In-Datacenter Performance Analysis of a Tensor Processing Unit

Presented by Josh Fried

Background: Machine Learning

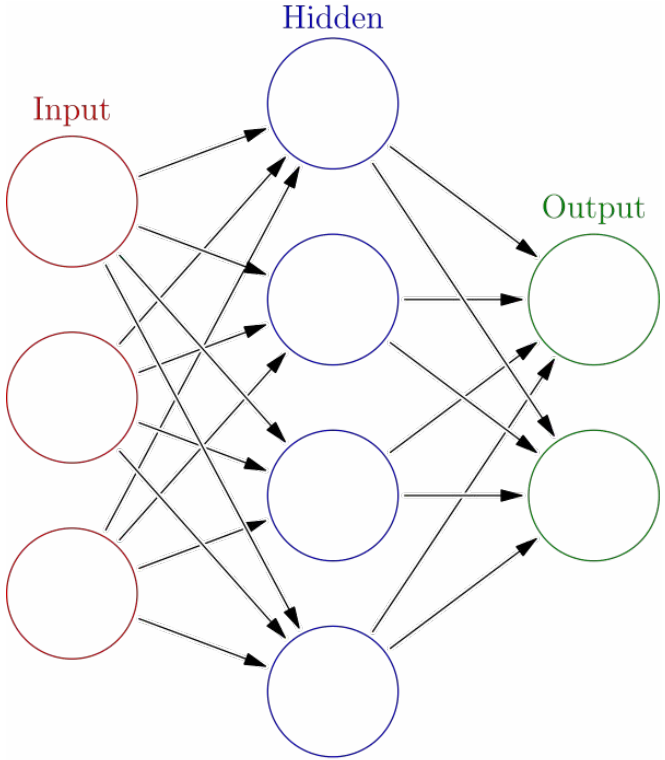
Neural Networks:

- Multi Layer Perceptrons
- Recurrent Neural Networks (mostly LSTMs)
- Convolutional Neural Networks

Synapse - each edge, has a weight

Neuron - each node, sums weights and uses non-linear activation function over sum

Propagating inputs through a layer of the NN is a matrix multiplication followed by an activation



Background: Machine Learning

Two phases:

- **Training** (offline)
  - relaxed deadlines
  - large batches to amortize costs of loading weights from DRAM
  - well suited to GPUs
  - Usually uses floating points
- **Inference** (online)
  - strict deadlines: 7-10ms at Google for some workloads
    - limited possibility for batching because of deadlines
  - Facebook uses CPUs for inference (last class)
  - Can use lower precision integers (faster/smaller/more efficient)

ML Workloads @ Google

90% of ML workload time at Google spent on MLPs and LSTMs, despite broader focus on CNNs

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	

Inception (image classification), AlphaGo

RankBrain (search)
Google Translate
(and others)

Background: Hardware Trends

End of Moore's Law & Dennard Scaling

- Moore - transistor density is doubling every two years
- Dennard - power stays proportional to chip area as transistors shrink

Machine Learning causing a huge growth in demand for compute

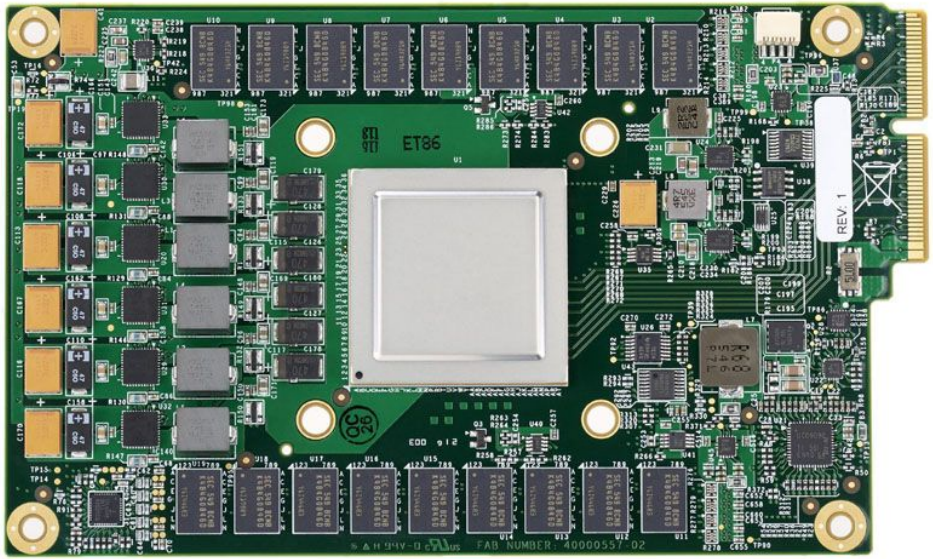
- 2006: Excess CPU capacity in datacenters is enough
- 2013: Projected 3 minutes per-day per-user of speech recognition
  - will require doubling datacenter compute capacity!

Google’s Answer: Custom ASIC

Goal: Build a chip that improves *cost-performance* for NN inference

What are the main costs?

- Capital Costs
- Operational Costs (power bill!)



TPU (V1) Design Goals

Short design-deployment cycle: ~15 months!

Plugs in to PCIe slot on existing servers

Accelerates matrix multiplication operations

Uses 8-bit integer operations instead of floating point

How does the TPU work?

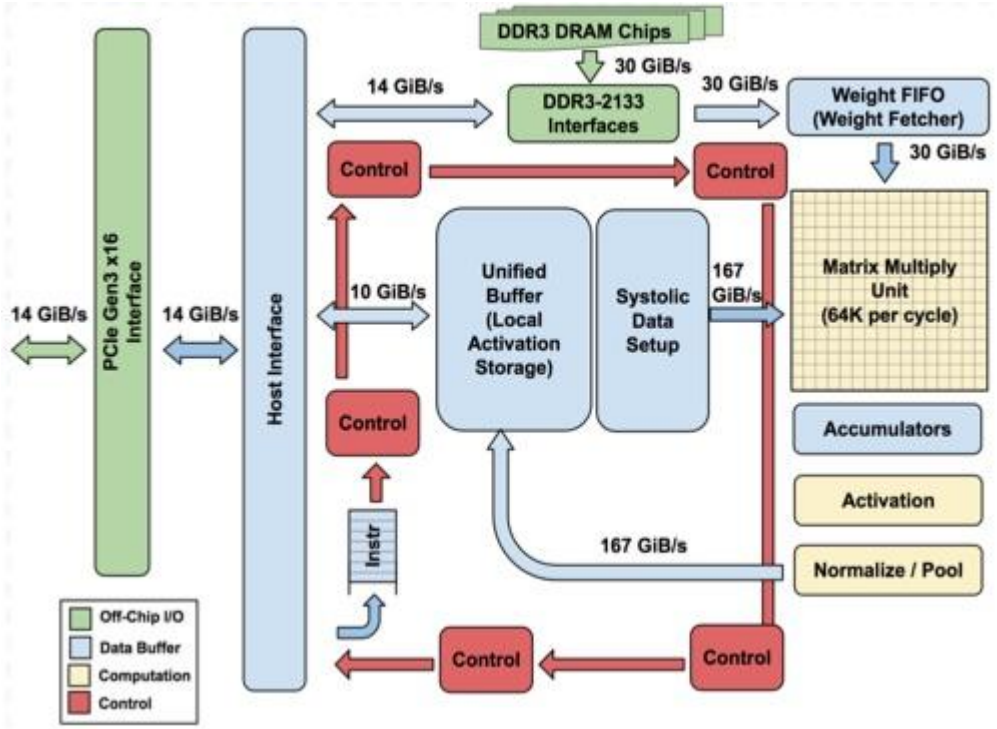
CISC instructions, issued by host.

Why CISC?

What are alternative designs?

<u>Main Instructions:</u>

- Read/Write Host Memory
- Read Weights
- Matrix Multiply/Convolve
- Activate



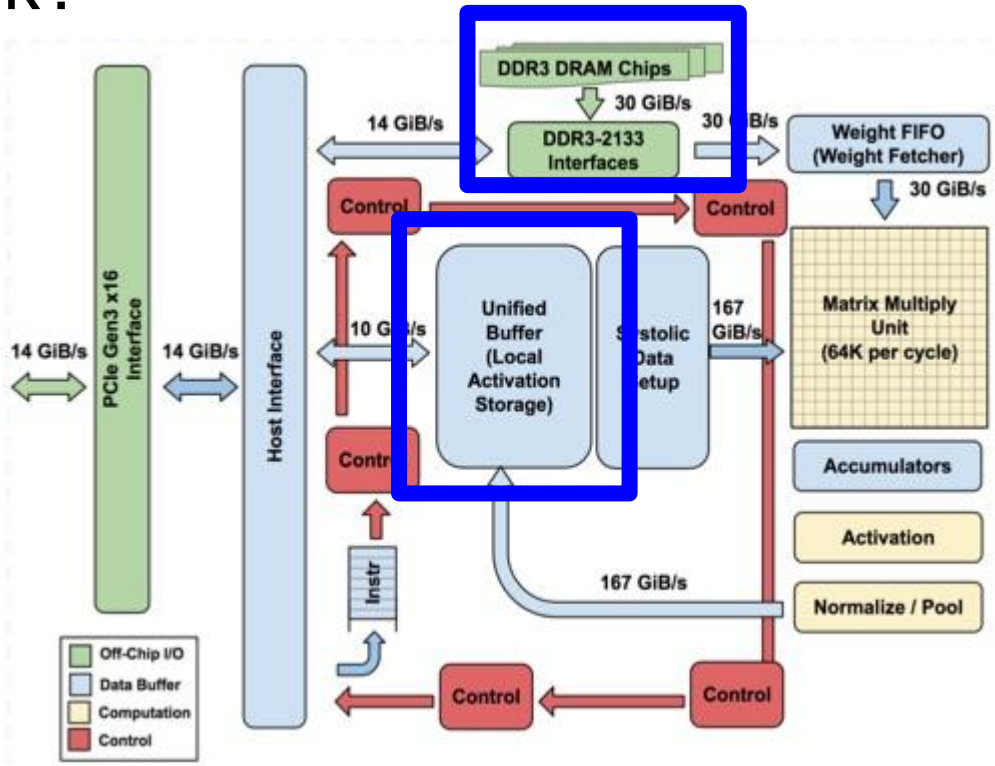
How does the TPU work?

24 MiB of on-chip SRAM

- 30% of chip space

8 GB off-chip DDR3 (30 GB/s)

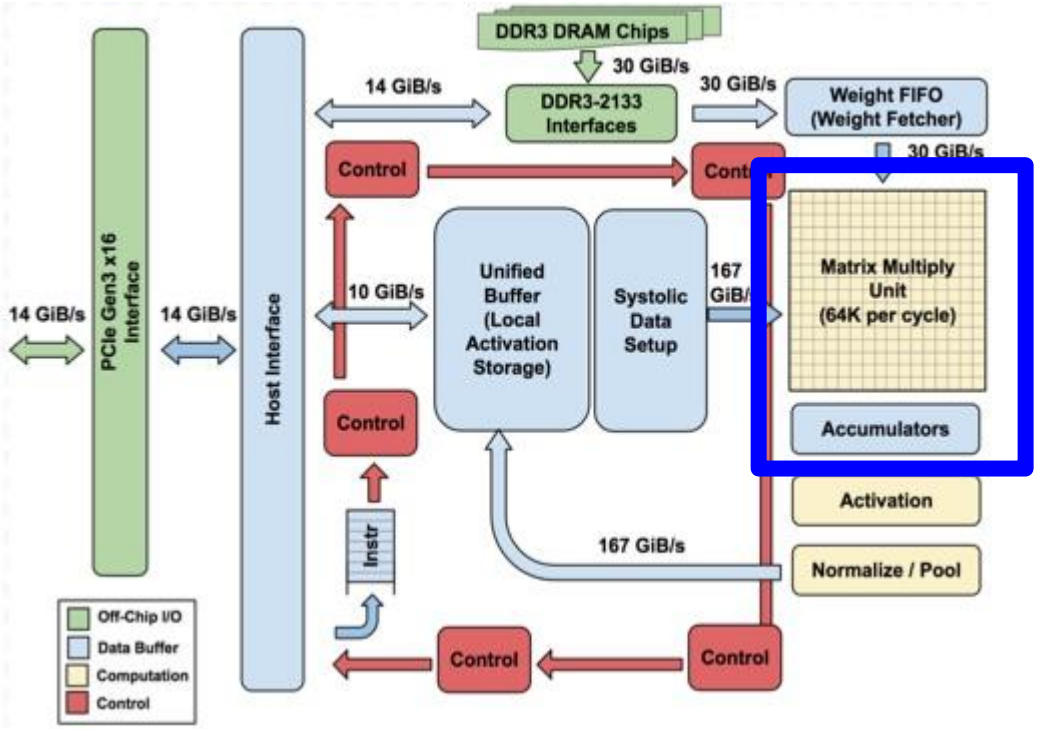
- stores weights for models



How does the TPU work?

Systolic Matrix Multiplication

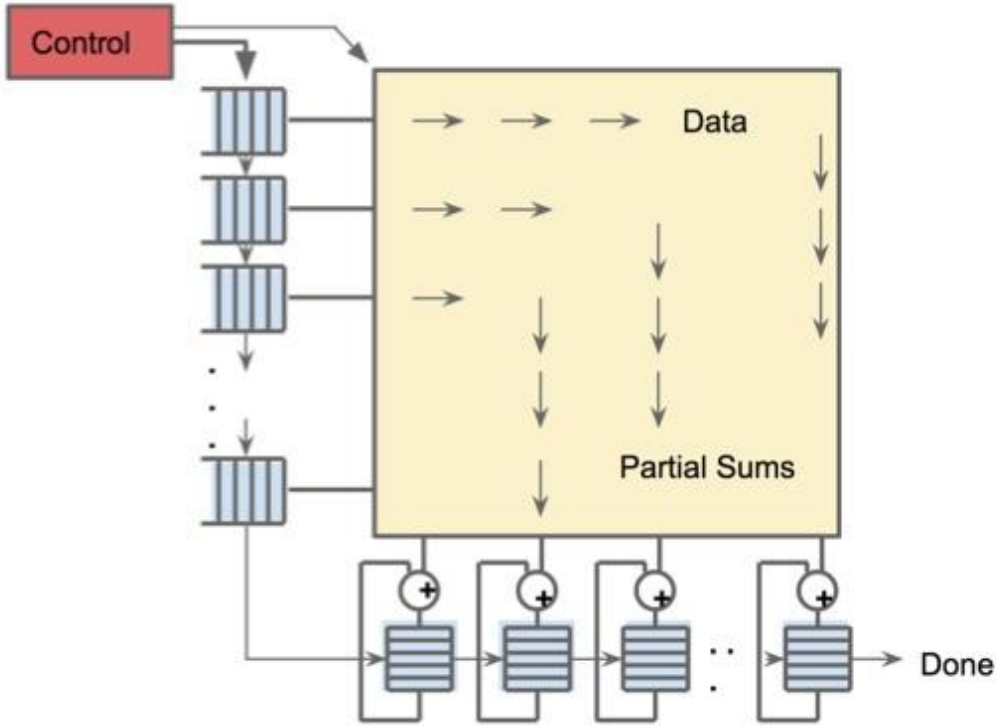
- Heart of the TPU
- 256 x 256 array of 8-bit multiplier-accumulators (MACs)
- Weights are preloaded
  - Matrix unit can hold 2 full sets of weights - Why?



How does the TPU work?

Systolic Matrix Multiplication

- Input data from unified buffer flows through unit in a wave
  - Each multiplier multiplies its input data and adds to accumulator
  - Each passes the input data to the MAC to its right, and the partial sum to the unit below



Systolic Matrix Multiplication

Q: How does systolic matrix multiplication contribute towards improved cost-performance?

Q: How is matrix multiplication performed on CPUs and GPUs?

Evaluating the TPU

- How do Google's ML workloads perform on the TPU and what are their bottlenecks?

- What are the overall improvements in cost-performance for the TPU?

- How could the TPU be improved in the future?

Comparison with CPUs and GPUs

3 Hardware Configurations Evaluated:

CPU Only (base machine): 2x Haswell processors

GPU: Base machine plus 4x NVIDIA K80 cards

TPU: Base machine plus 4x TPU cards

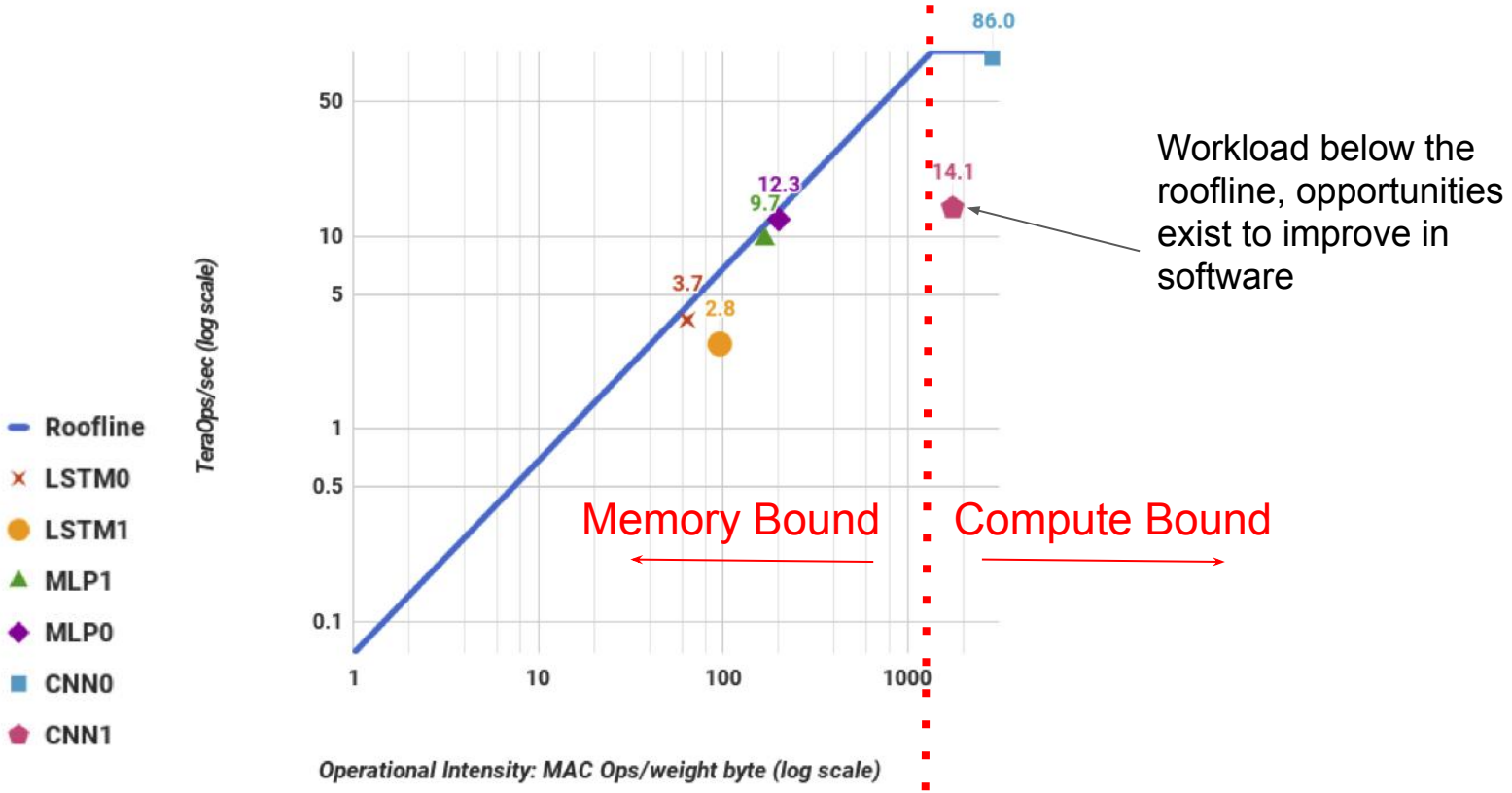
Model	*Die*										*Benchmarked Servers*				
	mm²	*nm*	*MHz*	*TDP*	*Measured*		*TOPS/s*		*GB/s*	*On-Chip Memory*	*Dies*	*DRAM Size*	*TDP*	*Measured*	
					Idle	*Busy*	8b	FP						*Idle*	*Busy*
Haswell E5-2699 v3	662	22	2300	145W	41W	145W	2.6	1.3	51	51 MiB	2	256 GiB	504W	159W	455W
NVIDIA K80 (2 dies/card)	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

Workload Performance: The Roofline Model

Illustrates performance bottlenecks for a given piece of hardware

- Is a workload compute bound or memory bound? Is the software underutilizing hardware?
- Term: **Operational intensity**
  - Operations performed per byte of memory read
    - For TPUs/NNs: MAC operations per byte of weight memory
  - Mostly fixed based on the workload - when is it not?

Workload Performance: The Roofline Model



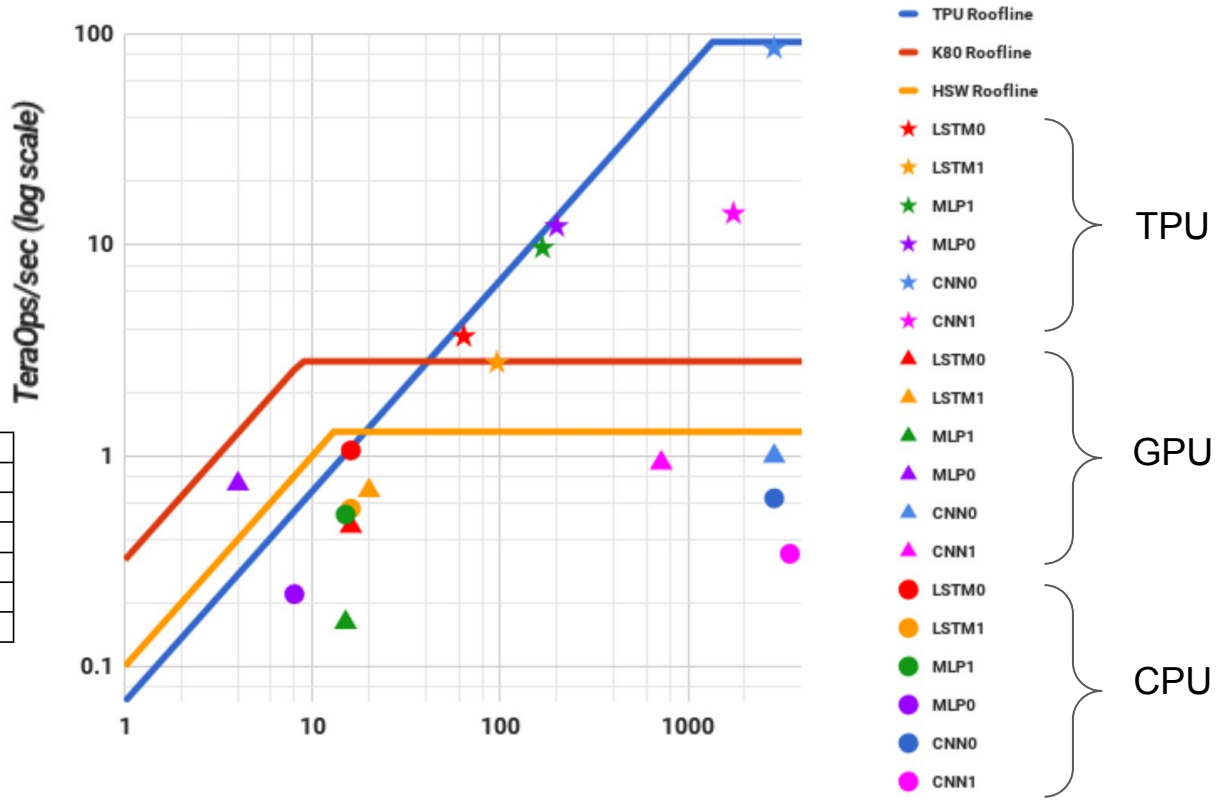
Workload below the roofline, opportunities exist to improve in software

Workload Performance: The Roofline Model

Q: Why are GPU/CPU workloads generally below their rooflines?

A: Latency limits prevent full utilization of hardware

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%



Overall Performance

Performance relative to CPUs (per-die)

Type	DNN		LSTM		CNN		GM	WM
	0	1	0	1	0	1		
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

Avg. Improvement over CPU

Avg. Improvement (weighted by workload popularity)

Host Server Overheads

Host time spent interacting with TPU (% of TPU time on workload)

MLP0	*MLP1*	*LSTM0*	*LSTM1*	*CNN0*	*CNN1*
21%	76%	11%	20%	51%	14%

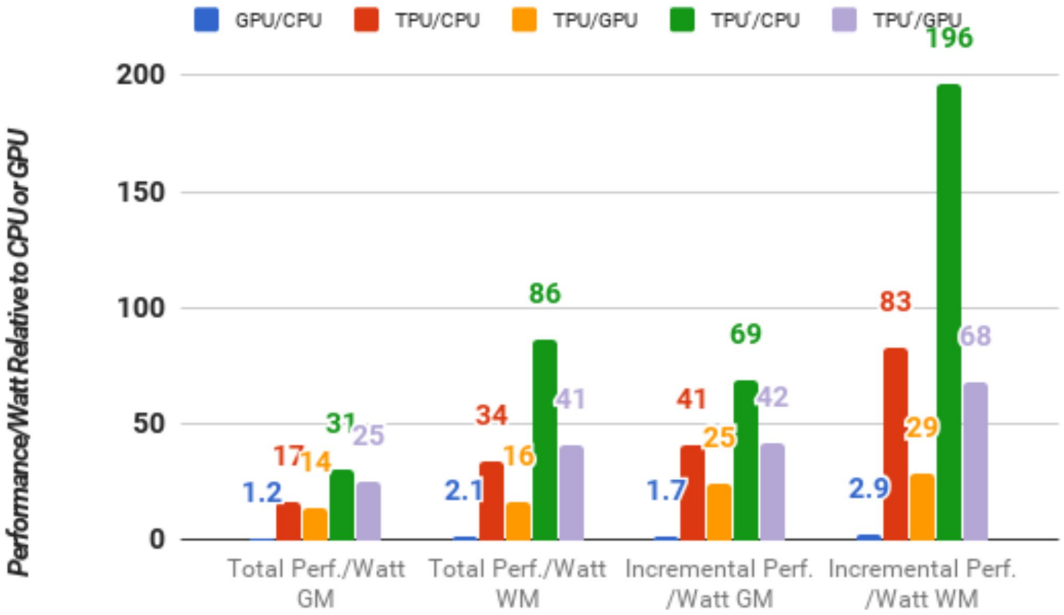
Overall Cost-Performance

Goal: Improve Perf/Total Cost of Ownership (TCO)

- Google can only show power consumption (correlates with TCO)

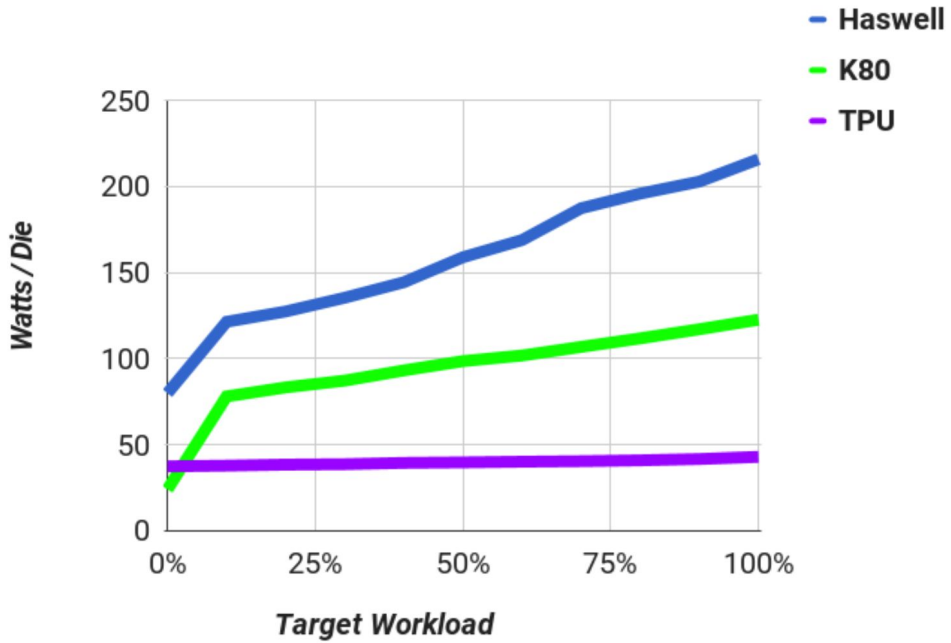
Total Perf/Watt:
17-34X over CPU
14-16X over GPU

Incremental:
41-83X over CPU
25-29X over GPU



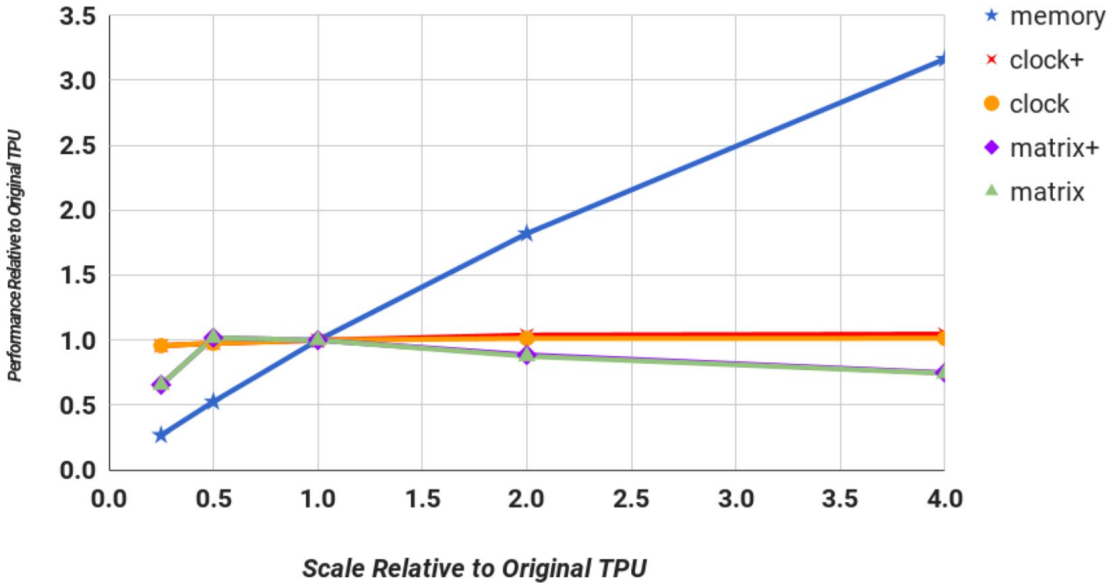
Energy Proportionality

TPUs do not scale energy consumption with utilization



Alternative TPU Configurations

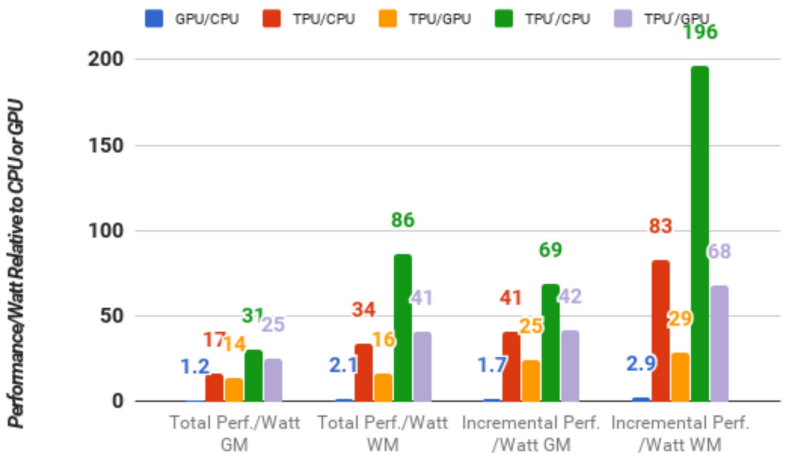
Using a model of the TPU to simulate the workloads running on alternate designs, what hardware changes would help most with performance?



Hypothetical TPU (TPU')

Uses GDDR5 memory (5x bandwidth improvement)

- Increases power budget by ~10 W / die
- Expand die size by around 10%
  - can recover from Unified Buffer - Why?
- Would dramatically improve Perf/Watt!
  - 31 - 86X improvement over CPU
  - 25 - 41X improvement over GPU



Newer GPUs?

	K80 2012	TPU 2015	P40 2016
Inferences/Sec <10ms latency	$^1/_{13}$X	1X	2X
Training TOPS	6 FP32	NA	12 FP32
Inference TOPS	6 FP32	90 INT8	48 INT8
On-chip Memory	16 MB	24 MB	11 MB
Power	300W	75W	250W
Bandwidth	320 GB/S	34 GB/S	350 GB/S

2X

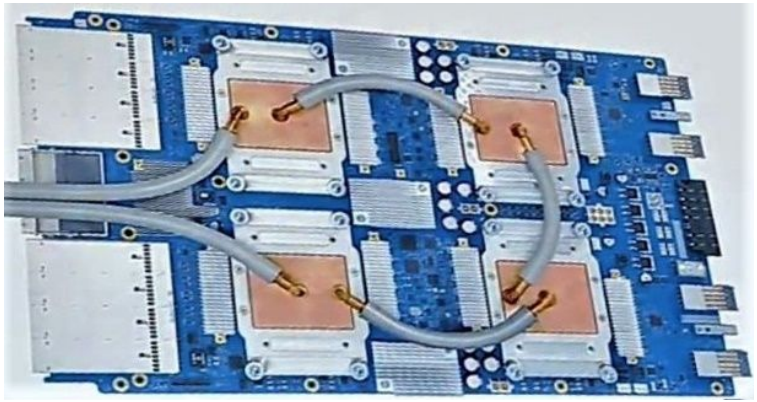
3.3X

https://blogs.nvidia.com/blog/2017/04/10/ai-drives-rise-accelerated-computing-datacenter/

TPU v2 and v3

V2 (“Cloud TPUs”):

- add High Bandwidth Memory (600 GB/s)
- Add support for floating point ops
  - new format, bfloat16, wider range + less precision
  - Allows for training to run on TPUs too
- Pods: 64 TPUs group with high speed interconnect
  - 11.5 Petaflops, 4TB memory

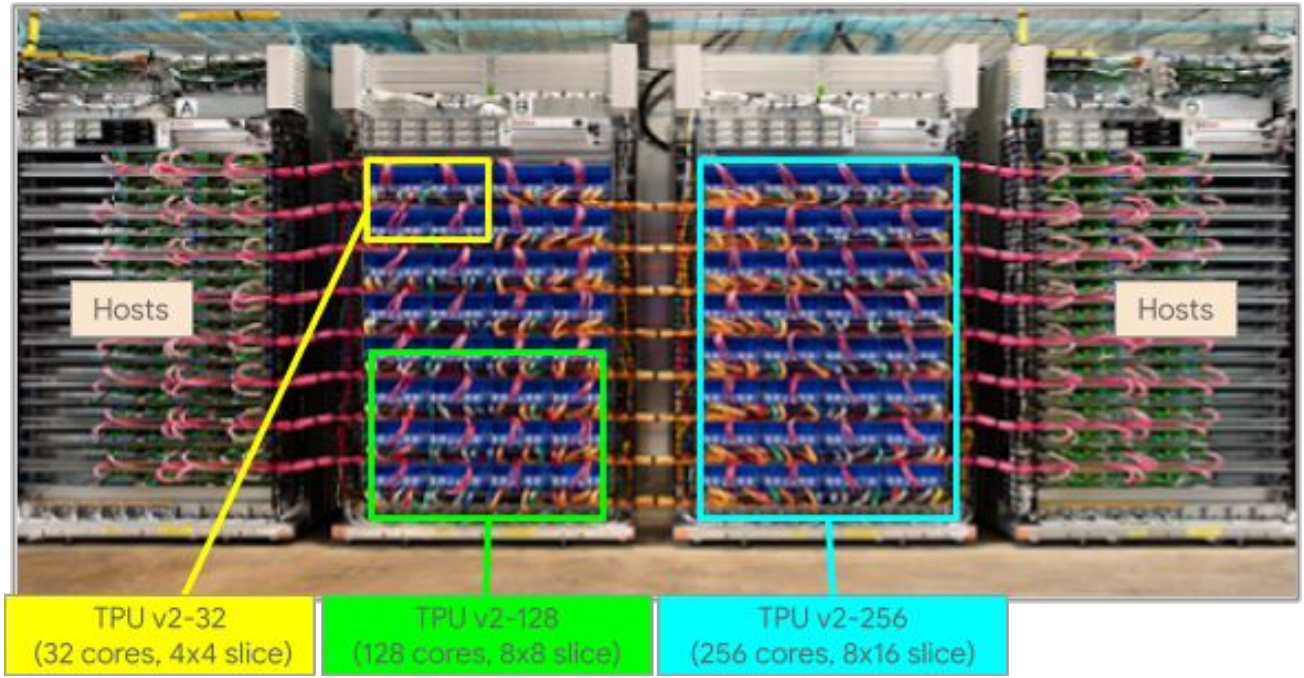


V3:

- Liquid cooled chips, larger pods
- >100 petaflops, 32TB memory

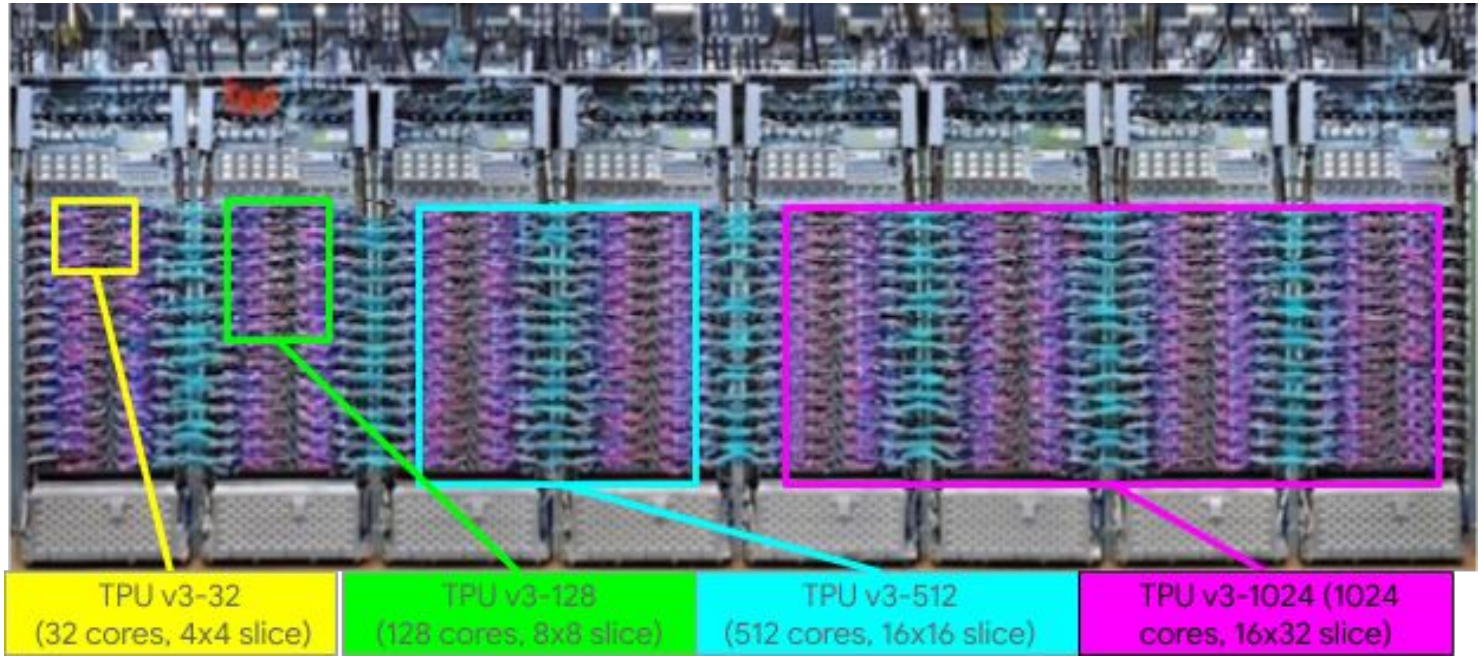
TPU Pods

Google Cloud lets you allocate slices of TPU pods



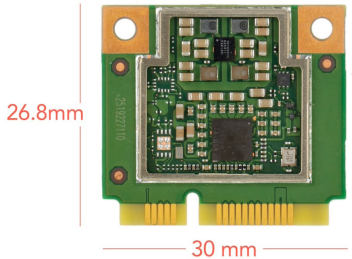
TPU Pods

Google Cloud lets you allocate slices of TPU pods



What else is out there?

- Optical Chip (Lightmatter)
  - Matrix processing unit that computes result in ~1 cycle!
- Cerebras
  - Single really large chip with fast communication between between cores
  - 400K cores, 18GB SRAM, interconnect is O(PB/s)
  - At least one is deployed in a supercomputing center
- Apple A11/12/13/14
  - Matrix accelerators (“neural engines”), 11 tera-op/s, 88 mm^2 area
- Edge TPU, 2 Tera-op/s



What else is out there?

Apple M1