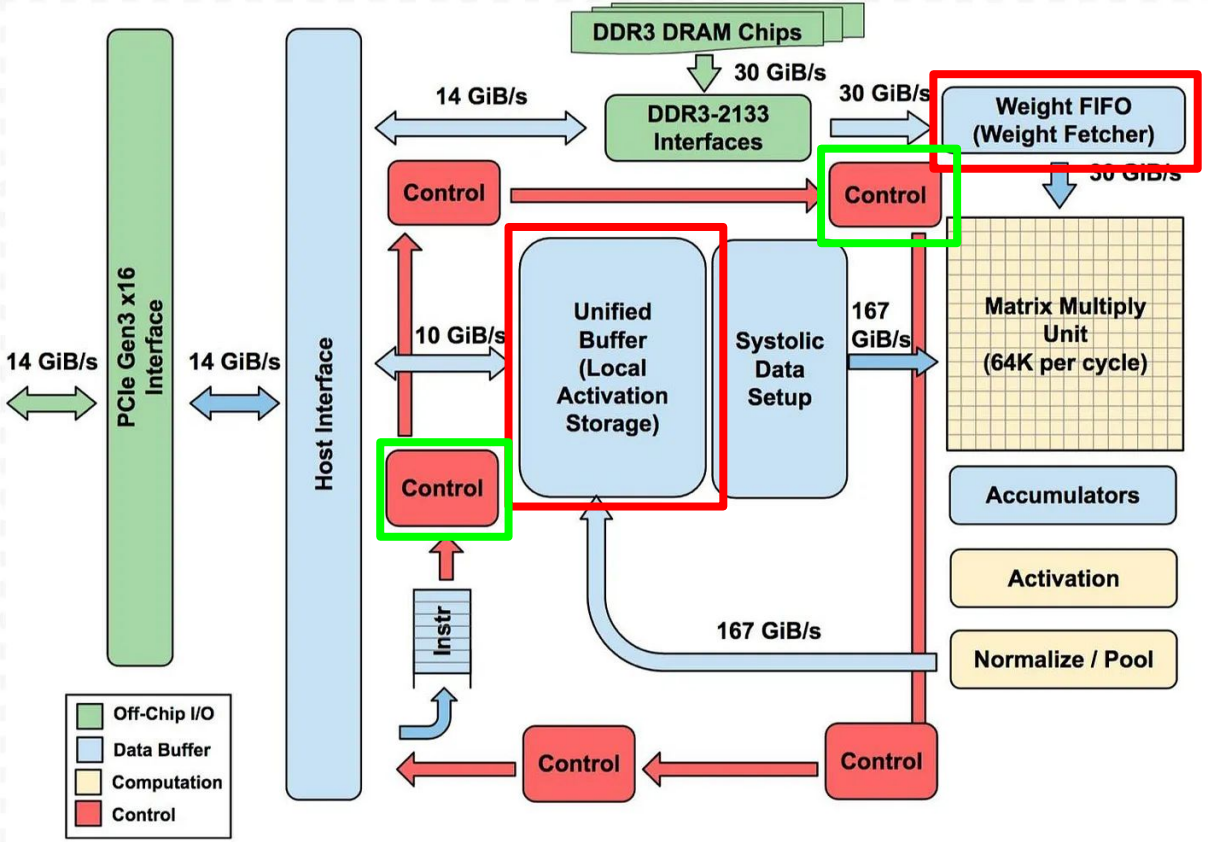


Carnegie Mellon University

Decoupled Access/Execute

Because of the highly deterministic workload

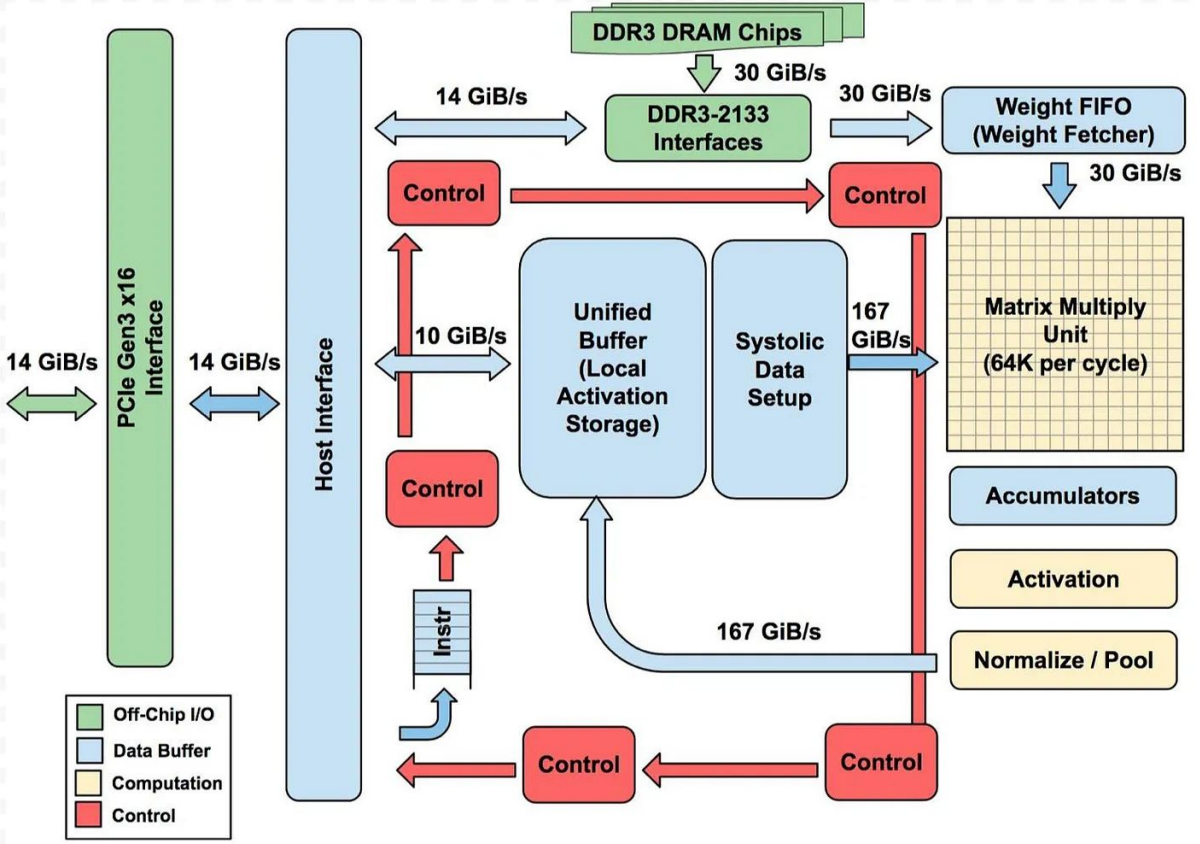
- Preemptively sent future read requests instead of waiting for read miss
- Allows for delay hiding and overlap of Access/Execute
- Double buffering of some execution blocks
  - Local array weight storage
  - Accumulators



Analysis

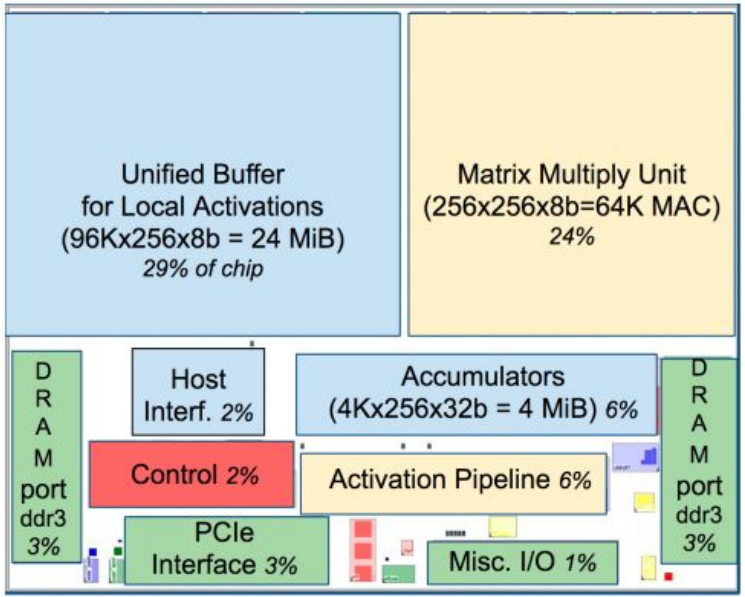
Highly specialized hardware designed for ML inference

- Specialized bit-width across the design
- Separate dedicated path for activation and weight
- Data forwarded between instructions, not always written back to shared buffers.
- Compute bottleneck located in systolic array, resulted in uneven pipelining
- Simple memory hierarchy



Area Allocation and Comparison

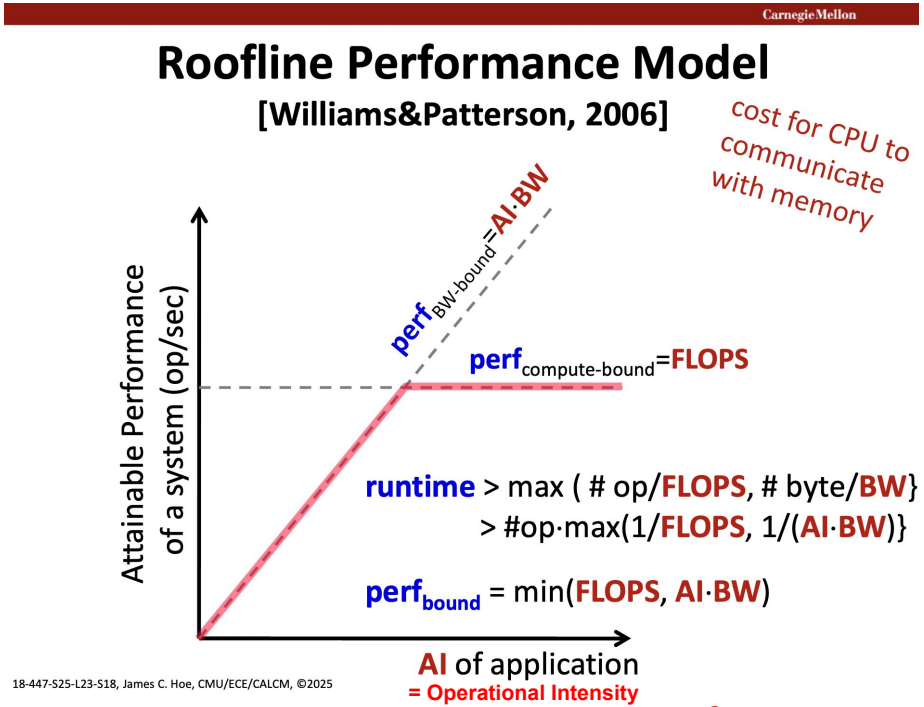
- Relatively low clock speed
- Large On-Chip memory
- Extremely large int8 TOPS/s
- Good amount of local DRAM
- Low per-die TDP
  - Small Idle vs Active power difference



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	<331*	28	700	75W	28W	40W	92	--	34	28 MiB	4	256 GiB (host) + 8 GiB x 4	861W	290W	384W

Performance: Roofline Model Introduction

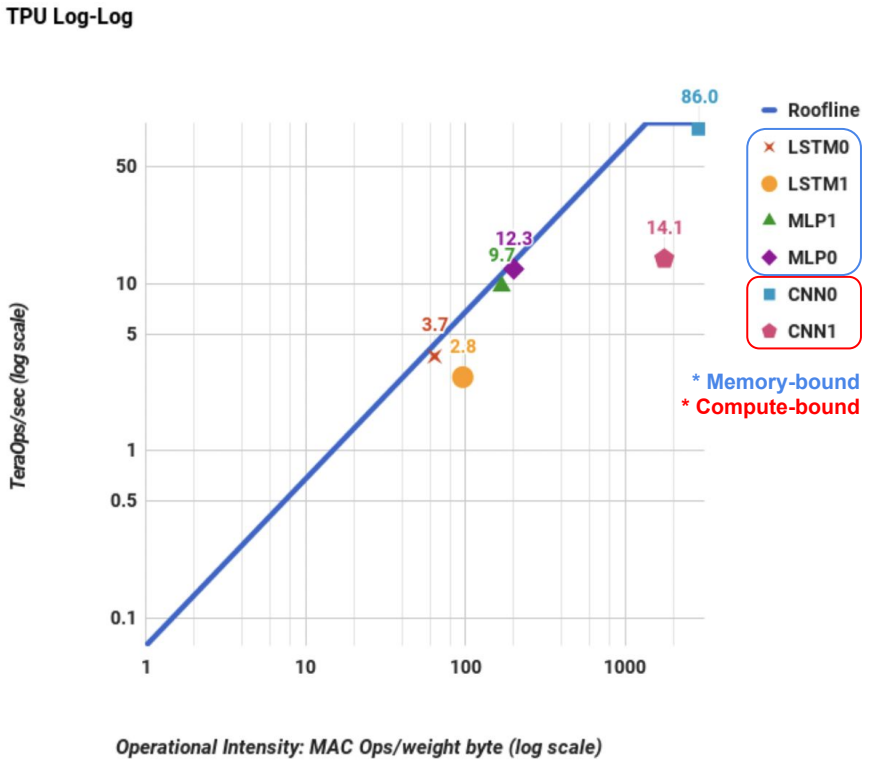
- Not perfect but used to give insights on bottlenecks
- Applications are either computation-limited (flat line) or memory bandwidth-limited (slanted line)
- Y-axis is operations per second (e.g. FLOPS)
- X-axis is arithmetic intensity or operational intensity (e.g. FLOPS/byte) of application
- Ideally datapoints are on or close to the ceiling to max utilization of hardware



Carnegie Mellon University

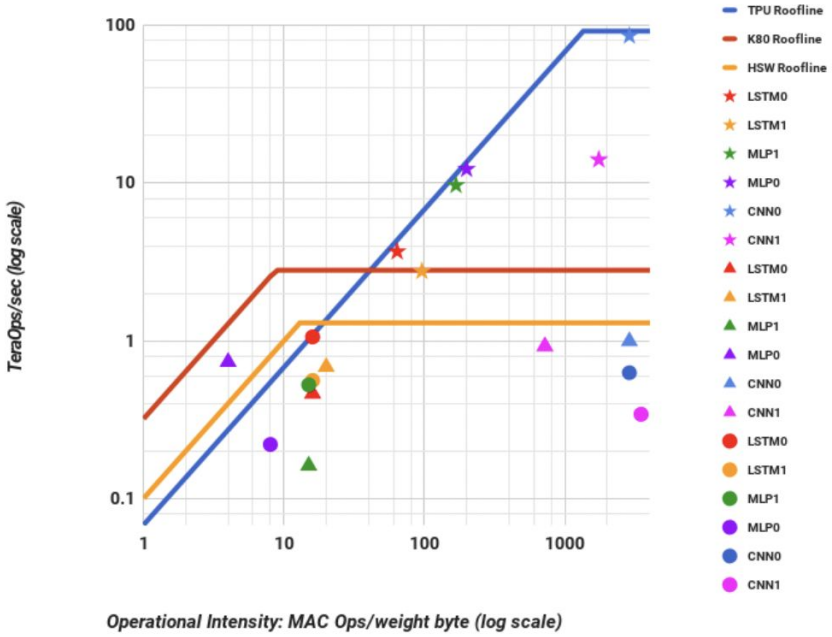
TPU Performance

- Five of the six applications are close to the ceiling (CNN1 is an outlier)
- CNN0/1 are compute-bound while the rest are memory-bound
- The ridge point is far to the right because of its high peak compute performance (92 TeraOps/s)



TPU Performance Comparison

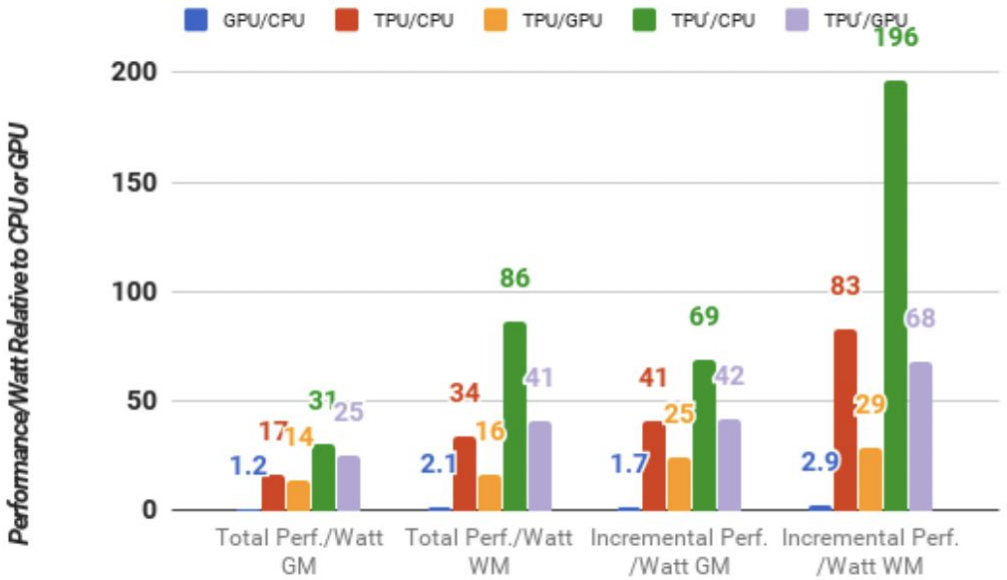
- Comparisons to CPU (Intel Haswell) and GPU (Nvidia K80) show their maximum throughput is underutilized because of response time limitations (e.g. 7ms for MLP0)**
- They run 2.3X-2.7X slower than if response time was unbounded, but for TPU the slowdown is only 1.2X
- Larger batch sizes and input queues increase throughput (IPS) but their longer response times exceed the limit



Carnegie Mellon University

Cost-Performance, TCO, and Performance/Watt

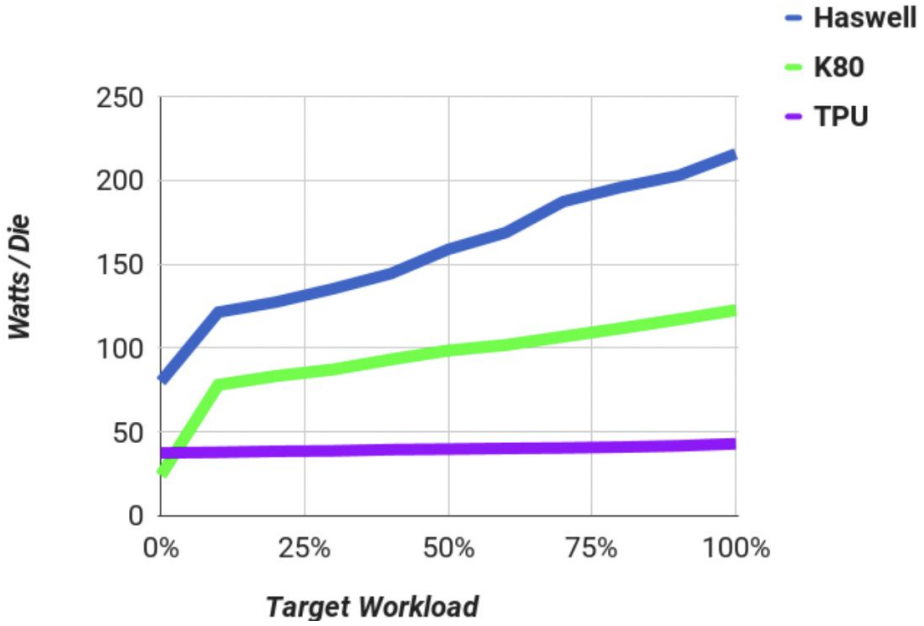
- Total Cost of Ownership (TCO) is important when buying thousands of computers
- They use performance/watt as a proxy for performance/TCO
- Incremental performance doesn't include host CPU power
- A TPU/CPU incremental power improvement of 41X – 83X was Google's key motivating figure for their custom ASIC



Carnegie Mellon University

Thermal Design Power and Energy Proportionality

- The TPU has the lowest power at 40W per die
- The cost of electricity is based upon the average consumed as the workload varies during the day
- Energy proportionality: servers should consume power proportional to the amount of work performed.
- But it also has poor energy proportionality: at 10% load, the TPU uses 88% of the power it uses at 100%.



Low Systolic Array Utilization

- Lacking Weight bandwidth
  - Systolic seems to have been optimized for more symmetric matrix multiplication workloads
- RAW Pipeline Hazards
  - We speculate layer to layer RAW issues
  - Output of systolic array results for layer i will be immediately needed for layer i+1
  - While the result still need to finish accumulation. activation. unified buffer

Application	*MLP0*	*MLP1*	*LSTM0*	*LSTM1*	*CNN0*	*CNN1*	*Mean*	*Row*
Array active cycles	12.7%	10.6%	8.2%	10.5%	78.2%	46.2%	28%	1
Useful MACs in 64K matrix (% peak)	12.5%	9.4%	8.2%	6.3%	78.2%	22.5%	23%	2
Unused MACs	0.3%	1.2%	0.0%	4.2%	0.0%	23.7%	5%	3
Weight stall cycles	53.9%	44.2%	58.1%	62.1%	0.0%	28.1%	43%	4
Weight shift cycles	15.9%	13.4%	15.8%	17.1%	0.0%	7.0%	12%	5
Non-matrix cycles	17.5%	31.9%	17.9%	10.3%	21.8%	18.7%	20%	6
RAW stalls	3.3%	8.4%	14.6%	10.6%	3.5%	22.8%	11%	7
Input data stalls	6.1%	8.8%	5.1%	2.4%	3.4%	0.6%	4%	8
TeraOps/sec (92 Peak)	12.3	9.7	3.7	2.8	86.0	14.1	21.4	9

Broader Industry Context and Critiques

- Utilization of ideas that didn't work in general-purpose computing may work for domain specific architecture (ie CISC, systolic arrays, decoupled access/execute)
- Software tuning could still lead to performance increases
  - Example of CNN1 could be further tuned to match TPU hardware
- At the time, architects were overly accelerating CNNs as opposed to other NNs
- Don't need to value throughput as much in inference in this case
  - And is a reason for moving away from CPU/GPU architecture
- Inference per second was a reductive metric

Discussion: Summary Question #1

What Did the Paper Get Right?

State the 3 most important things the paper says.

These could be some combination of the motivations, observations, interesting parts of the design, or clever parts of the implementation.

Discussion: Summary Question #2

What Did the Paper Get Wrong?

Describe the paper's single most glaring deficiency.

Every paper has some fault. Perhaps an experiment was poorly designed or the main idea had a narrow scope or applicability.

Shortcomings

- Inference only
- Less focus host server side delay
  - can’t tell when CPU idle
- Low systolic array utilization
  - Systolic array fragmentation?
  - RAW between layers
- Poor energy proportionality
- Not truly streaming
- Precision difference
  - Went the route of making everything floating point