welcome back!!

CS 744: TPU

Shivaram Venkataraman

Spring 2024

ADMINISTRIVIA

Midterm 2, April 25th

- Papers from SCOPE to HeMem
- Similar format as first midterm
- Details on Piazza

Poster session: May 2nd

- More details soon

MOTIVATION

Power

Capacity demands on datacenters

New workloads → ML workloads

Price → Commodity hardware
↳ Disks CPU
buy from a store

Metrics → Add or Multiply

- Power/operation
- Performance/operation
- Total cost of ownership

→ ML - specific accelerators

Goal: Improve cost-performance by 10x over GPUs ~K40s / K80s

WORKLOAD

→ ML inference workloads

handful of layers of compute kernels

how much compute for a byte of data (model weights)

model size

90% not CNN models

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	

larger → LLMs ~GBs

DNN: RankBrain, LSTM: subset of GNM Translate

CNNs: Inception, DeepMind AlphaGo

WORKLOAD: ML INFERNCE

representation of weights to integers

Quantization → Lower precision, energy use

Goal: Perf/watt

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

InFaaS → SLO satisfaction rate

→ hardware plays a key role

TPU DESIGN CONTROL

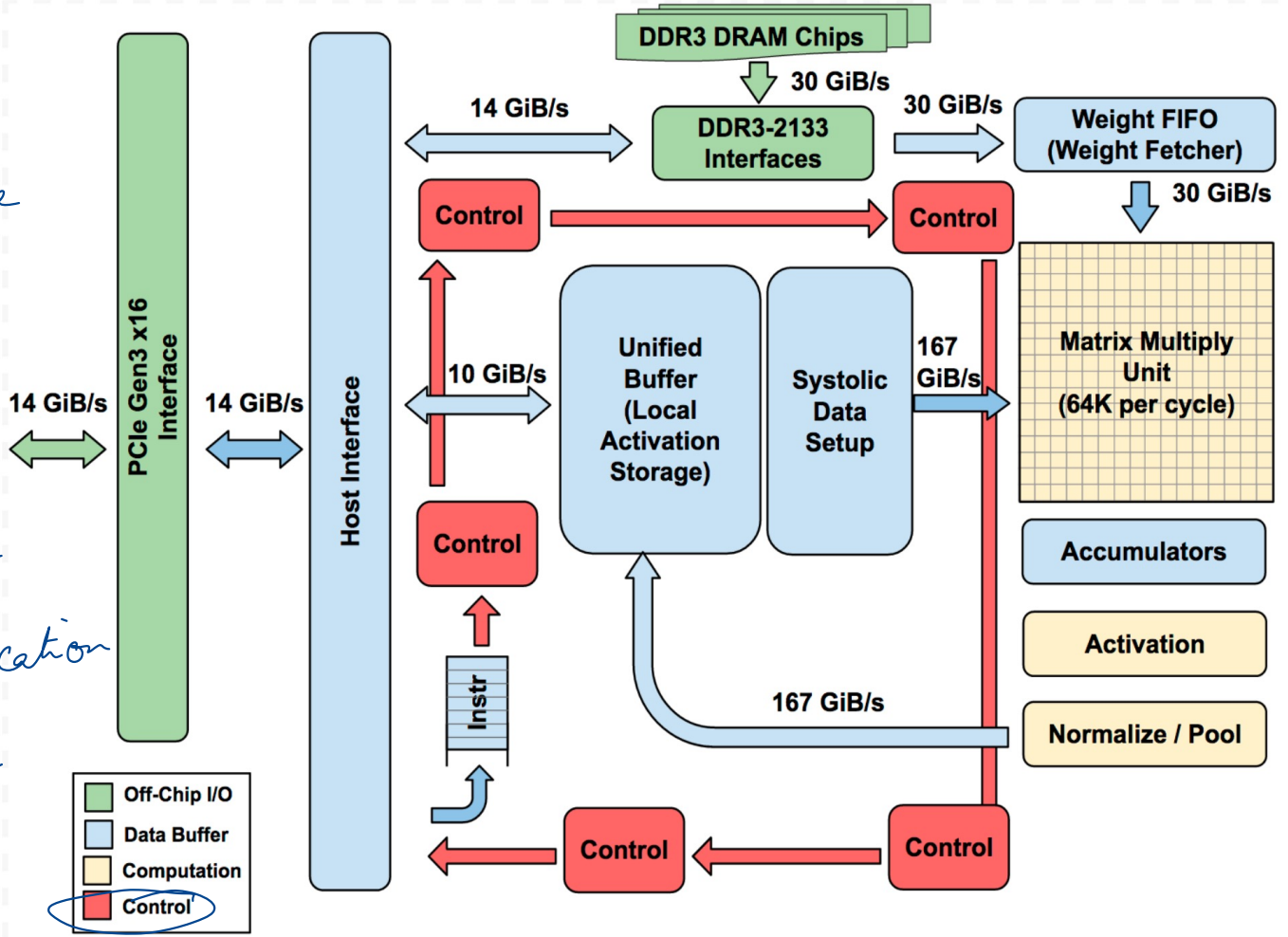
Mounted on a PCI-e

→ Compatibility

Low bandwidth when using PCI-e

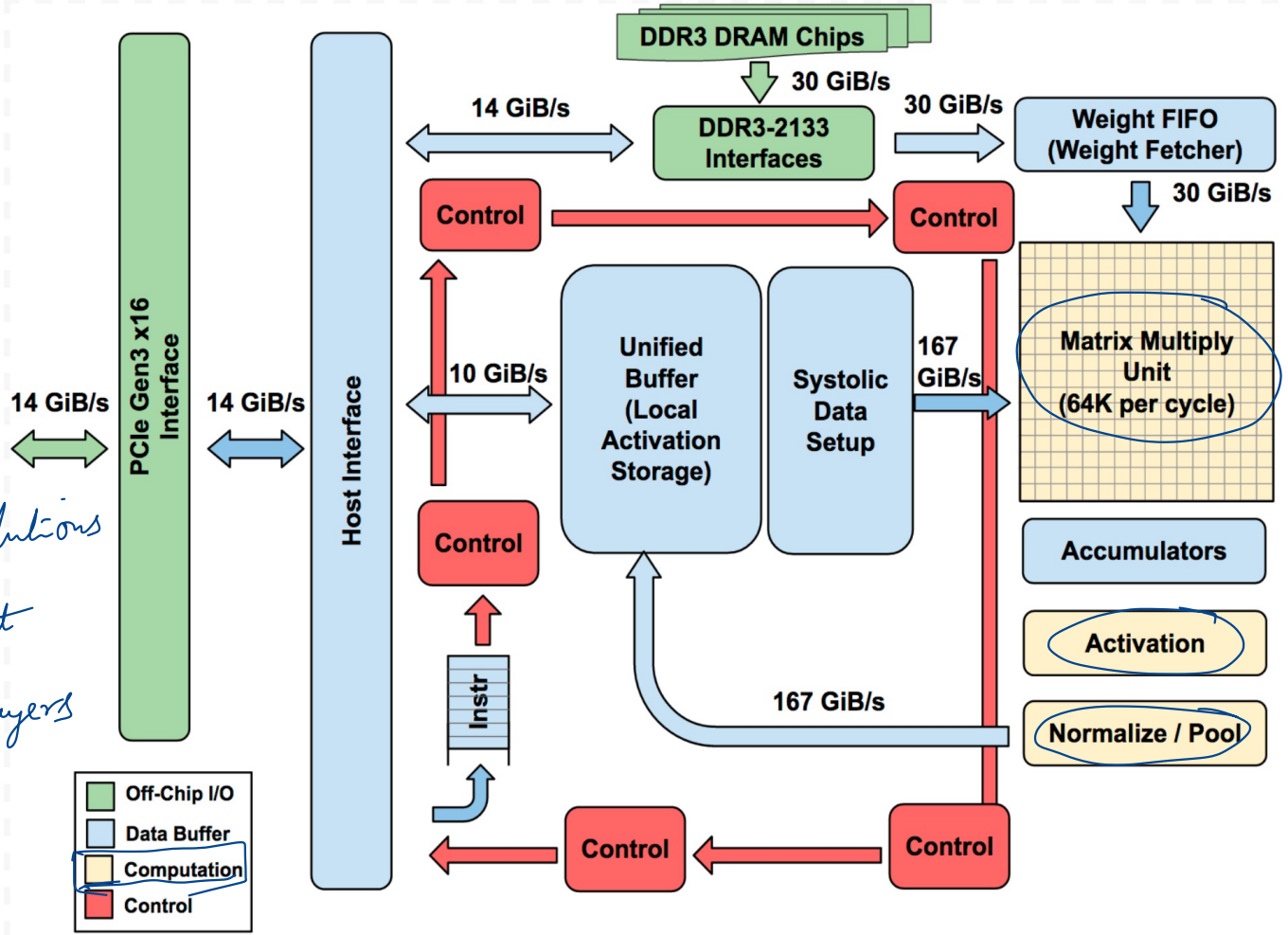
↳ minimize host to device communication

↳ Instructions are coarse grained



COMPUTE

- ML specific
  - → Activations / Pooling etc.
    - ↳ efficient
- → Matrix Multiply
  - MLPs & Convolutions
  - one unit that handles these layers



DATA

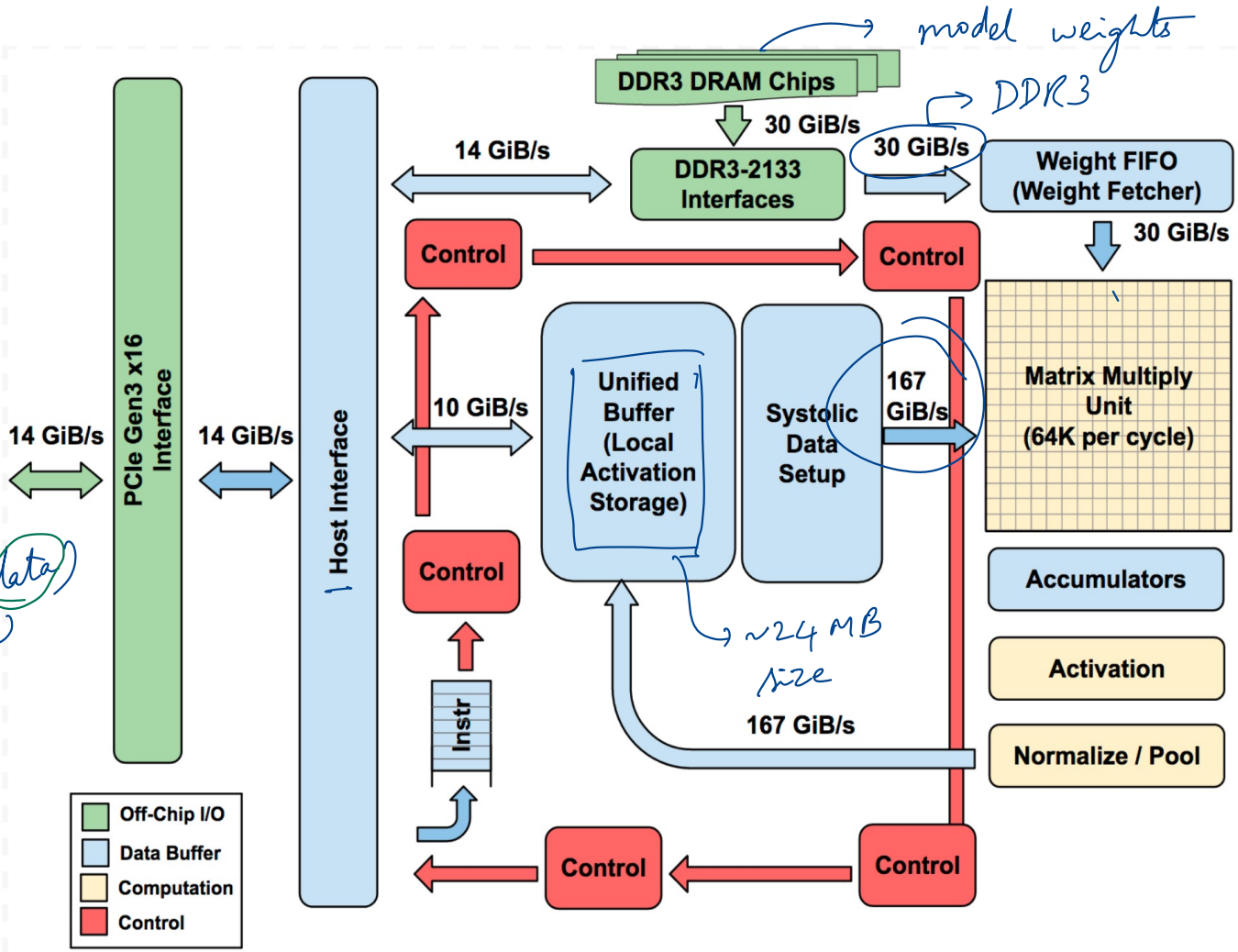
- Two kinds of data

(1) Model weights
→ static. load them ahead of time

(2) model forward (data)

input examples

Activations & layer outputs also go into unified buffer



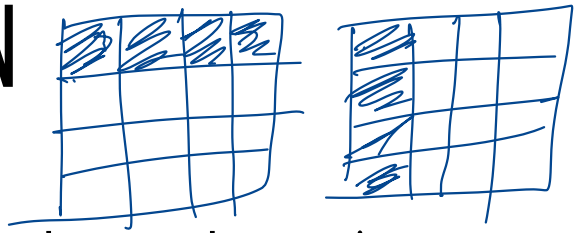
INSTRUCTIONS

CISC format (why ?) → specialized with complex instructions

1. Read_Host_Memory - input
2. Read_Weights
3. MatrixMultiply/Convolve → many ops to do to run this instruction
4. Activate
5. Write_Host_Memory → output

- Minimize host to device
- handful of layers

SYSTOLIC EXECUTION



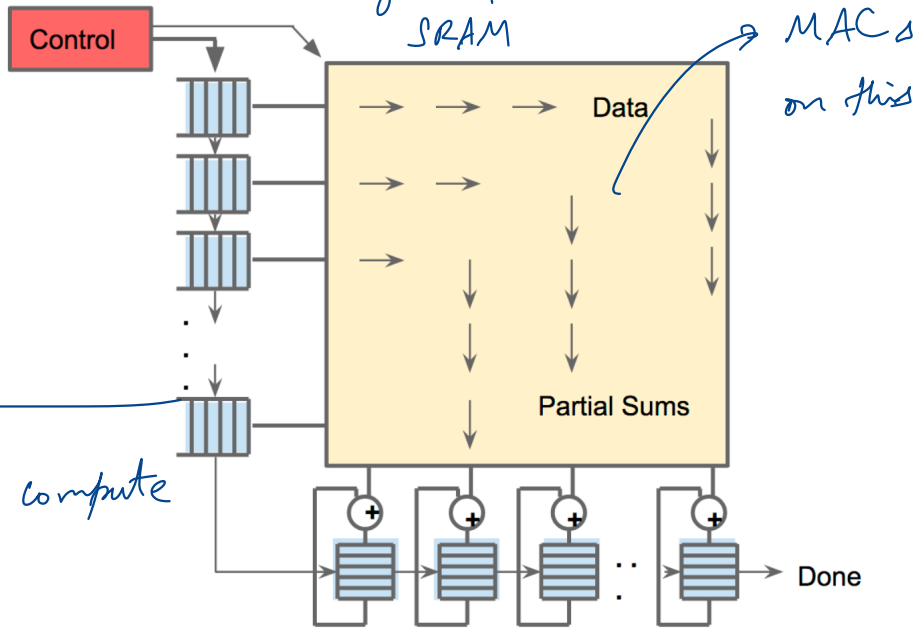
Problem: Reading a large SRAM uses much more power than arithmetic!

c[i] = a[i] * b[j]

read SRAM

multiply

write SRAM



Systolic arrays

- Data streamed through compute units

ROOFLINE MODEL

TPUs

x-axis: Computational intensity

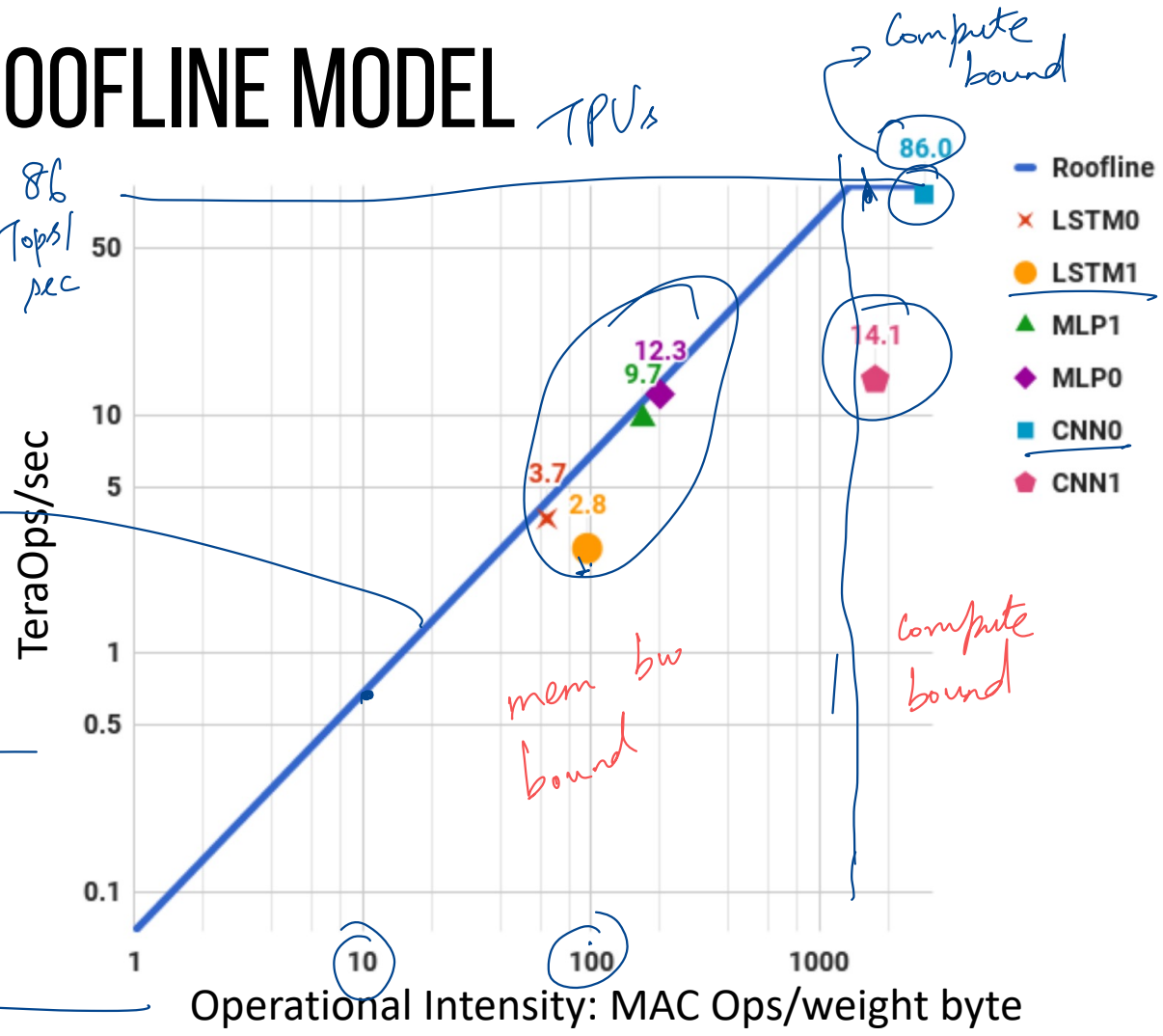
y-axis: tput or FLOPs that you get

Blue line → hardware

↳ peak performance for given operation intensity

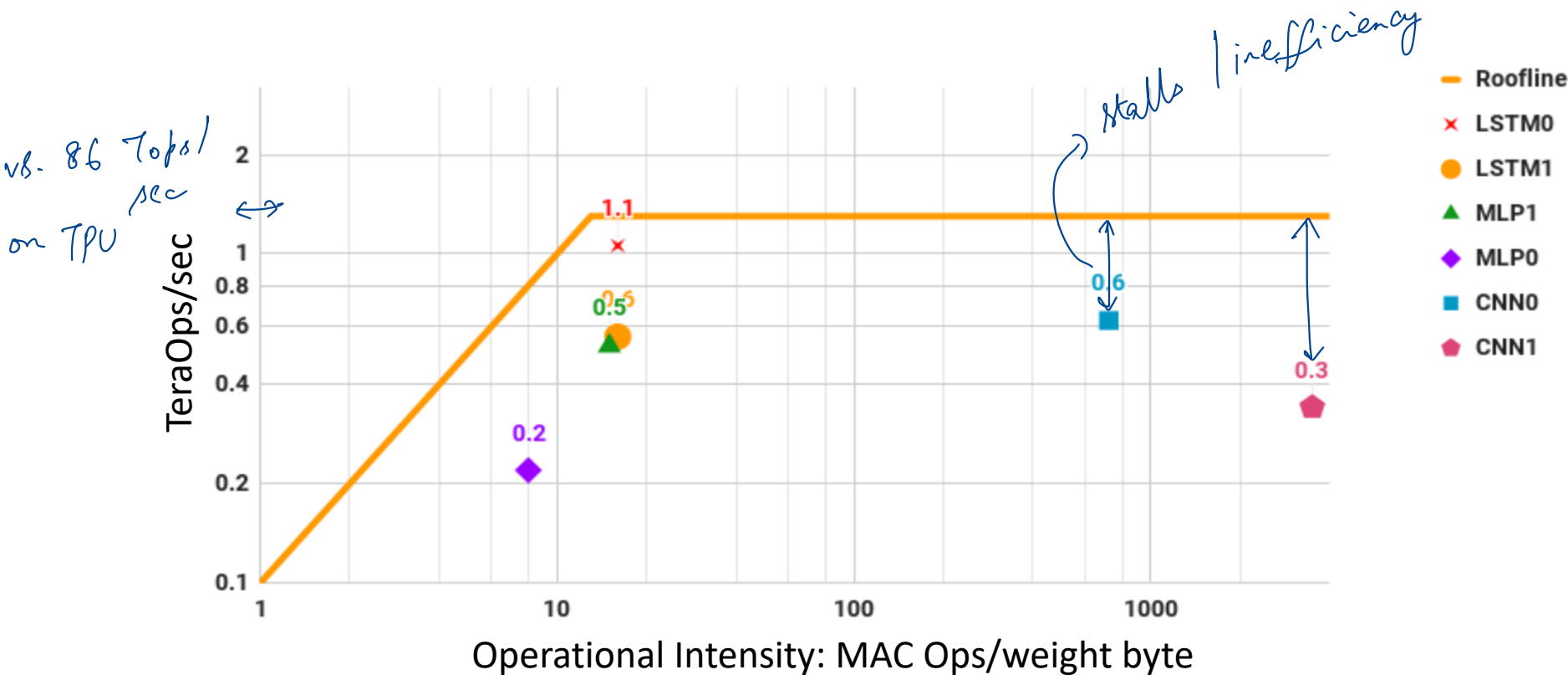
weight fetcher bottleneck

↳ most models mem BW



CPU

HASWELL ROOFLINE



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

50% better than GPU

8-bit integers

low mem bw

unified buffer

SELECTED LESSONS

Predictable

- Latency more important than throughput for inference
- LSTMs and MLPs are more common than CNNs → Surprising in 2016!
- Performance counters are helpful
- Remember architecture history → Systolic arrays

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/P7mZsfK44PemjkXa7

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%

7ms SLO

able to support larger batch size

slightly higher than GPU latency at 100%

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

As you add more hardware types
↳ need more profiling data

TPUs → quantization
↳ maintain more model variants

Sharing TPU might be tricky
coarse grained comp. units

- Additional need for Software maps from PyTorch ↳ TPU

NEXT STEPS

Next week schedule

- Tue: HeMem
- Thu: Midterm 2