

# CS 744: TPU

Shivaram Venkataraman

Spring 2025

# ADMINISTRIVIA

Midterm 2, April 24<sup>th</sup>

- Papers from FI Query to Luminix
- Similar format as first midterm
- Bring anything printed/written
- Details on Piazza

Poster session: May 1st

- More details soon

# MOTIVATION

Capacity demands on datacenters

New workloads

Metrics

- Power/operation

- Performance/operation

- Total cost of ownership

Goal: Improve cost-performance by 10x over GPUs

# WORKLOAD

<i>Name</i>	<i>LOC</i>	<i>Layers</i>					<i>Nonlinear function</i>	<i>Weights</i>	<i>TPU Ops / Weight Byte</i>	<i>TPU Batch Size</i>	<i>% of Deployed TPUs in July 2016</i>
		<i>FC</i>	<i>Conv</i>	<i>Vector</i>	<i>Pool</i>	<i>Total</i>					
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	

DNN: RankBrain, LSTM: subset of GNM Translate

CNNs: Inception, DeepMind AlphaGo

# WORKLOAD: ML INFERENCE

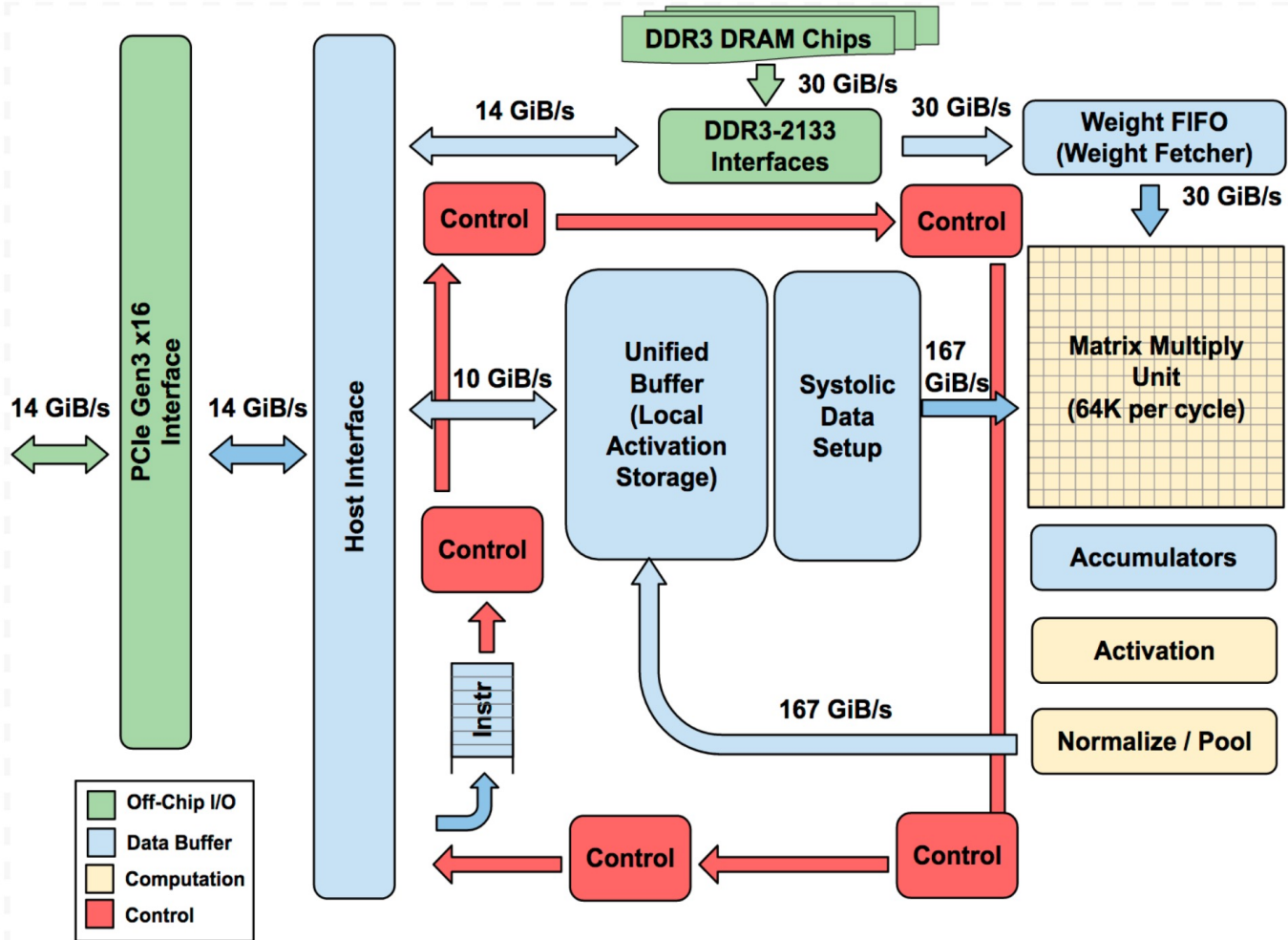
Quantization → Lower precision, energy use

8-bit integer multiplies (unlike training), 6X less energy and 6X less area

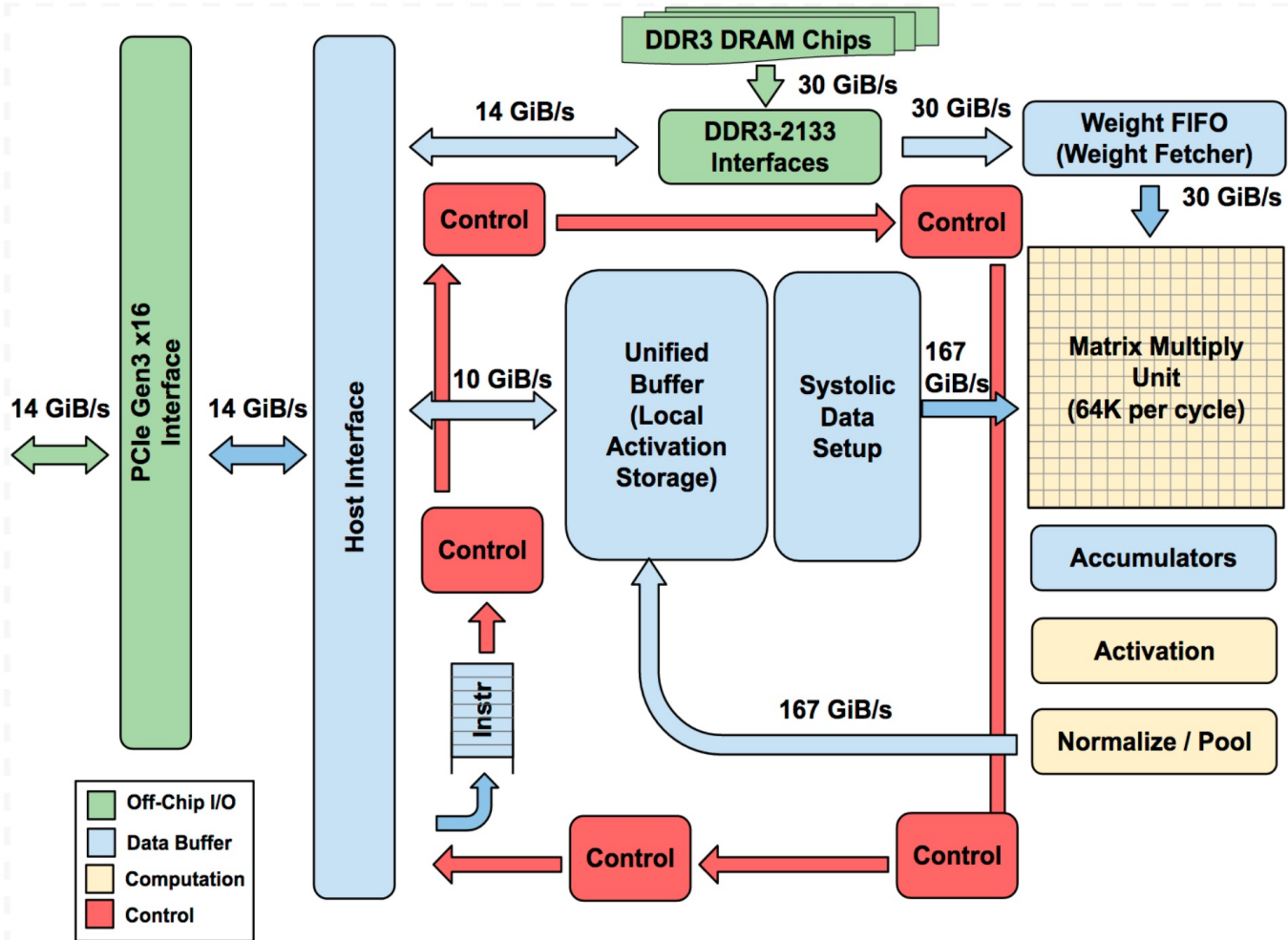
Need for predictable latency and not throughput

e.g., 7ms at 99th percentile

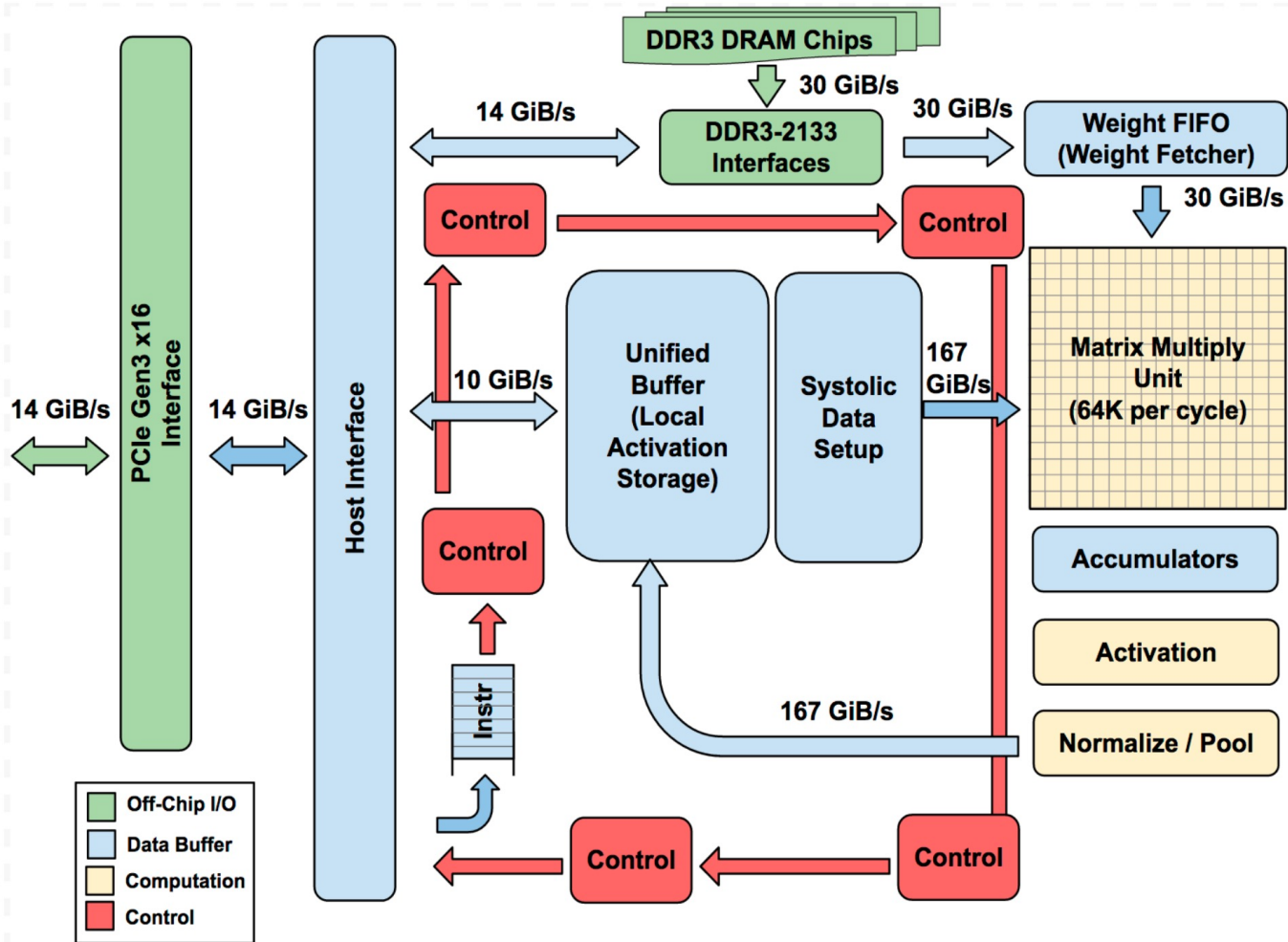
# TPU DESIGN CONTROL



# COMPUTE



# DATA



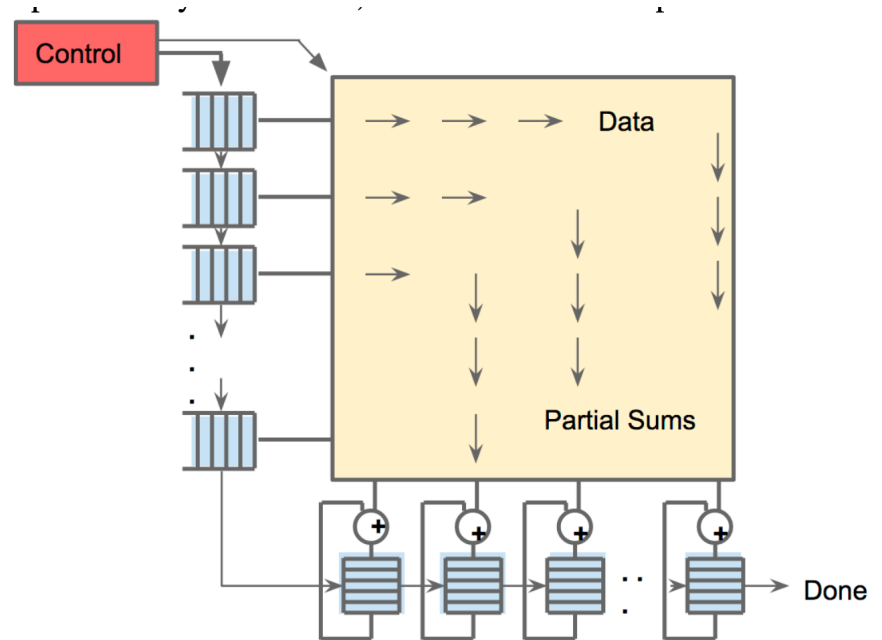
# INSTRUCTIONS

CISC format (why ?)

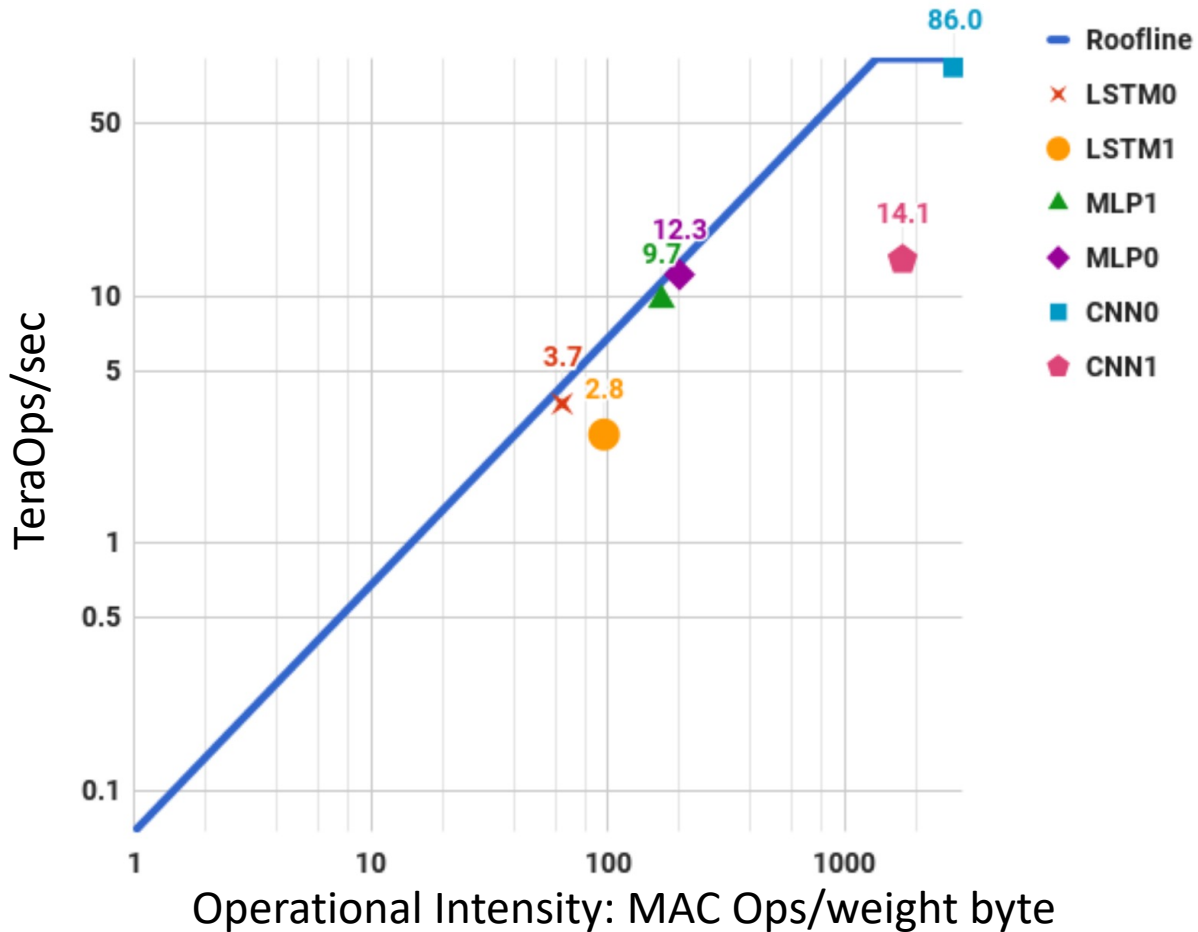
1. Read\_Host\_Memory
2. Read\_Weights
3. MatrixMultiply/Convolve
4. Activate
5. Write\_Host\_Memory

# SYSTOLIC EXECUTION

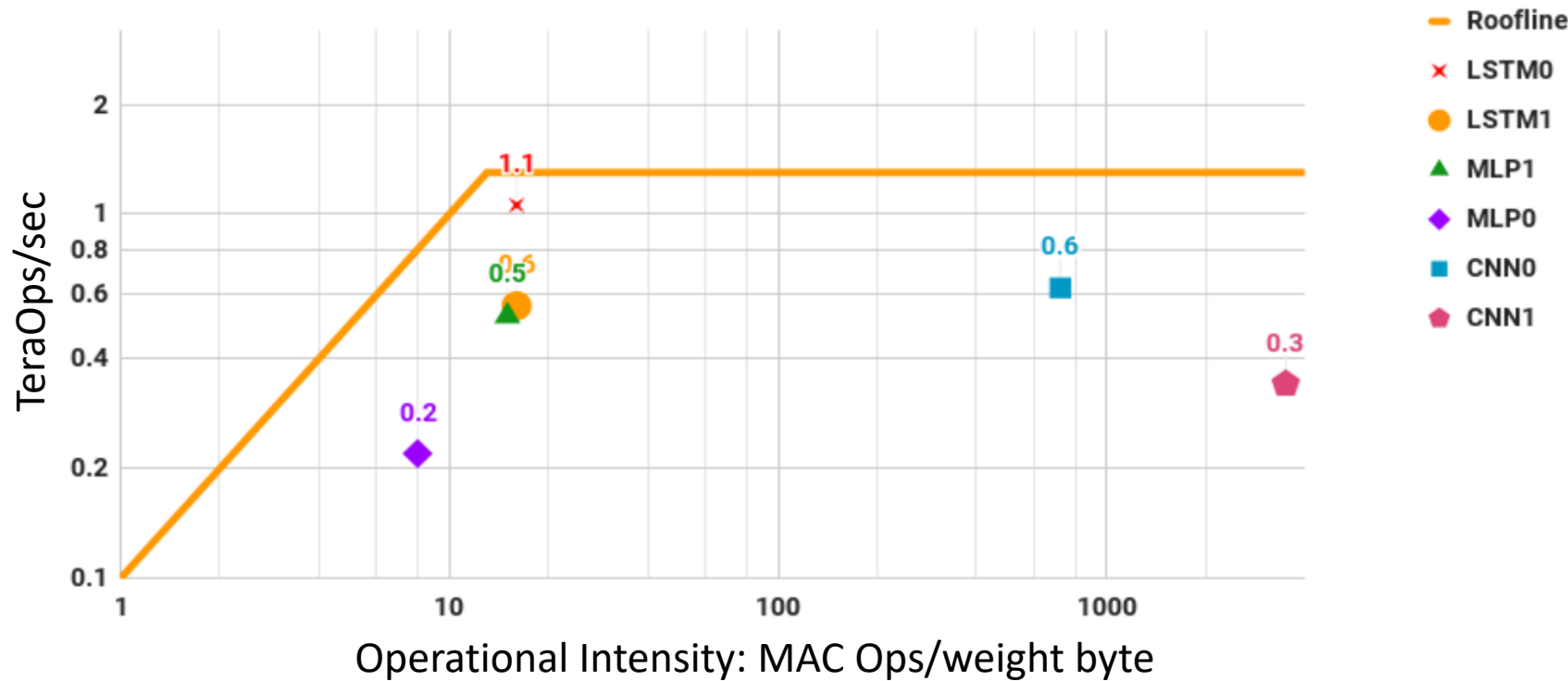
Problem: Reading a large SRAM uses much more power than arithmetic!



# ROOFLINE MODEL



# HASWELL ROOFLINE



# COMPARISON WITH CPU, GPU

<i>Model</i>	<i>Die</i>									
	<i>mm<sup>2</sup></i>	<i>nm</i>	<i>MHz</i>	<i>TDP</i>	<i>Measured</i>		<i>TOPS/s</i>		<i>GB/s</i>	<i>On-Chip Memory</i>
					<i>Idle</i>	<i>Busy</i>	8b	FP		
Haswell E5-2699 v3	662	22	2300	145W	41W	145W	2.6	1.3	51	51 MiB
NVIDIA K80 (2 dies/card)	561	28	560	150W	25W	98W	--	2.8	160	8 MiB
TPU	<331*	28	700	75W	28W	40W	92	--	34	28 MiB

# WHAT HAPPENED NEXT?

<i>DNN Model</i>	<i>TPU v1 7/2016 (Inference)</i>	<i>TPU v3 4/2019 (Training &amp; Inference)</i>	<i>TPU v4 Lite 2/2020 (Inference)</i>	<i>TPU v4 10/2022 (Training)</i>
MLP/DLRM	61%	27%	25%	24%
RNN	29%	21%	29%	2%
CNN	5%	24%	18%	12%
Transformer	--	21%	28%	57%
<i>(BERT)</i>	--	--	<i>(28%)</i>	<i>(26%)</i>
<i>(LLM)</i>	--	--	--	<i>(31%)</i>

Feature	TPUv1	TPUv2	TPUv3
Peak TeraFLOPS/ Chip	92 (8b int)	46 (16b) 3 (32b)	123 (16b) 4 (32b)
Network links x Gbits/s/Chip	--	4 x 496	4 x 656
Max chips/supercomputer	--	256	1024
Peak PetaFLOPS/supercomputer	--	11.8	126
Bisection Terabits/supercomputer	--	15.9	42.0
Clock Rate (MHz)	700	700	940
TDP (Watts)/Chip	75	280	450
TDP (Kwatts)/supercomputer	--	124	594
Die Size (mm <sup>2</sup> )	<331	<611	<648
Chip Technology	28nm	>12nm	>12nm
Memory size (on-/off-chip)	28MiB/8GiB	32MiB/16GiB	32MiB/32GiB
Memory GB/s/Chip	34	700	900
MXUs/Core, MXU Size	1 256x256	1 128x128	2 128x128
Cores/Chip	1	2	2
Chips/CPU Host	4	4	8

# SUMMARY

New workloads → new hardware requirements

Domain specific design (understand workloads!)

- No features to improve the average case

- No caches, branch prediction, out-of-order execution etc.

- Simple design with MACs, Unified Buffer gives efficiency

Drawbacks

- No sparse support, training support (TPU v2, v3)

- Vendor specific ?



# DISCUSSION

<https://forms.gle/6t8EE4DPCKvszjHu7>

<i>Type</i>	<i>Batch</i>	<i>99th% Response</i>	<i>Inf/s (IPS)</i>	<i>% Max IPS</i>
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%

How would TPUs impact serving frameworks like Nanoflow? What specific effects could it have on LLM serving architecture

# NEXT STEPS

Next week schedule

Tue: Sinan (ML for Systems)

Thu: Luminix (Inference scheduler)