CS 744: TPU

Shivaram Venkataraman

Fall 2019

ADMINISTRIVIA

Review class - Thursday → LAST CLASS

Midterm 2, Dec 10th

- Papers from Dataflow Model to TPU
- Similar format, cheat sheet etc.

Poster session Dec 13th → 3pm to 5pm

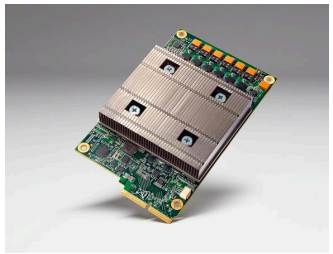
- Template
- Printing instructions
- Reimbursement

Dec 17th - final reports

Multi core
large main memory
→ weld

Serverless Computing

↳ PyWren



Compute Accelerators

Infiniband Networks

Non-Volatile Memory

MOTIVATION

Capacity demands on datacenters

New workloads

Metrics

Total cost of ownership (Depends on price ?)

- Power/operation
- Performance/operation

Goal: Improve cost-performance by 10x over GPUs

WORKLOAD

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	

DNN: RankBrain, LSTM: subset of GNM Translate

CNNs: Inception, DeepMind AlphaGo

WORKLOAD: ML INFERNCE

Quantization → Lower precision, energy use

Quantization
32-bit or
16-bit float → 8-bit integer

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

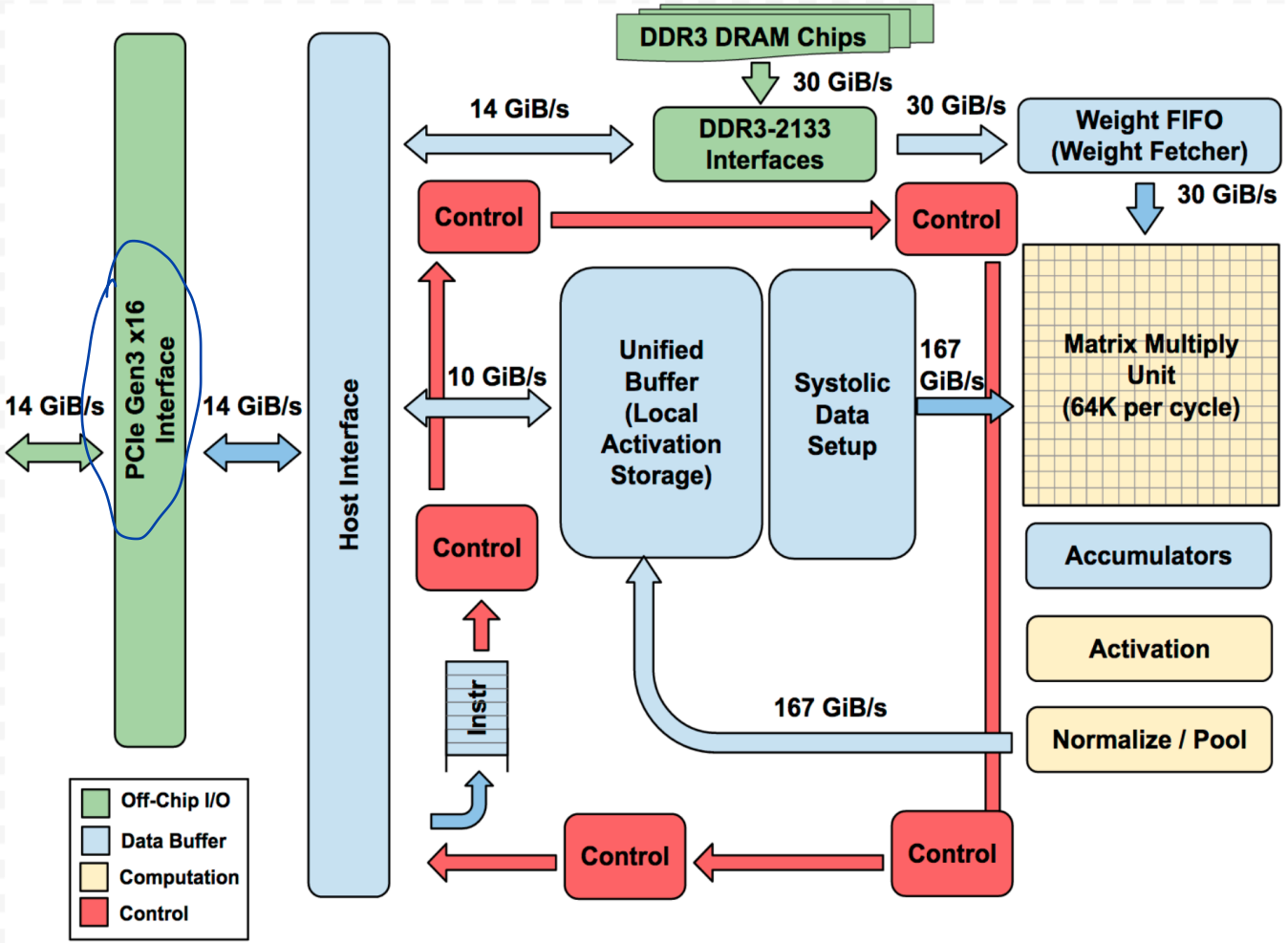
CONTROL

Compatibility
→ PCIe

"Instructions"

CPU → TPU
PCIe

Easier deployment



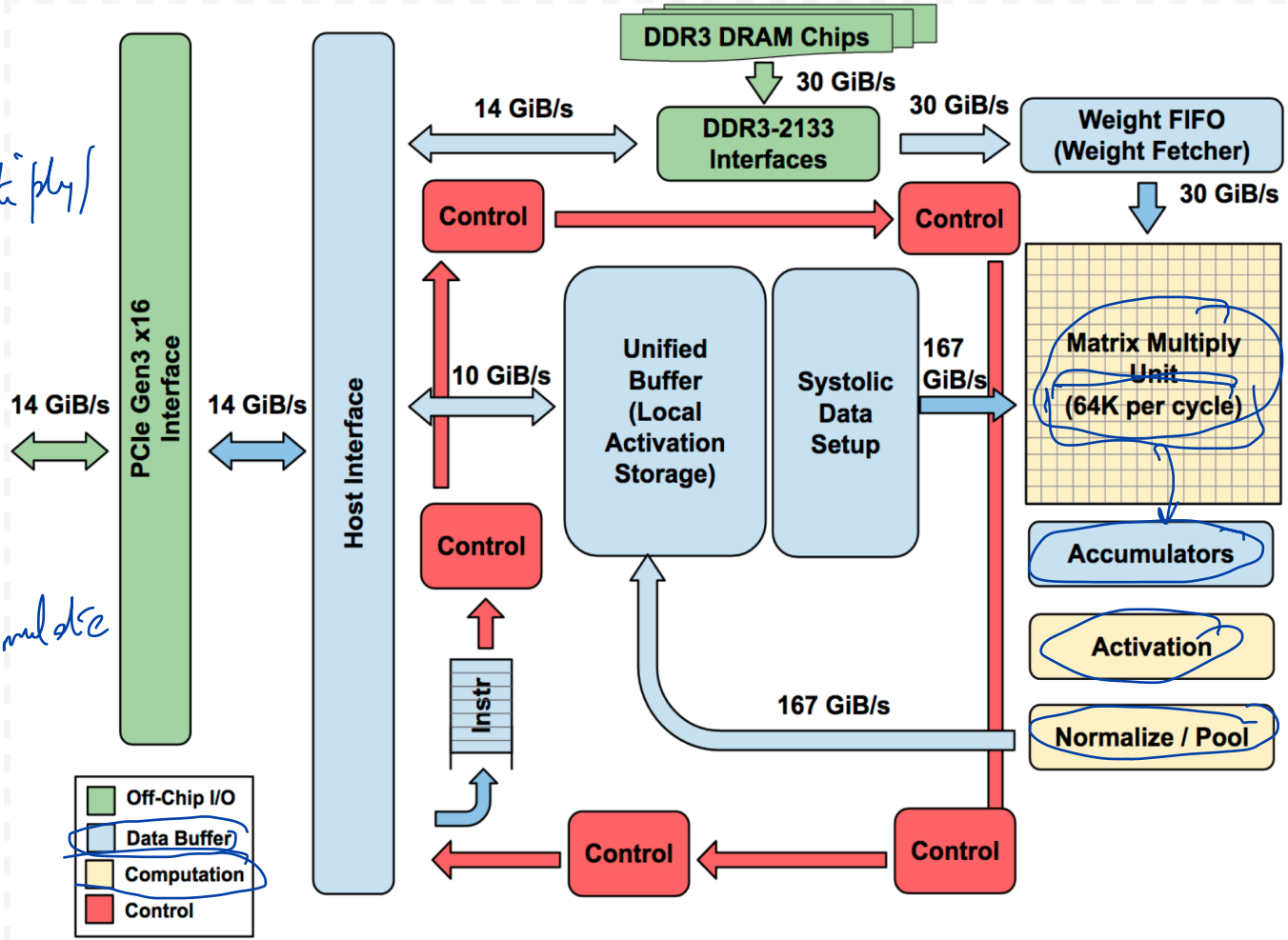
COMPUTE

Specialized
→ Matrix Multiply
Conv
→ Activation

64K per cycle?

MACs
↳ Multiply Accumulate

25% of the area
8 bit / 16-bit



DATA = Example Model weights

→ Weights & Examples

8GB mem to "cache" weights in TPU DRAM

Pipeline weight reads

Weight FIFO

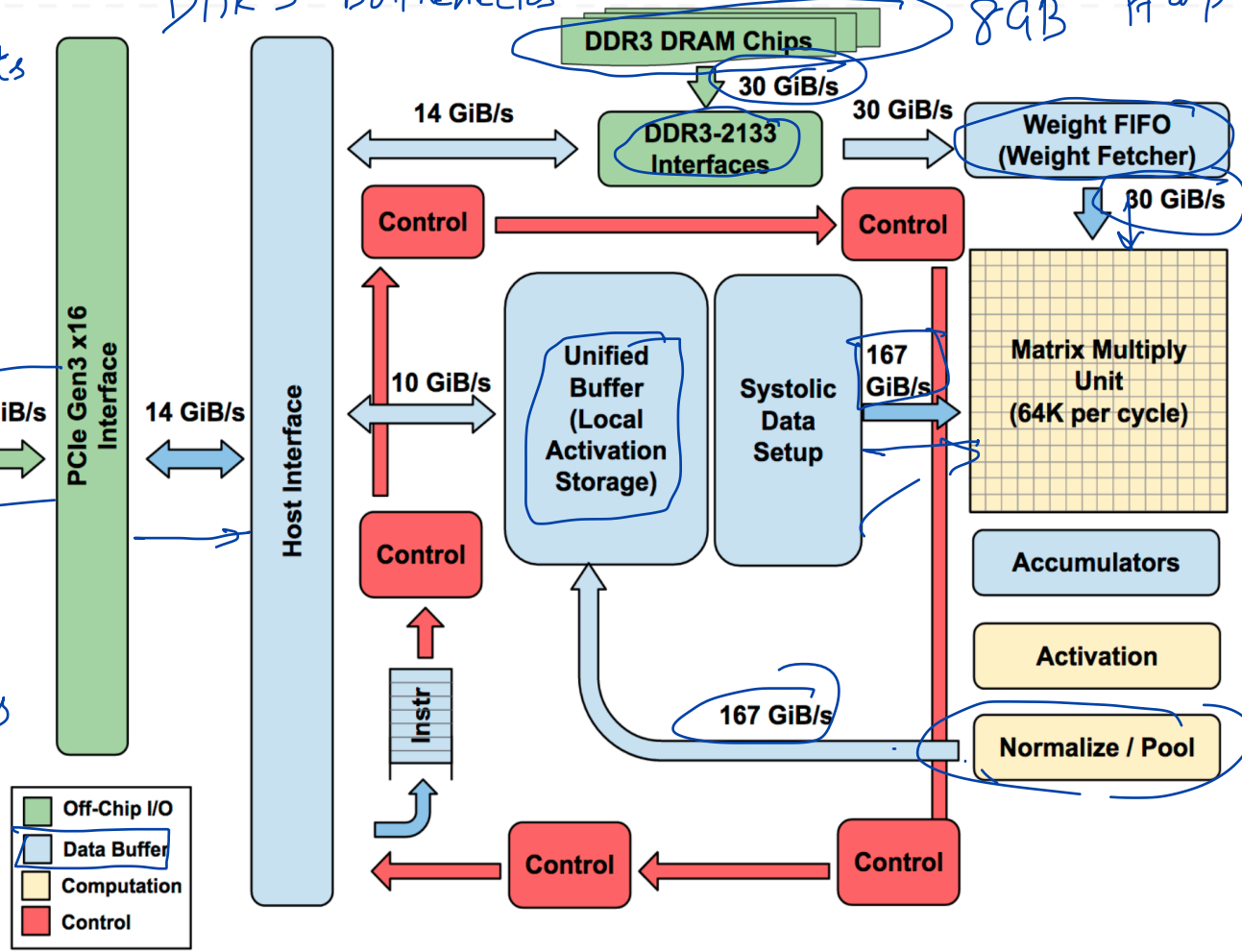
Unified Buffer → Examples → output from prev layers

29% area

24 MB

DDR3 bottlenecks

8GB off chip



INSTRUCTIONS

CISC format (why ?)

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

→ Transfer examples → x86 instruction set
→ CISC instruction is pretty complex
LD, ST, ADD, MUL
Inst → lot of work
↓
next Inst
→ PCIe
→ specialized

SYSTOLIC EXECUTION

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

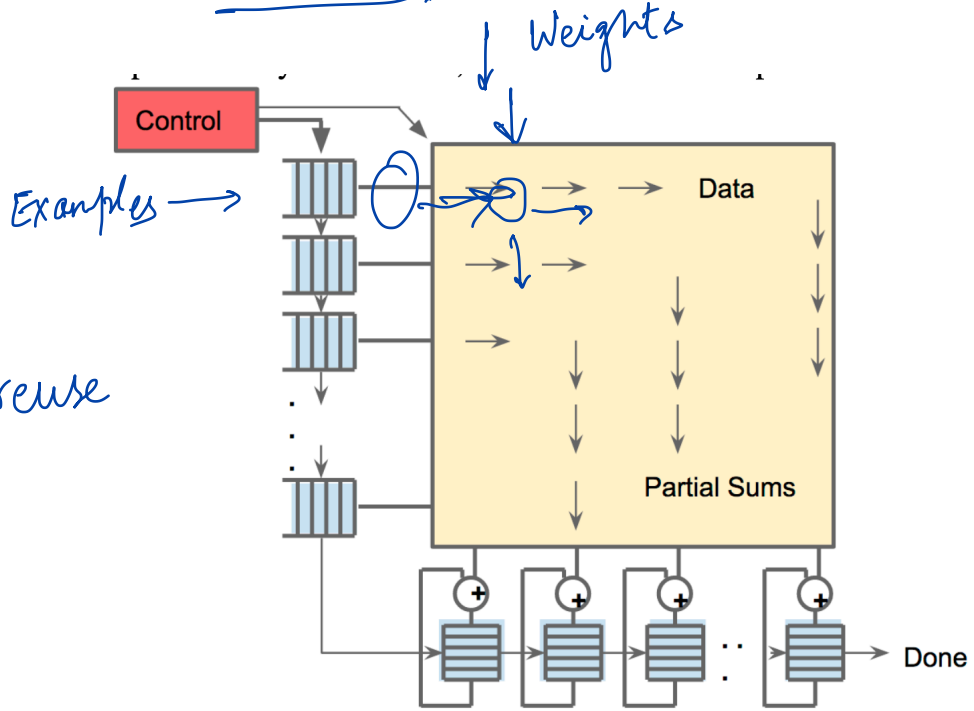
Systolic arrays

↓

more structured

execution and data reuse

helps reduce power



ROOFLINE MODEL

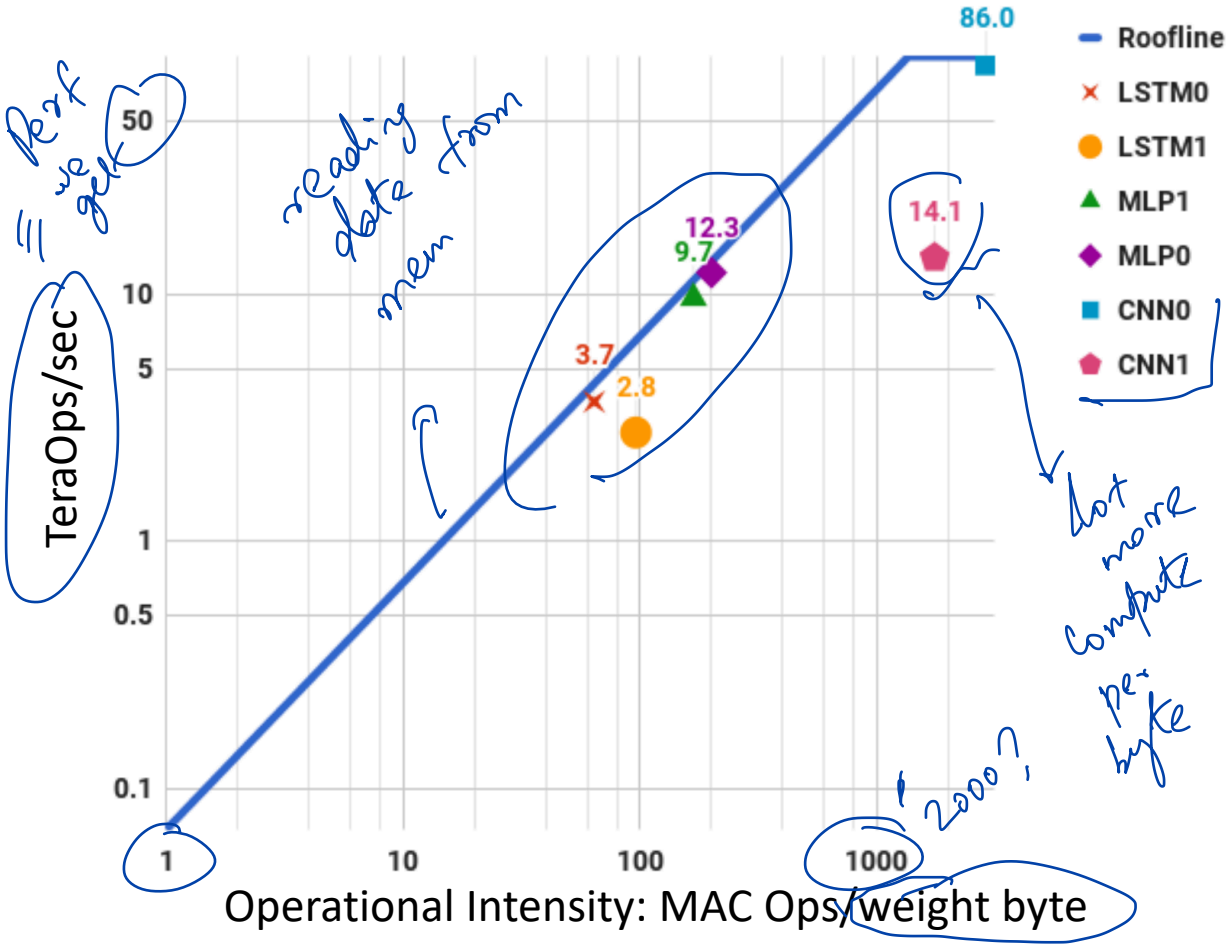
slow — stalling

fast — slow

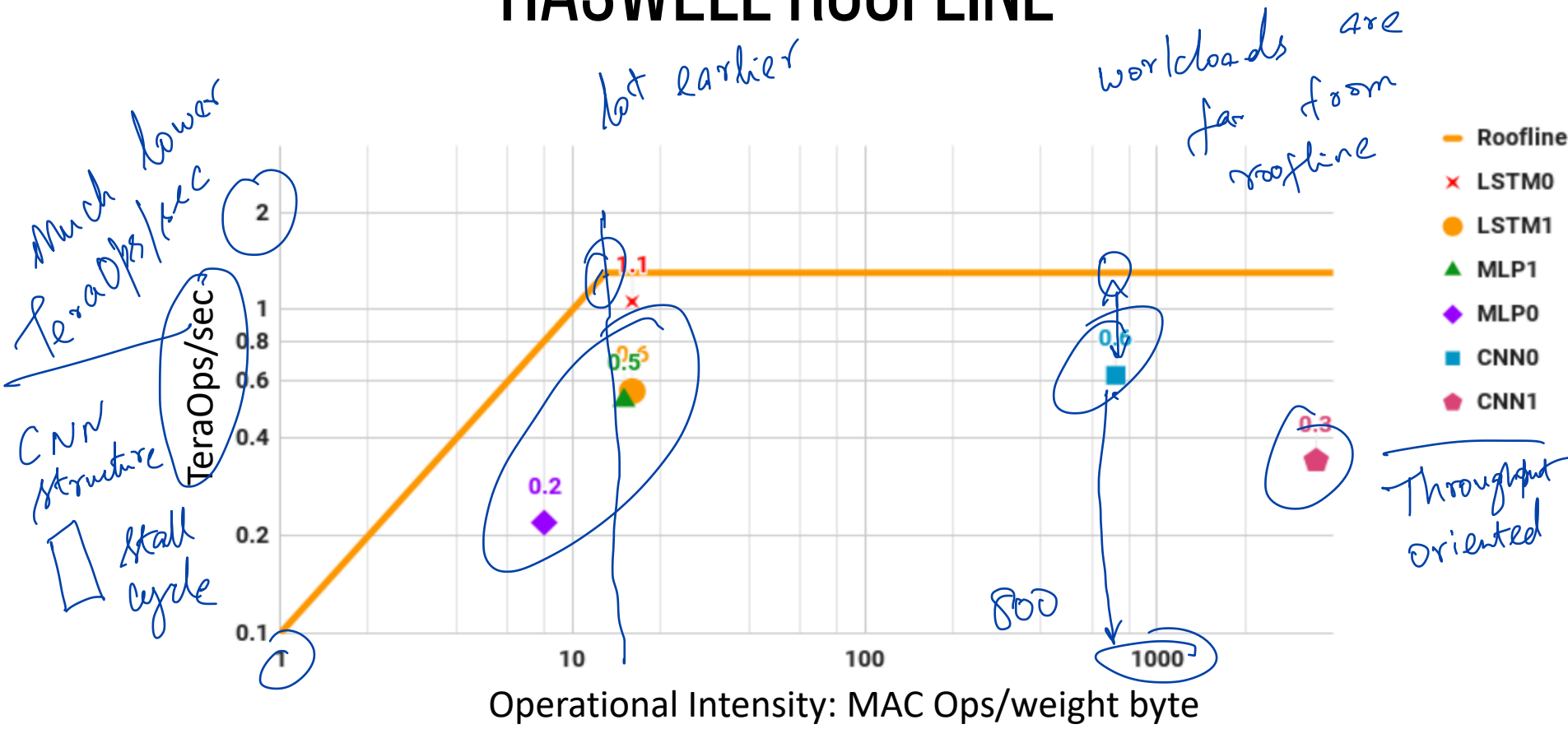
1 byte from memory

how many Ops do you do with it?

Many workloads are close to roofline



HASWELL ROOFLINE

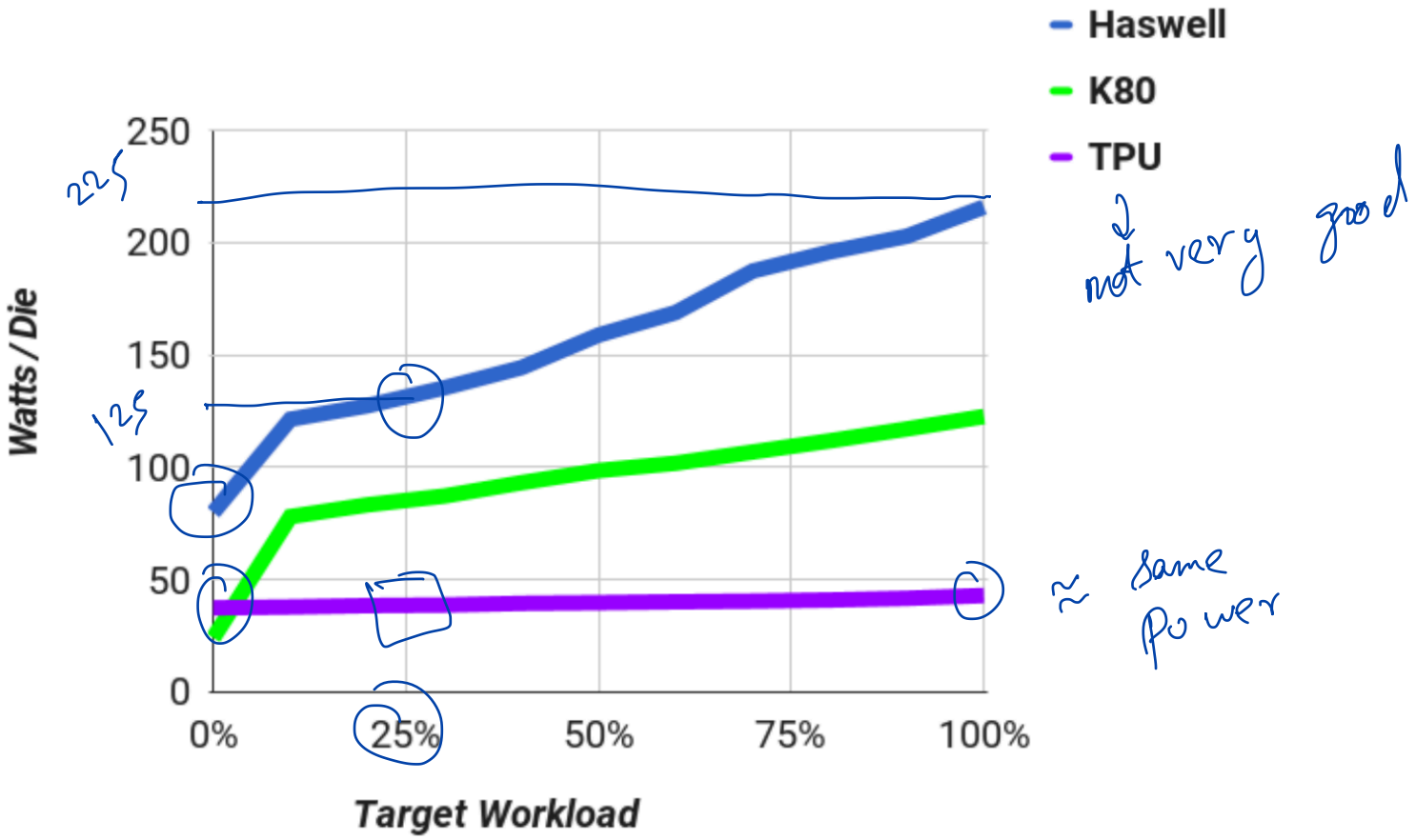


COMPARISON WITH CPU, GPU

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

2x Watts 7 30x

ENERGY PROPORTIONALITY



SELECTED LESSONS

- Latency more important than throughput for inference
- LSTMs and MLPs are more common than CNNs
- Performance counters are helpful
- 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/zhH9eCbdjMnaRLRB8

① Larger batch size → Increase % Max IPS

② TPU can have much larger batch

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%

Tput friendly

③ CPU batch size increase results in 2x higher latency

GPU & TPU have lower latency increases

TPU batch → IPS is linear

How would TPUs impact serving frameworks like Clipper? Discuss what specific effects it could have on distributed serving systems architecture

1. Arrival pattern is bursty
   → TPU is not energy proportion
2. Heterogeneity increases with TPUs
   ↳ CPU models could become stragglers
3. Same goal = Low Tail latency → Could make it easier
4. You can send upto 200 examples

5. Quantization vs. Accuracy
   Large memory