Hello!

CS 744: TPU

Shivaram Venkataraman

Fall 2021

ADMINISTRIVIA

Midterm 2, Dec 7rd

- Papers from SCOPE to TPU →
- Similar format etc. → questions more specified
- Details on Piazza writing your assumptions etc.

Poster session: Dec 14th

- More details soon

MOTIVATION

Capacity demands on datacenters

New workloads → voice search (Google Assistant, Siri) → lots of new queries

Metrics

- Power/operation
- Performance/operation
- Total cost of ownership → acquiring hardware

Custom ASICs for workloads

Goal: Improve cost-performance by 10x over GPUs

For every byte of data (model) how many arithmetic ops

WORKLOAD → ML inference

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	

smaller

large

mixture of models with low % of CNNs

DNN: RankBrain, LSTM: subset of GNM Translate

CNNs: Inception, DeepMind AlphaGo

WORKLOAD: ML INFERNCE

Quantization → Lower precision, energy use → model weights could be converted to integers

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

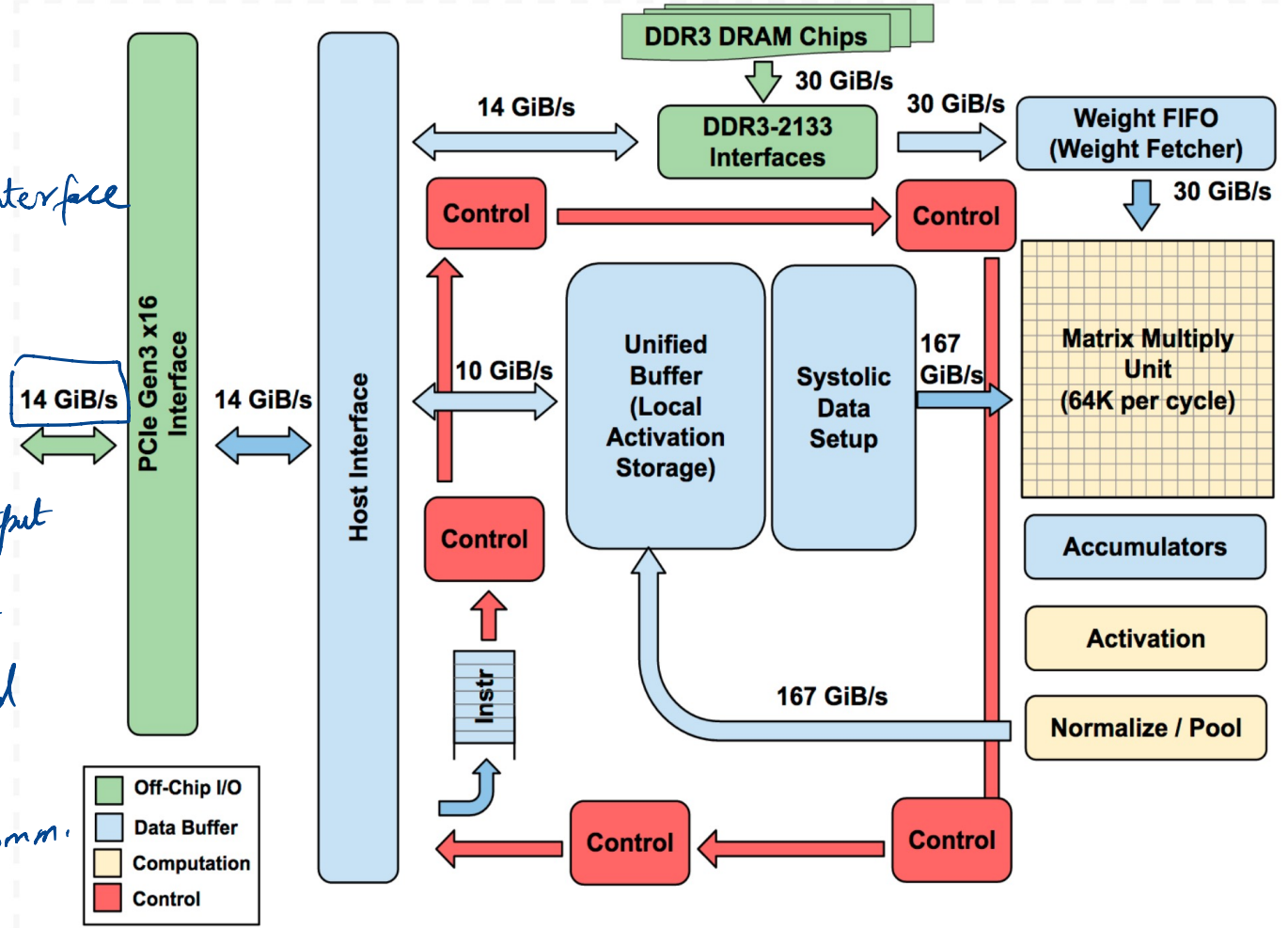
Need for predictable latency and not throughput

e.g., 7ms at 99th percentile

Particularly important for serving
- not just low avg latency
- but we want predictability!

TPU DESIGN CONTROL

- Mounted on PCIe interface
  - compatibility
  - PCIe BW is low and this inputs latency / tput
- Instructions need to be coarse grained to minimize Host to Device Comm.

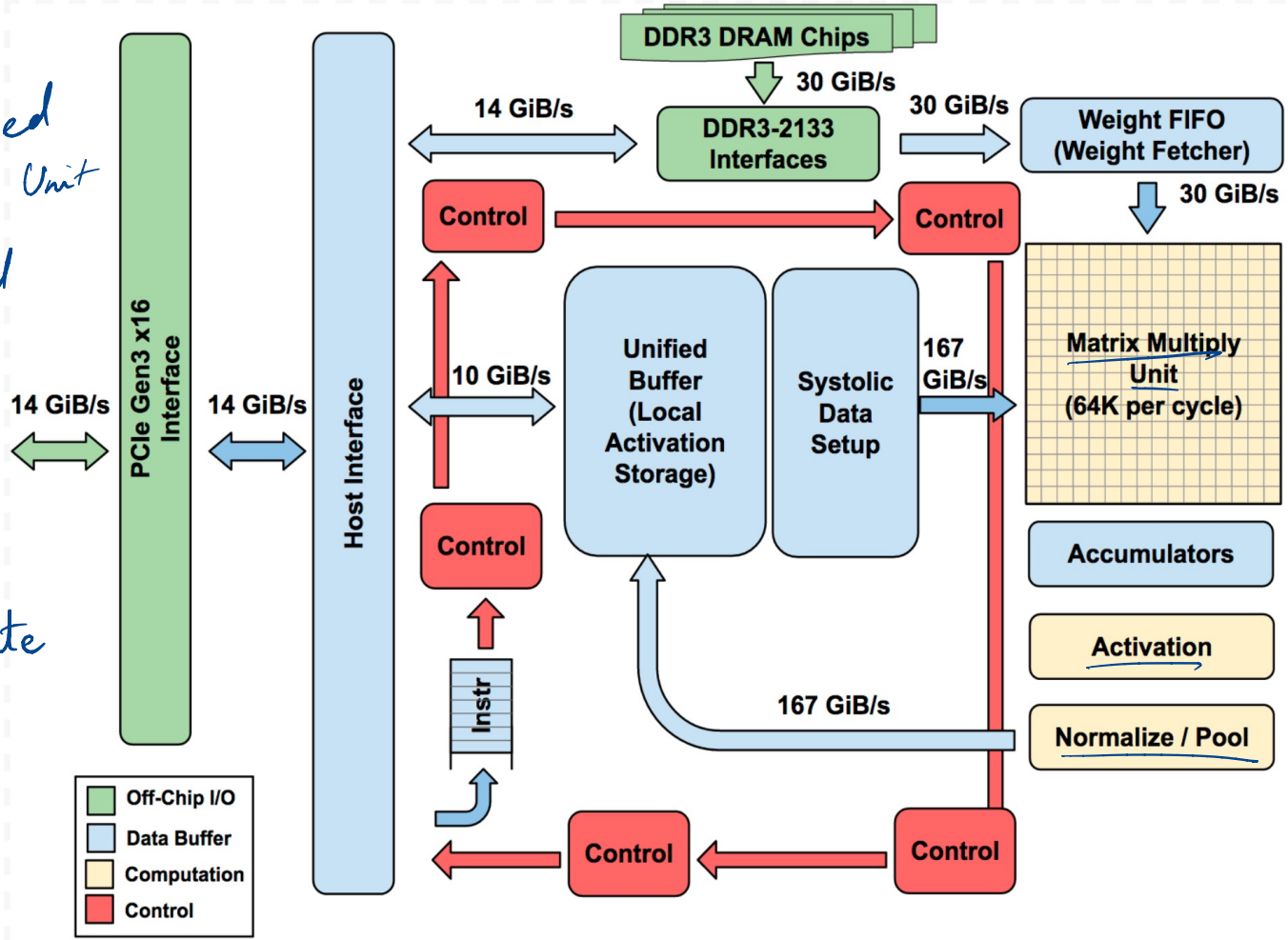


COMPUTE

- Biggest area occupied by Matrix Multiply Unit
  - ↳ Fully connected
  - ↳ Convolutions ...

→ Activation ReLU, tanh etc...

→ Specialized compute Units



DATA → model (data) = output

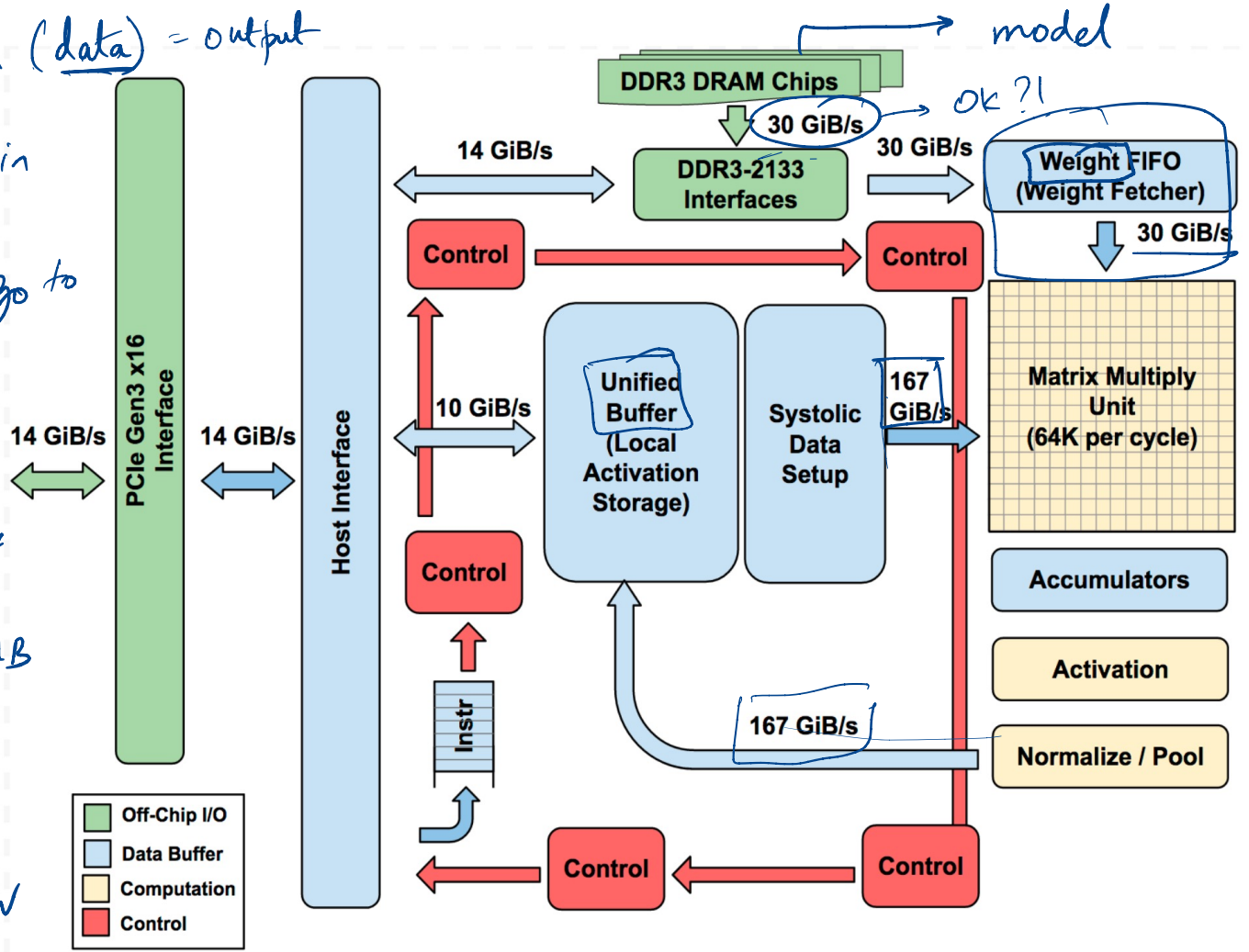
① Models are stored in DDR3

② Input examples go to Unified buffer

Both are inputs to Matrix Multiply

Unified buffer = 24 MB
~ L3 cache size

→ DDR 3 BW is lower than Unified buffer BW



INSTRUCTIONS

CISC format (why ?) → Complex Instruction Set

↳ PCIe bw not that high

↳ Each instruction can take a long time!

1. Read_Host_Memory ← input
2. Read_Weights ← model
3. MatrixMultiply/Convolve
4. Activate
5. Write_Host_Memory

→ maps very closely to the workload

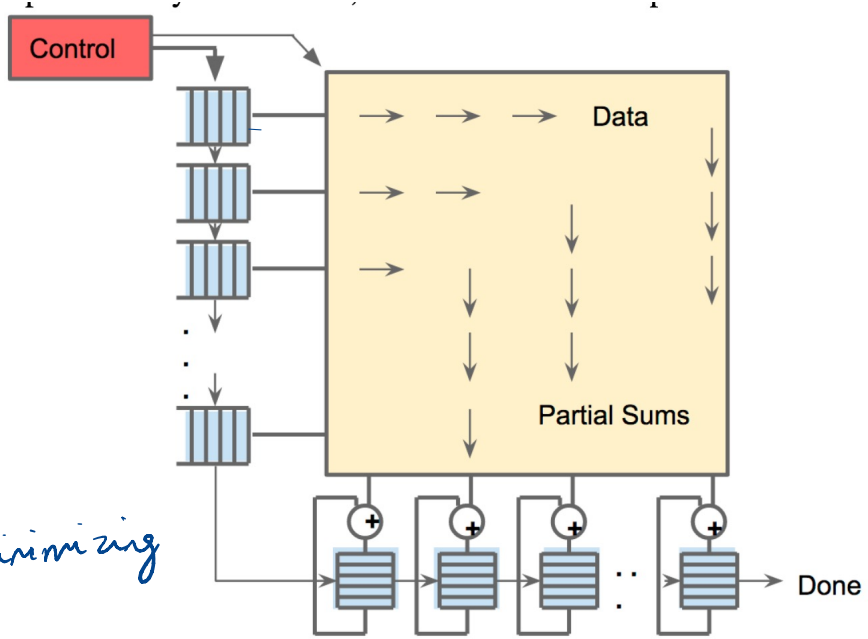
SYSTOLIC EXECUTION

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

Typical implementation
- Read from cache / register
- FPU to do MAC
- Write back to cache / register

Structured propagation of data between compute units

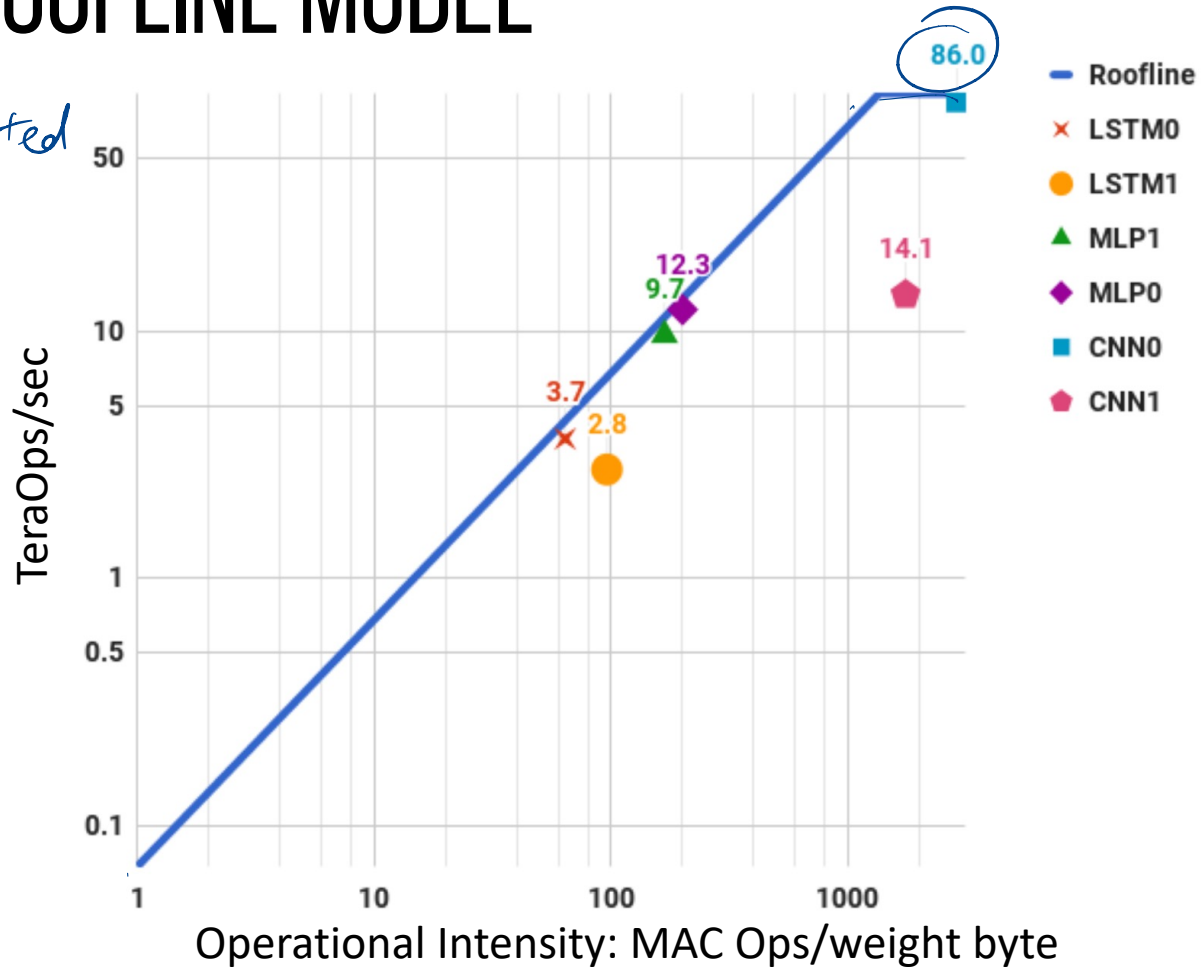
Predictable performance, while minimizing power



ROOFLINE MODEL

Capture the performance that is available / supported given arithmetic intensity

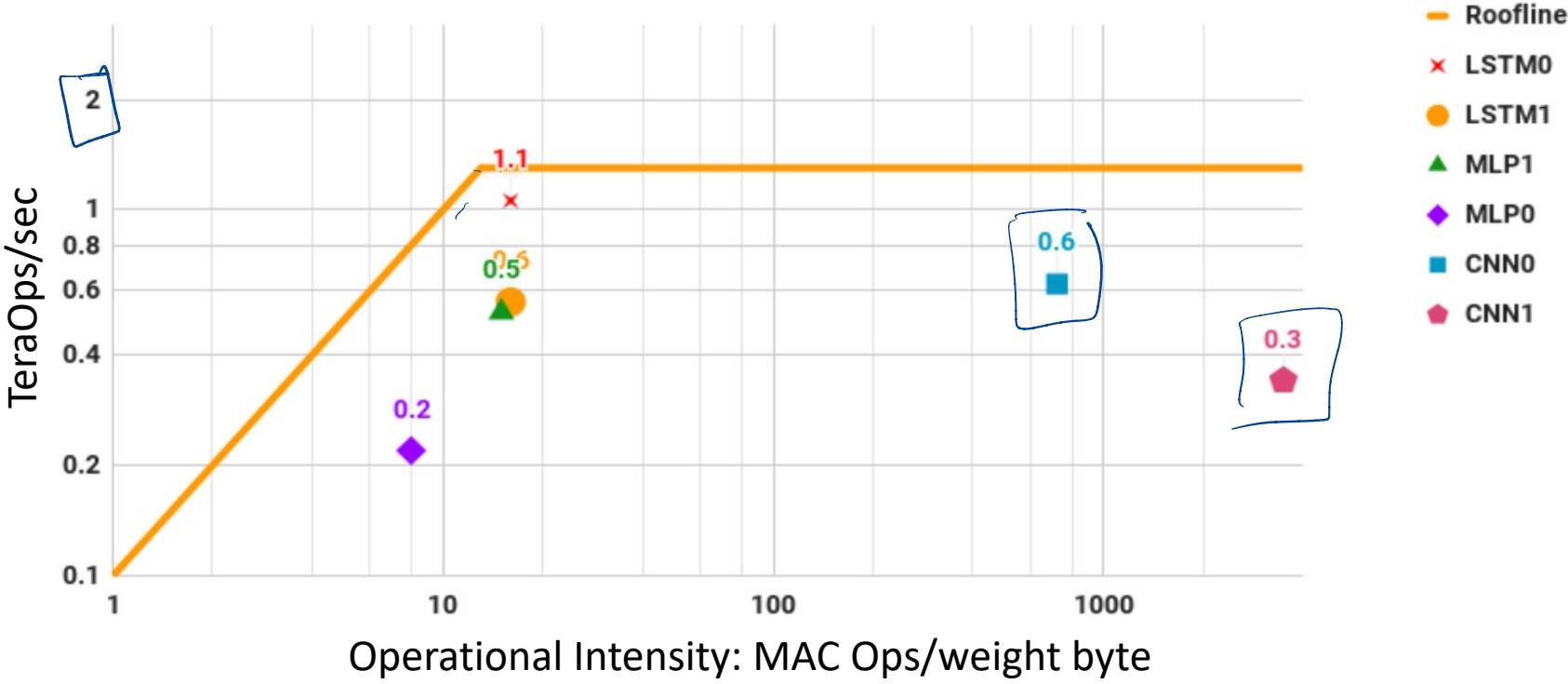
- Slope and a flat line
  - points along the slope ≡ mem access bound
  - flat line ≡ compute bound



State of the art at that time

HASWELL ROOFLINE

86 for TPUs



COMPARISON WITH CPU, GPU

Model	*Die*									
	mm²	*nm*	*MHz*	*TDP*	*Measured*		*TOPS/s*		*GB/s*	*On-Chip Memory*
					Idle	*Busy*	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

SELECTED LESSONS

Tail / Predictable

- Latency more important than throughput for inference
- LSTMs and MLPs are more common than CNNs
- Performance counters are helpful → for benchmarking / debugging
- Remember architecture history

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/LFeaeME4pFdHZdMV6

For all hardware incr batch size

⇒ incr util & tput but at cost of latency

GPU, CPU sacrifice utilization to reach 7ms SLO

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%

Can handle much larger batch size

Able to meet 7ms while ~200K tput

How would TPUs impact serving frameworks like Nexus? What specific effects it could have on distributed serving systems architecture

- Goals

  Nexus: high Acc util

  TPU: pred. latency

  ↳ have predictable latency for a range of input sizes

  could make Nexus profiles more stable

- less chance for stragglers

NEXT STEPS

Next week schedule

- Tue: Midterm 2
- Thu: Last class! (Fairness in ML, Summary)