Lecture 2: Domain Specific Architectures

CS 256: Systems and Machine Learning

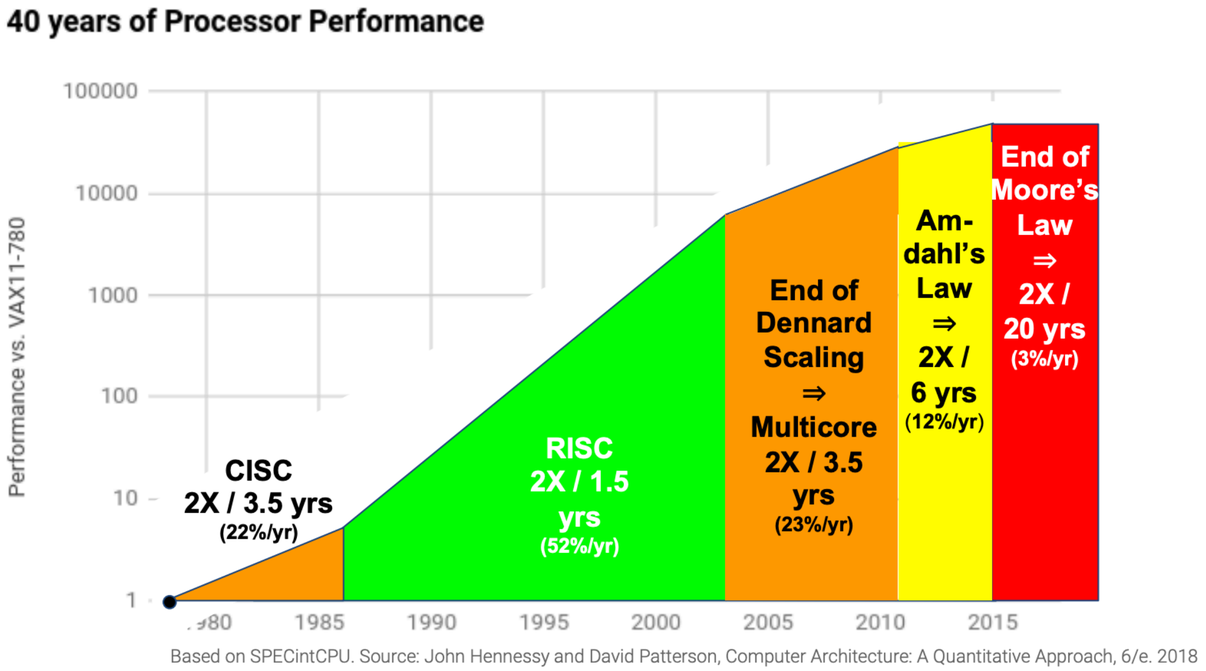
Sangeetha Abdu Jyothi

UCIrvine

This lecture uses material adapted from David Patterson's talks on DSA

Need for Domain Specific Architectures

- Architecture goals
  - Maximize performance
  - Minimize cost
  - Improve energy efficiency
- But Moore's law and Dennard scaling are ending
- Hence, we need Domain Specific Architectures for performance improvement



Based on SPECintCPU. Source: John Hennessy and David Patterson, Computer Architecture: A Quantitative Approach, 6/e. 2018

Deep Learning Workload

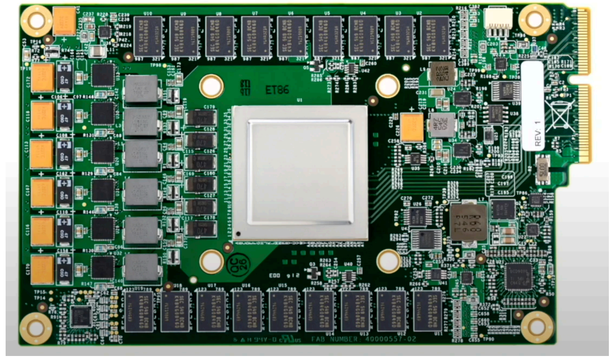
- Types of Workloads
  - DNN training (Learning weights of a DNN model)
  - Inference (Using the learned model to make predictions)
- Workload characteristics
  - Compute intensive
    - Large matrix multiplications, convolutions, etc.
  - Memory intensive
    - Several layers with millions to billions of parameters

Origin of TPU

- Observation at Google in 2013: If 100 million people talked to Google phones 3 min a day, Google will need to double their data center capacity

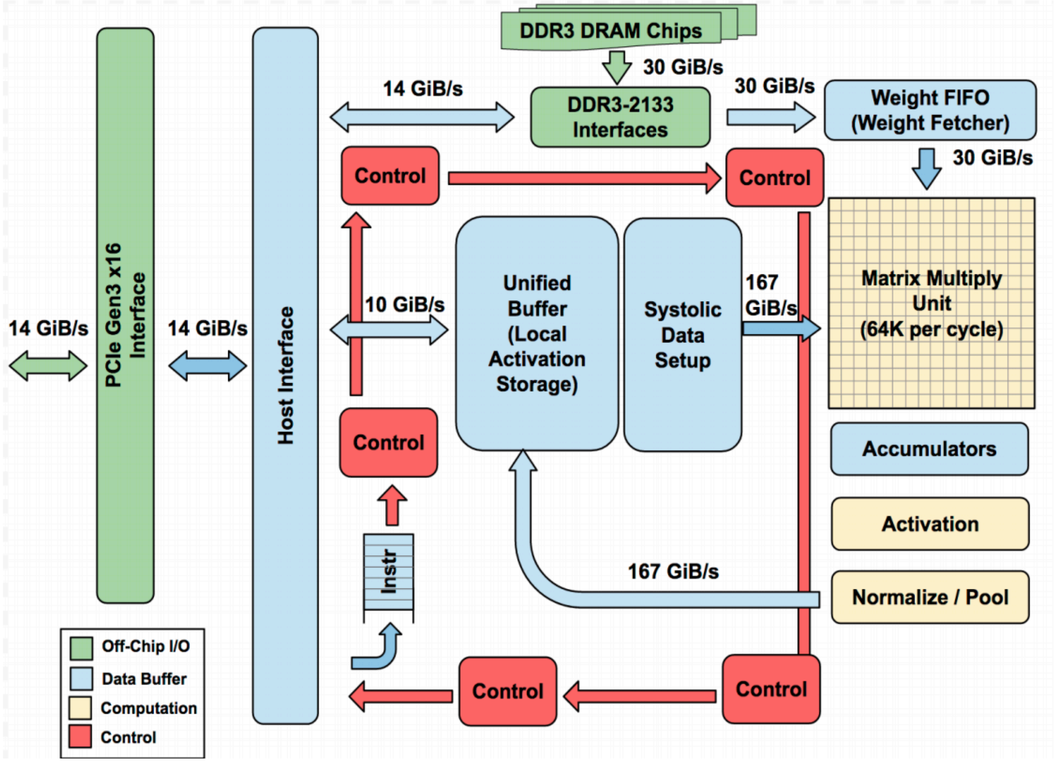
- New project: Custom hardware to reduce Total Cost of Ownership (TCO) of DNN inference by 10X

- This led to TPUv1 which was designed and deployed in production in just 15 months



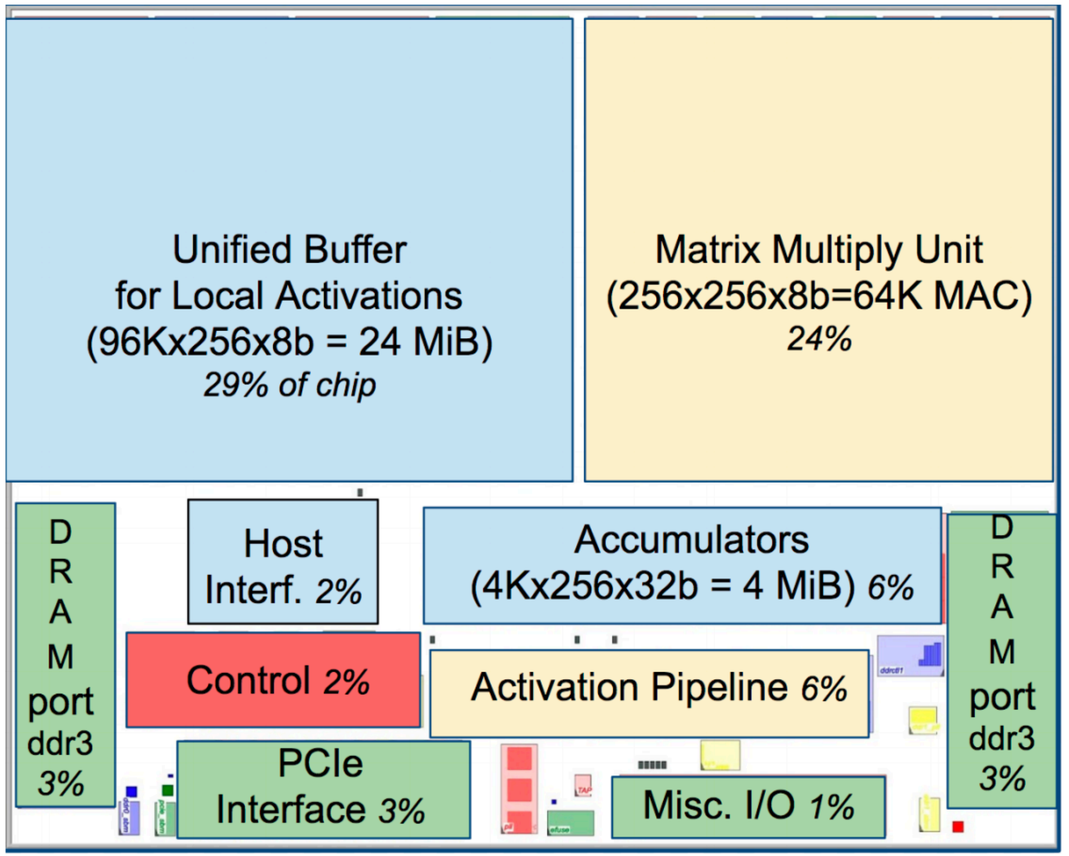
High-Level architecture

- Matrix Multiply Unit
  (65536 8-bit Multiple-Accumulate units)
- 25X MACs compared to GPUs at that time
- 700 MHz clock rate
- 92T operation/sec (65536 * 2 * 700M)
- 4 MiB on-chip accumulator memory
- 3.5X memory compared to GPUs
- Two 2133MHz DDR3 DRAM channels for weights (8GiB)



Floorplan of TPU die

- Datapath is 67%
- I/O is 10%
- Control is 2%



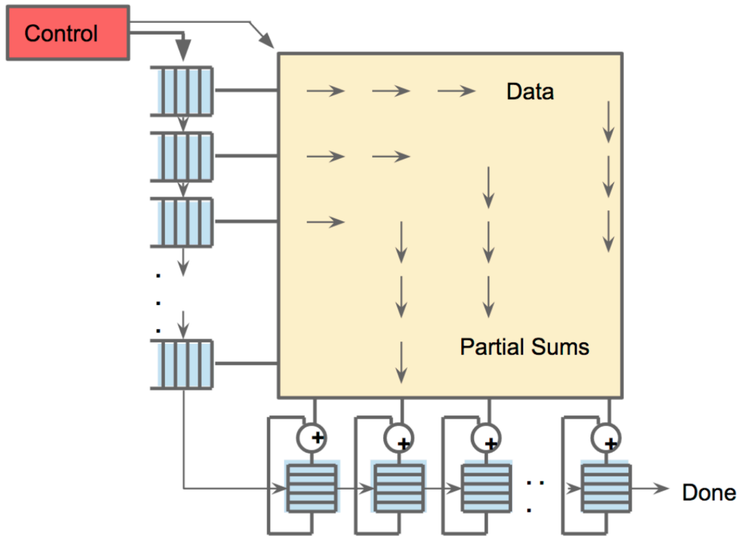
Systolic Execution

- Problem: During matrix multiplication, multiple SRAM/register accesses which leads to energy/time inefficiencies
- Solved by systolic execution: pipelining of control and data
- Intermediate results available at cells exactly at the instance they are required

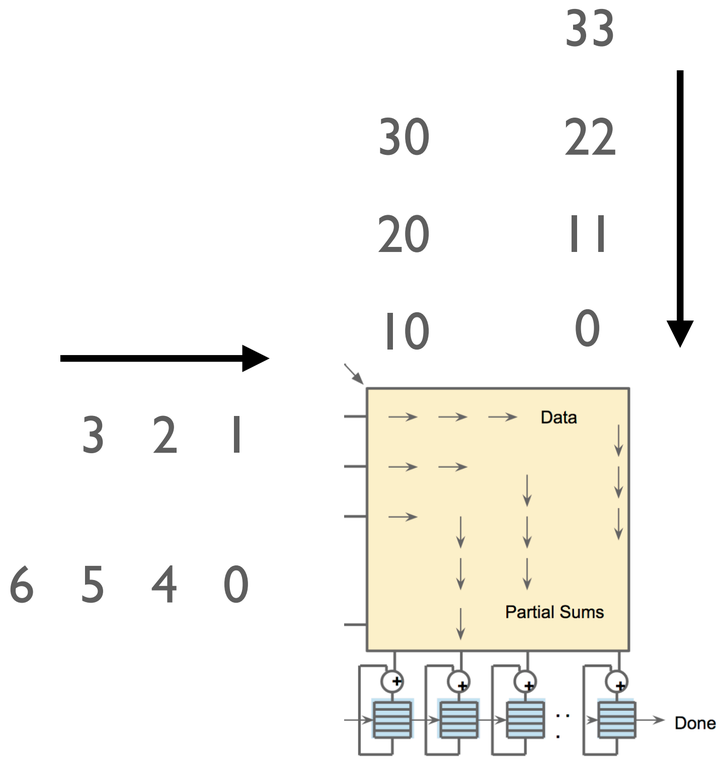
$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 10 & 11 \\ 20 & 21 \\ 30 & 31 \end{bmatrix}$$

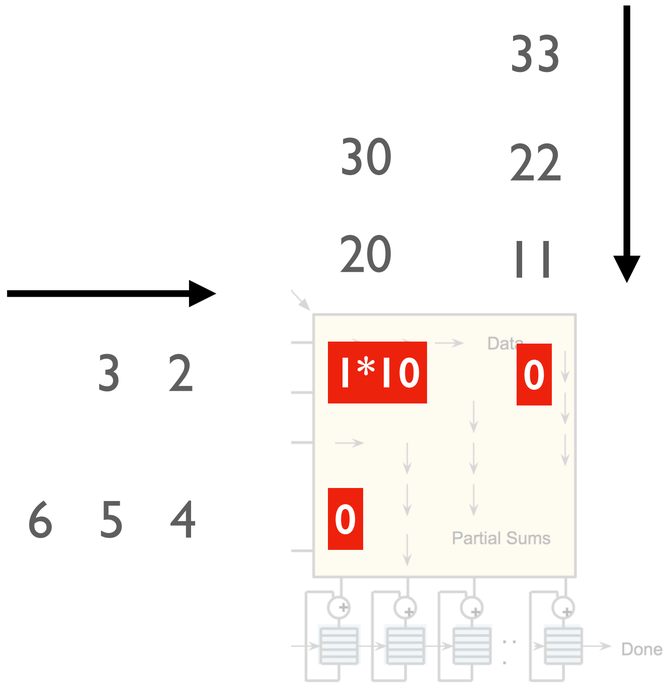
$$= \begin{bmatrix} 1x10 + 2x20 + 3x30 & 1x11 + 2x21 + 3x31 \\ 4x10 + 5x20 + 6x30 & 4x11 + 5x21 + 6x31 \end{bmatrix}$$

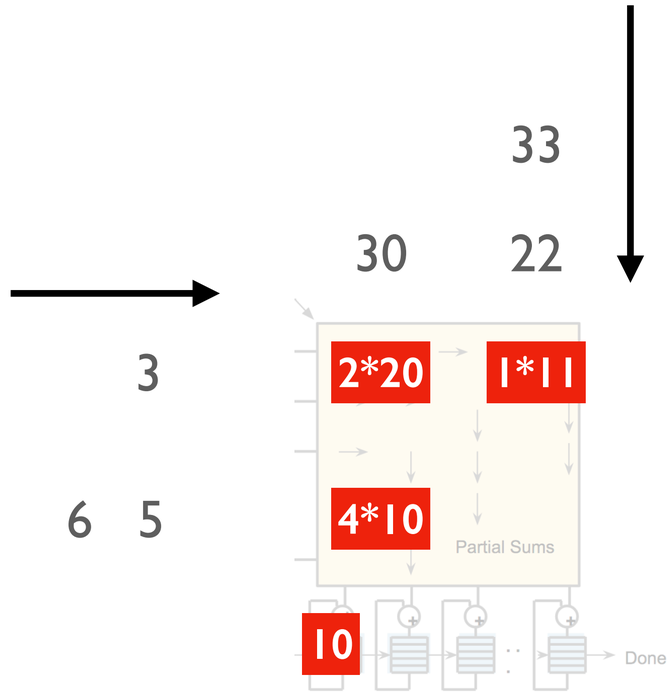
$$= \begin{bmatrix} 10+40+90 & 11+42+93 \\ 40+100+180 & 44+105+186 \end{bmatrix} = \begin{bmatrix} 140 & 146 \\ 320 & 335 \end{bmatrix}$$



Systolic Execution





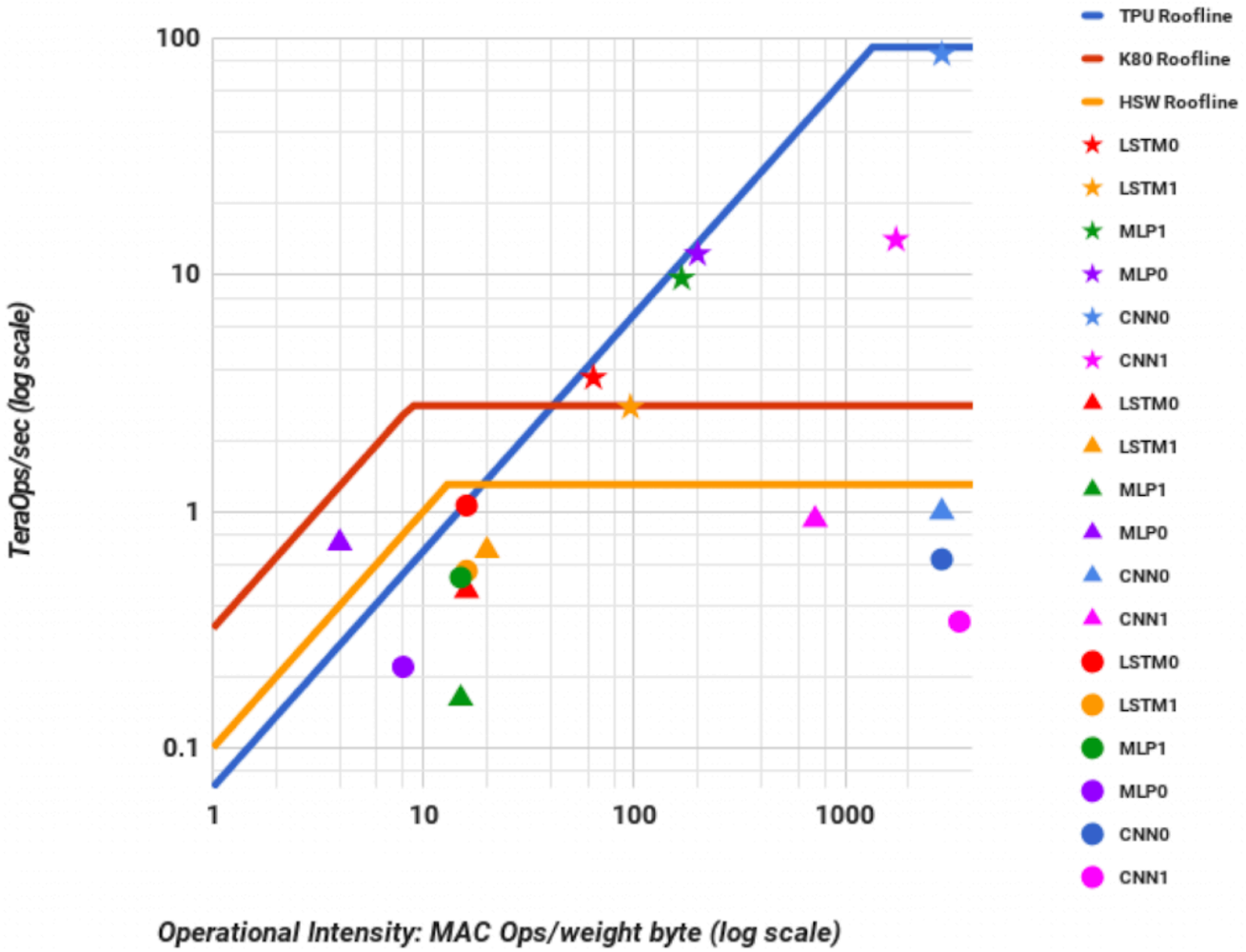


$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 10 & 11 \\ 20 & 21 \\ 30 & 31 \end{bmatrix}$$

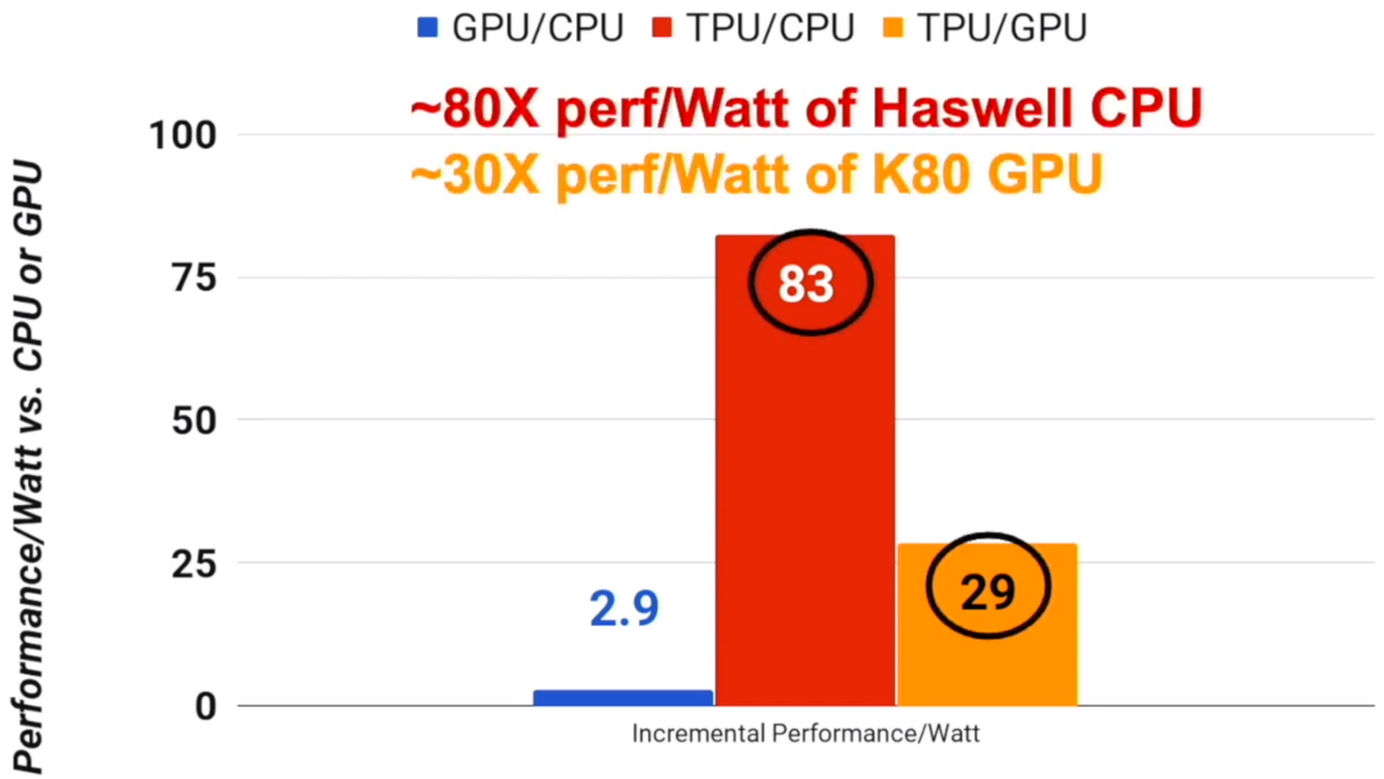
$$= \begin{bmatrix} 1x10 + 2x20 + 3x30 & 1x11 + 2x21 + 3x31 \\ 4x10 + 5x20 + 6x30 & 4x11 + 5x21 + 6x31 \end{bmatrix}$$

$$= \begin{bmatrix} 10+40+90 & 11+42+93 \\ 40+100+180 & 44+105+186 \end{bmatrix} = \begin{bmatrix} 140 & 146 \\ 320 & 335 \end{bmatrix}$$

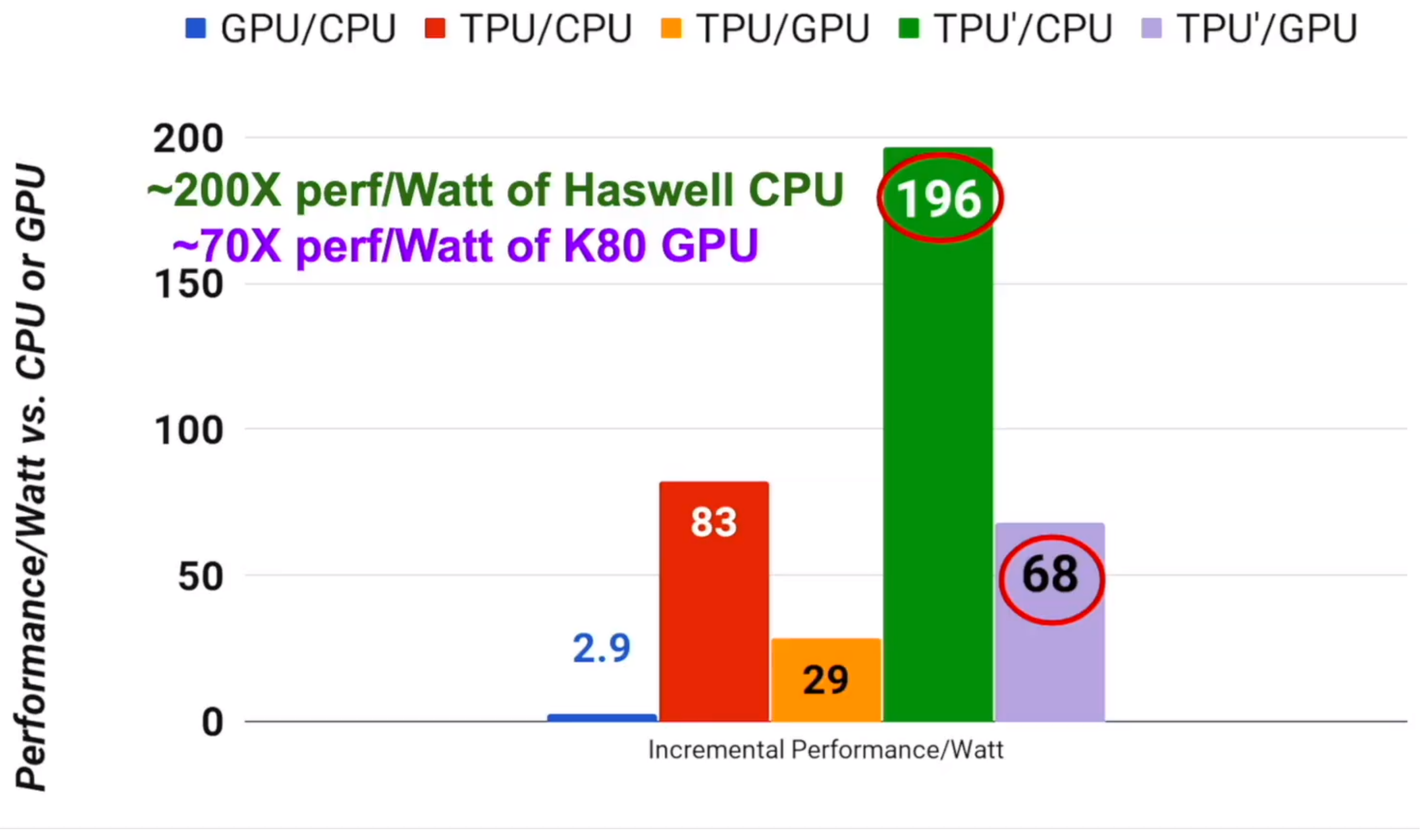
Roofline Model



Performance improvement



Can it be improved further?



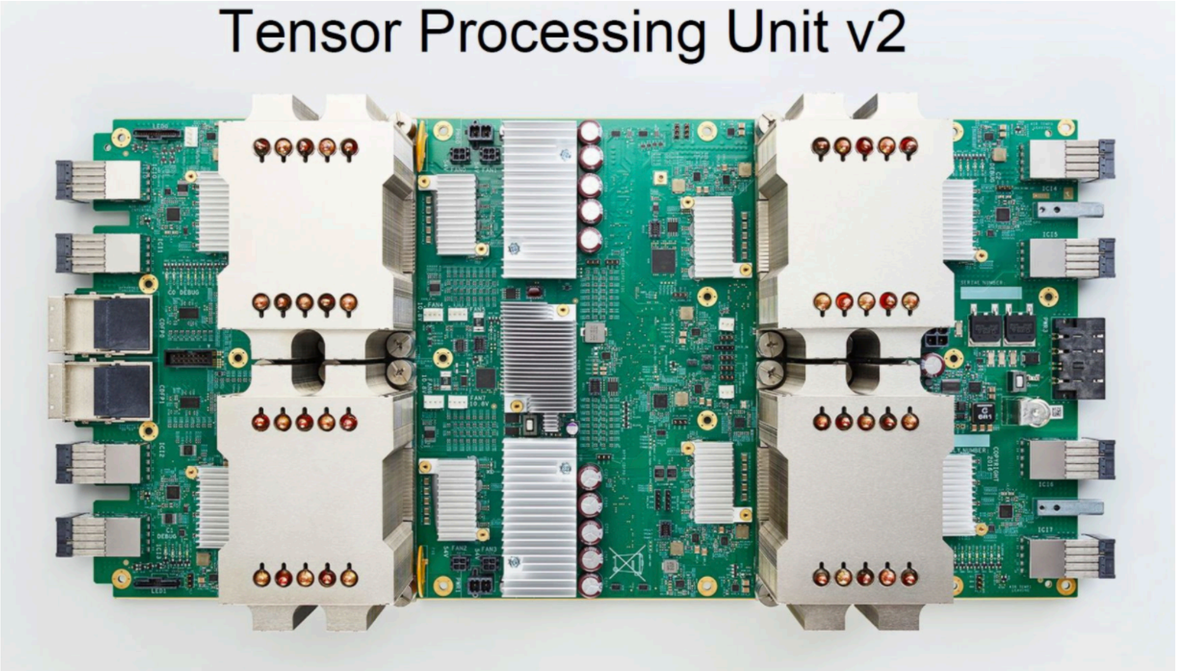
Simulated results with bigger MXU, faster clock, faster memory

Reasons for TPUv1 success

- 1 large 2D multiplier instead of several smaller 1D multipliers
- 8-bit ints vs 32-bit FP => more efficient computation / memory
- Systolic array => fewer registers (less energy)
- TPUv1 drops several CPU/GPU features => saves energy, reuse transistors for domain-specific on-chip memory

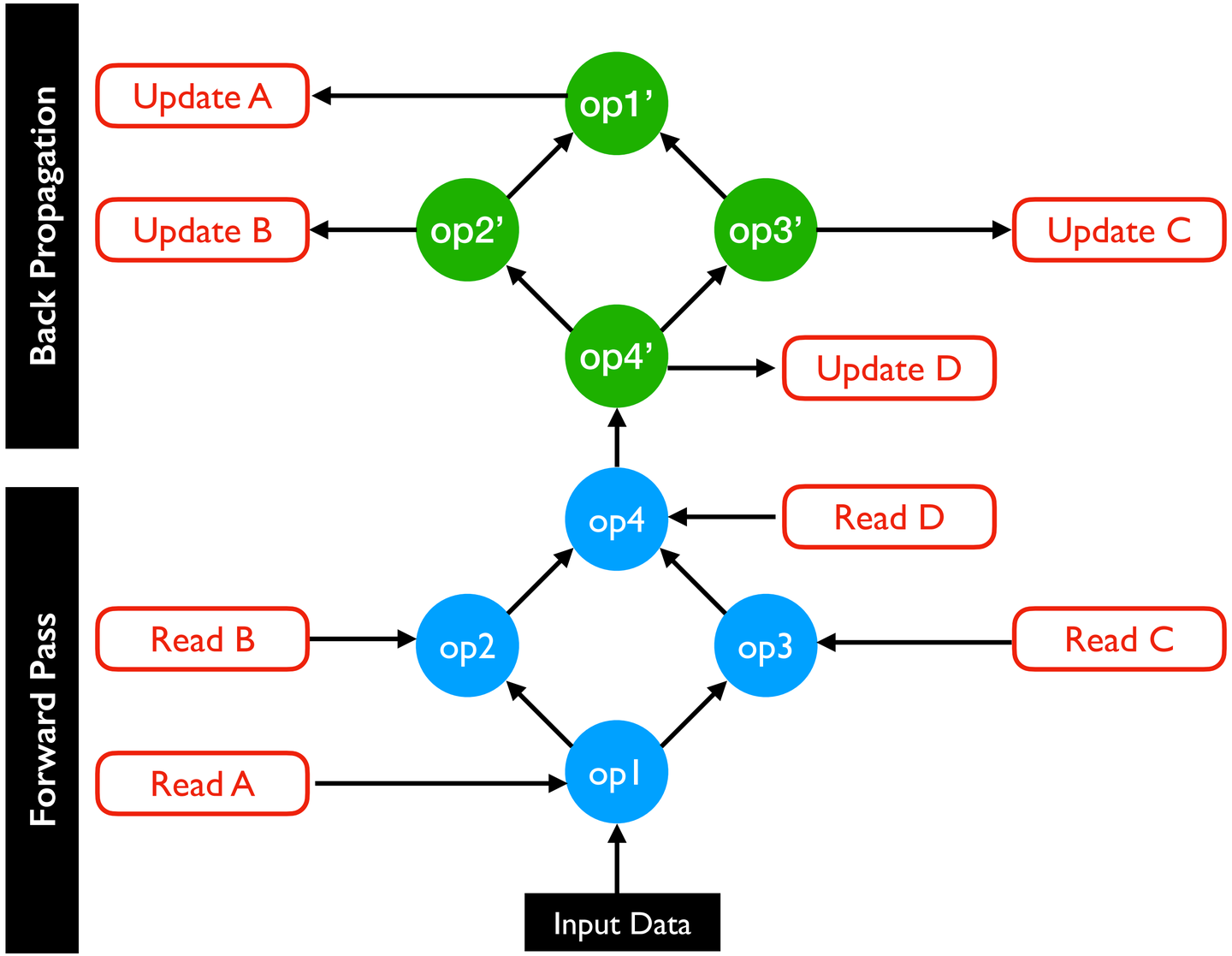
TPUv2

- TPUv1: tailored for inference
- TPUv2: targets training workloads
- Training is more difficult to handle

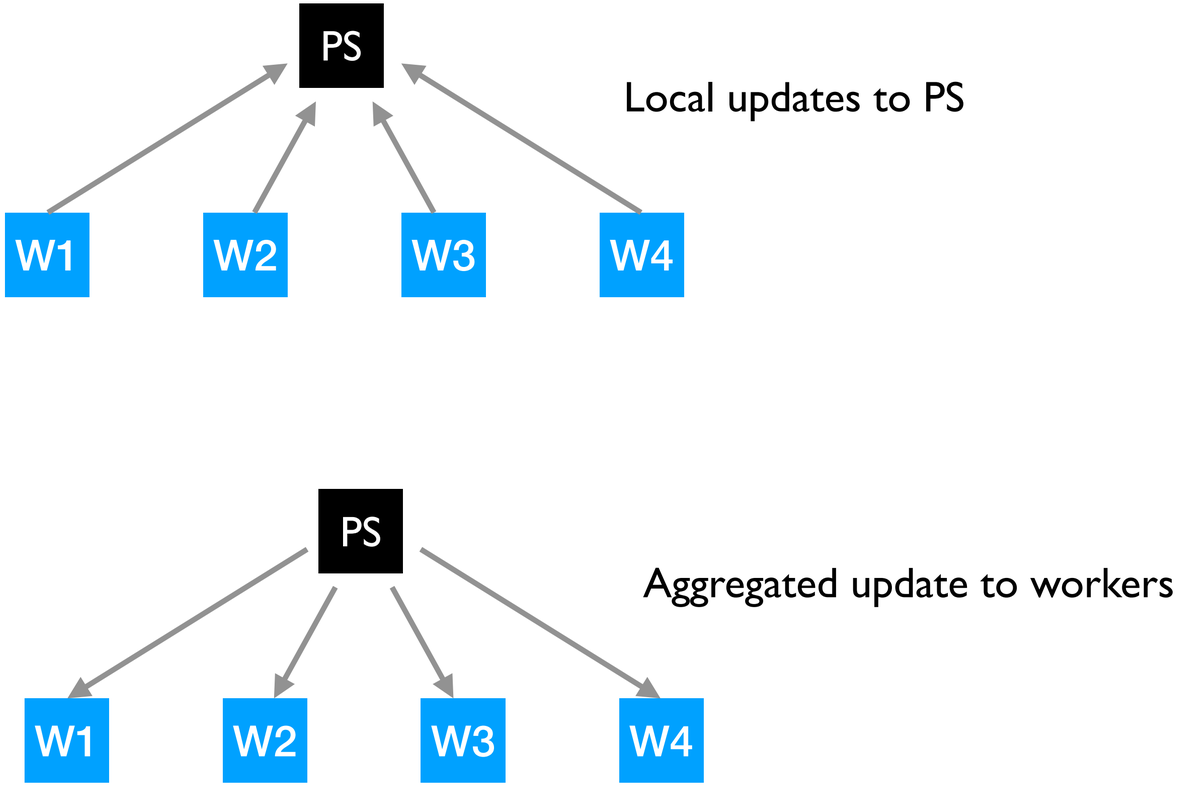


Training Process

- Initialize model with random weights
- SGD to learn weights
- 2 steps:
  - Forward pass
  - Backpropagation
- Weights updated after backpropagation
- Inference ~= Forward Pass Alone



Distributed Training



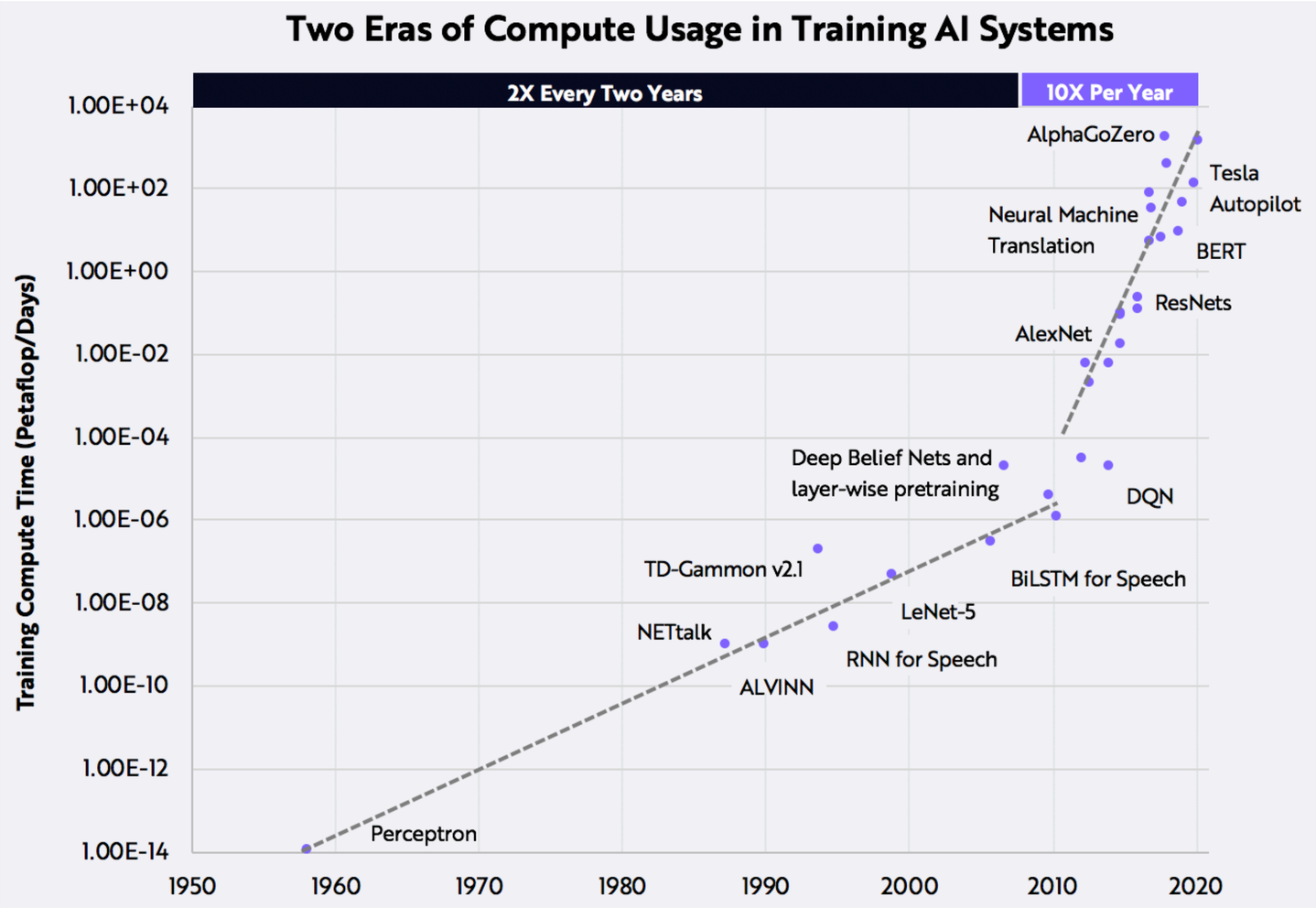
Discussion

- How are requirements of training different from inference?
    - Compute?
    - Memory?
    - Network?
- Other considerations? What would you build?

Why is training more difficult?

- More compute
- More memory
- More programmable
- Bigger numerics
- Parallelization is harder

ML Training trends



What to build?

- 2-16 months to train production DNNs on 1 chip
- Bigger machines + more data => bigger ML breakthroughs
- Build a NN supercomputer (TPUv2) instead of a NN coprocessor (TPUv1)

ML Training Quality Determines Correctness

- ML Quality ≈ Correctness: Fast but Incorrect Uninteresting
- 1% quality loss to ML practitioners can be like getting wrong answer
    - Aiming for intelligent app for a billion people, so lower quality can mean worse experience for millions of people / loss of income
- For datacenter production apps, training has to be in floating point
    - Researchers exploring fixed point for training but at cost in quality
    - Production remains floating point (but FP32 sufficient, no need for FP64)

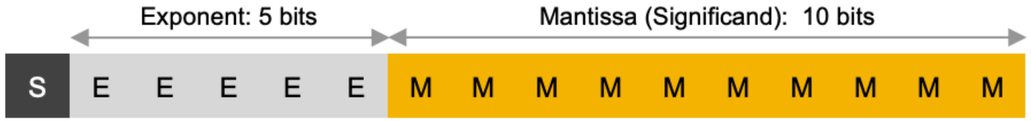
bfloat16 (Brain Floating Point)

fp32: Single-precision IEEE Floating Point Format

Range: ~1e^{-38} to ~3e^{38}

fp16: Half-precision IEEE Floating Point Format

Range: ~5.96e^{-8} to 65504



bfloat16: Brain Floating Point Format

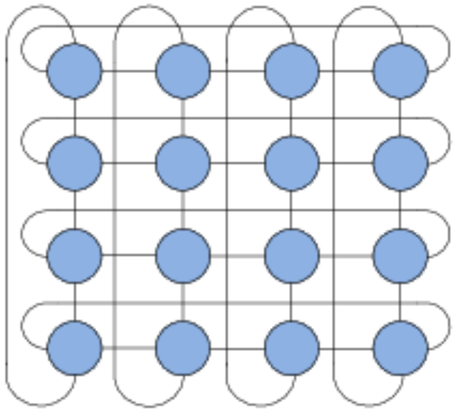
Range: ~1e^{-38} to ~3e^{38}

bfloat16

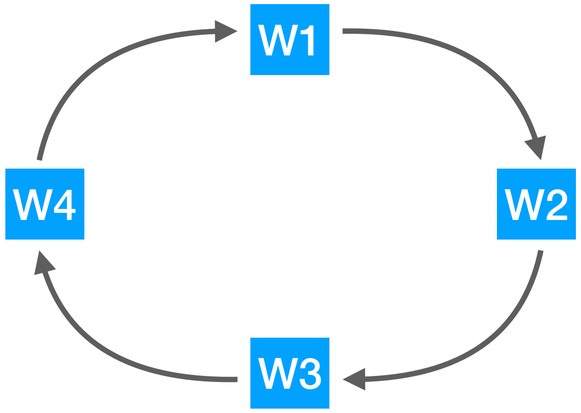
- Hardware: small mantissa reduces multiplier power, area
  - float32: $23^2 = 529$
  - float16: $10^2 = 100$
  - bfloat16: $7^2 = 49$
- Software: same dynamic range on number line
  - same Inf/NaN behavior as float
- Numerics: Unlike IEEE fp16, bfloat16 trains without loss scaling [Micikevicius 2017]
- System: bfloat16 as an implementation technique inside the matrix multiplier.
  - Can also expose it to save memory capacity and bandwidth, with more work

TPUv2 Supercomputer Network Interconnect

- TPUv2 chips have 4 custom Inter-Core Interconnect (ICI) Links
  - 500 Gbps in both directions
  - Allows direct wire connection between TPUv2 chips
  - Uses only 13% of the die
- On-device switch provides deadlock-free routing in 2D Torus topology
  - Inspired shift to ring AllReduce based aggregation which can be mapped easily to 2D Torus



2D Torus (4-ary 2-cube)

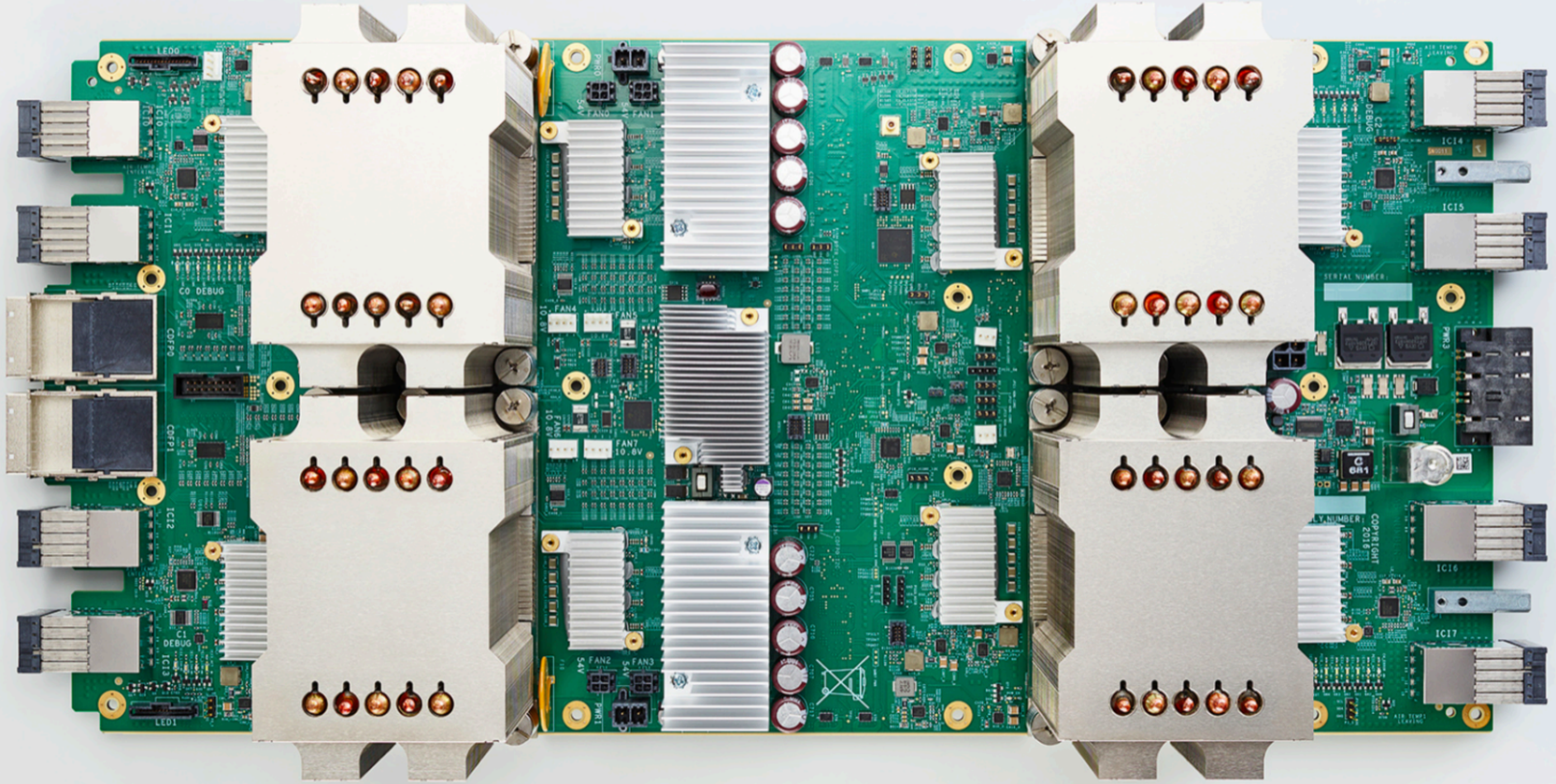


Ring AllReduce

Supercomputer Node: How many cores?

- Global wires don't scale with shrinking feature size, so relative delay increases
- 2 smaller TensorCores/chip prevented excessive latencies of a 1 full-chip core
- Easier to generate programs on 2 cores rather than several wimpier ones

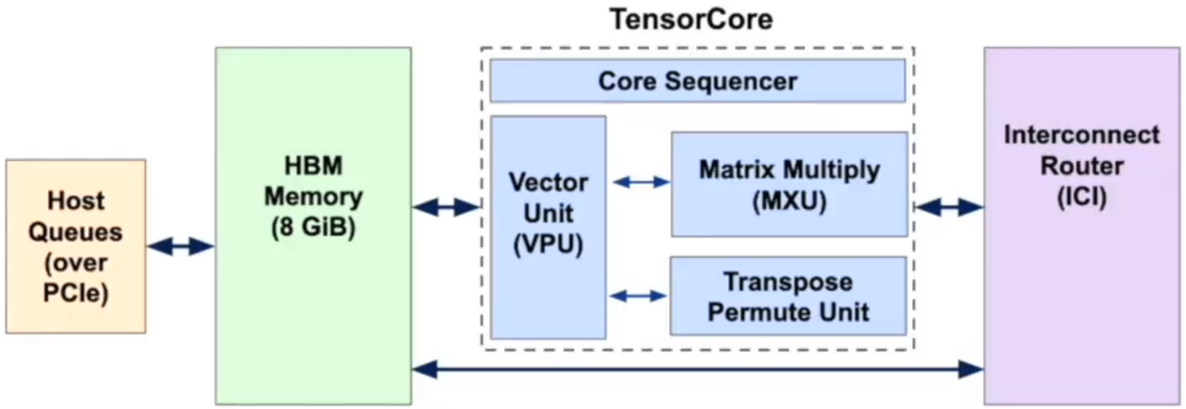
Cloud TPU (v2)



- 180 teraflops of computation, 64 GB of HBM memory, 2400 GB/s mem. BW
- Designed to be connected together into larger configurations

TPUv2 Block Diagram

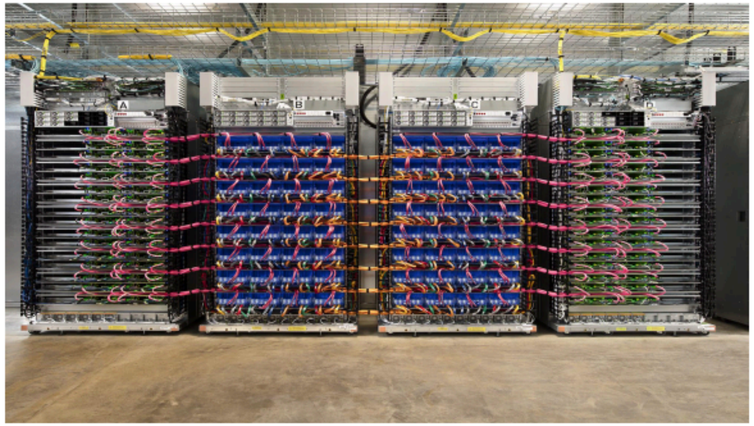
- 1 128*128 systolic MXU
  - bfloat16 multipliers
  - float32 accumulate
- Special hardware for Transpose Reduction Permute (TRP)
- Vector Processing Unit: 32 2D vector registers + 2D Vector memory (16MiB)
- Inter-core Interconnect
- High Bandwidth Memory



TPUv3

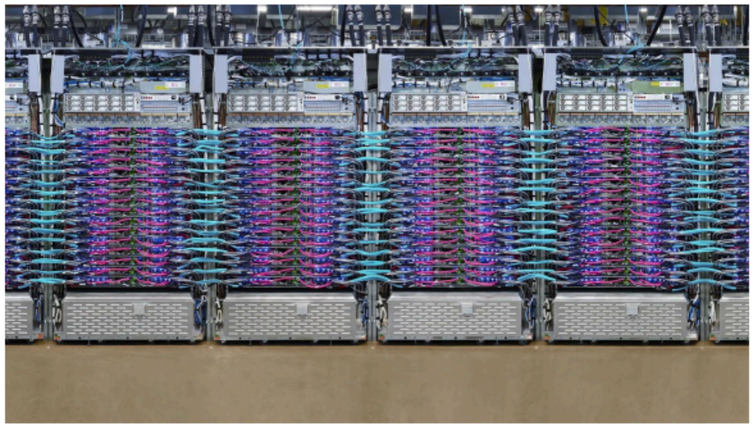
- Improvements on same technology
- 1.35X clock rate, ICI bandwidth, HBM bandwidth
- 2 MXUs/core
- 2.7X peak multiply performance per chip
- 1.6X more power
- 2X HBM memory capacity
- Die size grew only 6%
- 4X bigger supercomputer (1024 vs 256 chips)

TPUv2 vs TPUv3 clouds



Cloud TPU Pod (v2, 2017)

11.5 petaflops
4 TB HBM
2-D toroidal mesh network
Training and inference
Alpha



TPU v3 Pod (2018)

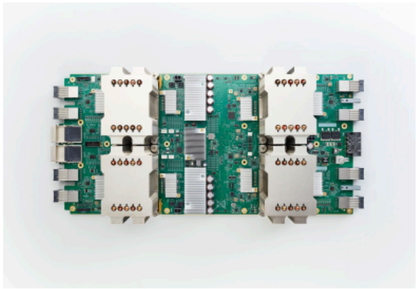
> 100 petaflops!
32 TB HBM
Liquid cooled
New chip architecture + larger-scale system

The Evolution..

TPU v1
(deployed 2015)

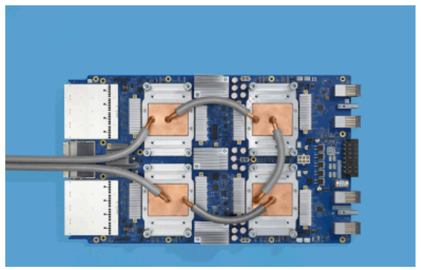
92 teraops
Inference only

Cloud TPUv2



180 teraflops
64 GB HBM
Training and inference
Generally available (GA)

Cloud TPUv3



420 teraflops
128 GB HBM
Training and inference

TPU v4 (2021) vs. TPU v3 (2018)

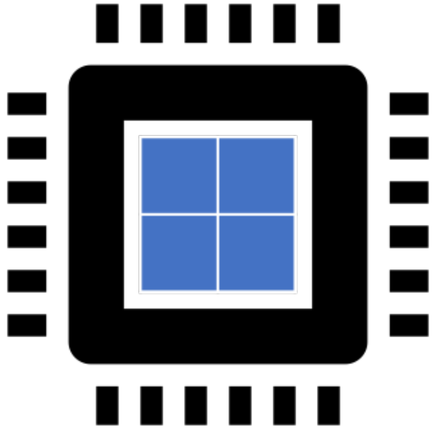
Chip feature	Cloud TPU v3	Cloud TPU v4
Peak compute per chip	123 teraflops (bf16)	275 teraflops (bf16 or int8)
HBM2 capacity and bandwidth	32 GiB, 900 GB/s	32 GiB, 1200 GB/s
Measured min/mean/max power	123/220/262 W	90/170/192 W
TPU pod size	1024 chips	4096 chips
Interconnect topology	2D torus	3D torus
Peak compute per pod	126 petaflops (bf16)	1.1 exaflops (bf16 or int8)
All-reduce bandwidth per pod	340 TB/s	1.1 PB/s
Bisection bandwidth per pod	6.4 TB/s	24 TB/s

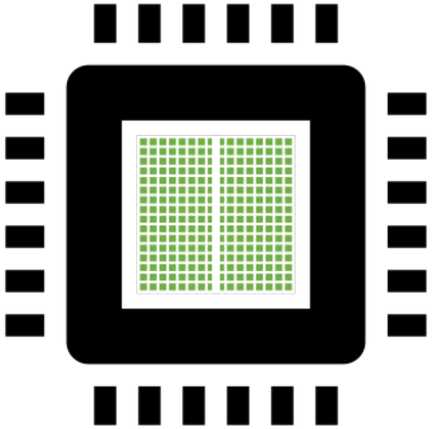
What next?

- DNNs are improving really fast
  - LSTM (1997) → Transformer (2017) → BERT (2018) → GPT-3 (2020)
- Demand for ML accelerators growing
- Many more open problems in Domain Specific Architectures compared to general purpose computing
- Efficiency will matter a lot for new designs
- Opportunities for hardware/software codesign

Graphics Processing Unit (GPU) [Not a DSA!]

- SIMD (Single Instruction, Multiple Data)
- Large number of cores
- High memory bandwidth
- High throughput





CPU	GPU
Central Processing Unit	Graphics Processing Unit
4-8 Cores	100s or 1000s of Cores
Low Latency	High Throughput
Good for Serial Processing	Good for Parallel Processing
Quickly Process Tasks That Require Interactivity	Breaks Jobs Into Separate Tasks To Process Simultaneously

Related Information

- MLPerf / ML Commons (https://mlcommons.org/en/)
  - Consortium of companies and universities
  - Benchmarking, datasets, and best practices information for ML practitioners
  - Benchmarks on different hardware for both training and inference (e.g., https://mlcommons.org/en/training-normal-11/)

Thanks!