Hello!

# CS 744: TPU

Shivaram Venkataraman

Fall 2022

# ADMINISTRIVIA

Midterm 2, Dec 6th → Tuesday

- Papers from SCOPE to TPU
- Similar format as first midterm → handwritten / printed notes
- Details on Piazza

Poster session: Dec 13th → Template

- More details soon

Print poster

venue etc.

# MOTIVATION

Capacity demands on datacenters

New workloads → voice assistants → query volume to search
↳ new hardware

Metrics

- Power/operation → Energy is expensive → minimize power consumed
- Performance/operation → latency, +throughput
- Total cost of ownership
  ↳ TCO

Goal: Improve cost-performance by 10x over GPUs

# WORKLOAD → ML Inference

- Workload mix evolves over time → new architectures
- Batch sizes vary across 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 | |

pretty small models

MLPs + LSTM dominate

lot of ops / weight byte

$= \frac{\text{FLOPs}}{\text{weight bytes}}$

CNN batch sizes are small

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

# WORKLOAD: ML INFERNCE

not so much for training

Quantization → Lower precision, energy use

quantization works well for inference,

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

as compared to 32-bit floating point numbers

Need for predictable latency and not throughput

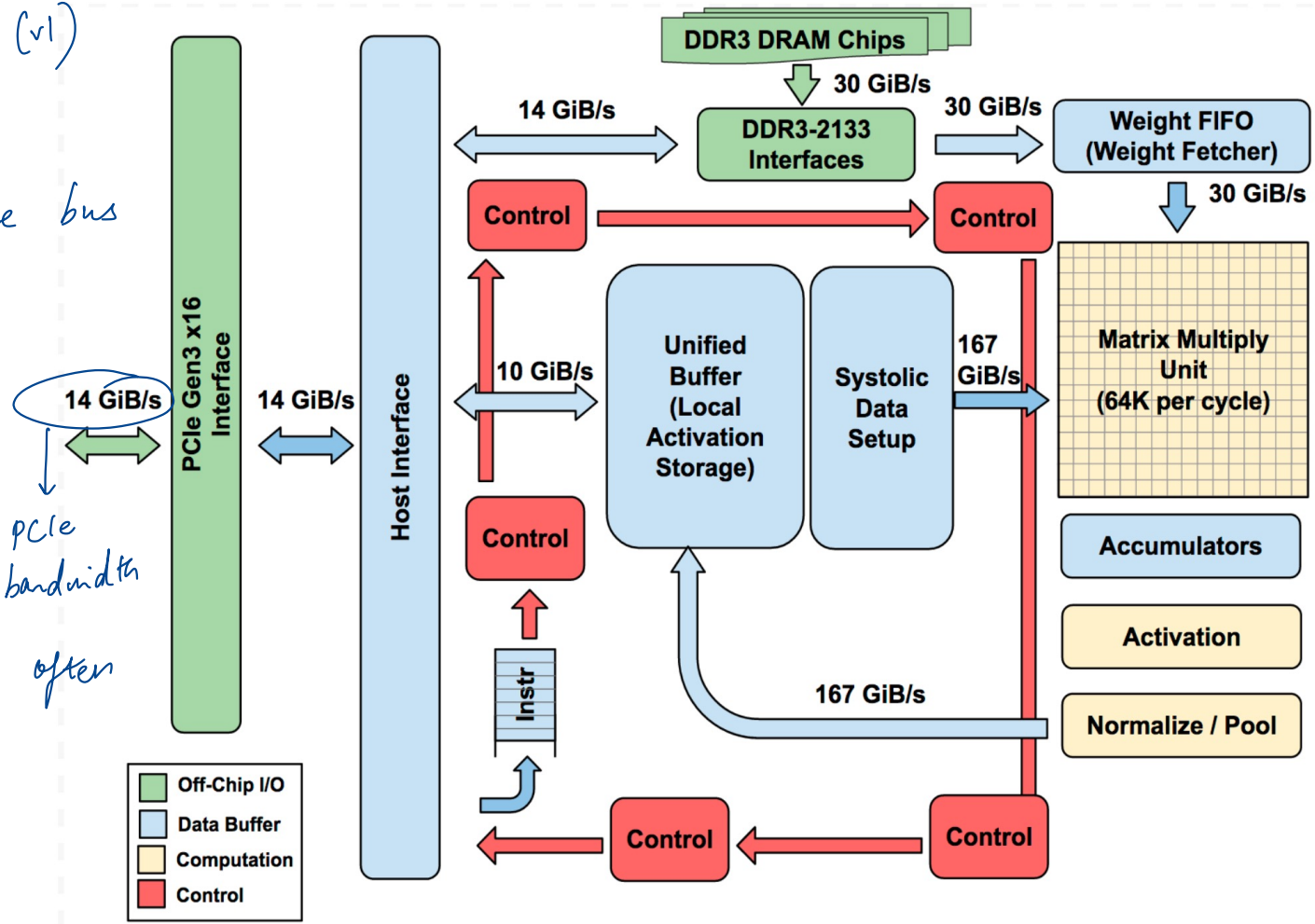
e.g., 7ms at 99th percentile

SLO that needs to be met

# TPU DESIGN CONTROL

TPU (v1)

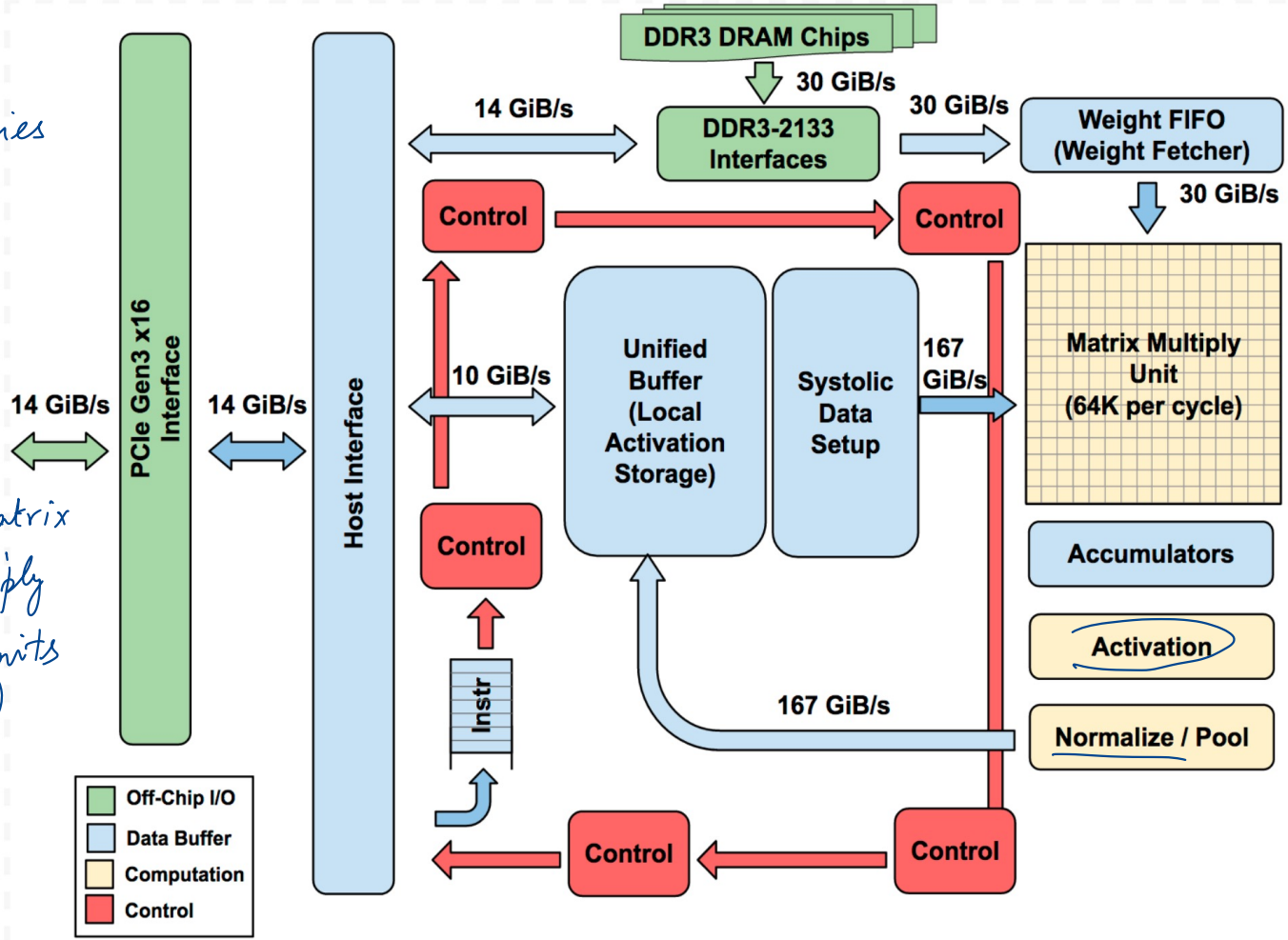
- Mounted on a PCIe bus
  - Compatibility
  - Bandwidth is low
  - You don't want to communicate very often
- Instructions are coarse grained



# COMPUTE

- Matrix Multiply occupies most area in this chip.
- ML model layers
  - fully connected,
  - Convolutions → Matrix multiply
- Activation compute units (Relu, tan-h etc.)

→ Specialized!



# DATA

Inference:
- model, model weights
- data, examples

model.forward(data)

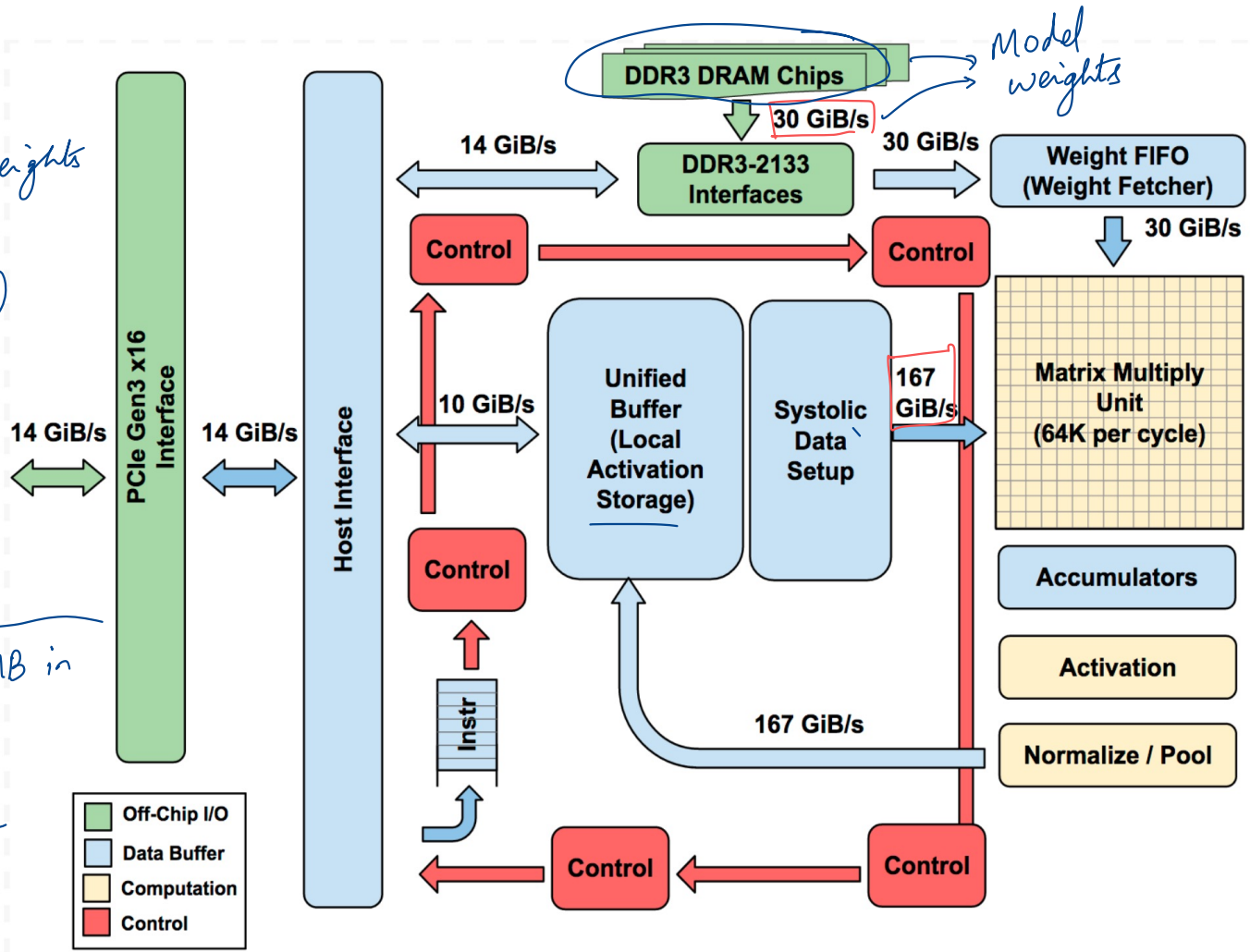
---

examples come from host, put in Unified buffer

---

Unified buffer ~ 24 MB in size

→ SRAM → Inputs Intermediate



# INSTRUCTIONS

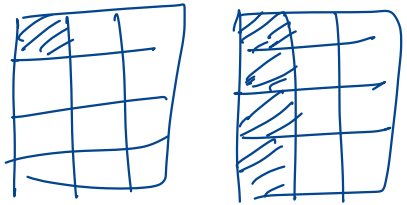
CISC format (why ?)

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

All the model layers are translated into these instruction

Very small number of instructions - very specific to workload

# SYSTOLIC EXECUTION



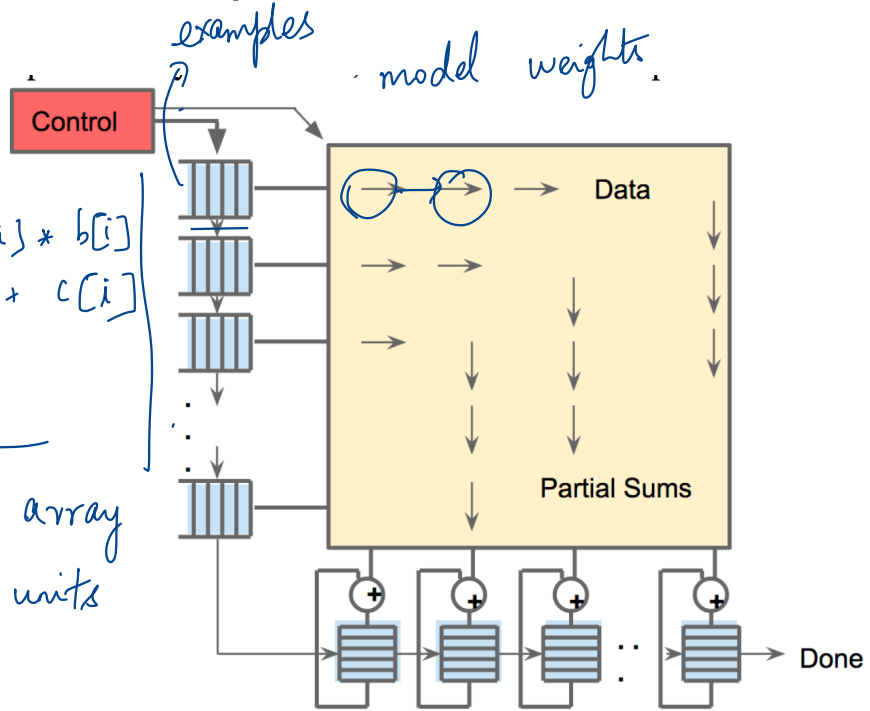
Problem: Reading a large SRAM uses much more power than arithmetic!

CPU / Von Neumman arch

- Read matrix slice from DRAM / cache
- Floating Point Unit → c[i] = a[i] * b[i] + c[i]
- Return values are in DRAM / cache

Systolic arrays

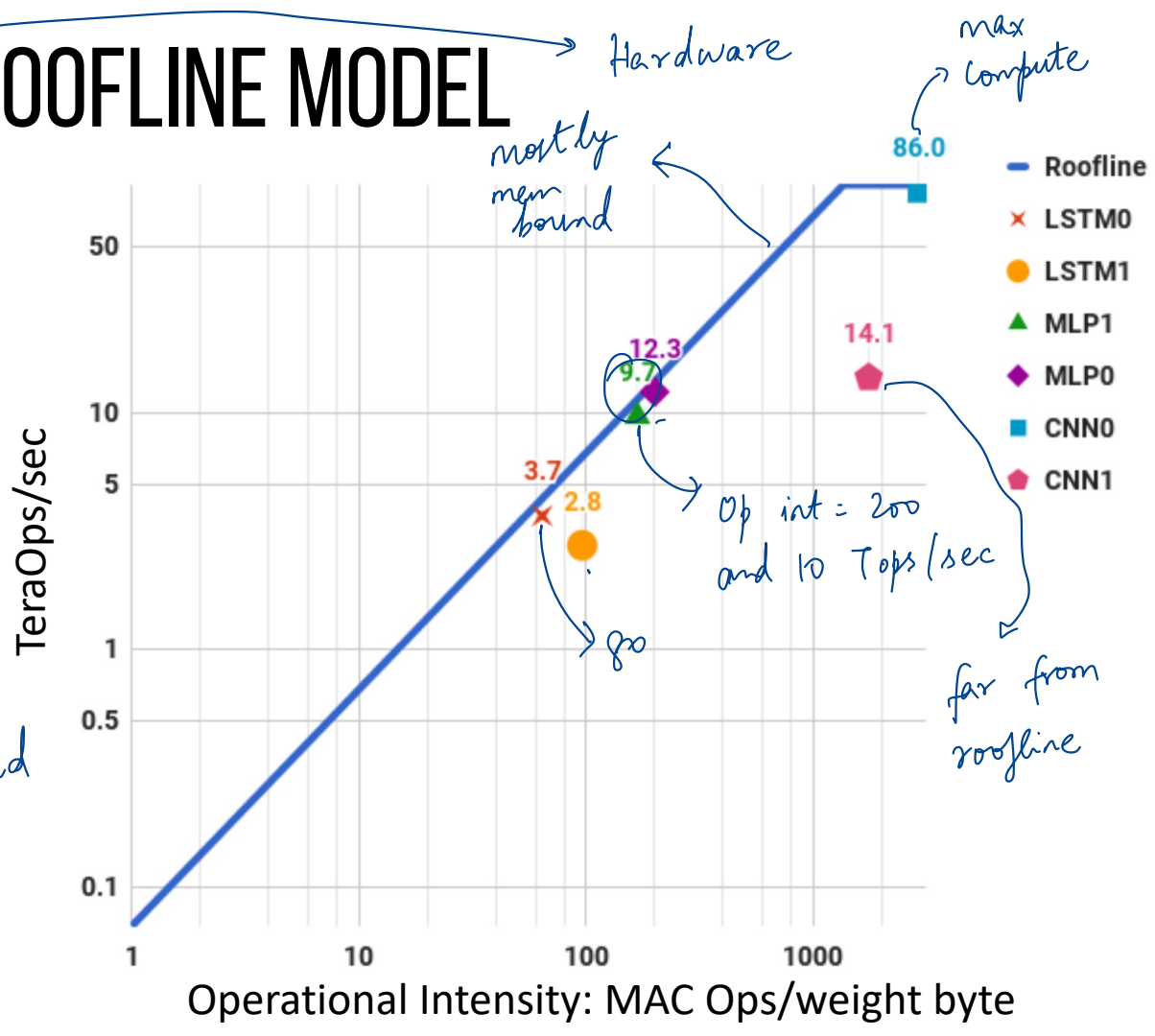
- Data streaming through systolic array
- Not general purpose compute units



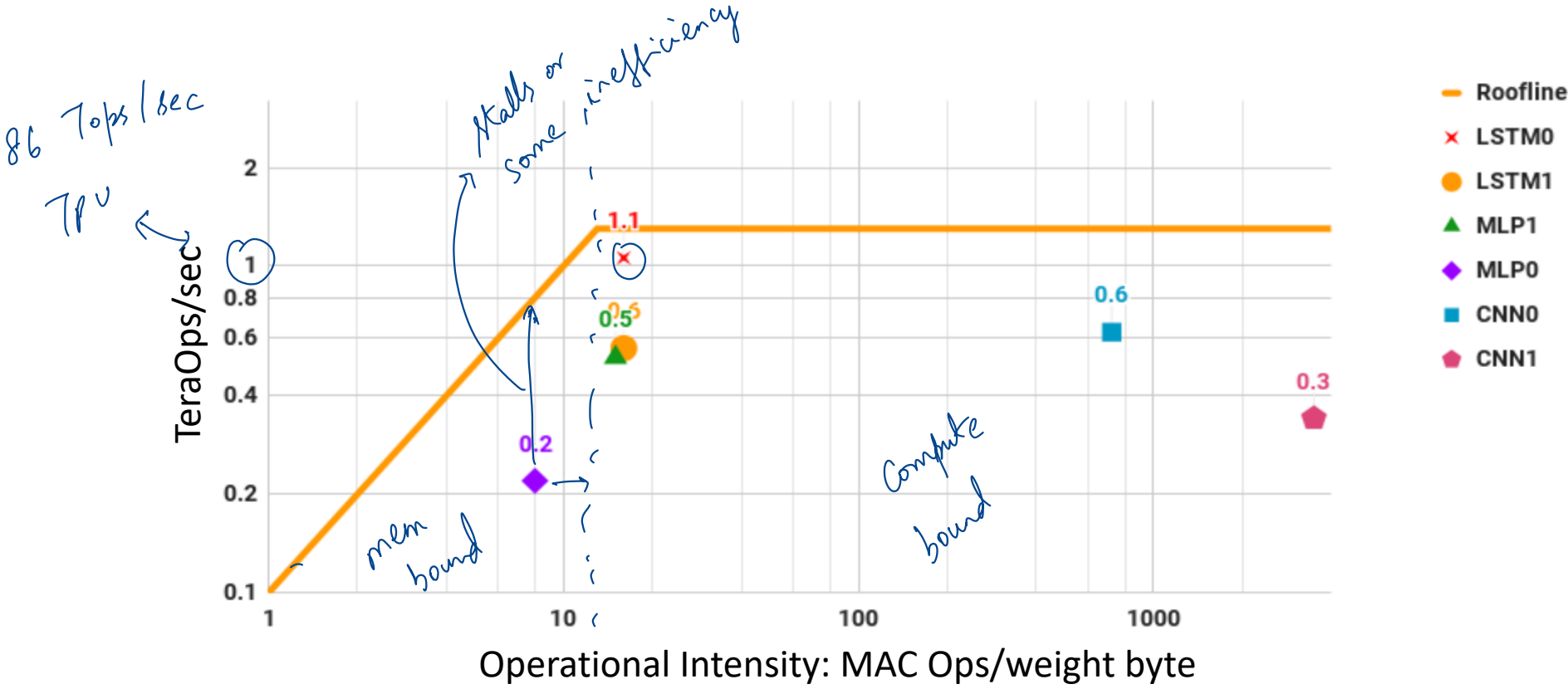
# ROOFLINE MODEL

Performance model → Hardware

- tells you how much peak performance for given "operational intensity"
  → Multiply or Add ops per byte read from memory
- Slope and flat part
  ↓ ↓
  mem bound  comp. bound



# HASWELL ROOFLINE



# COMPARISON WITH CPU, GPU

max power device draw ← Total drawn Power

quantized integer

| Model | Die | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $mm^2$ | 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 |

clock speed

Performance / Watt is very high!

much larger

# SELECTED LESSONS

Predictable

- Latency more important than throughput for inference
- LSTMs and MLPs are more common than CNNs
- 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/Gx5M7NhMjTvKtuUB6

User → latency SLO = 8 ms

smaller batch to meet SLO

need relatively large batch size to fill TPU

| *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% |

batch size 200 satisfies 8 ms

200,000 inf/s target = 1 TPU → lower power, lower TCO

high utilization

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

↳ Can we share TPU for diff models

(a) Store Resnet-18, MLP on the TPU → share DRAM

(b) Compute inference for Resnet 18 | Compute one after another

(c) Compute inference for MLP

→ fewer TPUs

→ Batch size is important → early dropping to ensure high batch size

→ 99th percentile latency predictable

# NEXT STEPS

Next week schedule

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