CS 744: BIG DATA SYSTEMS

Shivaram Venkataraman

Fall 2018

ADMINISTRIVIA

- Final class on Dec 6th , No class on Dec 11th
- Poster session Dec 13th, details on Piazza
- Course Project final report
- Course feedback form

ACCELERATORS

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: 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	

DNN: RankBrain, LSTM: subset of GNM Translate
CNNs: Inception, DeepMind AlphaGo

What are some notable points from this table ?

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

OUTLINE

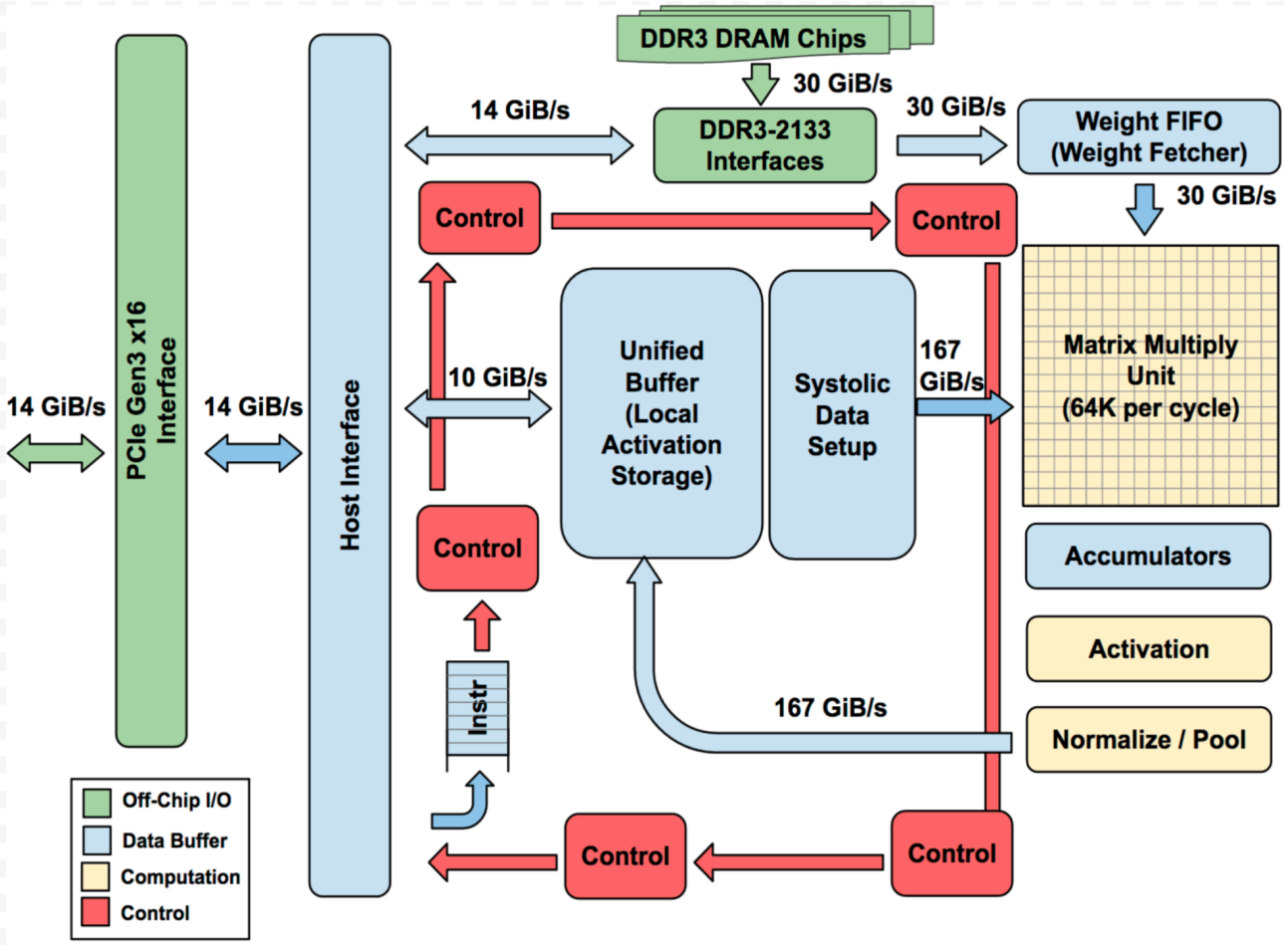
Motivation

TPU Design

Comparison with CPU, GPU

Selected Lessons ?

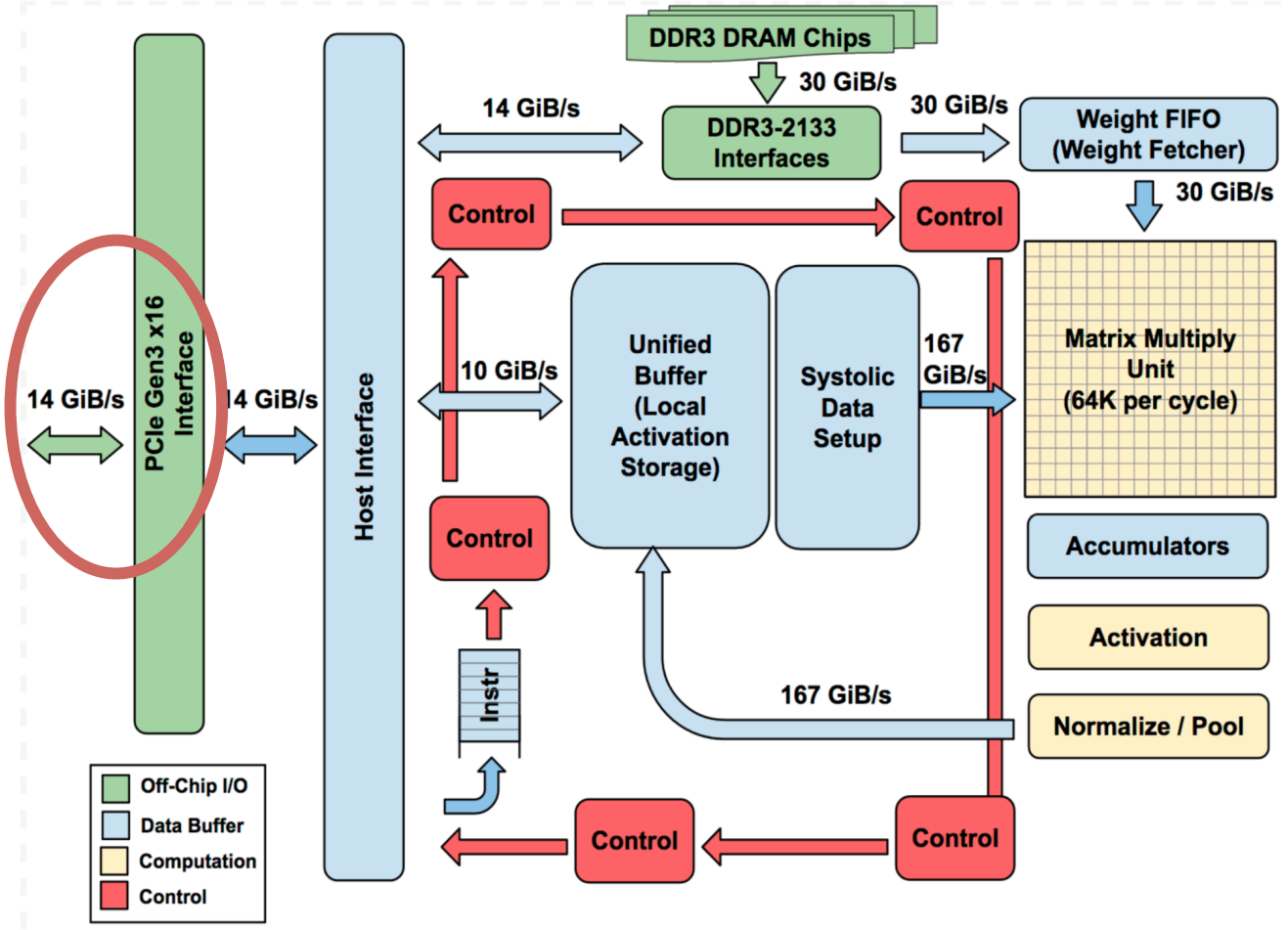
TPU DESIGN


DDR3 DRAM Chips
30 GiB/s
14 GiB/s
DDR3-2133 Interfaces
30 GiB/s
Weight FIFO (Weight Fetcher)
30 GiB/s
Control
Control
14 GiB/s
PCIe Gen3 x16 Interface
14 GiB/s
Host Interface
10 GiB/s
Unified Buffer (Local Activation Storage)
Systolic Data Setup
167 GiB/s
Matrix Multiply Unit (64K per cycle)
Control
Accumulators
Activation
Instr
167 GiB/s
Normalize / Pool
Control
Control
Off-Chip I/O
Data Buffer
Computation
Control

CONTROL

Instructions sent over PCIe

Easy deployment



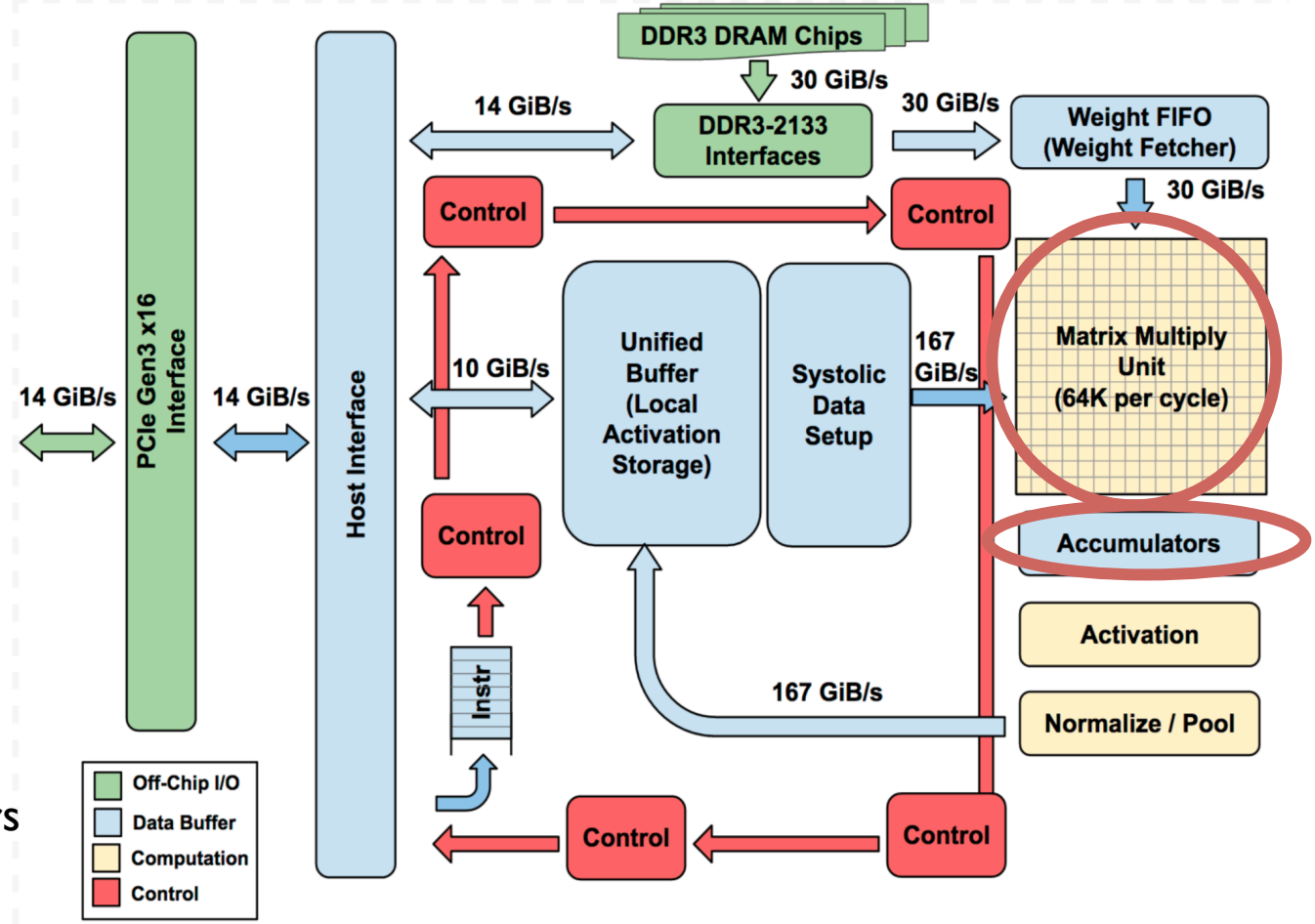
COMPUTE

Matrix Multiply Unit
24% of the area!

256x256 MACs
8-bit multiply-and-adds

RW 256 values/cycle

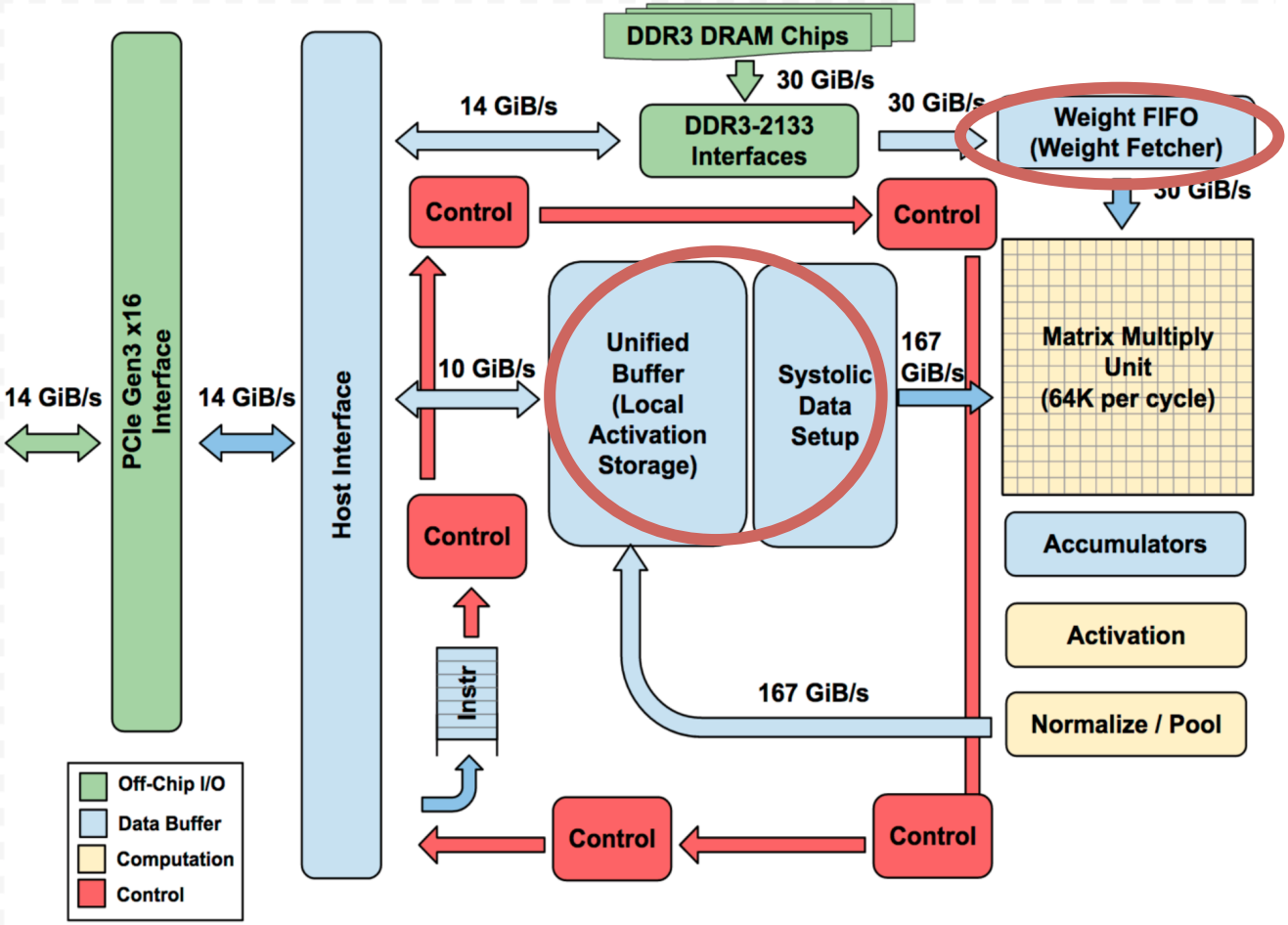
16-bit products are collected in Accumulators



DATA

Read weights from off-chip
8GB off-chip memory
Pipeline reads, compute

Unified Buffer: 24MB
29% of the chip!
Store intermediate results



INSTRUCTIONS

CISC format (why ?)

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

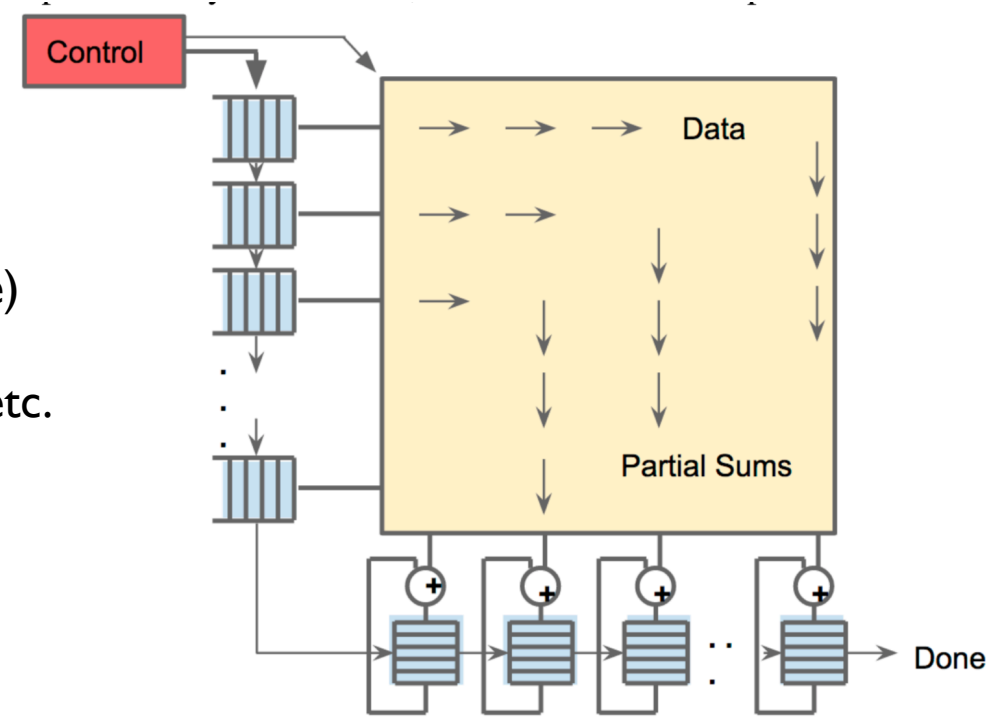
SYSTOLIC EXECUTION

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

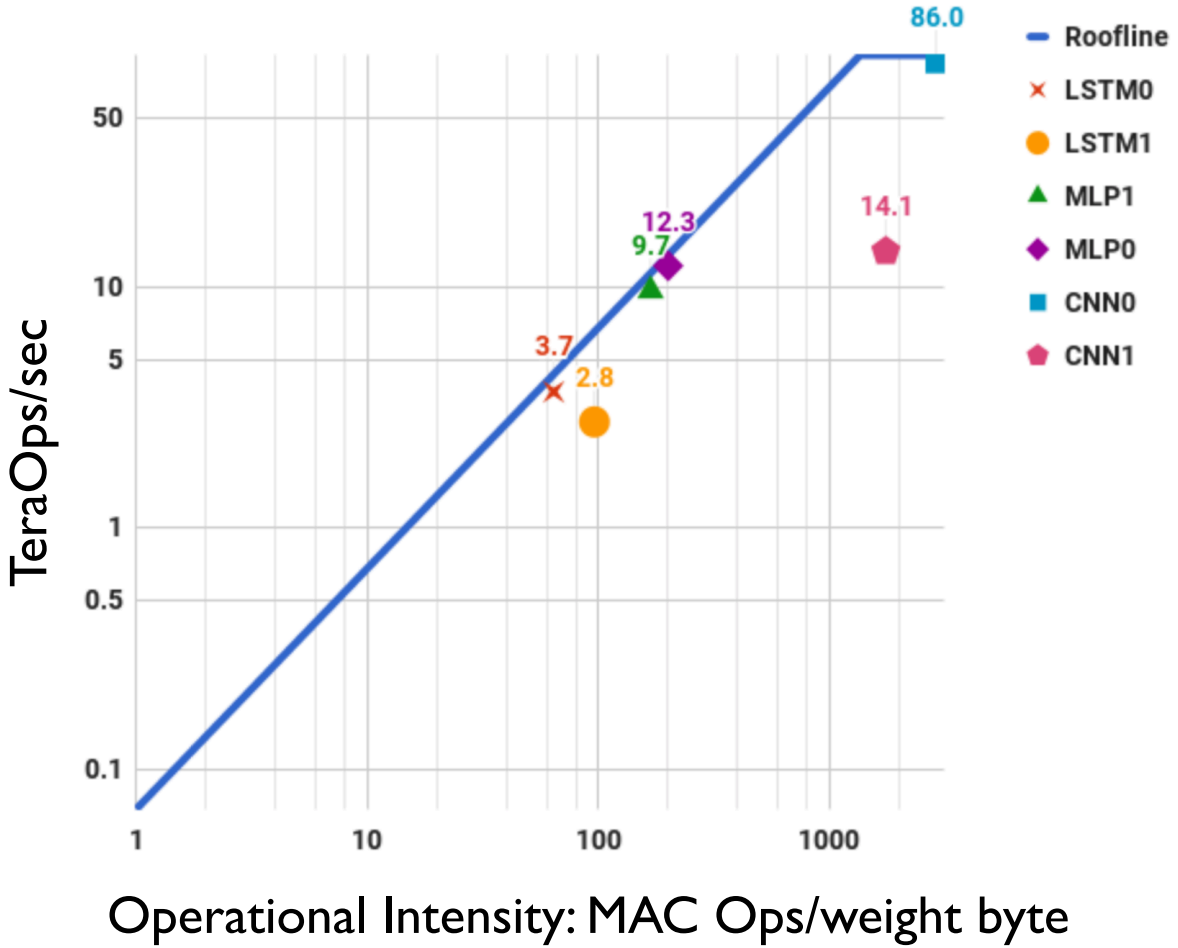
Wave-like propagation of data

Not Von Neumann (fixed compute)

No memory access / cache / bus etc.



PERFORMANCE: ROOFLINE MODEL



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

COMPARISON WITH GPU, CPU

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%

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 ?