EE-508: Hardware Foundations for Machine Learning
TPU Architecture

University of Southern California

Ming Hsieh Department of Electrical and Computer Engineering

Instructors:
Arash Saifhashemi

TPU Architecture

Acknowledgement:

- Many slides are taken Google's Training Chips Revealed: TPUv2 and TPUv3, by Thomas Norrie et al. HC32, 2020
- Several diagrams are taken from Google's TPU paper ISCA'17

Google’s Projection in 2013

*“DNNs could become so popular that they might **double** computation demands on our datacenters, which would be very expensive to satisfy with conventional CPUs”*

COMPANIES > GOOGLE (ALPHABET)

Google Has Spent $21 Billion on Data Centers

Inside a Google data center in the Dalles, Oregon. (Photo: Google)

Google has invested more than $21 billion in its Internet infrastructure since the company began building its own custom data centers in 2006.

Rich Miller | Sep 17, 2013

Source: “datacenterknowledge.com”

Reuters

World Business Markets Sustainability Legal Breakingviews Technology Investigations

Microsoft plans to invest $80 billion on AI-enabled data centers in fiscal 2025

By **Reuters**

January 3, 2025 3:22 PM PST · Updated 2 months ago

Investopedia

INVESTING SIMULATOR BANKING PERSONAL F

COMPANY NEWS › TECH SECTOR NEWS

Google Parent Alphabet Plans to Spend $75B This Year, as Big Tech Goes All in on AI

By ANDREW KESSEL Published February 04, 2025 07:55 PM EST

TECHNOLOGY | ARTIFICIAL INTELLIGENCE Follow

Meta Spending to Soar on AI, Massive Data Center

Social-media giant to spend between $60 billion and $65 billion, Zuckerberg says

By Meghan Bobrowsky Follow

Updated Jan. 24, 2025 12:04 pm ET

Share Resize Listen (1 min)

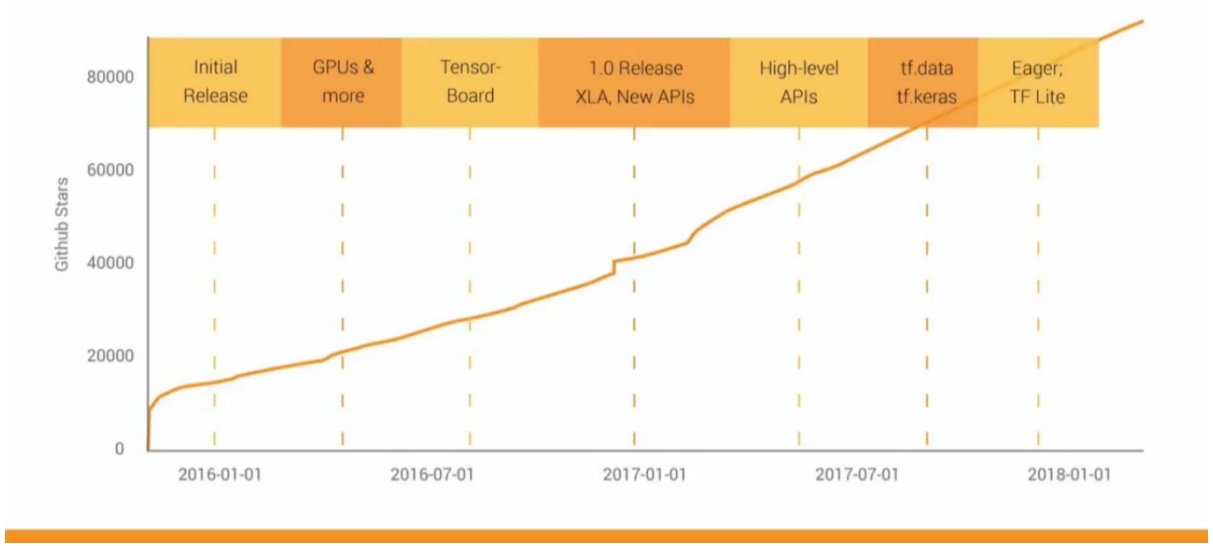
Mark Zuckerberg, chief executive officer of Meta. PHOTO: DAVID PAUL MORRIS/BLOOMBERG NEWS

The Problem TPU Aimed to Solve

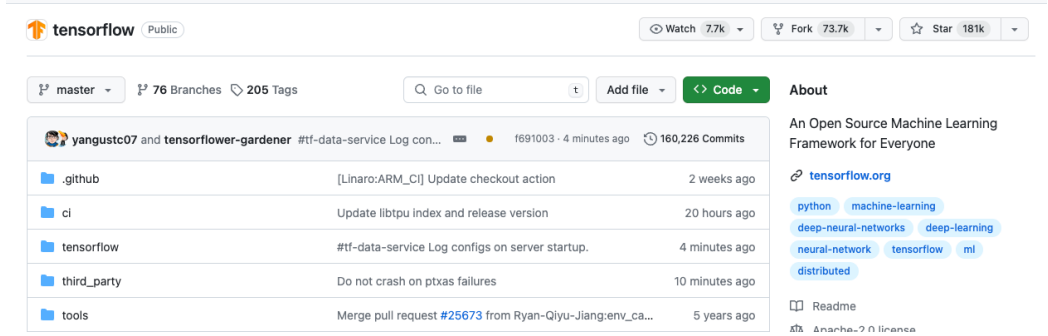
- **Computational Demands of Deep Learning**
  - Training deep neural networks requires massive amounts of computation.
  - Traditional CPUs and GPUs, while powerful, were not optimized specifically for neural network operations.
- **Latency & Throughput**
  - Real-time applications, such as voice and image recognition, require low latency.
  - Need for high throughput to process large datasets efficiently.
- **Power Efficiency**
  - Data centers consume significant amounts of energy.
  - Efficient computation per watt is crucial for both environmental and economic reasons.

The Problem TPU Aimed to Solve

- **Cost Constraints**
  - A need for a cost-effective solution tailored for machine learning tasks.
- **Integration with Machine Learning Frameworks**
  - Many hardware accelerators require significant changes to existing software.
  - A seamless integration with popular frameworks, like TensorFlow, was desired.



Source: “analyticsvidhya.com”



Source: https://github.com/tensorflow/tensorflow

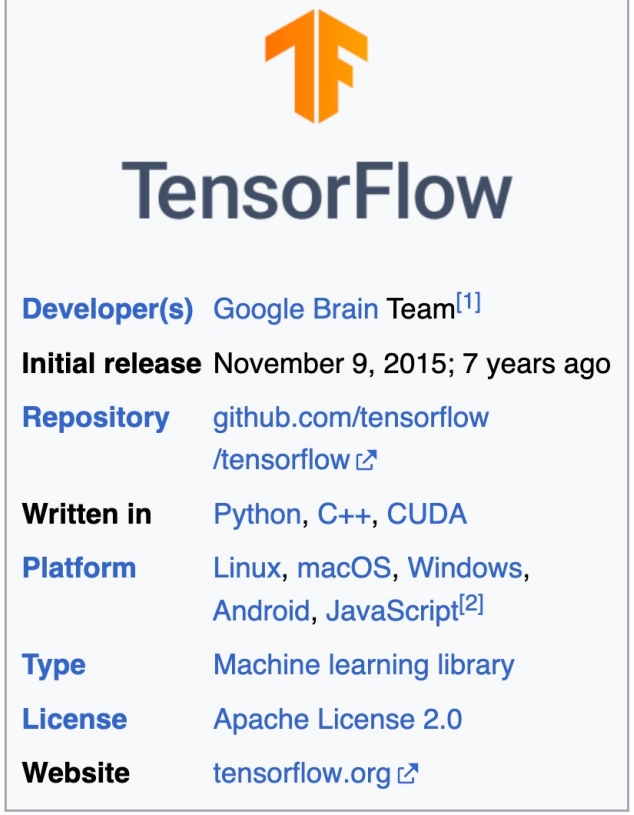
TPU Goals

- **High Throughput for Neural Network Computation**
  - Designed to accelerate matrix multiplications, core in deep learning.
  - Optimized for large batches and high computational demands.
- **Used for Inference Only**
  - Use off-the-shelf GPUs for training
- **Cost Efficiency & TCO**
  - Custom ASIC (Application-Specific Integrated Circuit) design.
  - More operations per watt than traditional CPUs and GPUs.
  - **Total Cost of Ownership:** Designed to be cost-effective over its lifecycle, considering acquisition, operation, and maintenance costs.
  - Improve cost-performance by **10X** over GPUs.

TPU Goals

- **Integration with TensorFlow**
  - Seamless integration with Google's TensorFlow framework.
  - Harness the power of TPUs easily.
- **Scalability**
  - Designed for easy scaling in data centers.
  - Can be used in tandem for larger models and datasets.

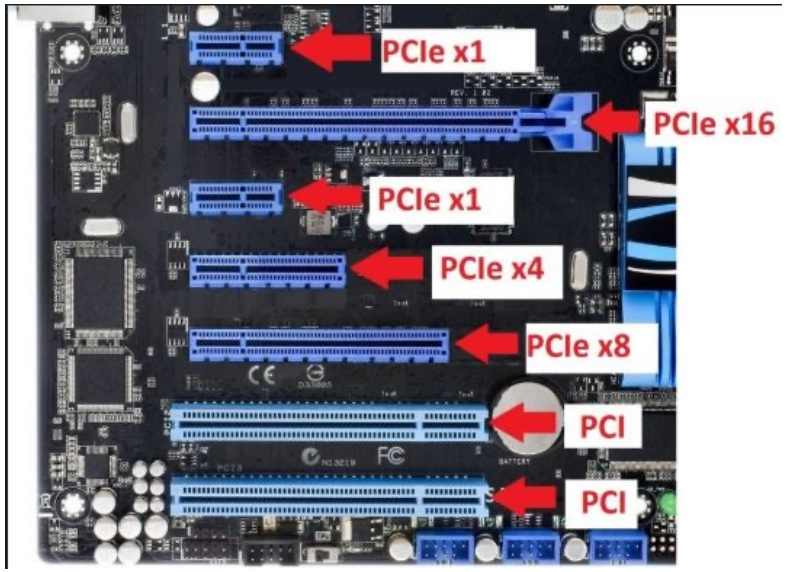
TensorFlow



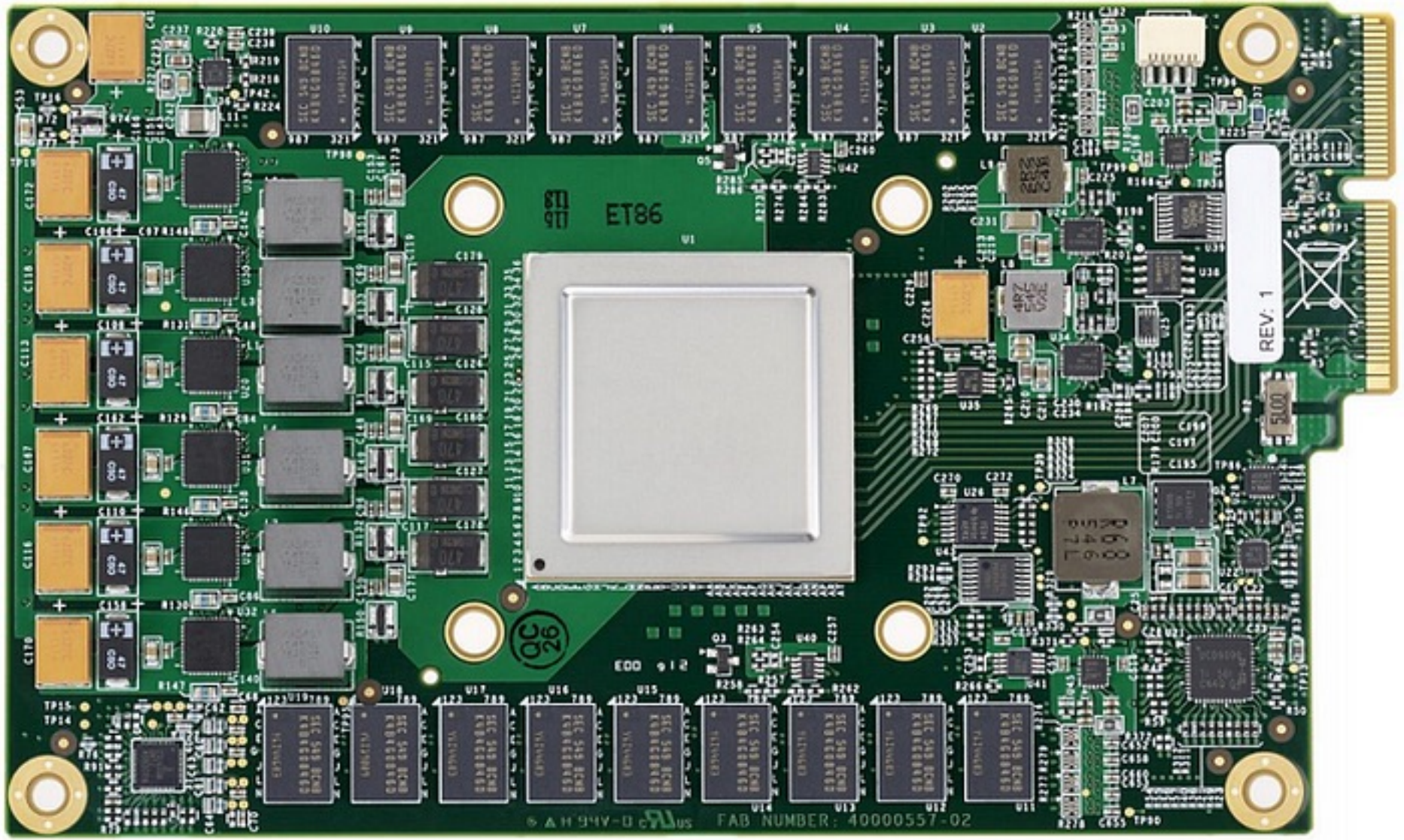
Source: “Wikipedia”

TPU Goals

- **Fast Design**
  - Start in 2014 deployed 15 months later in datacenter
- **Simplify Design:**
  - Rather than integrated with CPU, it was a coprocessor on the PCIe I/O bus
  - Allowing it to plug into existing servers just as a GPU does.
  - Receive instructions from CPU rather than fetching them directly



Source: “ccboot.com”



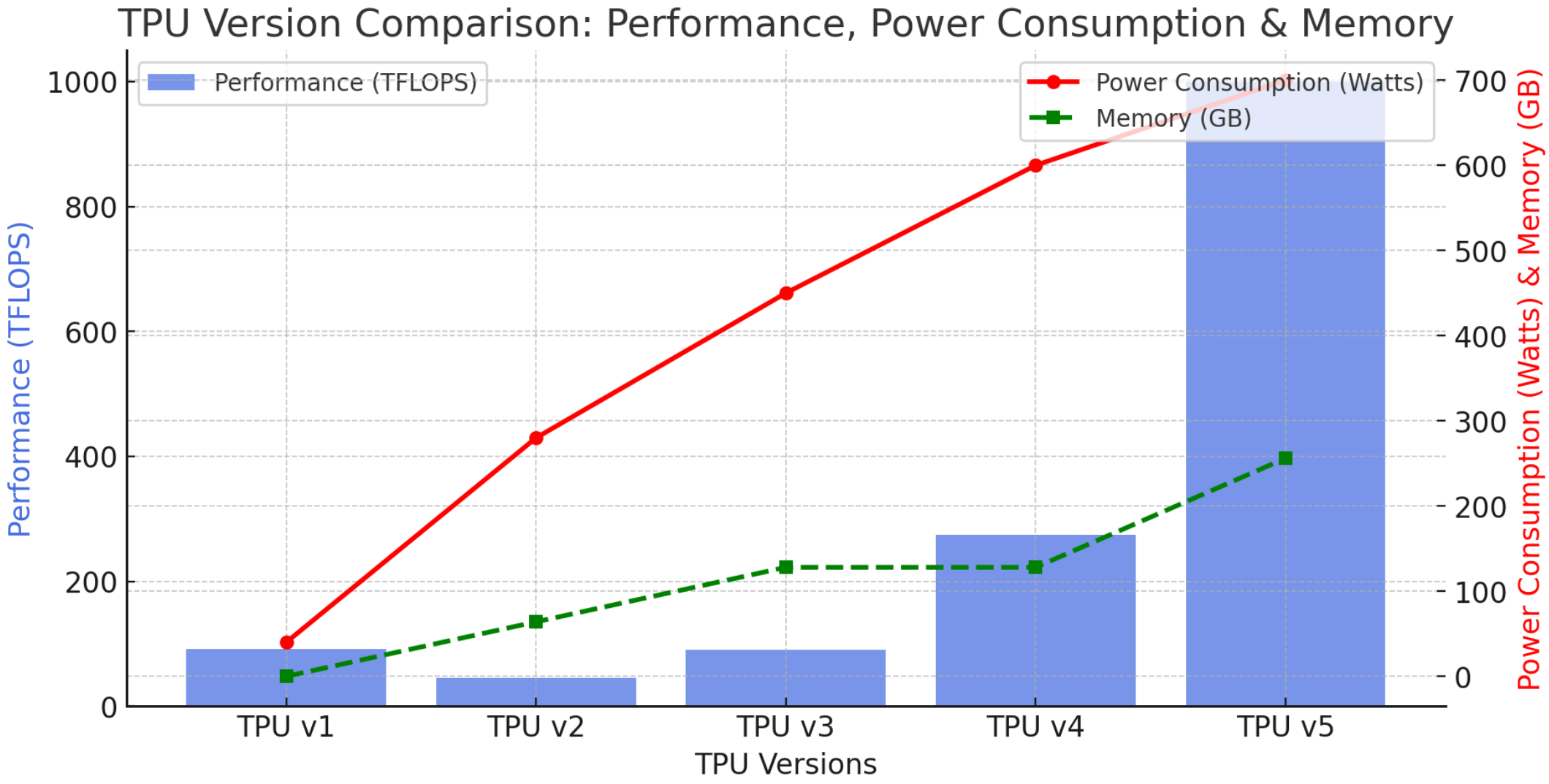
Google TPU v1 printed circuit card that fits into a SATA hard disk slot for drop-in installation.

Source: “In-Datacenter Performance Analysis of a Tensor Processing Unit”, Google, ISCA’17

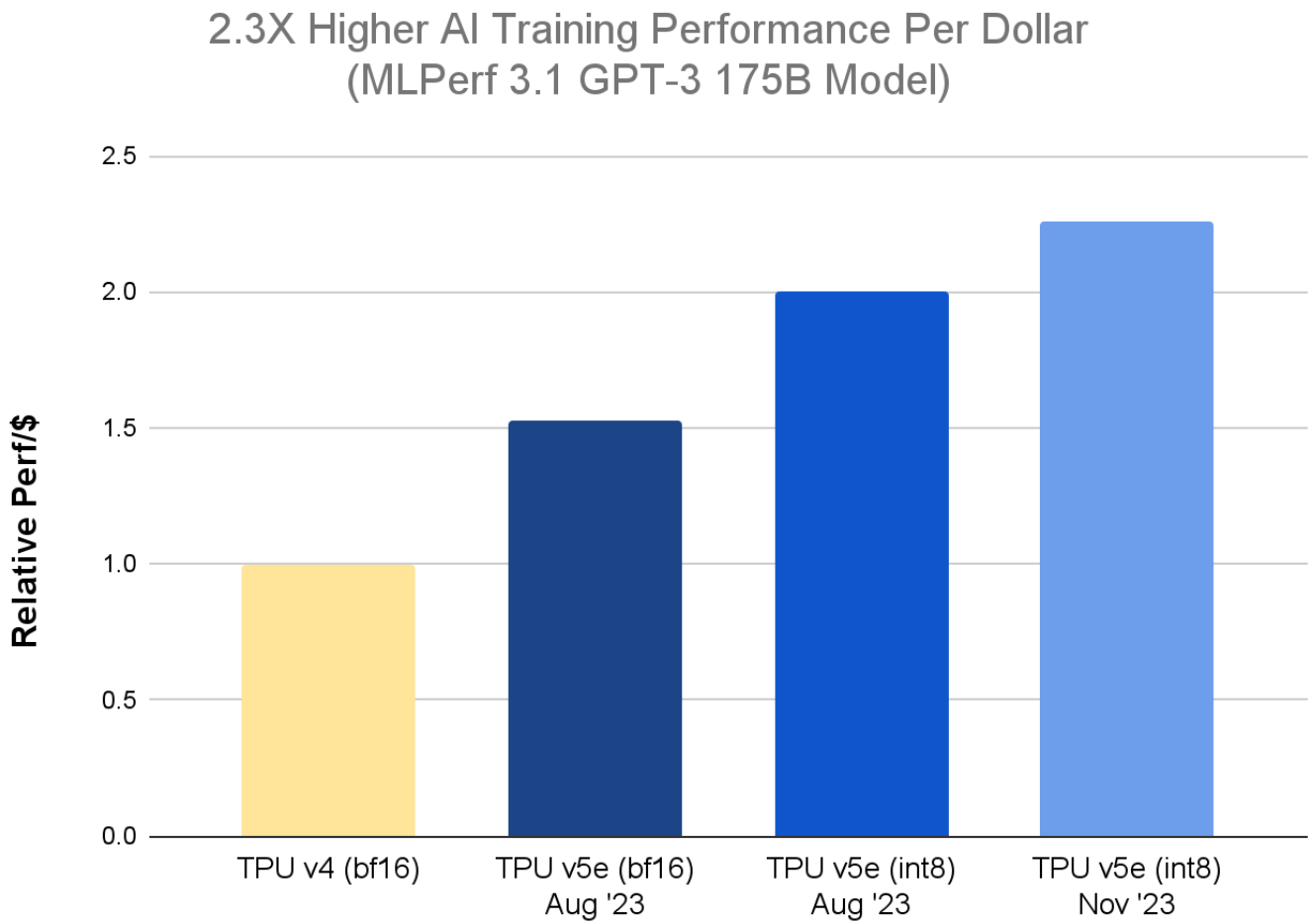
TPU Version Comparison

Feature	TPU v1	TPU v2	TPU v3	TPU v4	TPU v5
Release Year	2015 (announced 2016)	2017	2018	2021	2023
Interface	PCIe	PCIe	PCIe	Optical interconnect (TPU pod)	Optical interconnect (TPU pod)
Peak Performance (TOPS)	92	45 TFLOPS (per chip)	90 TFLOPS (per chip)	275 TFLOPS (per chip)	1,000+ TFLOPS (per chip)
Precision	8-bit INT	FP16, BF16	FP16, BF16	BF16, INT8	BF16, INT8
Memory	Unknown	64 GB HBM	128 GB HBM	128 GB HBM	256 GB HBM
Power Consumption	~40W	~280W	~450W	~600W	Unknown
Scalability	Single-chip	TPU Pod (256 TPUs)	TPU Pod (512 TPUs)	TPU SuperPod (4096 TPUs)	TPU SuperPod (1000s of TPUs)
Key Use Case	TensorFlow acceleration	Cloud AI	Larger ML models	Large-scale AI, GPT models	Advanced AI, deep learning at extreme scale

TPU Version Comparison



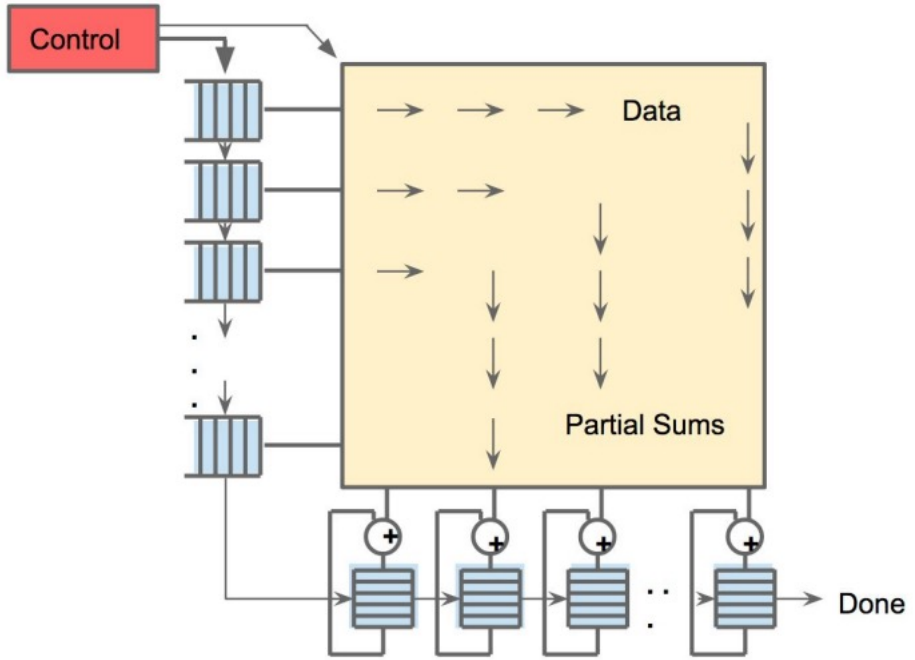
TPU 4 and 5



*MLPerf™ 3.1 Training Closed results for v5e, Google Internal data for TPU v4. As of November, 2023: All numbers normalized per chip seq-len=2048 for GPT-3 175 billion parameter model implemented using relative performance using public list price of TPU v4 ($3.22/chip/hour) and TPU v5e ($1.2/chip/hour).*1*

TPU Design Choices

- **Prioritizing Matrix Operations**
  - TPU v1 features a large matrix multiply unit at its core.
  - Optimized for large batches to maximize throughput.

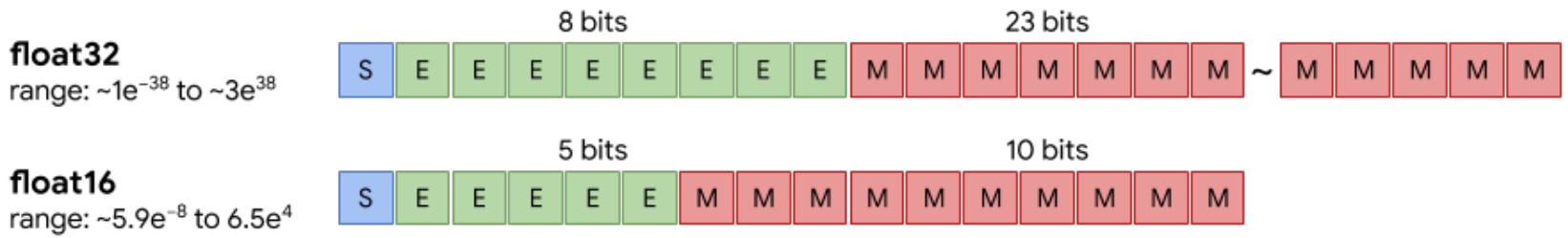


Systolic data flow of the Matrix Multiply Unit.

- Data flows in from the left side.
- Weights are loaded from the top.
- Weights are preloaded.
- Control and data mechanisms are designed to be pipelined.
- This pipelining creates the illusion that all 256 inputs are read simultaneously.
- It also gives the impression that these inputs instantly update one location in each of the 256 accumulators.

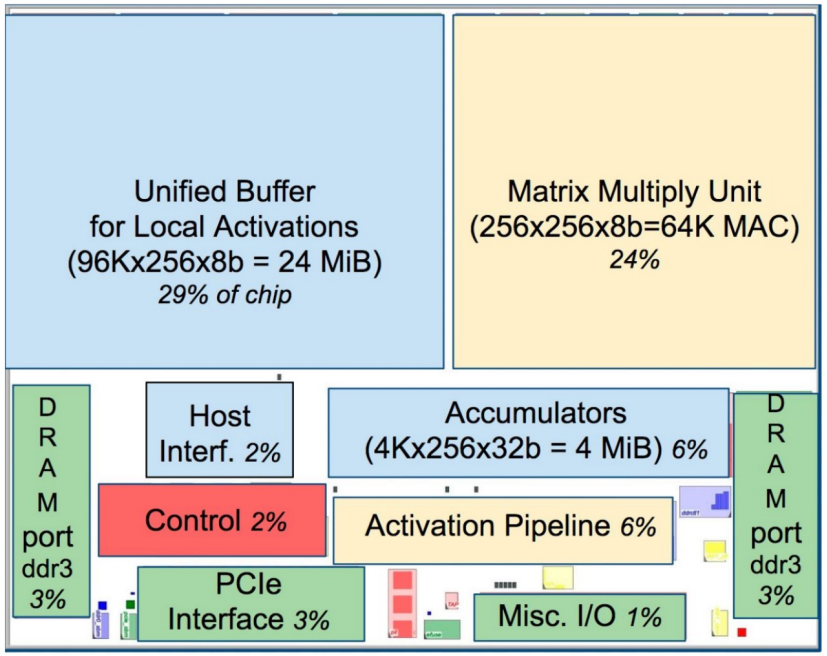
TPU Design Choices

- **Low-Precision Arithmetic**
  - 8-bit Quantized Integers
  - Traditional hardware often uses 32-bit floating points.
  - TPU v1 uses 8-bit quantized integers for computations.
- **Benefits:**
  - Speeds up computation
  - Reduces power consumption

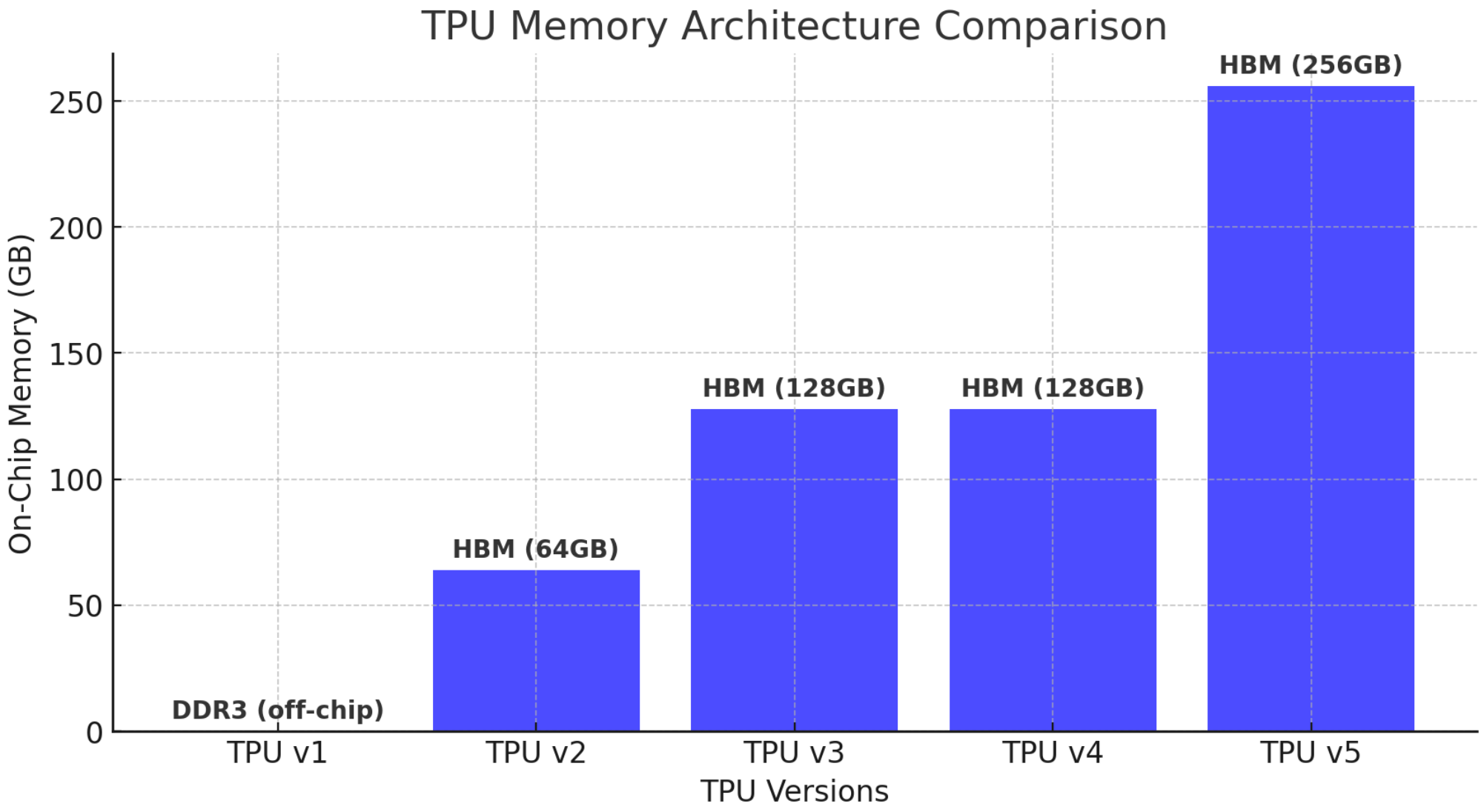


TPU Design Choices

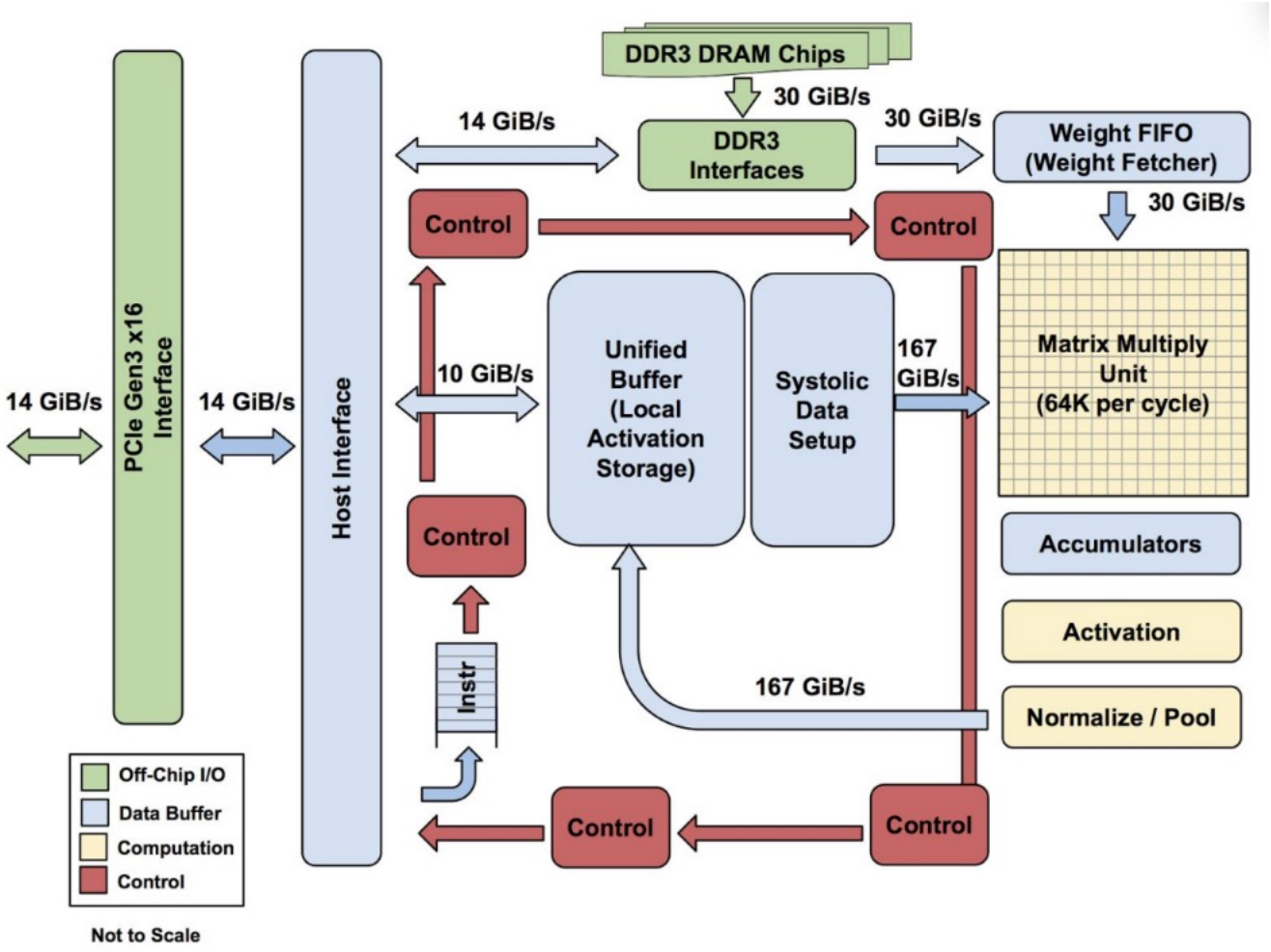
- **On-Chip Memory**
  - TPU v2 incorporates high-bandwidth on-chip memory.
  - Reduces the need to fetch data from external memory.
  - TPU v2 did not have on-chip memory (used for inference)
- **Benefits:**
  - Reducing External Memory Fetches
  - **Decreased latency** and **increased speed**



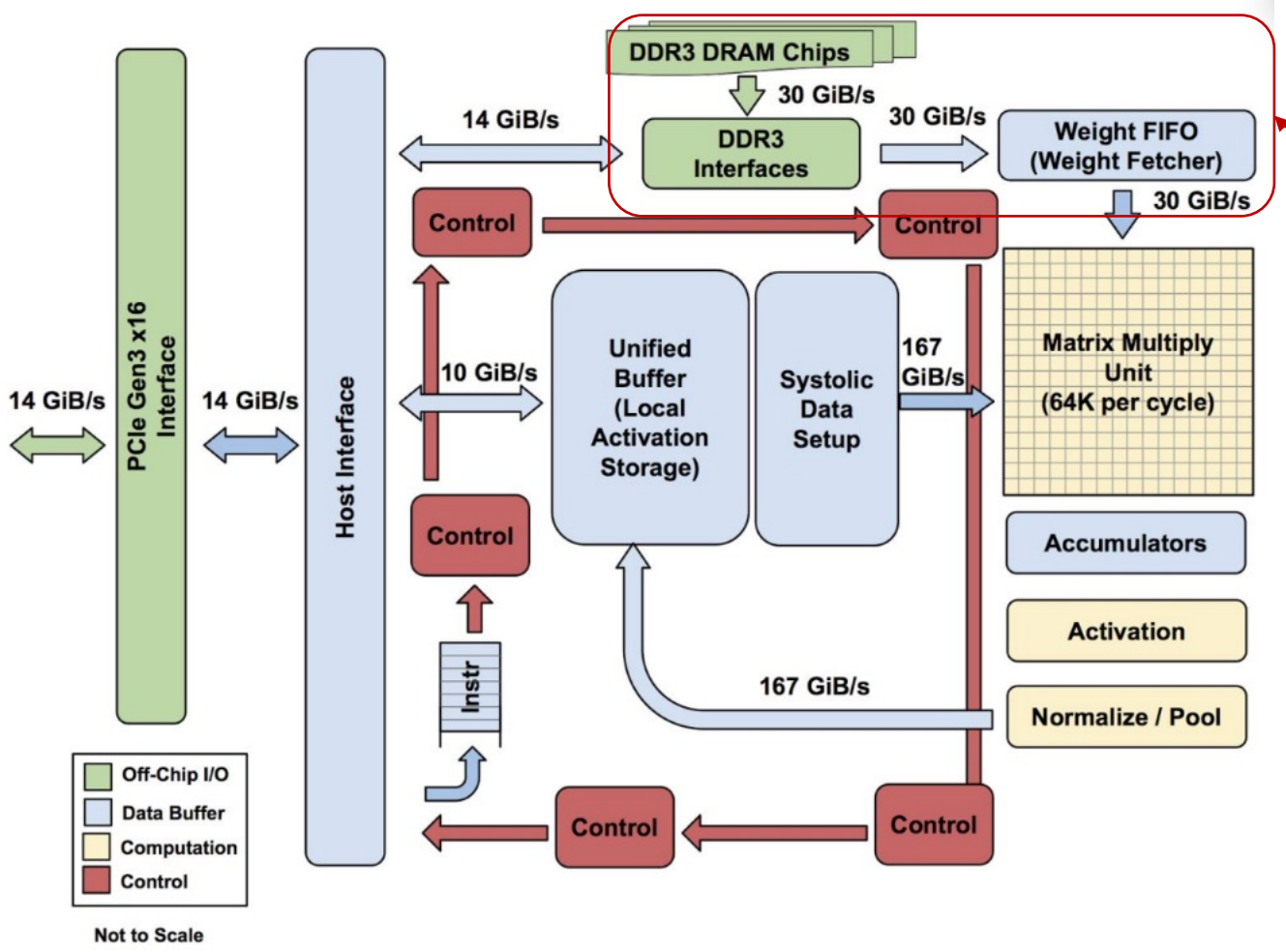
TPU Memory Comparison



TPU 1 Block Diagram

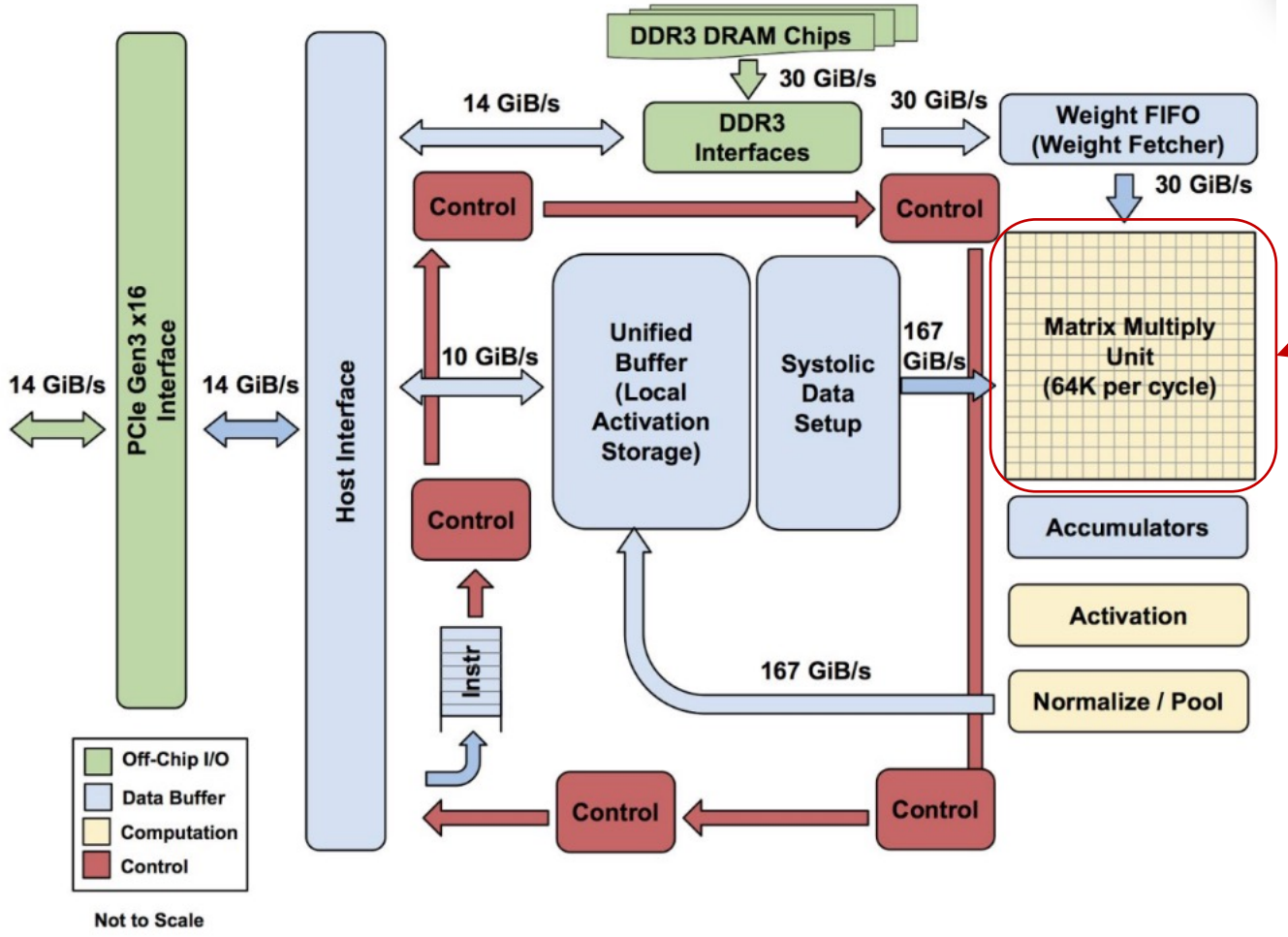


TPU Block Diagram

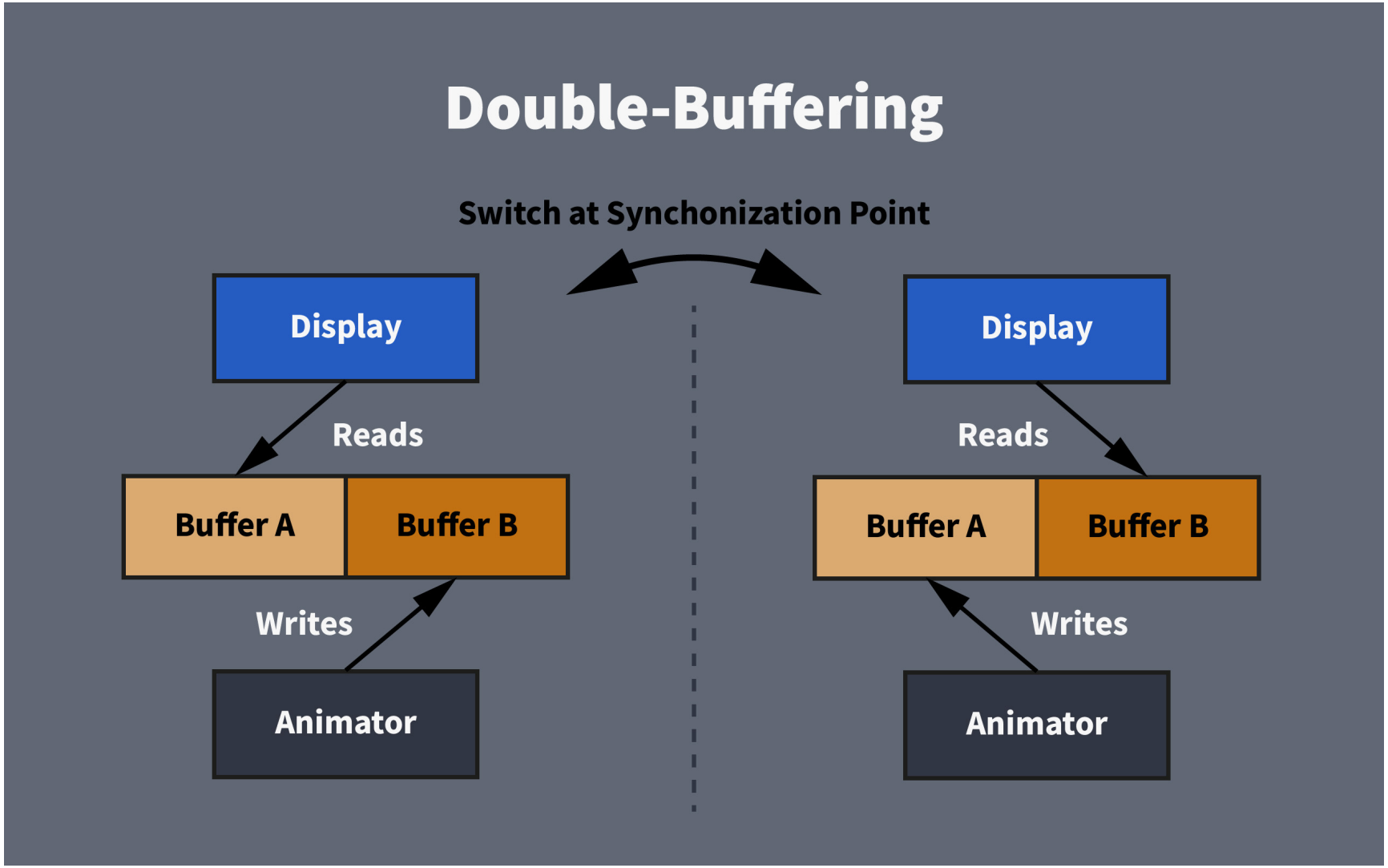


- Off-chip 8 GiB DRAM
  - Support many models simultaneously
- Four tiles deep on chip weight FIFO
- Same weights reused across multiple inputs
  - Weight stationary

TPU Block Diagram

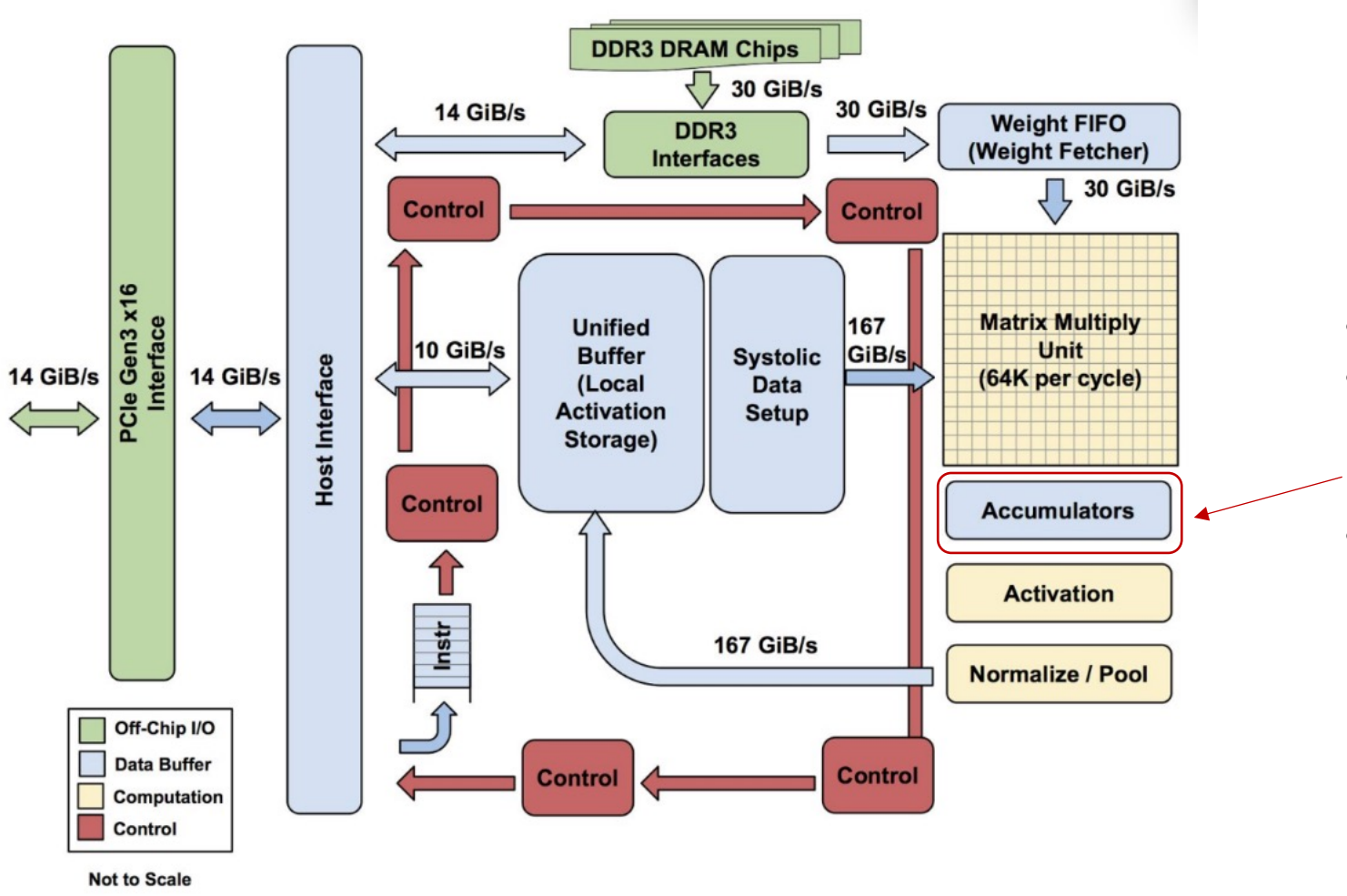


- 256x256 MACs (64K)
- Systolic Array
- **8-bit** integer multiply-and-adds
- Produces one **256-element** partial sum per clock cycle.
- Holds one 64KiB tile of weights plus one for double-buffering (to hide the 256 cycles it takes to shift a tile in).



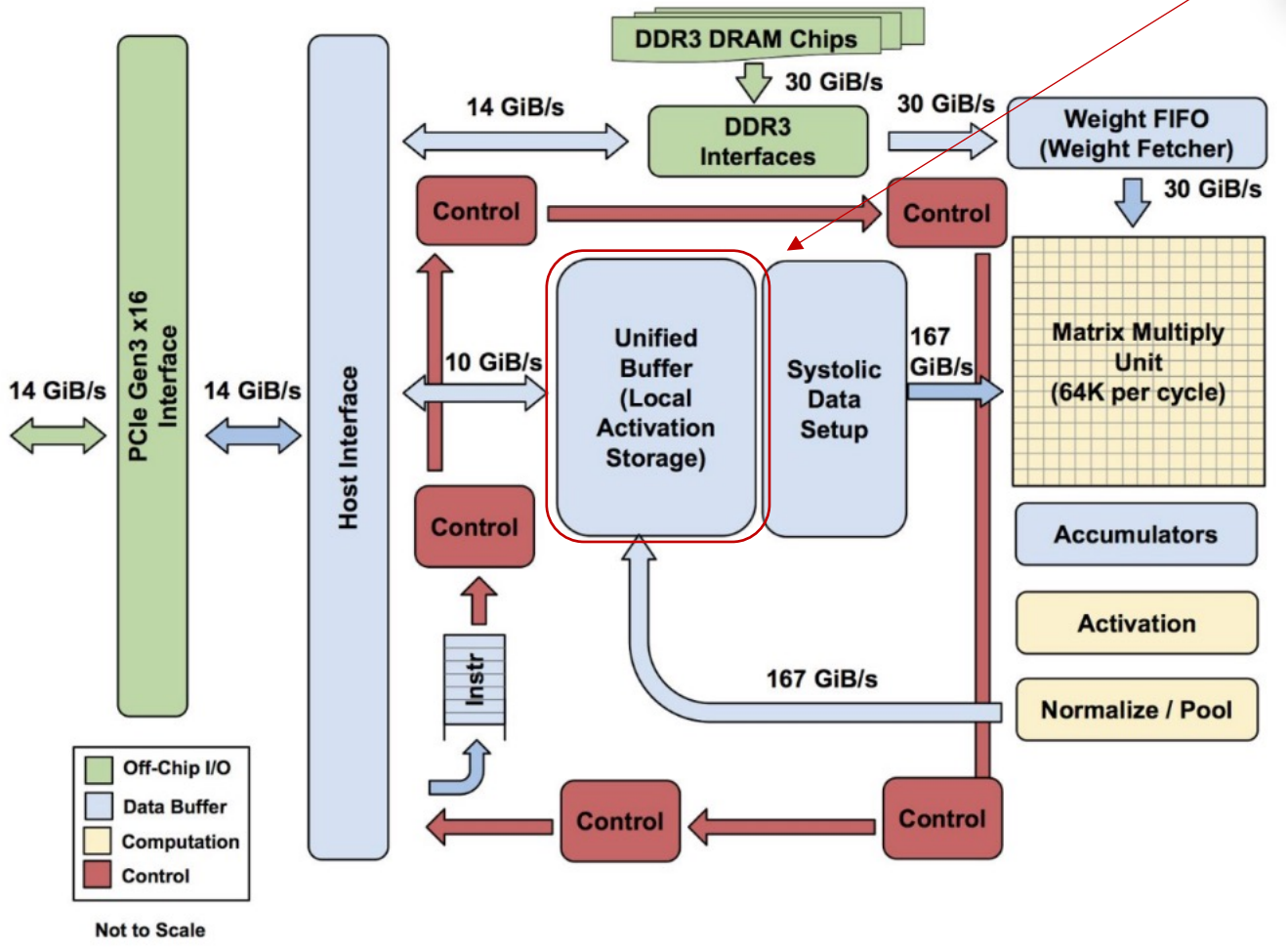
Source: luckyresistor.me

TPU Block Diagram



- 16-bit products are collected here
- 4MiB of 32-bit accumulators
  - 4096, 256-element, 32-bit accumulator
- Why 4MiB?
  - Peak performance observed at 1350 operations/byte
  - Round up to 2048
  - Double for double buffering

TPU Block Diagram

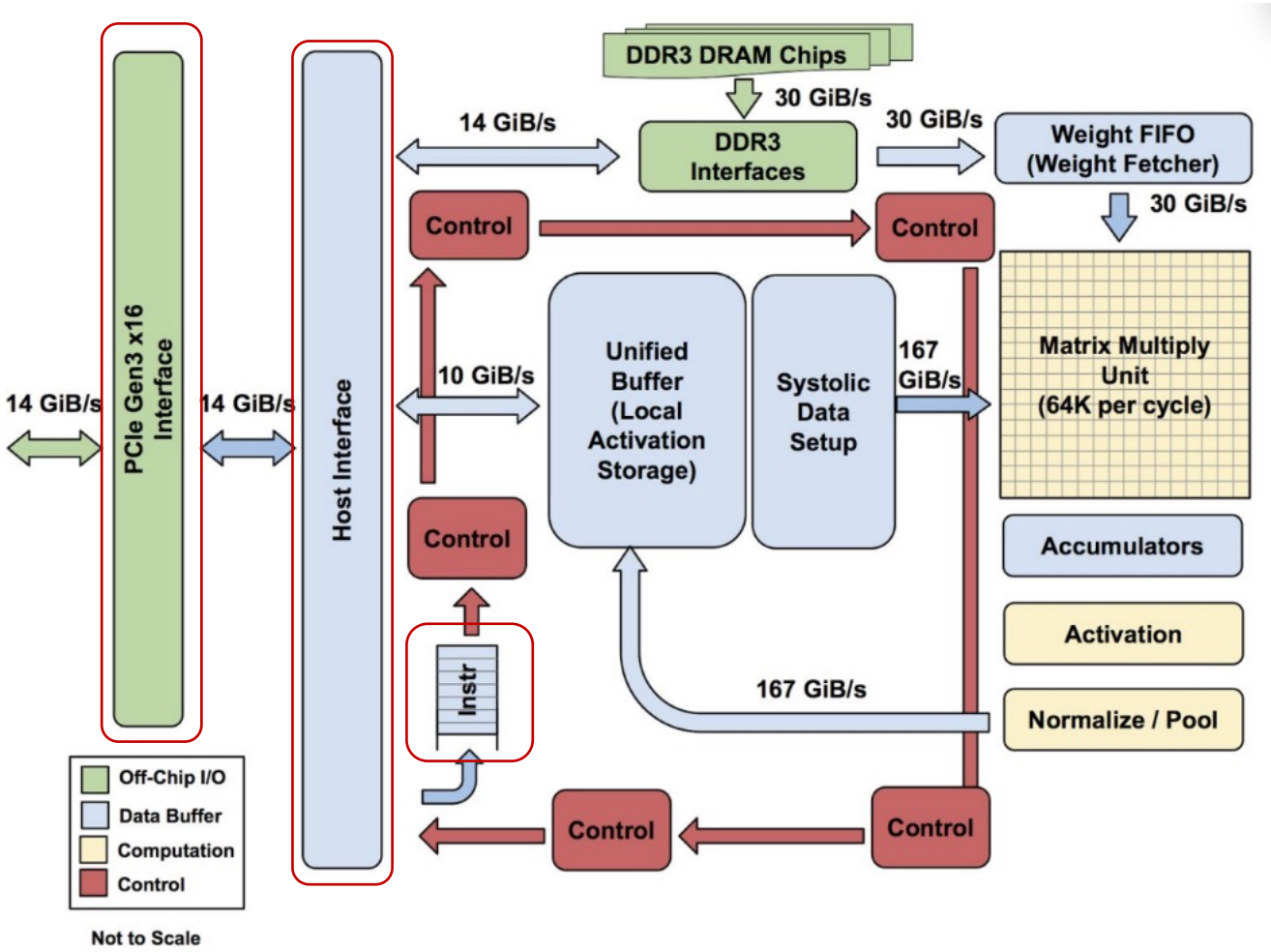


- 24 MiB on-chip for intermediate results
- Inputs to the MMU
- Communicates with host CPU using DMA

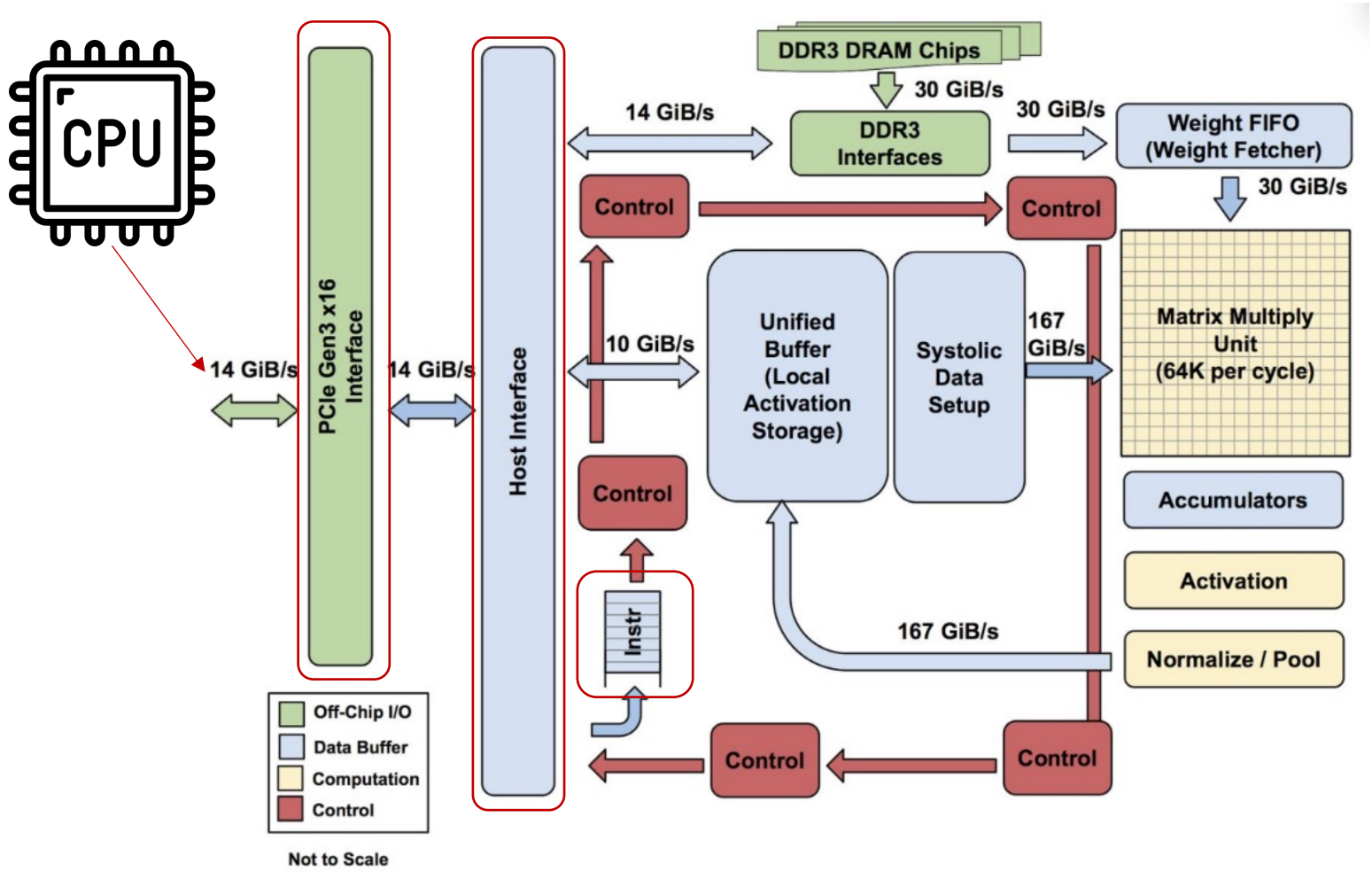
TPU Block Diagram

- PCIe stands for Peripheral Component Interconnect Express.
- PCIe has a data rate of 16GB/s

Feature	**PCIe Interface (Gen3 x16)**	**Host Interface**
Purpose	Transfers data/model weights	Handles control and coordination
Bandwidth	16 GB/s bidirectional	Lower than PCIe
Function	High-speed communication	Task management & synchronization
Used for	Moving large AI data sets	Ensuring TPU execution stability



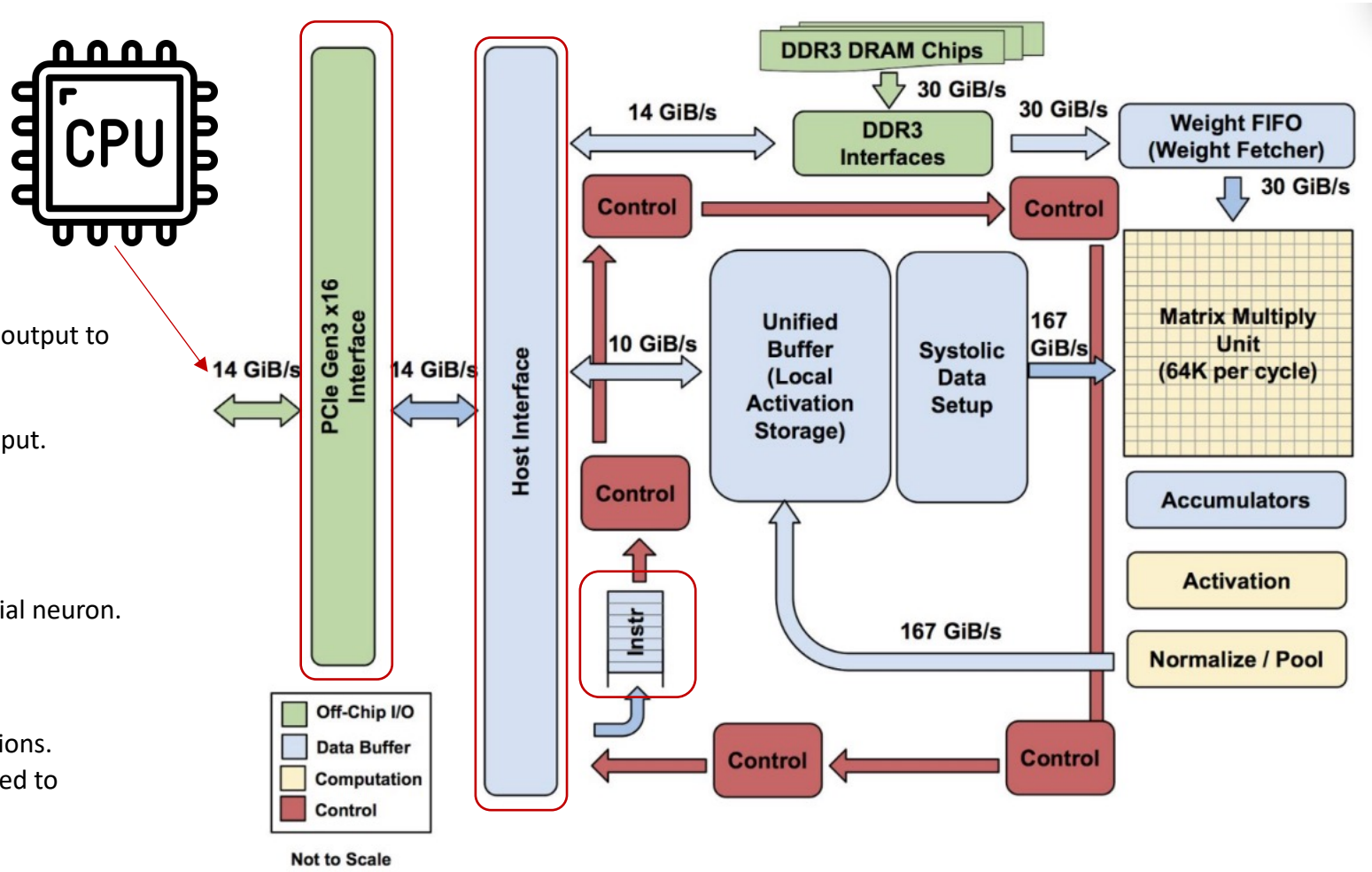
TPU Block Diagram



- TPU instructions are sent from the host into an instruction buffer
- CISC Instructions with average CPI 10 to 20

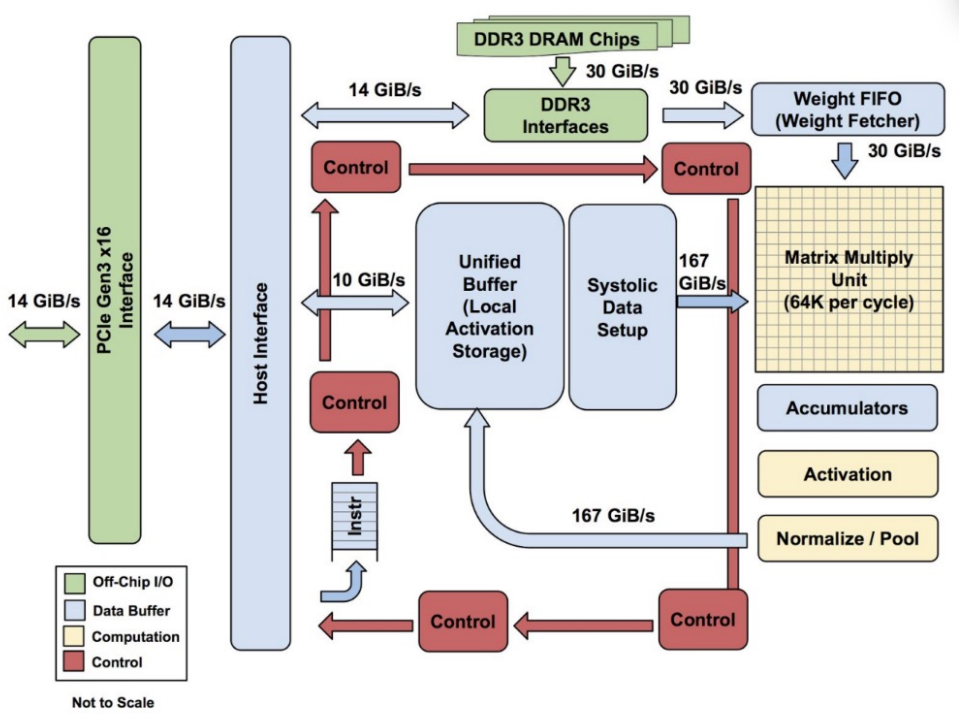
TPU Block Diagram

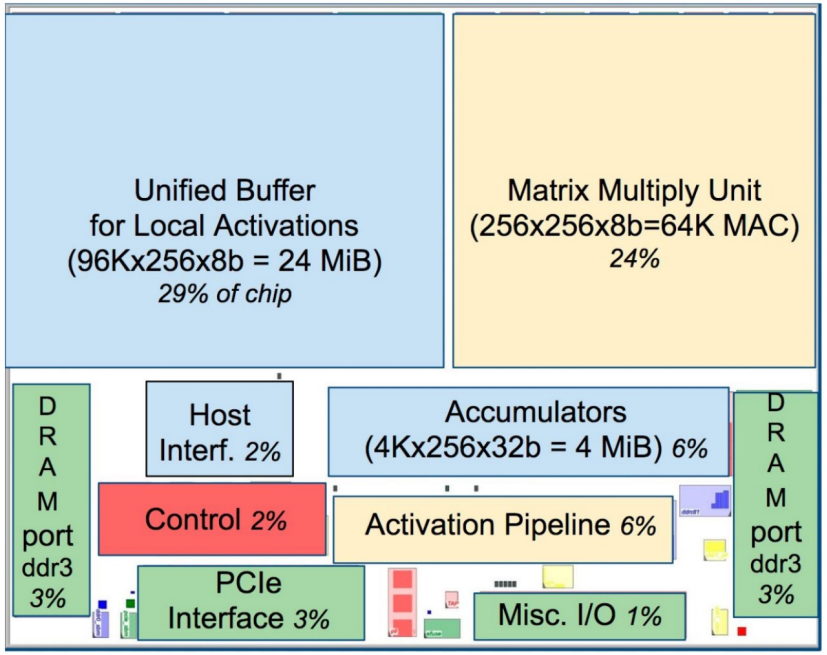
- **Read_Host_Memory**
  - Initial step to fetch data.
- **Read_Weights**
  - Reads weights from Weight Memory.
  - Loads them into the Weight FIFO.
  - Serves as input to the Matrix Unit.
- **MatrixMultiply/Convolve**
  - Performed by the Matrix Unit.
  - Uses data from the Unified Buffer and sends output to the Accumulators.
  - Takes a variable-sized B*256 input.
  - Multiplies it by a 256x256 constant weight input.
  - Produces a B*256 output.
  - Requires B pipelined cycles to complete.
    - **(B is batch size)**
- **Activate**
  - Executes the nonlinear function of the artificial neuron.
  - Options include ReLU, Sigmoid, etc.
  - Inputs sourced from the Accumulators.
  - Outputs to the Unified Buffer.
  - Can perform pooling operations for convolutions.
  - Uses dedicated hardware on the die connected to nonlinear function logic.
- **Write_Host_Memory**
  - Writes data from the Unified Buffer.
  - Sends it to the CPU host memory.



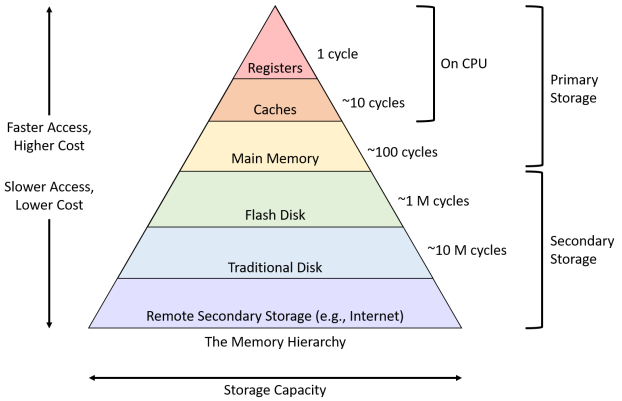
- TPU instructions are sent from the host over the PCIe into an instruction buffer
- CISC Instructions with average CPI (Cycles per instruction) 10 to 20

TPU Block Diagram and Floorplan





Memory Hierarchy



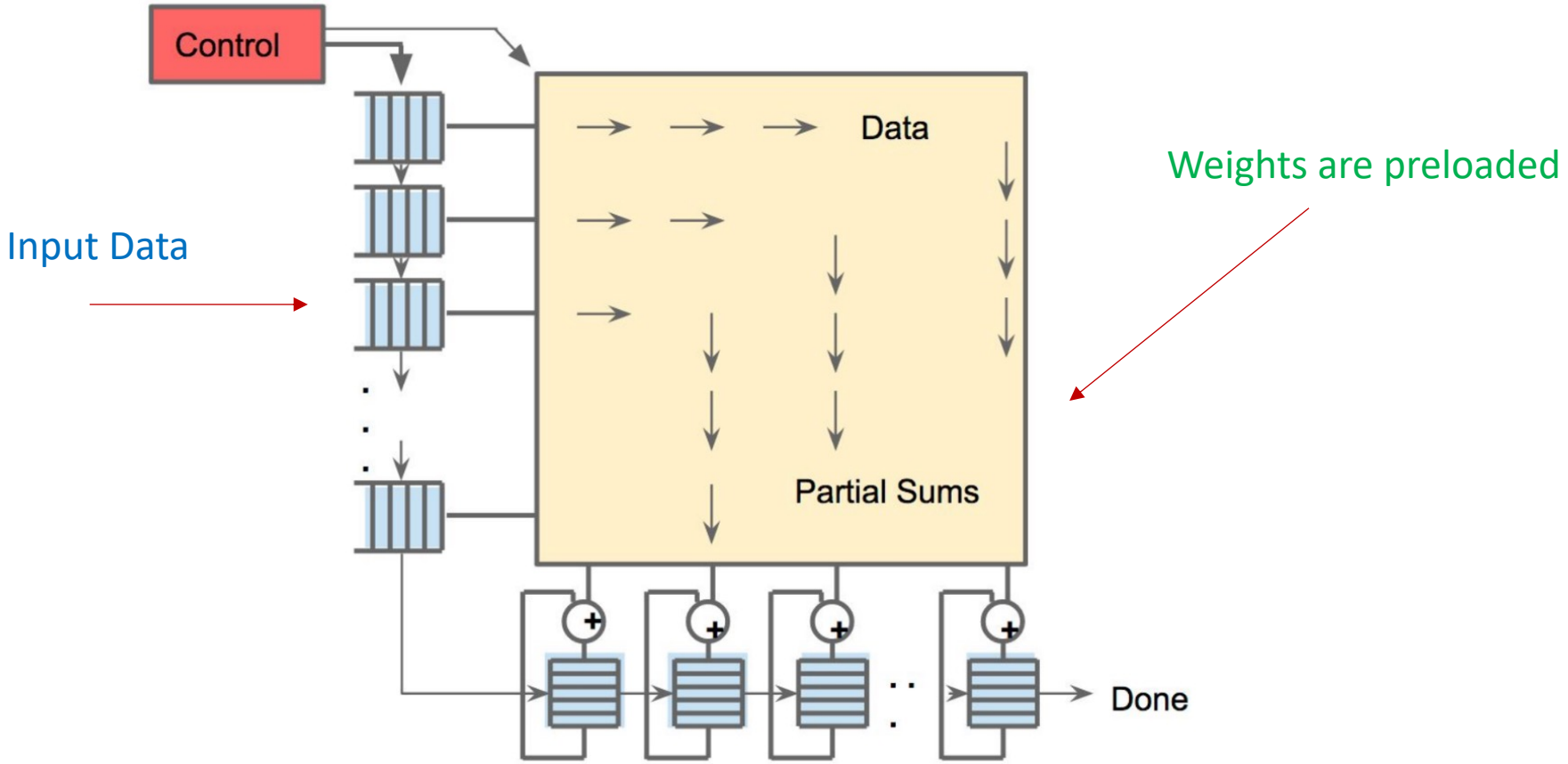
CPU, Implicit

TPU, Explicit

The CISC **MatrixMultiply** instruction is 12 bytes

	Bytes
Unified Buffer Address	3
Accumulator Address	2
Length	2-4
Opcode and Flag	The rest

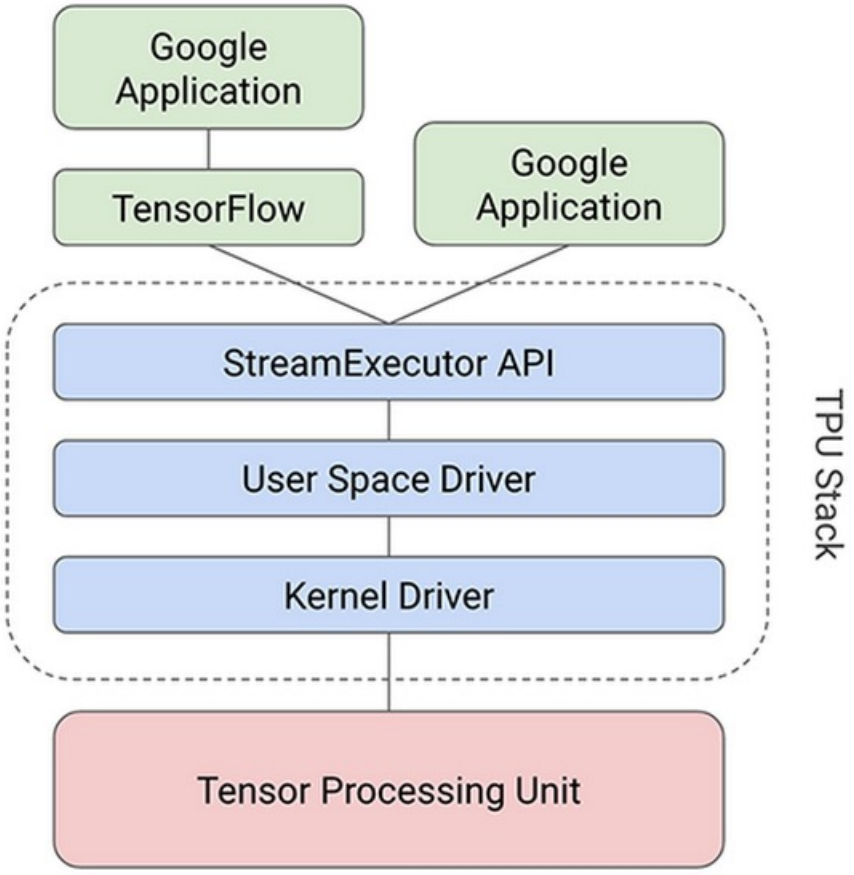
Systolic Array



Software has the illusion that each 256B input is read at once, and they instantly update one location of each of 256 accumulator RAMs.

TPU Application Execution

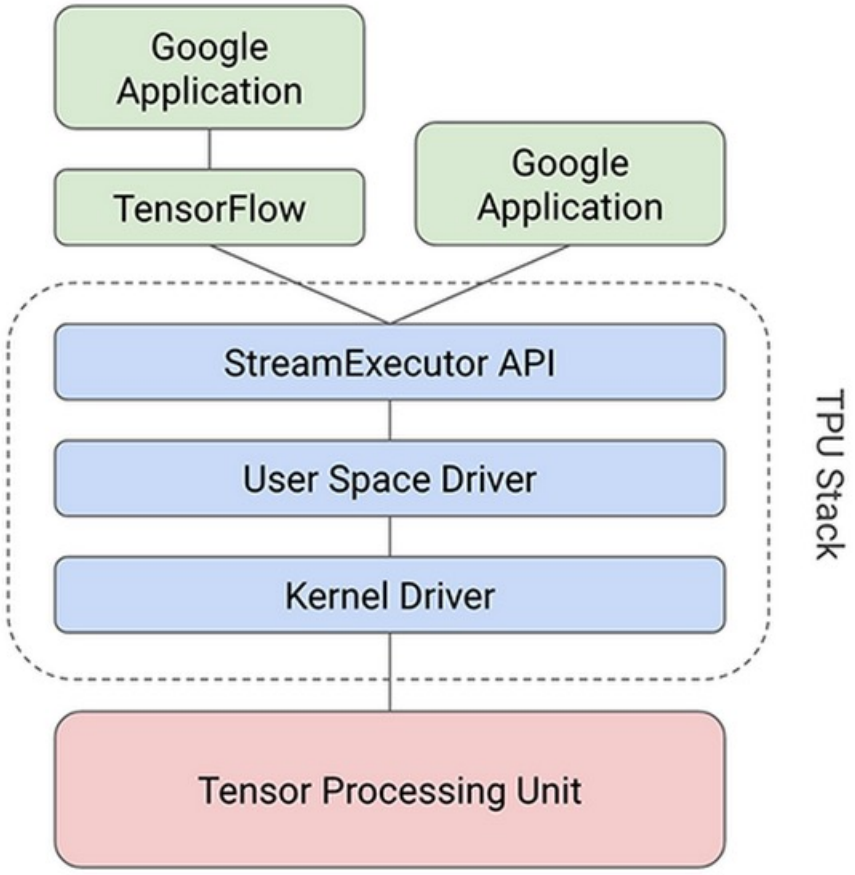
- Applications on TPU typically written in TensorFlow.
- Compiled into an API compatible with GPUs or TPUs.
- TPU stack comprises of:
  - User Space Driver
  - Kernel Driver
  - StreamExecutor API: This is an API (Application Programming Interface) layer that provides a higher level of abstraction for programming the TPU.
    - It allows the software above it to execute commands on the TPU without having to deal directly with the intricacies of the hardware.



Source: Google cloud blog

Distinguishing Kernel Driver and User Space Driver

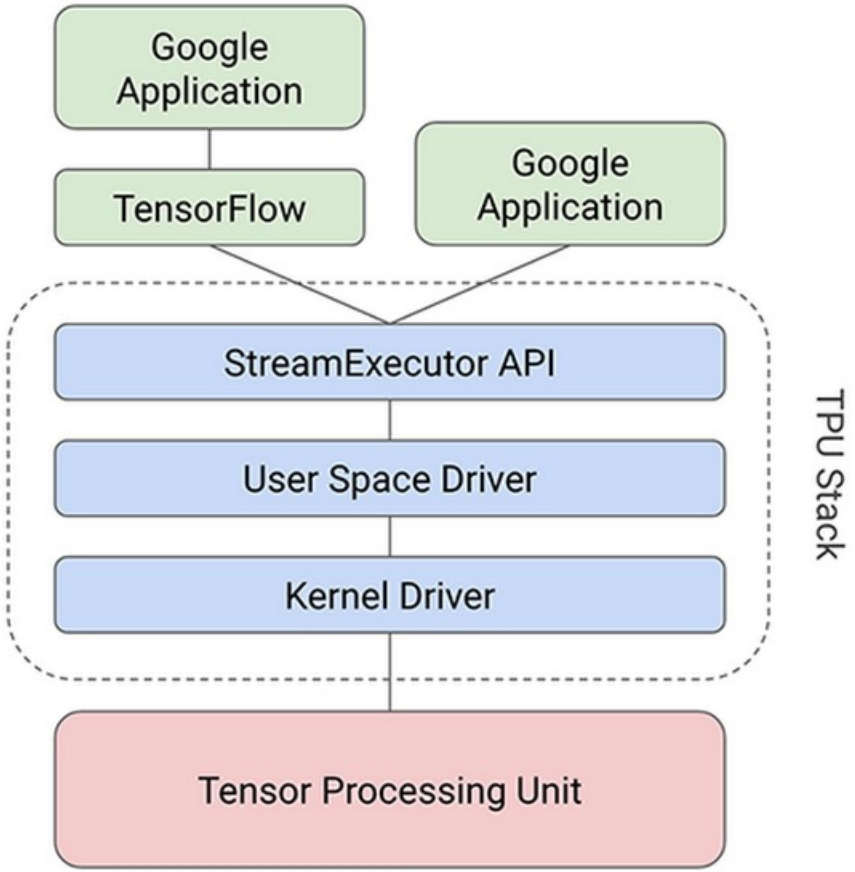
- **Kernel Driver:**
  - Lightweight component.
  - Manages memory and handles interrupts.
  - Designed for long-term stability.
    - Robust
    - Compatibility
    - Limited updates
    - Support commitment
- **User Space Driver:**
  - More dynamic, changes frequently.
  - Sets up and controls TPU execution.
  - Reformats data, translates API calls into TPU instructions.
  - Compiles models and manages TPU's weight memory.
  - Not explicitly detailed in public domain



Source: Google cloud blog

Distinguishing Kernel Driver and User Space Driver

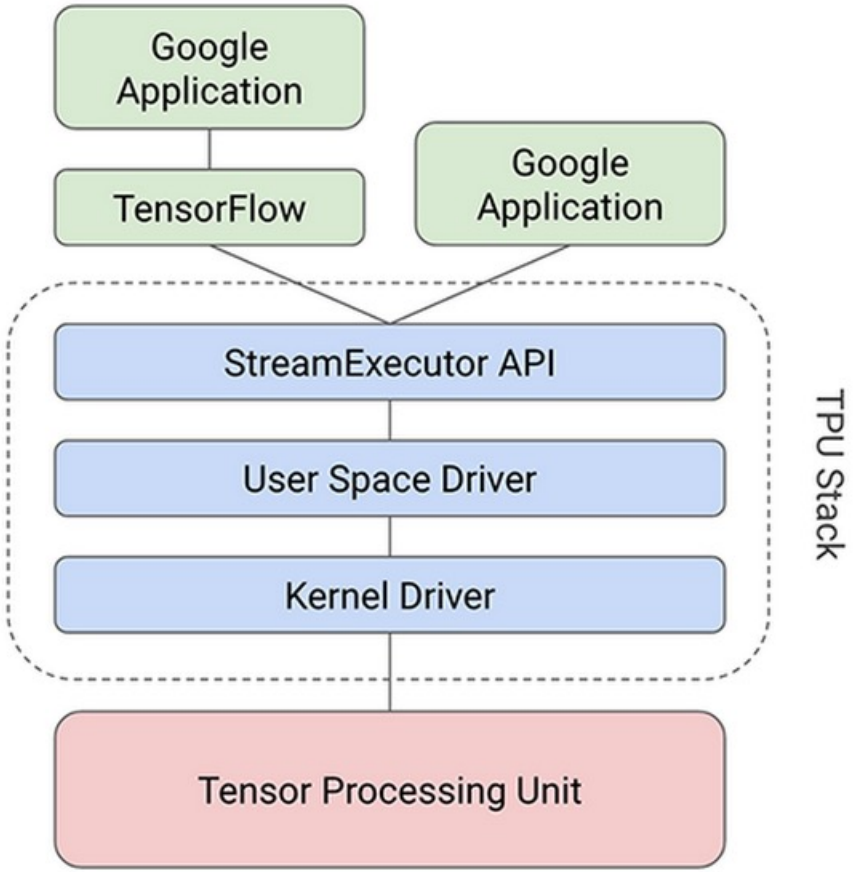
Step	Component	What Happens?
1	**Google App**	Requests AI task (e.g., translate image).
2	**TensorFlow**	Converts AI model to TPU-compatible ops.
3	**StreamExecutor API**	Manages TPU execution and memory.
4	**User Space Driver**	Converts API calls into low-level instructions.
5	**Kernel Driver**	Sends final commands to TPU.
6	**TPU Hardware**	Processes AI task using its systolic array.
7	**User Space Driver**	Collects TPU results.
8	**Google App**	Receives output and displays results.



Source: Google cloud blog

From Compilation to Full-Speed Execution

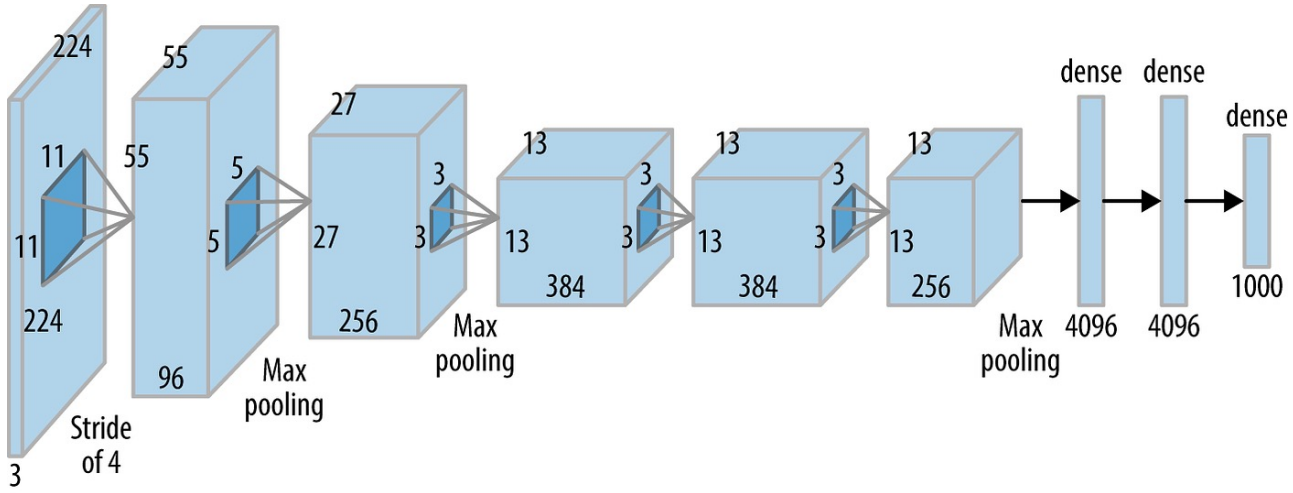
- Initial model evaluation:
  - Compilation of the model.
  - Caching of the program image.
  - Writing weight image into TPU's weight memory.
- Subsequent evaluations run at maximum speed.



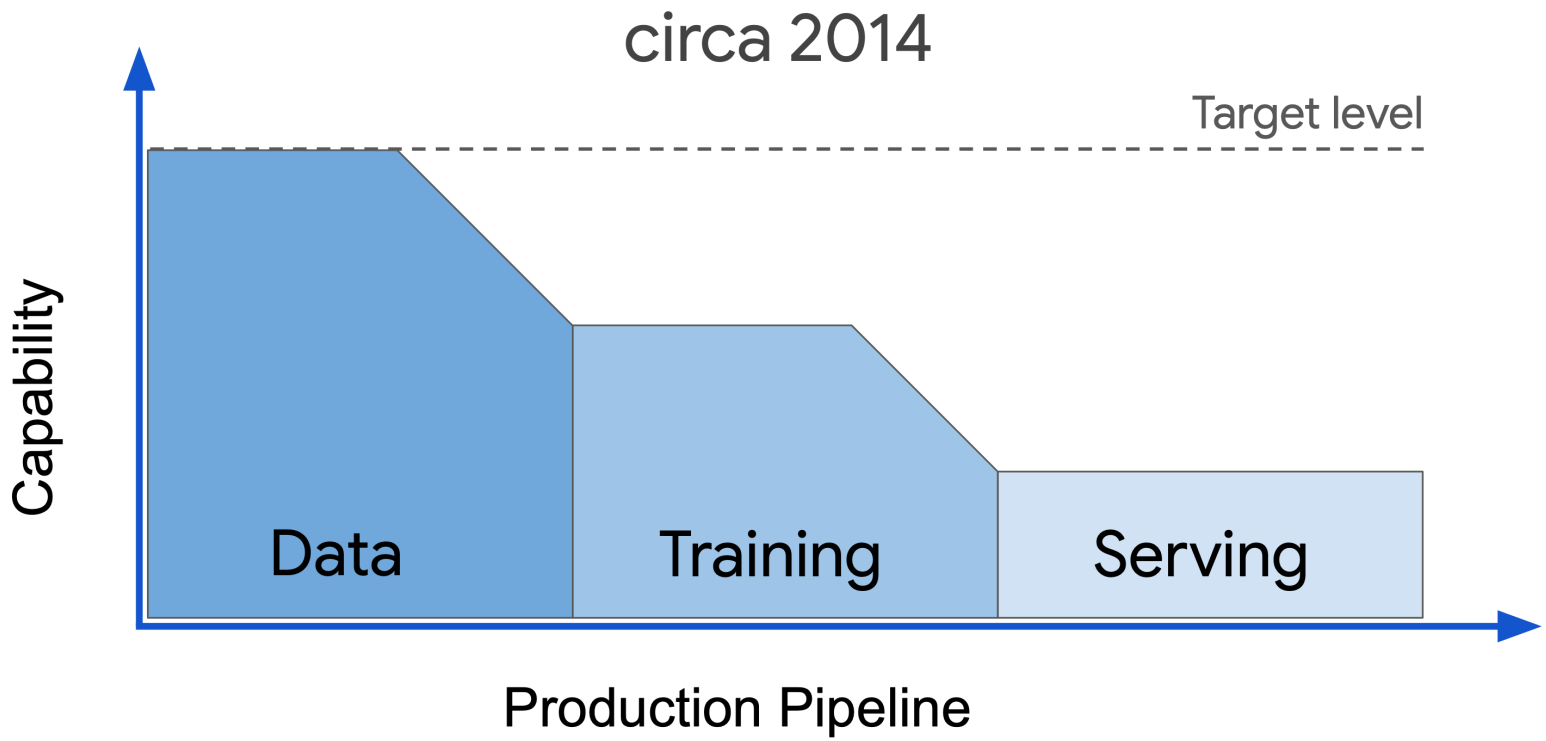
Source: Google cloud blog

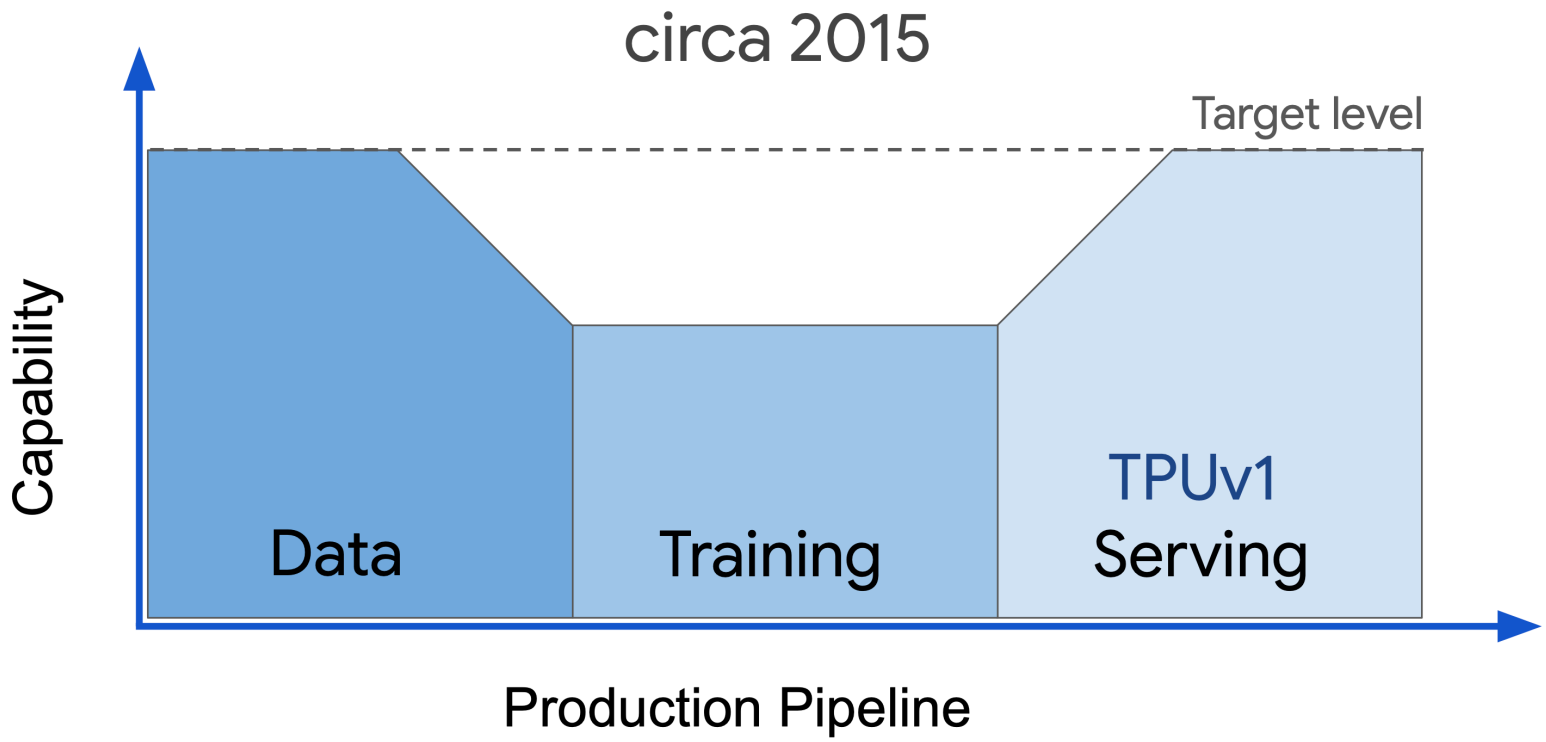
Efficient Computation on TPU

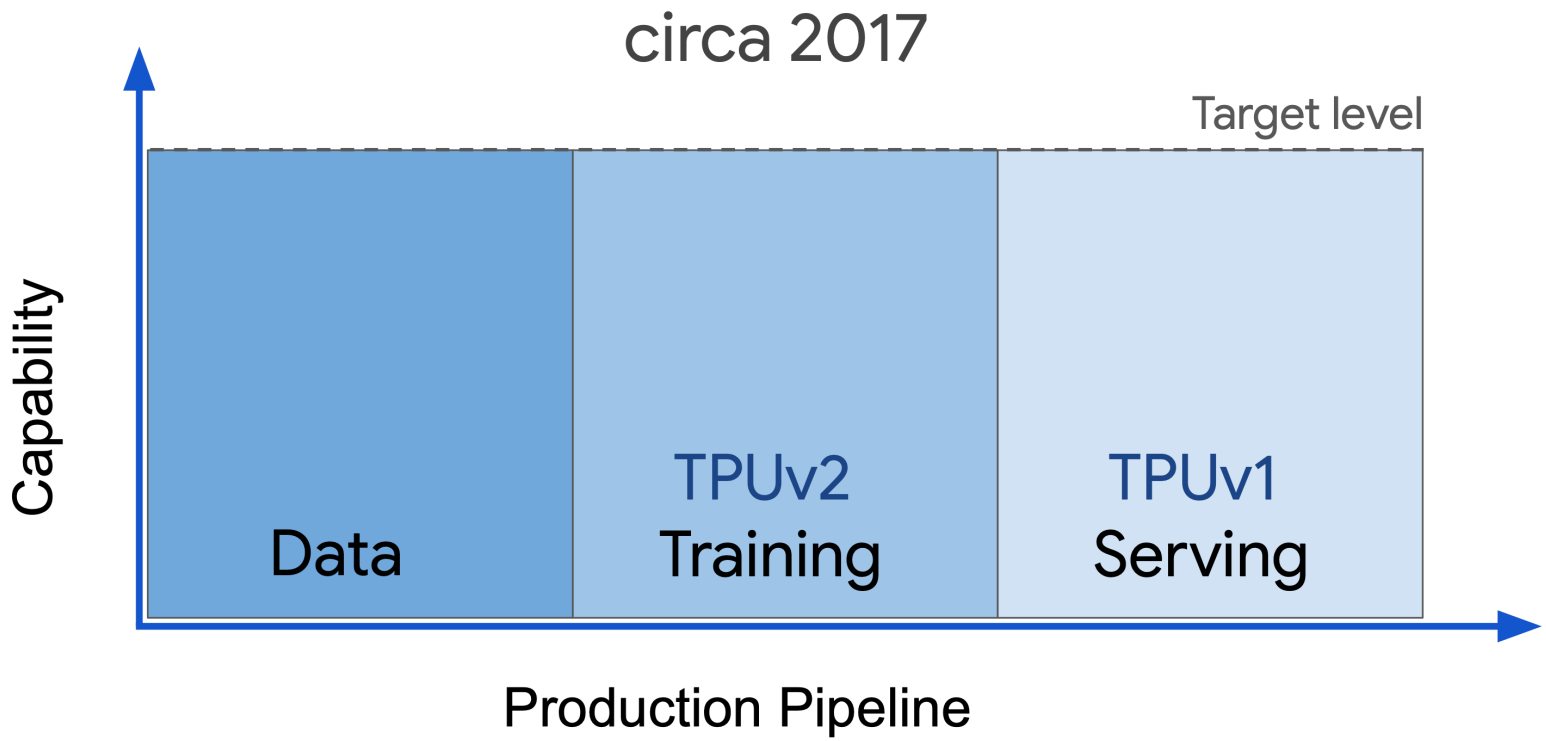
- TPU processes most models from start to finish (inputs to outputs).
  - Aims to maximize TPU compute time relative to I/O time.
  - Computation often segmented by layers.
  - Overlapped execution: Matrix multiply unit hides most non-critical-path operations.



What About Training?


circa 2014
Target level
Capability
Data
Training
Serving
Production Pipeline

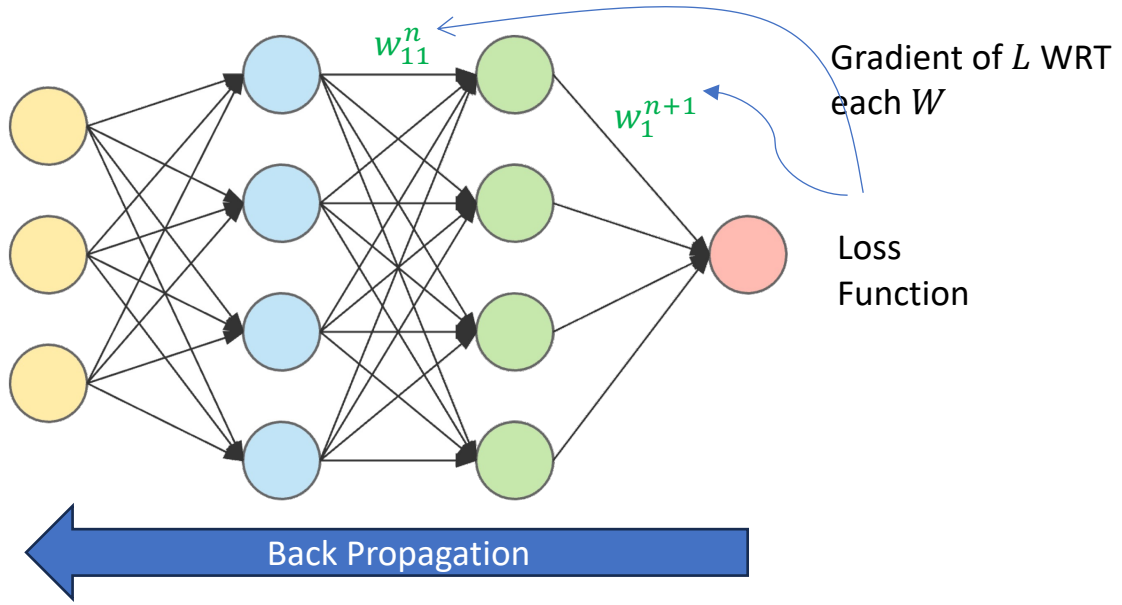

circa 2015
Target level
Capability
Data
Training
TPUv1
Serving
Production Pipeline


circa 2017
Target level
Capability
Data
TPUv2
Training
TPUv1
Serving
Production Pipeline

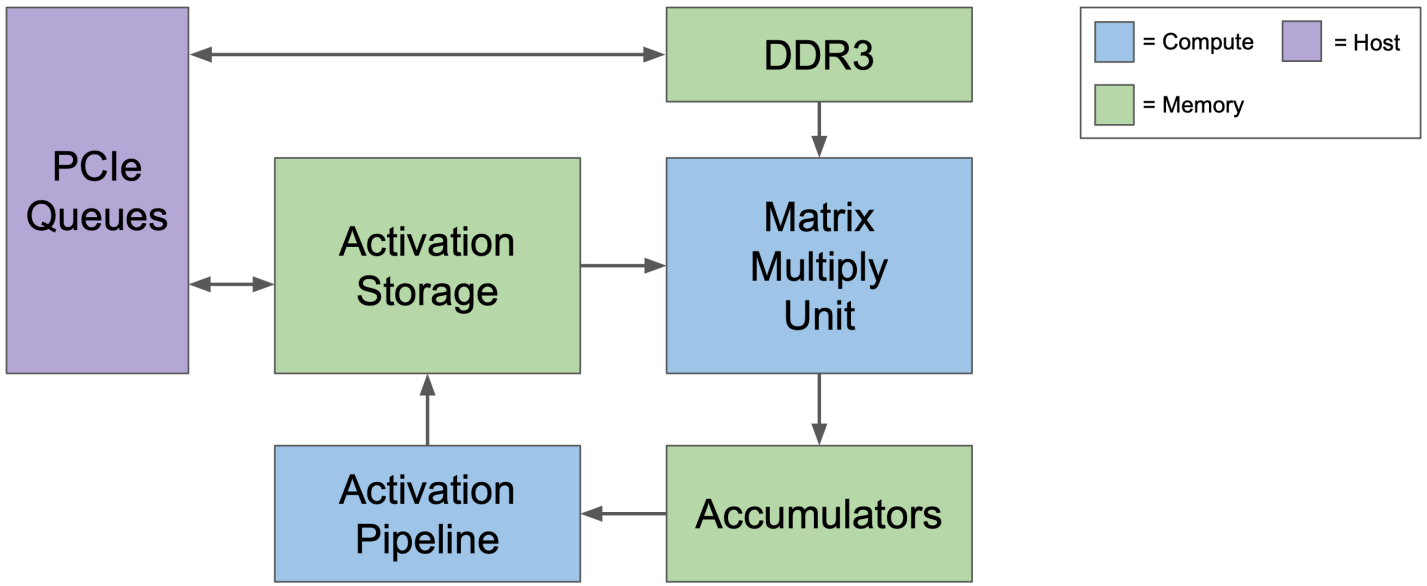
Challenges of ML Training

- More **Computation**
  - Backprop, transpose, derivatives
- More **Memory**
  - Keep data around for backprop
- Wider **Operands**
  - Need dynamic range (more than int8)
- More **Programmability**
  - User experimentation, optimizers
- **Parallelization**
  - Scale-up instead of scale-out

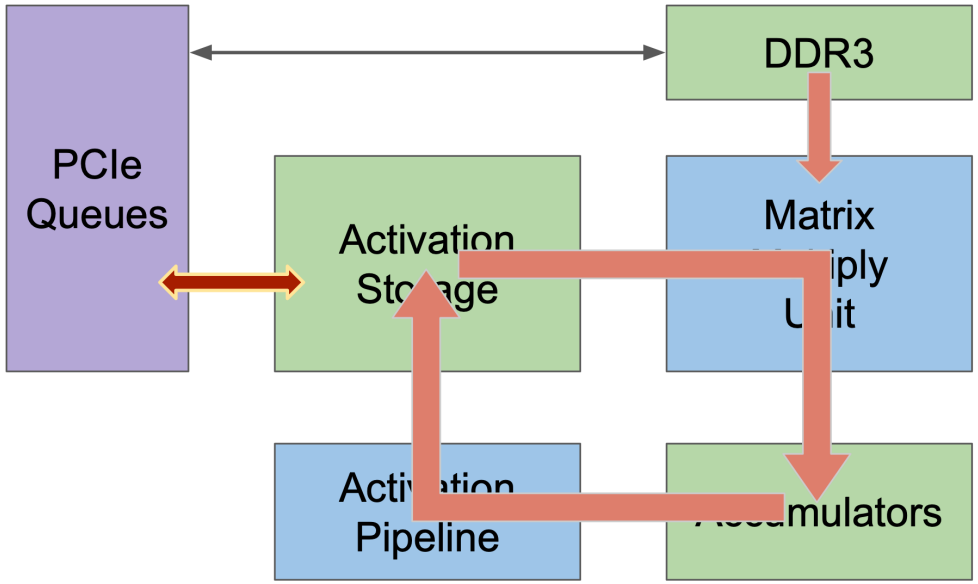
Source:
premioinc.com

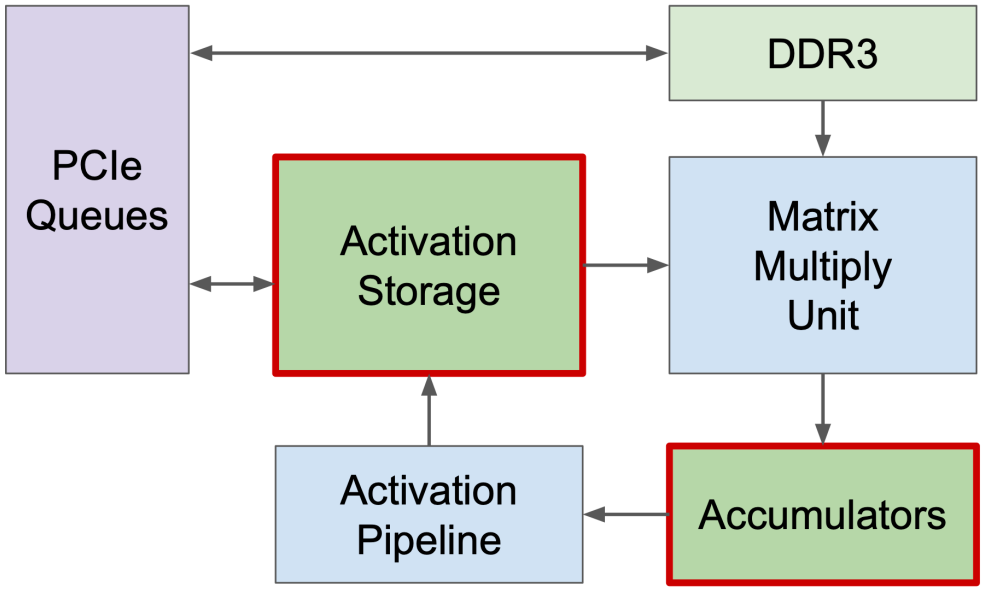


TPUv1 Recap

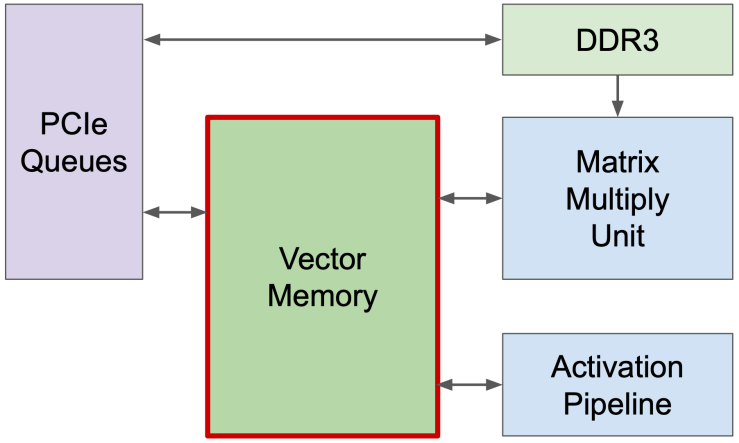

PCIe
Queues
DDR3
Activation
Storage
Matrix
Multiply
Unit
Activation
Pipeline
Accumulators
= Compute
= Host
= Memory

TPUv1 Recap

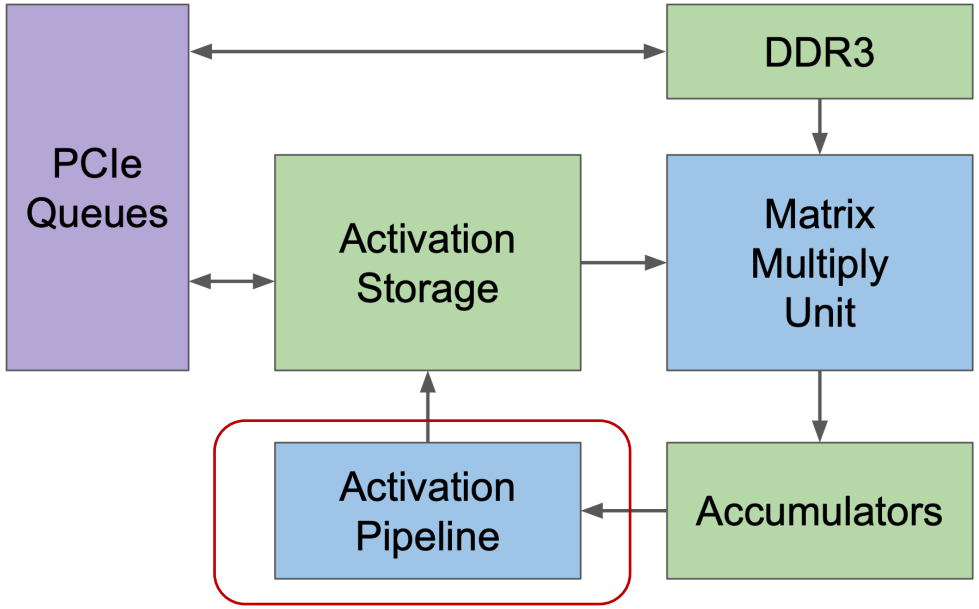

DDR3
PCIe
Queues
Activation
Matrix
Pipeline
Accumulators



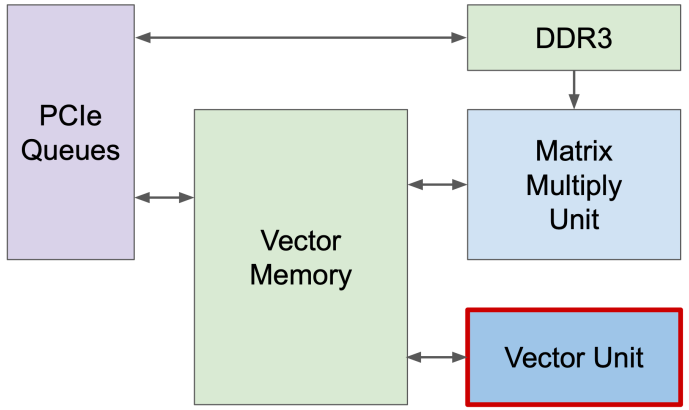
TPUv2 Changes



Single vector memory instead of buffers between fixed function units.



TPUv2 Changes

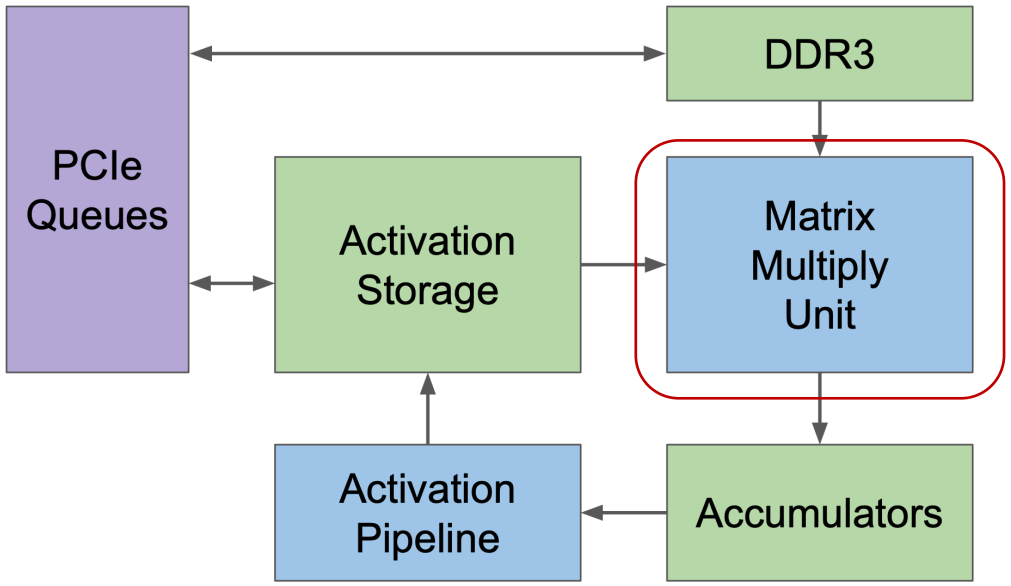


General purpose vector unit instead of a fixed function activation pipeline.

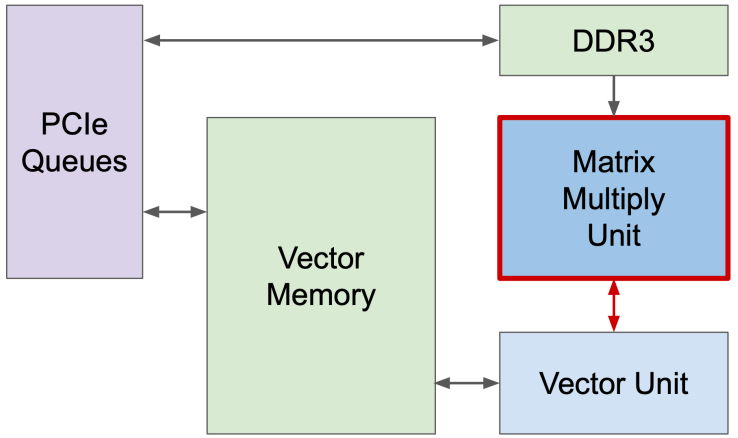
Operations like activations, pooling, and certain element-wise operations can be handled efficiently by the VPU.

Vector Unit vs. Matrix Multiply Unit (MXU)

Feature	**Matrix Multiply Unit (MXU)**	**Vector Unit**
Primary Function	Matrix multiplications (MatMul)	Element-wise operations (e.g., ReLU, Softmax)
Execution Type	Large matrix processing using **128×128 systolic array**	Scalar, vector, and element-wise operations
Common Use Cases	Dense layer multiplications, Convolutions	Activations, Normalization, Loss functions
Parallelism	Highly parallel (fixed-size systolic array)	Parallel, but processes individual elements



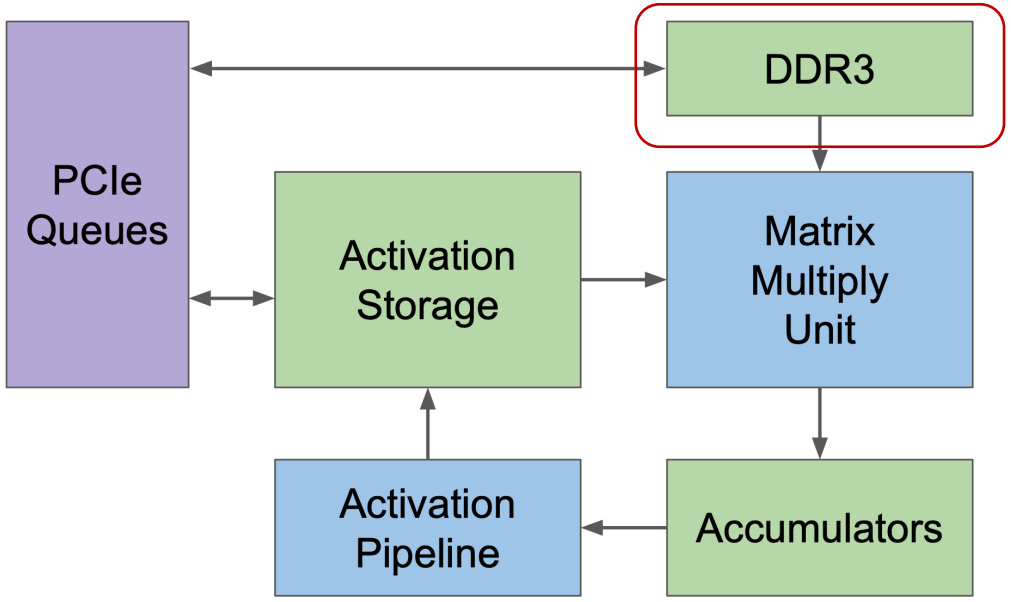
TPUv2 Changes



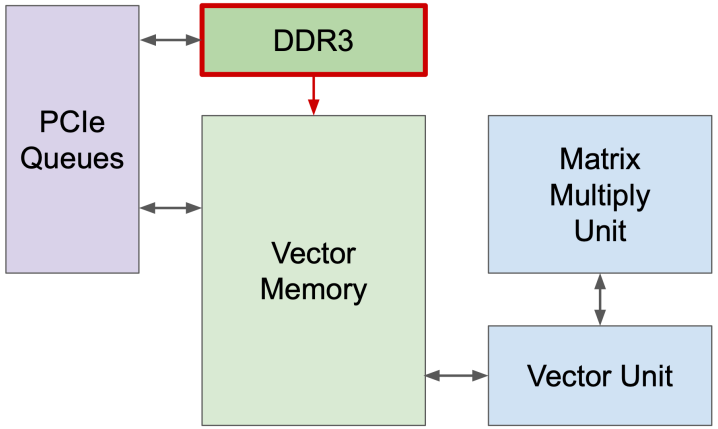
Connect matrix unit as an offload for the vector unit

How TPU v2 Uses the Vector Unit

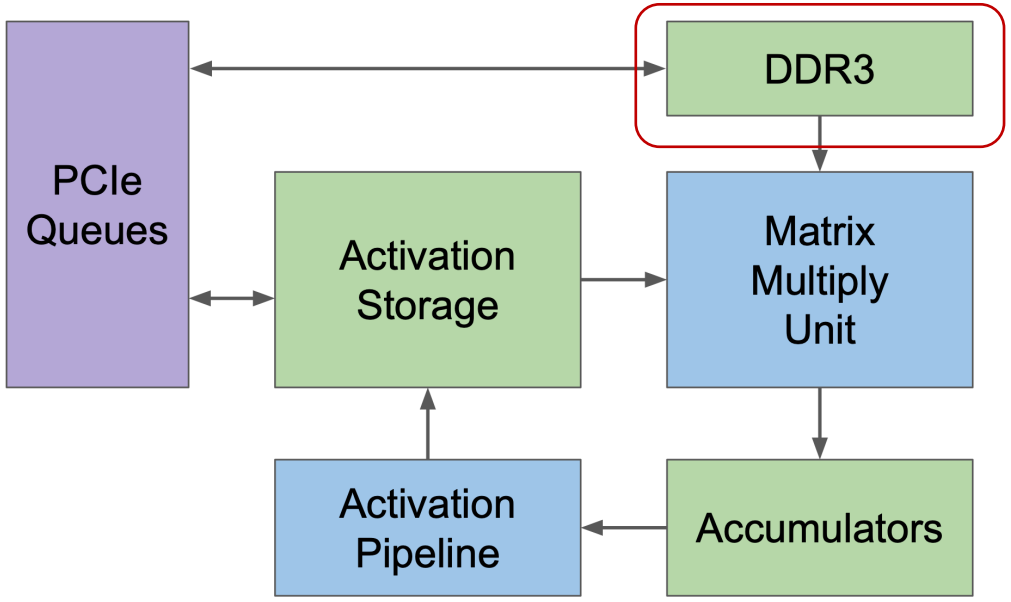
- **Pre-Processing** → The **Vector Unit** prepares input data (e.g., normalizing values before MXU).
- **Matrix Multiplication (MXU)** → TPU performs large **matrix multiplications** using systolic arrays.
- **Post-Processing** → The **Vector Unit applies activation functions** and **reduces outputs**.
- For example, in a **DNN**:
  - The **MXU** computes **weights × inputs**.
  - The **Vector Unit** applies **ReLU activation** to produce the final neuron output.



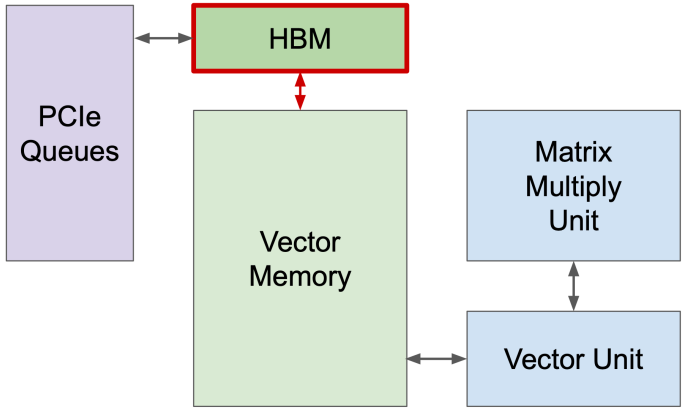
TPUv2 Changes



Connect DRAM into the memory system instead of directly into the matrix unit



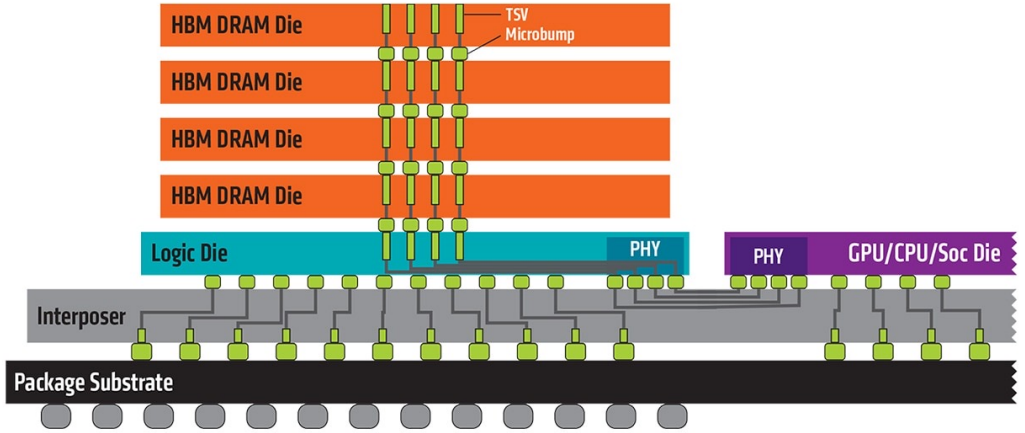
TPUv2 Changes



Move to HBM for bandwidth

High Bandwidth Memory

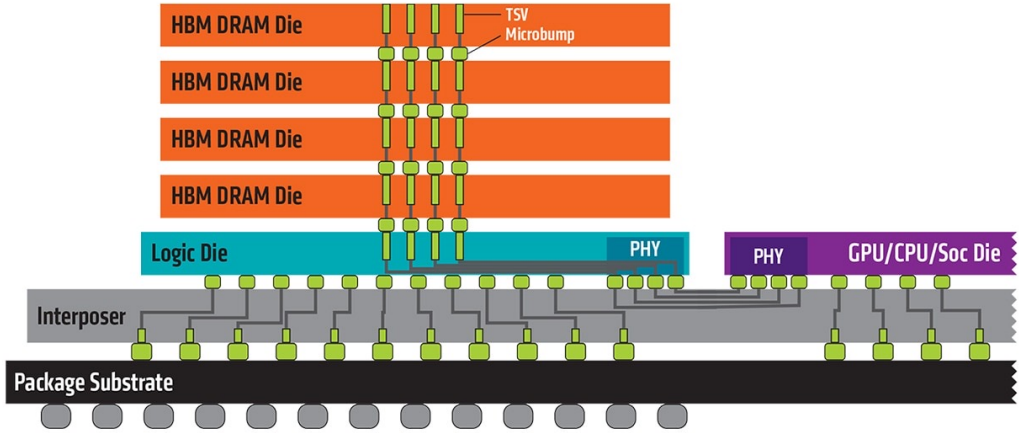
- **3D Stacking**: Compact structure with vertically stacked memory dies.
- **Wide Interface**: Typically 1024 bits or wider, far exceeding traditional memory.



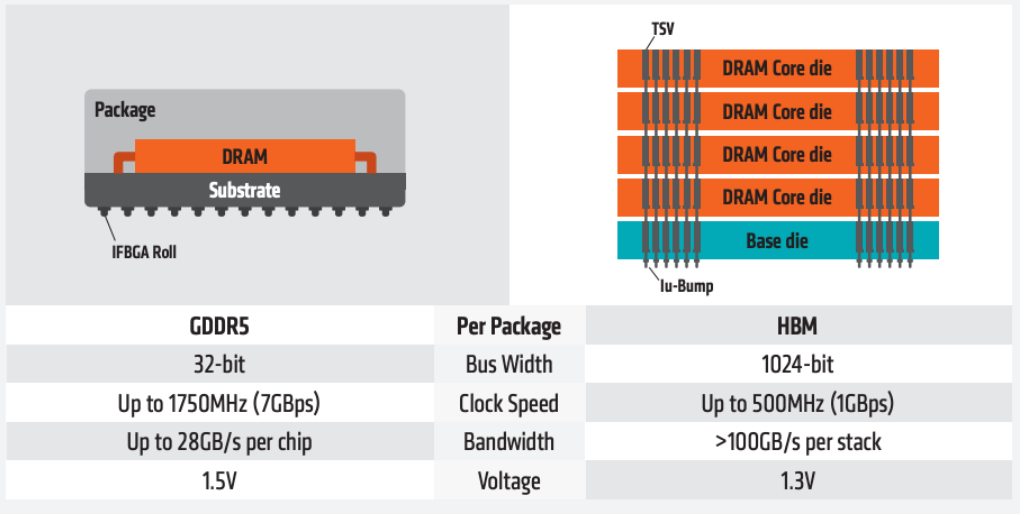
Source: AMD

High Bandwidth Memory

- **3D Stacking**: Compact structure with vertically stacked memory dies.
- **Wide Interface**: Typically 1024 bits or wider, far exceeding traditional memory.
- **High Bandwidth**: Ideal for data-intensive tasks like graphics and machine learning.
- **Low Power Consumption**: Efficient design with shorter, dense connection paths.
- **Enabled by TSVs**: Vertical electrical connections pass through the silicon.

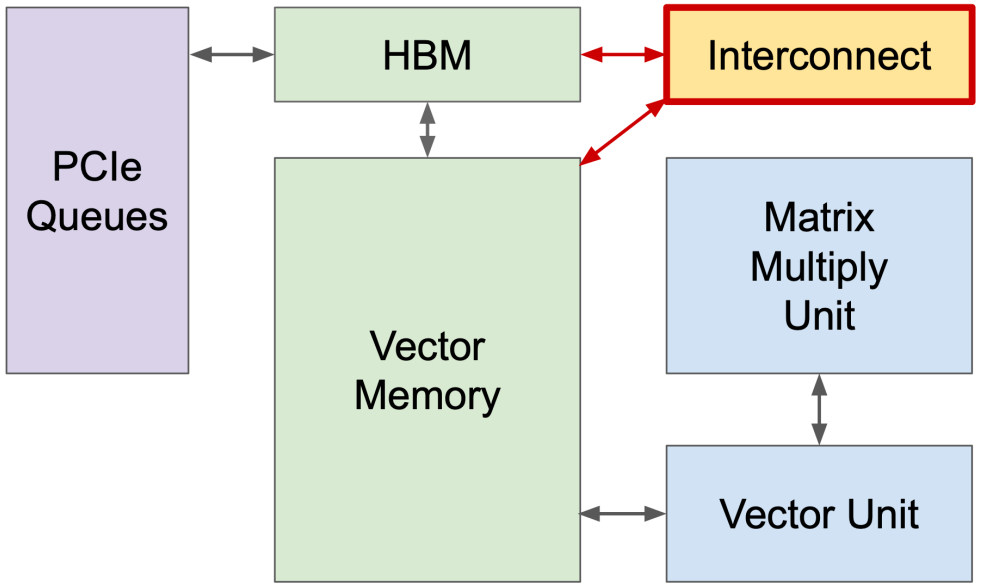


Source: AMD



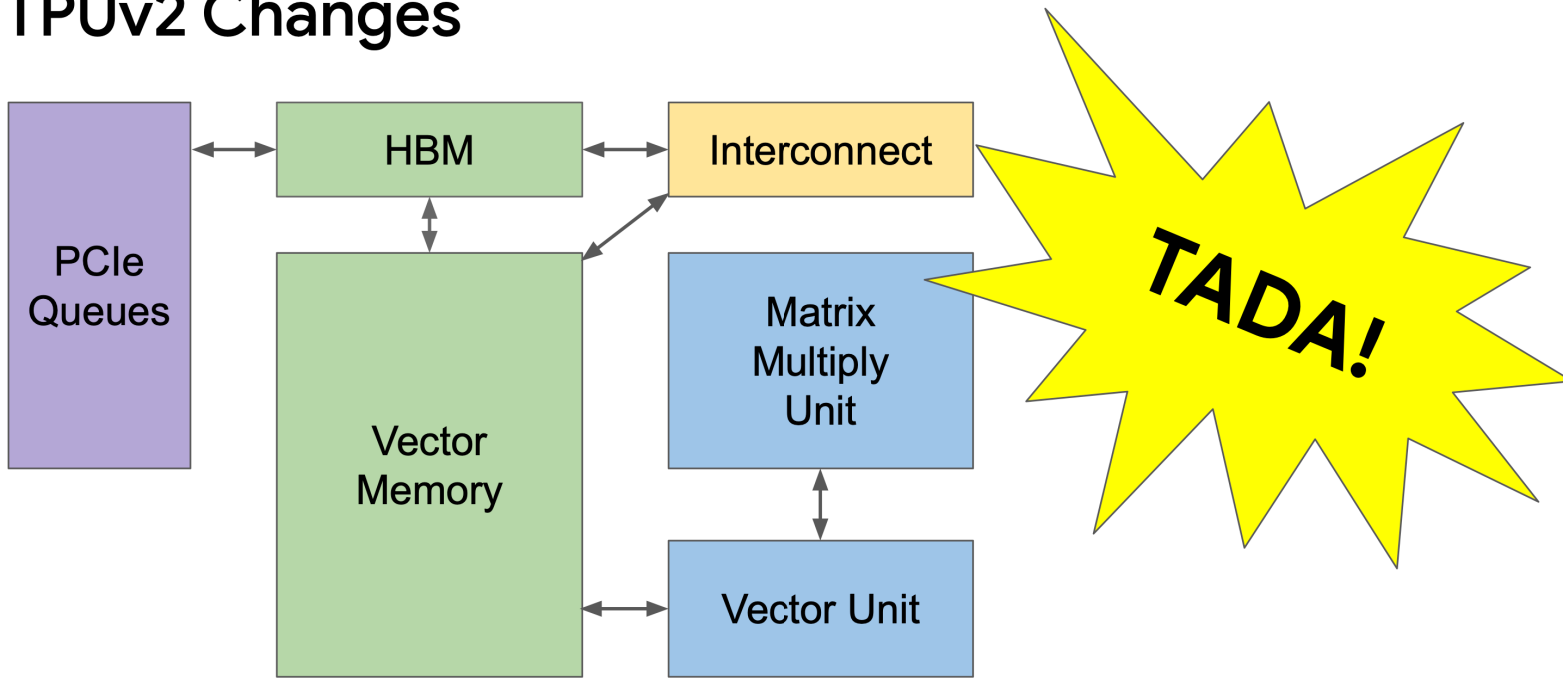
GDDR5	Per Package	HBM
32-bit	Bus Width	1024-bit
Up to 1750MHz (7GBps)	Clock Speed	Up to 500MHz (1GBps)
Up to 28GB/s per chip	Bandwidth	>100GB/s per stack
1.5V	Voltage	1.3V

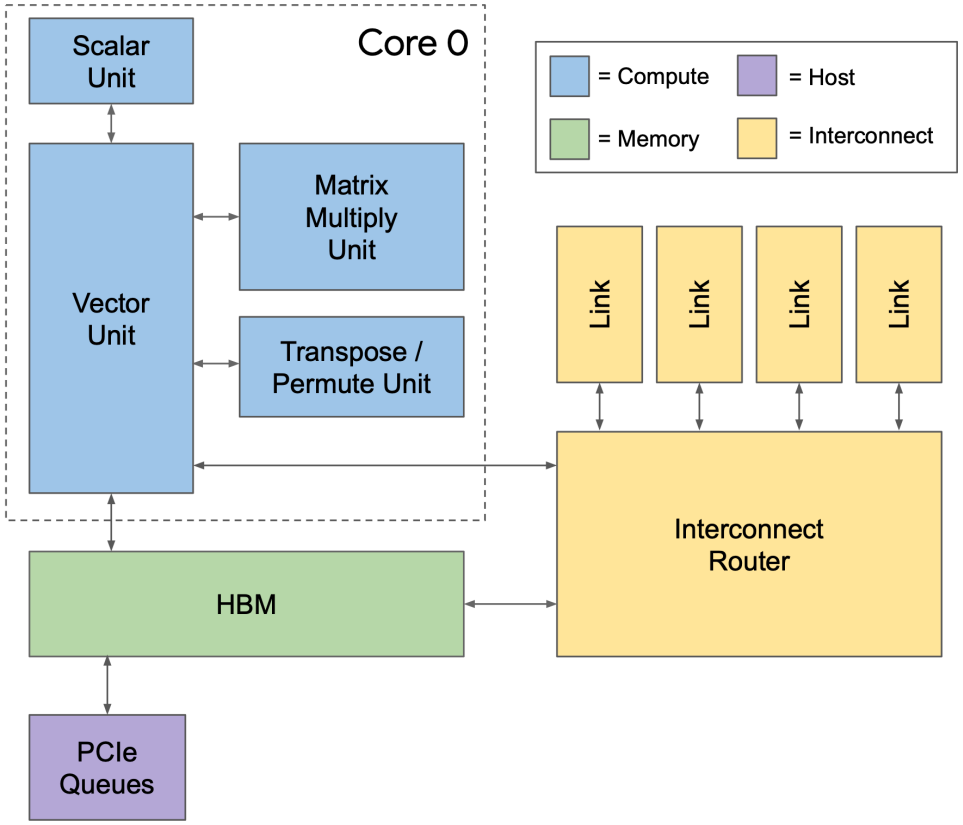
TPUv2 Changes



Add interconnect for high-bandwidth scaling

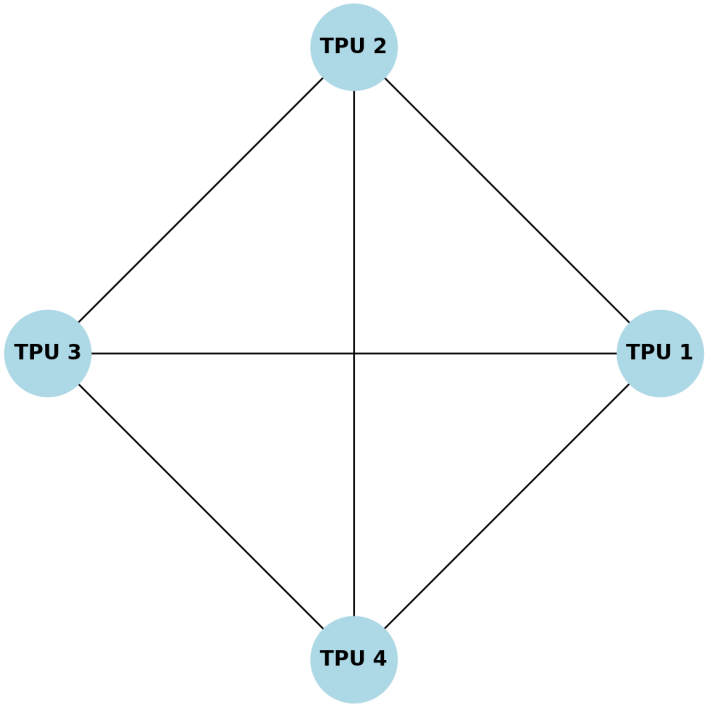
TPUv2 Changes

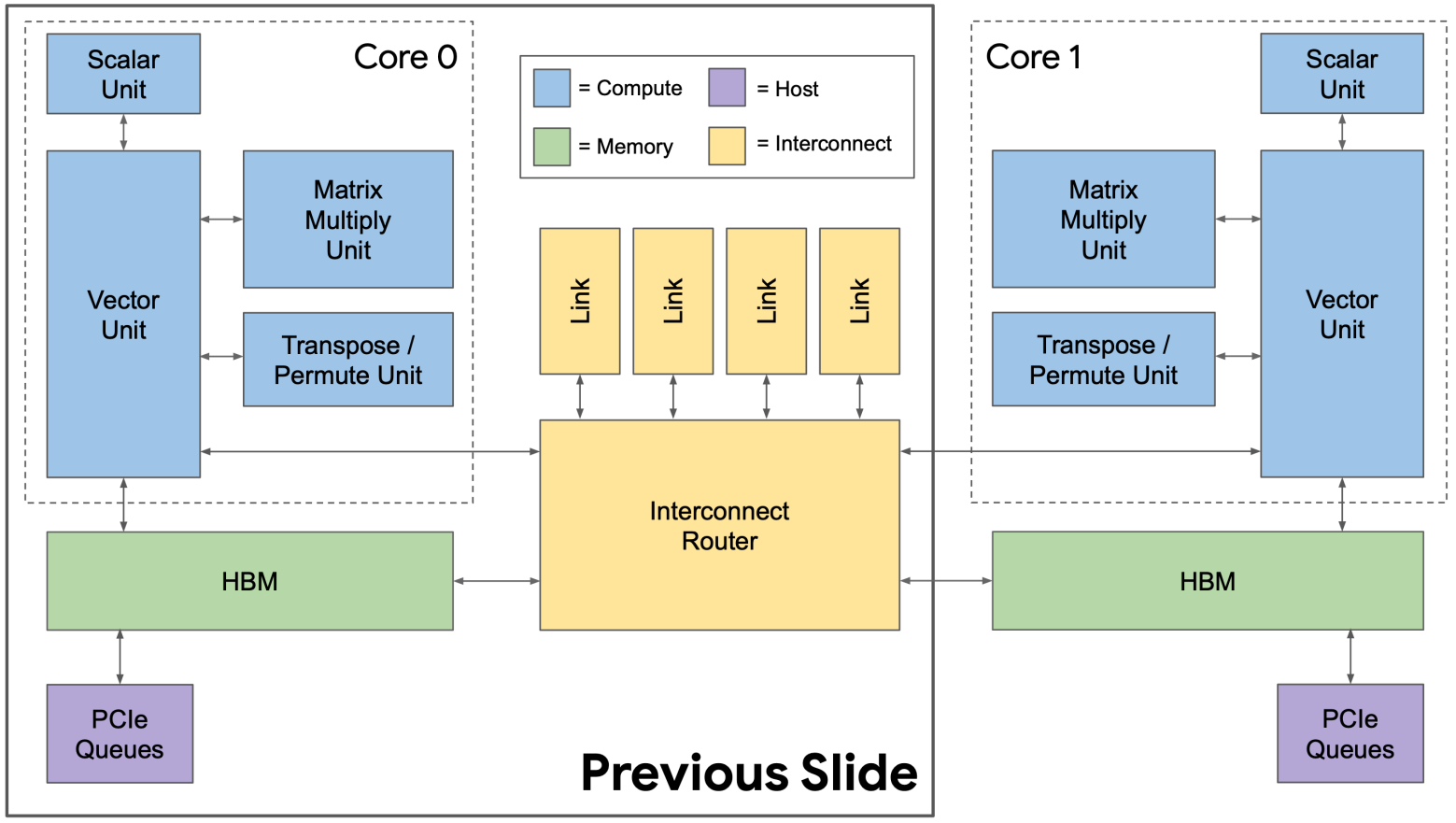




Redrawn with more detail...

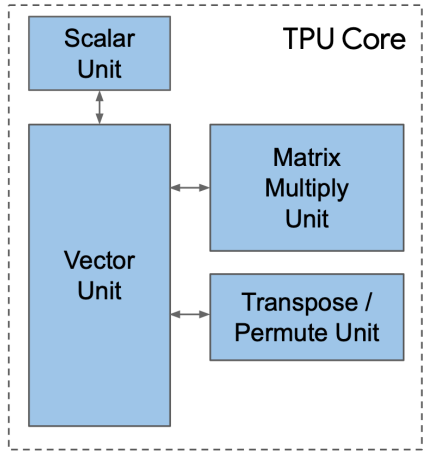
TPU v2 Interconnect (4 High-Speed Links)





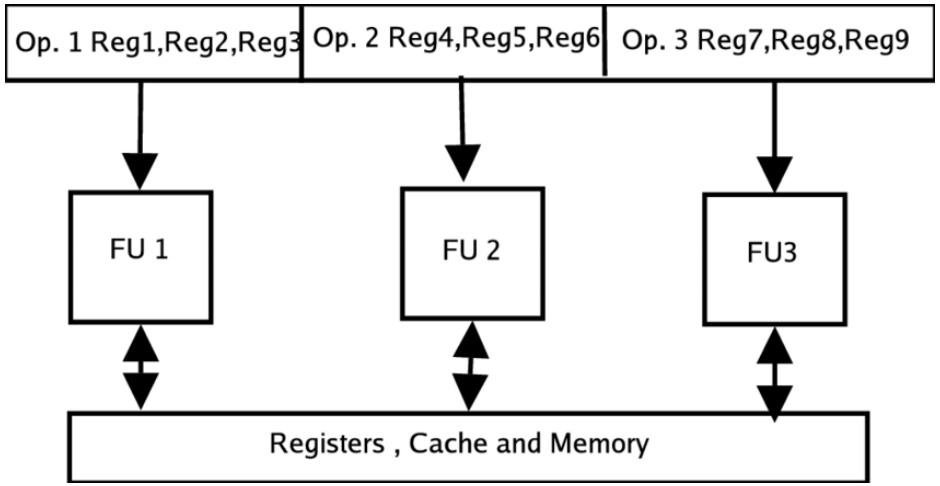
TPU Core

- VLIW Architecture
  - Leverage known compiler techniques
- Linear Algebra ISA
  - Scalar, vector, and matrix
  - Built for generality

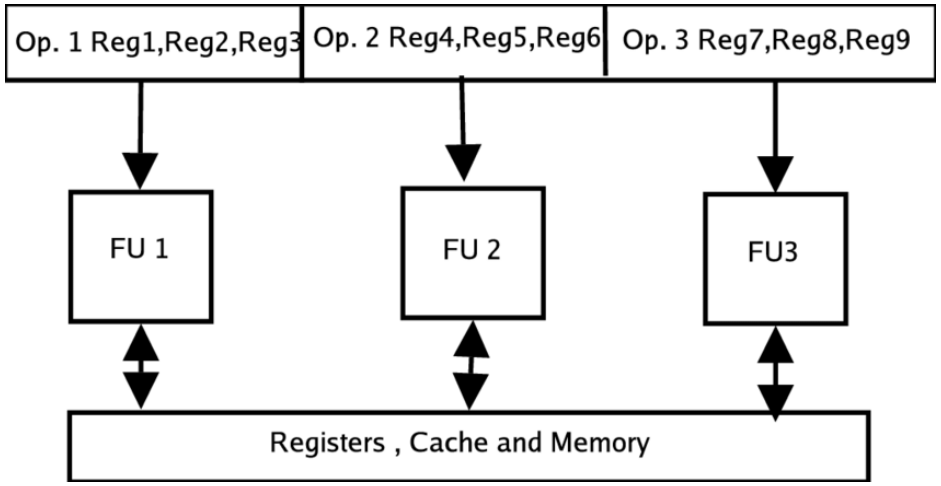


Very Long Instruction Word (VLIW) Overview

- Definition:
  - A computer processor architecture where multiple operations are encoded in a single, long instruction word.
- Compile-Time Parallelism:
  - Offloads the responsibility of instruction scheduling and identifying parallelism to the compiler, not the processor.
- Simplicity and Efficiency:
  - By relying on the compiler to handle parallelism, the hardware can be simplified. No need for complex circuitry to manage out-of-order execution or dynamic scheduling.
- Usage:
  - Common in certain embedded systems and digital signal processors (DSPs).



Very Long Instruction Word (VLIW) Overview

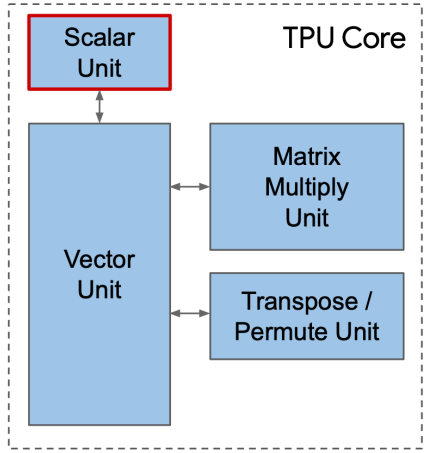



```
for (i=0; i<N; i++){
   B[i] =A[i] + C;
}
```

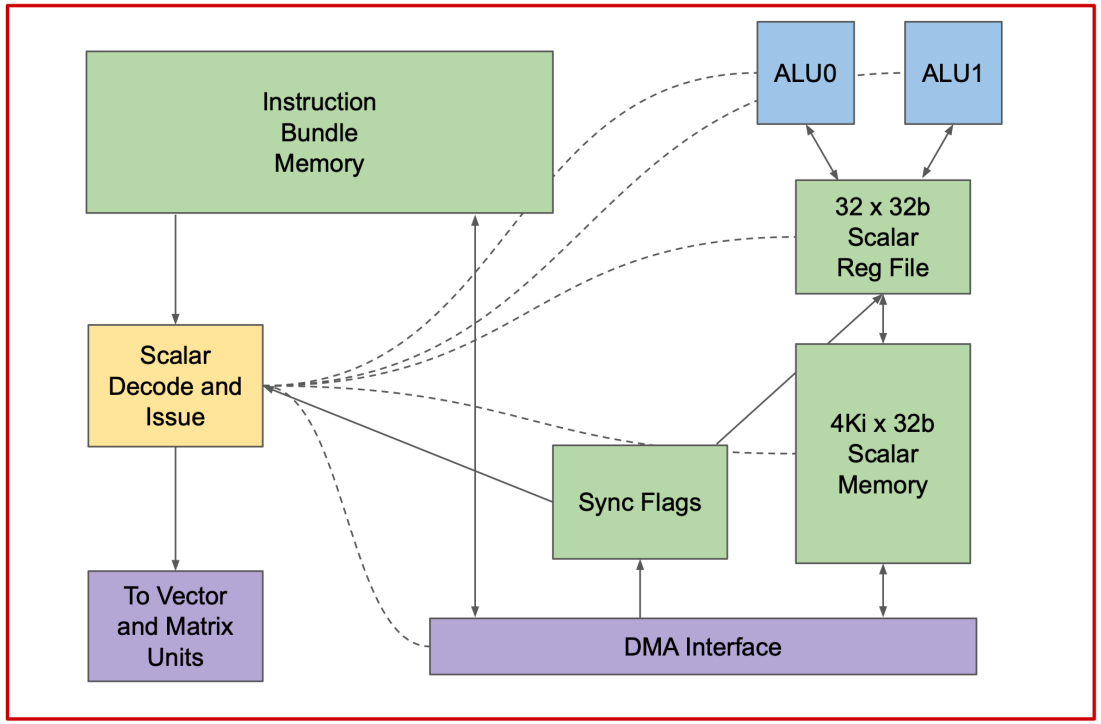
```
// Unroll inner loop to perform 3 iterations
for (i=0; i<N; i+=3) {
  B[i] = A[i] + C;
  B[i+1] = A[i+1] + C;
  B[i+2] =A[i+2] + C;
}
```

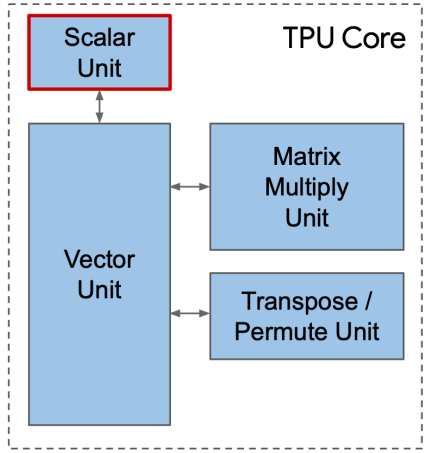
TPU Core: Scalar Unit

- 322b VLIW bundle
  - 2 scalar slots
  - 4 vector slots (2 for load/store)
  - 2 matrix slots (push, pop)
  - 1 misc slot
  - 6 immediates
- Scalar Unit performs:
  - Full VLIW bundle fetch and decode
  - Scalar slot execution

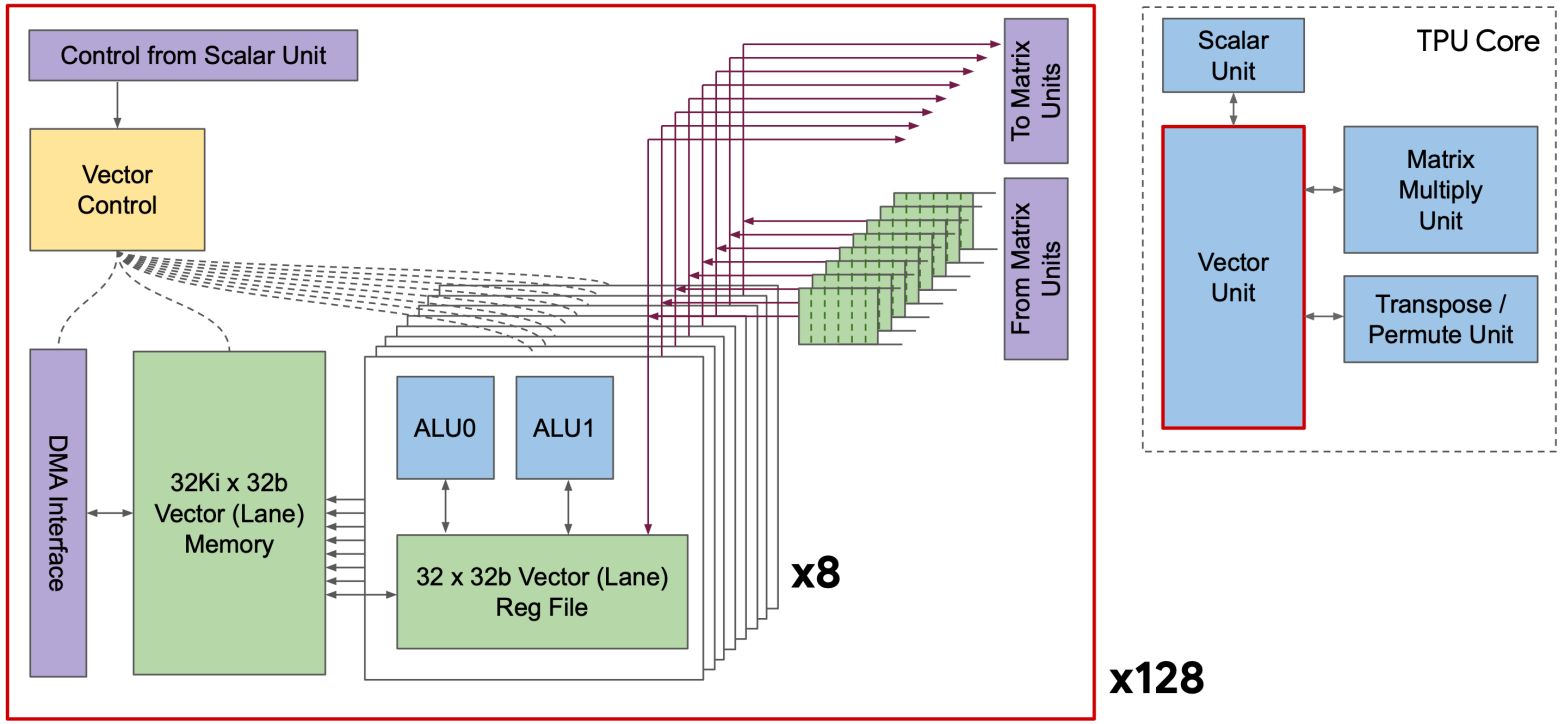


TPU Core: Scalar Unit





TPU Core: Vector Unit (Lane)



The **Vector Unit** in a TPU has multiple **lanes**, each responsible for processing a subset of vector elements. Each lane performs **SIMD (Single Instruction, Multiple Data) operations**, meaning one instruction operates on multiple data elements in parallel.

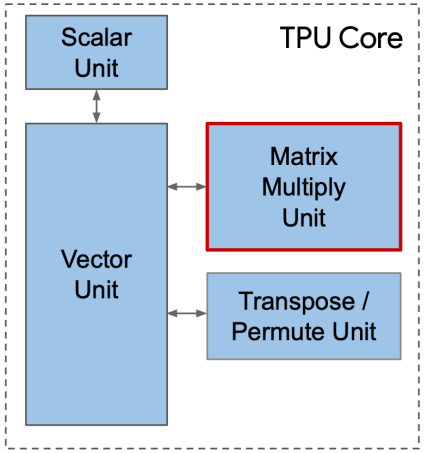
Example: Vector Unit with 8 Lanes

- Let’s say we have **8 vector lanes** inside a TPU:
- Each lane processes **one element of an 8-element vector** per cycle.
- If we need to apply **ReLU(x) = max(0, x)** on 8 numbers:
  - The **8 vector lanes** process all elements in **one cycle**, making execution much faster than processing elements one-by-one.

```
Input Vector:   [-3, 2, -1, 5, 6, -2, 4, 0]
Output Vector:  [ 0, 2,  0, 5, 6,  0, 4, 0]
```

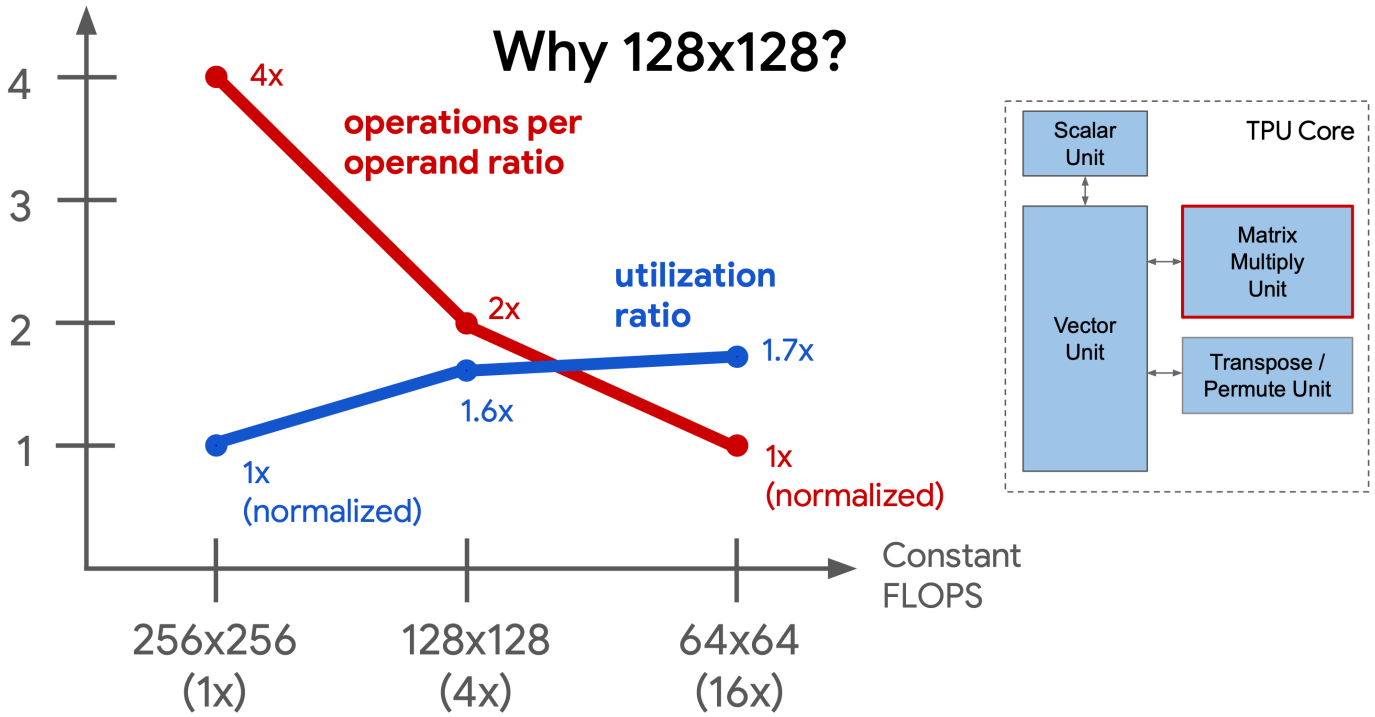
TPU Core: Matrix Multiply Unit

- 128 x 128 systolic array
  - Streaming LHS and results
  - Stationary RHS (w/ optional transpose)
- Numerics
  - bfloat16 multiply
    - {s, e, m} = {1, 8, 7}
    - The original!
  - float32 accumulation



The numerics {s, e, m} = {1, 8, 7} describe the bit allocation for sign (s), exponent (e), and mantissa (m) in the bfloat16 format, with 1 bit for sign, 8 bits for the exponent, and 7 bits for the mantissa.

$$\text{Operations-per-operand} = \frac{\text{Total Compute Operations}}{\text{Total Data Loads}}$$

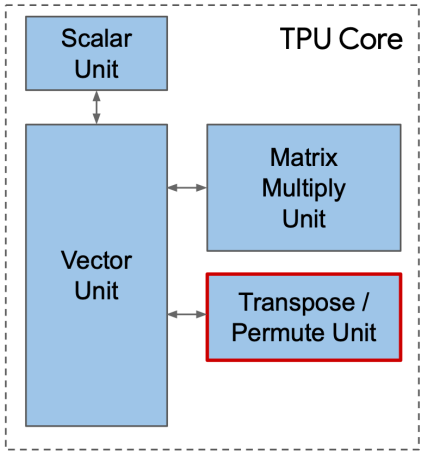


- The **operations-per-operand** ratio provides insight into the computational efficiency of the array.
- Relates to the reusability of data operands within the array.
- A higher operations-per-operand ratio means that the data is being used more efficiently.

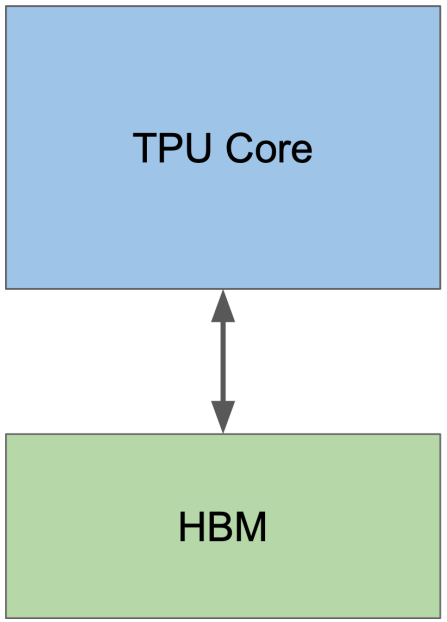
Once an operand is loaded into the array, it can be used by multiple processing elements as it propagates through the array. In a smaller array, such as the 128x128, each operand is reused more times within the array without needing to be reloaded. This increases the operational efficiency for the 128x128 array as compared to the 256x256 array.

TPU Core: Transpose, Reduction, Permute Unit

- Efficient common matrix transforms
  - Transpose
  - Reduction
  - Permutation
- Generally, allow reshuffling of data across vector lanes



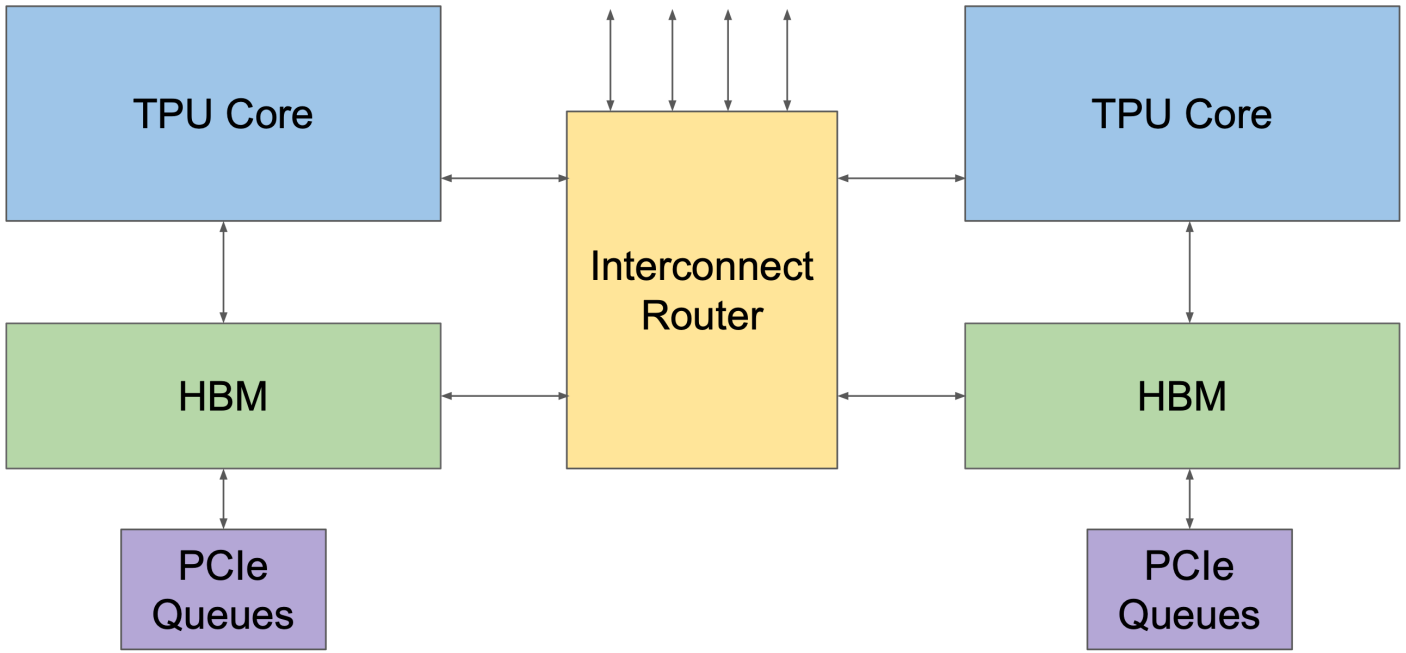
Memory System



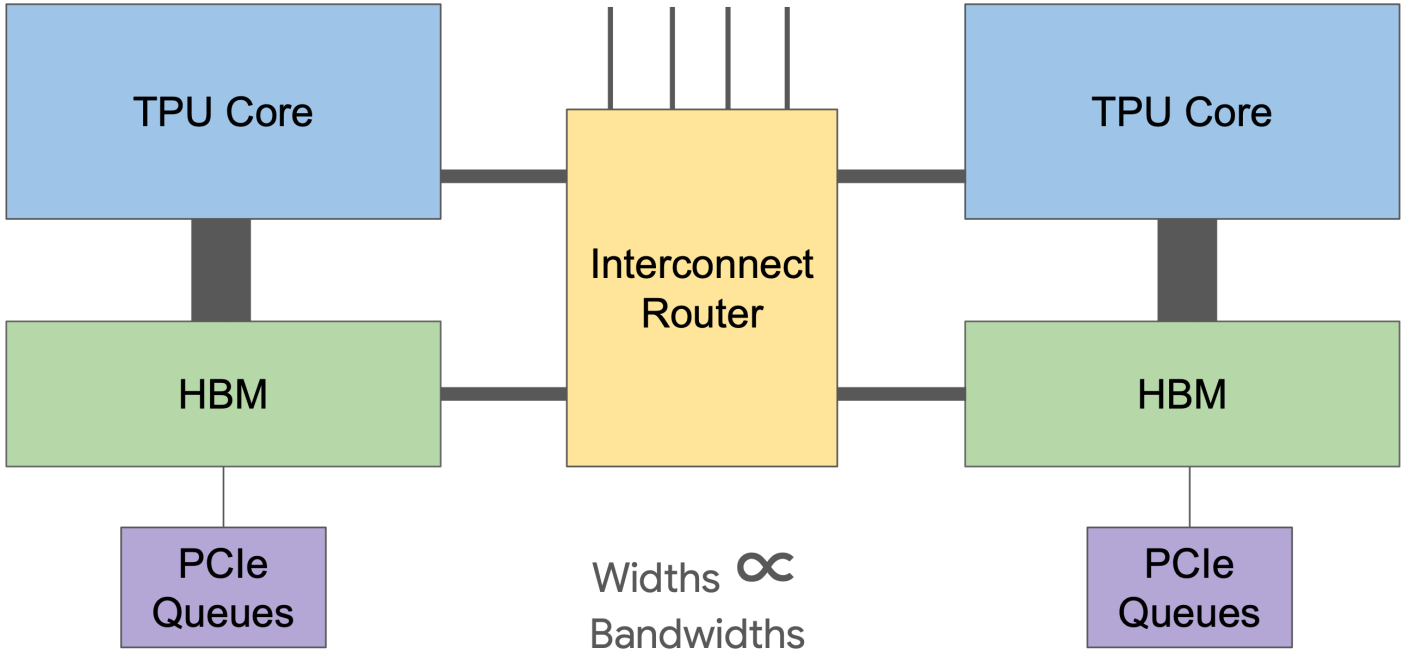
- Loads and stores against SRAM scratchpads
- Provides predictable scheduling within the core
- Can stall on sync flags

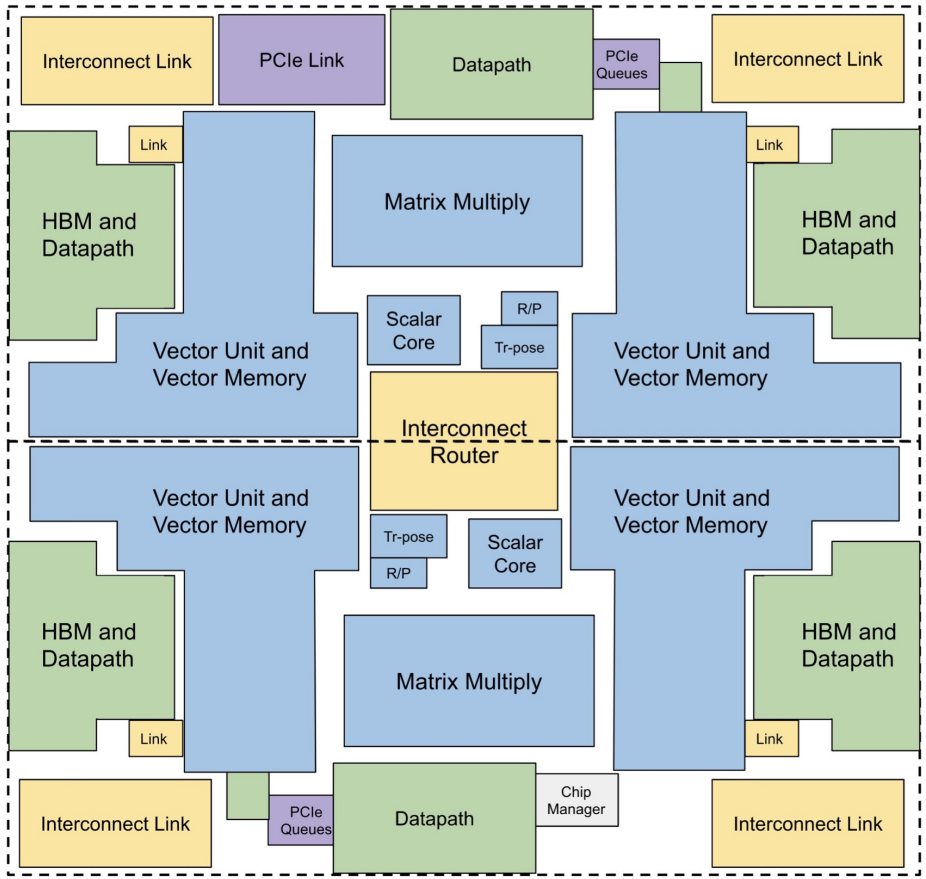
- Accessible through asynchronous DMAs
- Indicate completion in sync flags

Memory System



Memory System





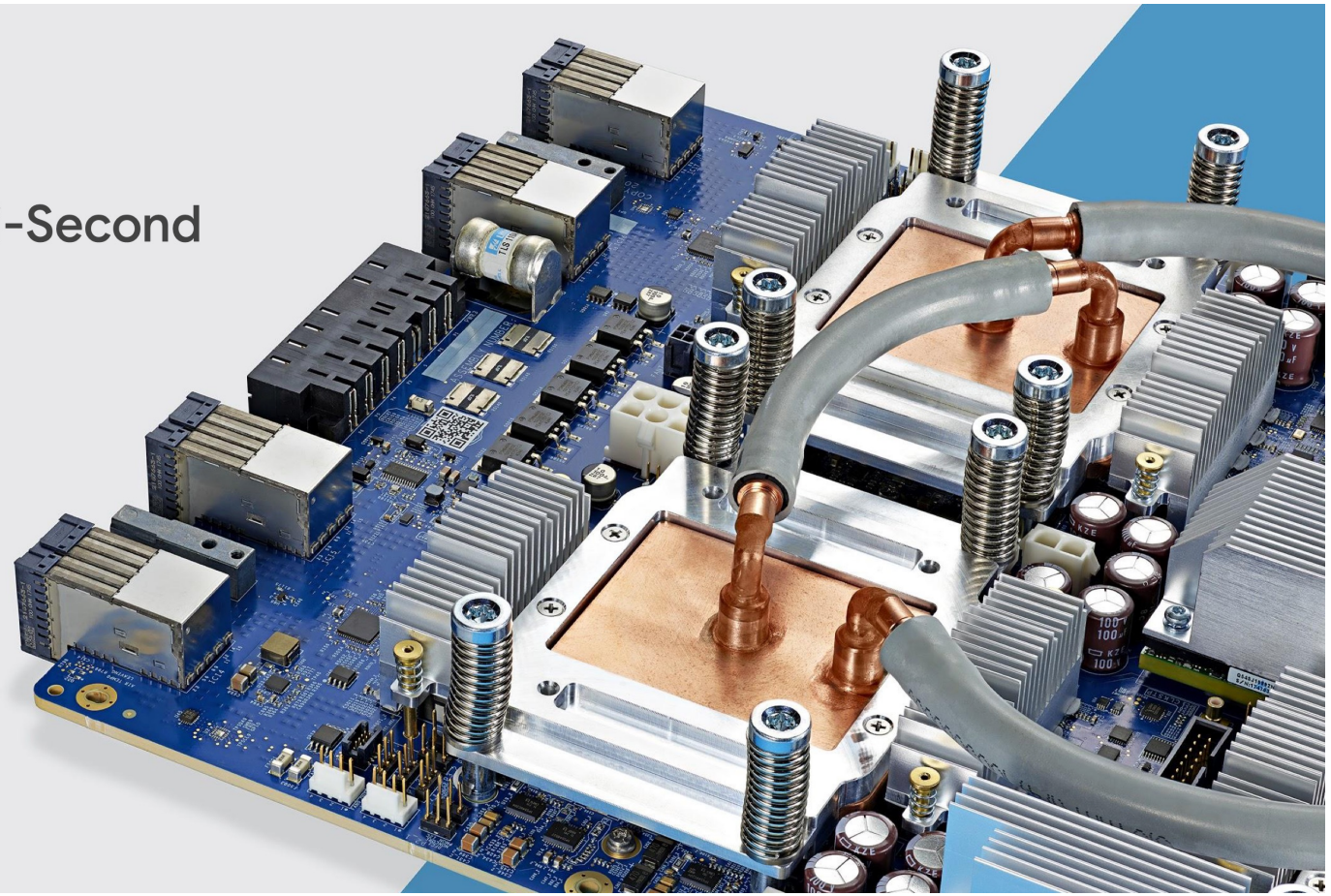
Floorplan

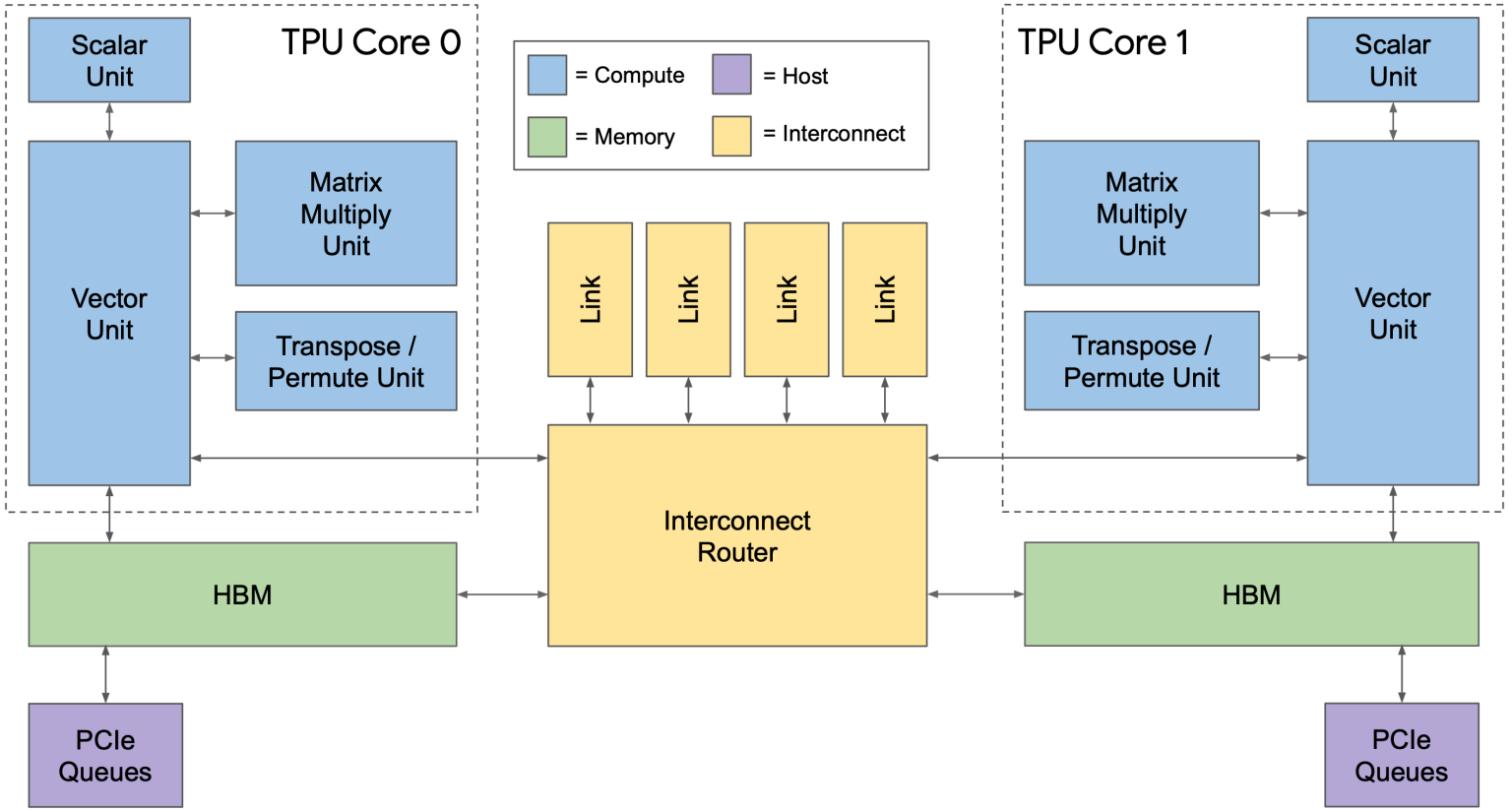
- = Compute
- = Memory
- = Interconnect
- = Host
- = Routing

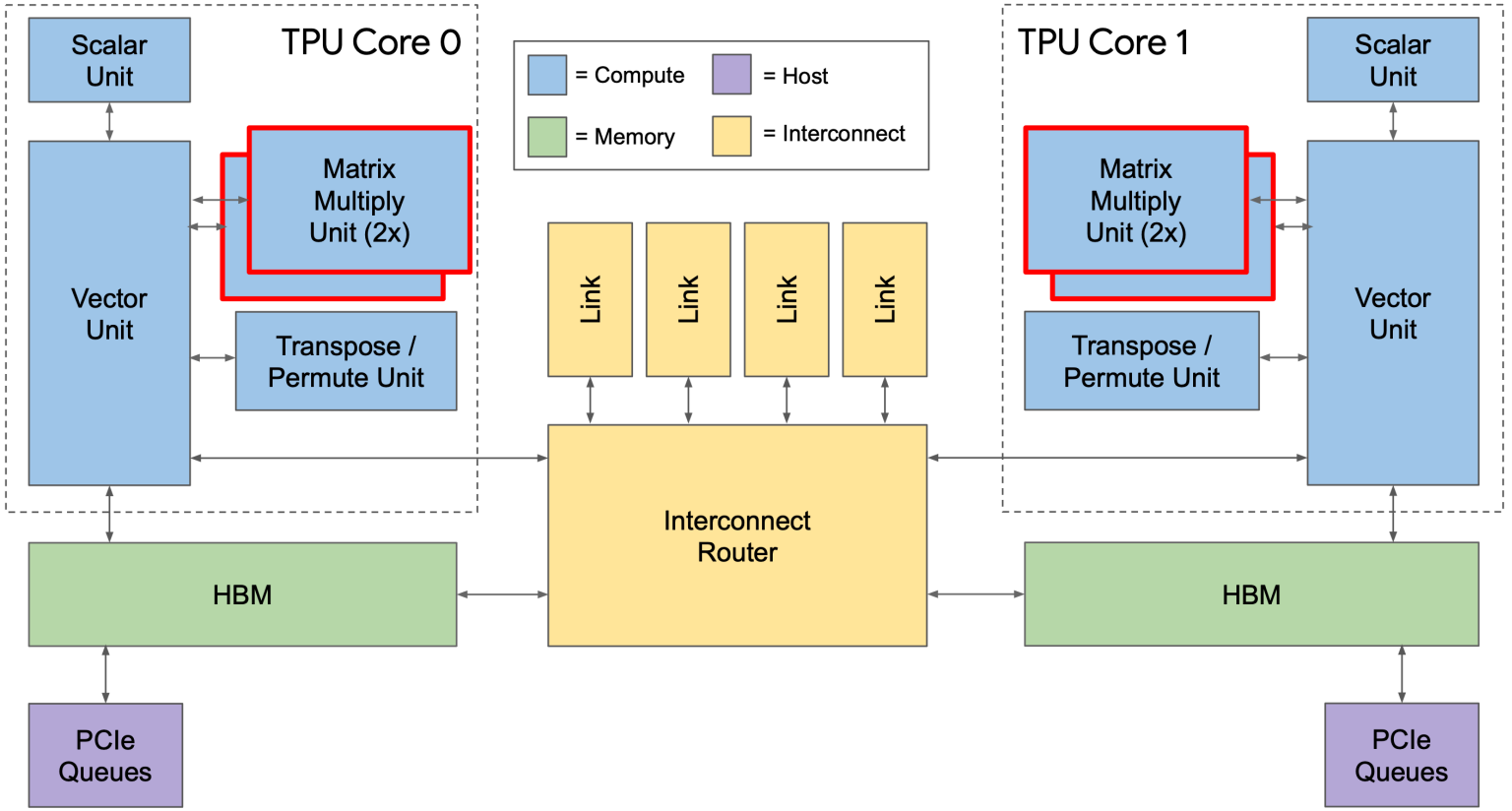
TPU2 has 2 cores per chip

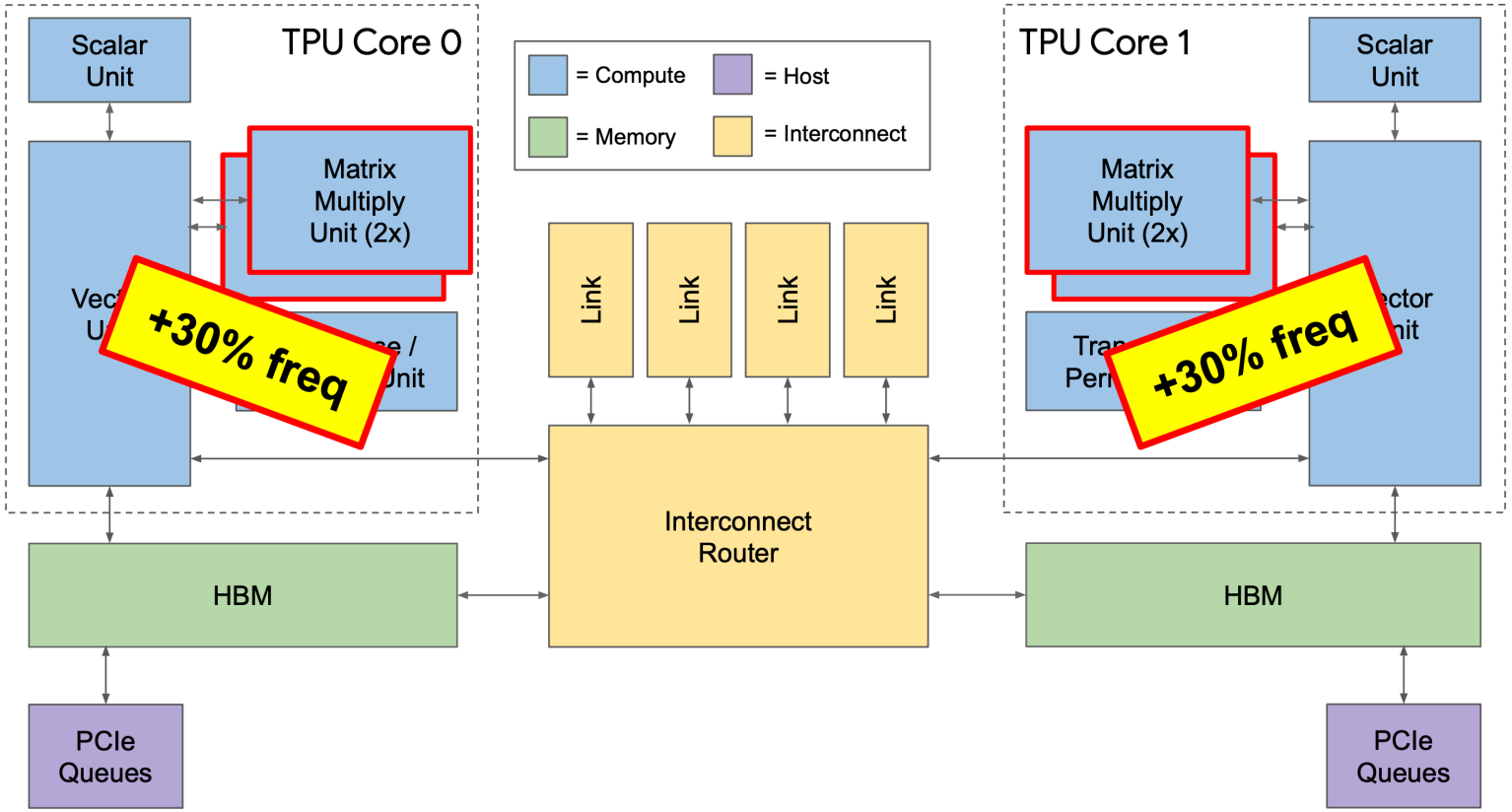
TPUv3

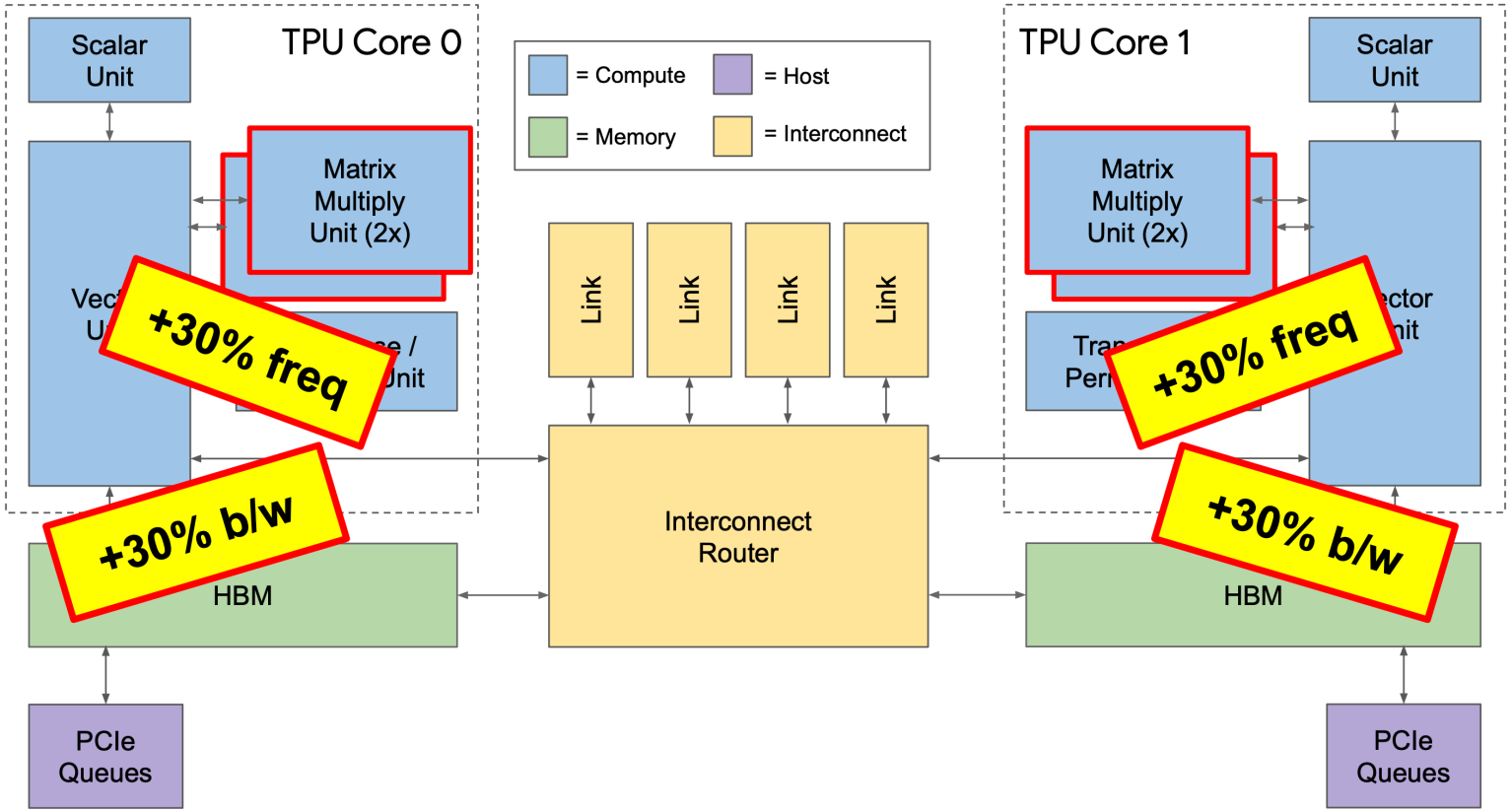
The Anti-Second System

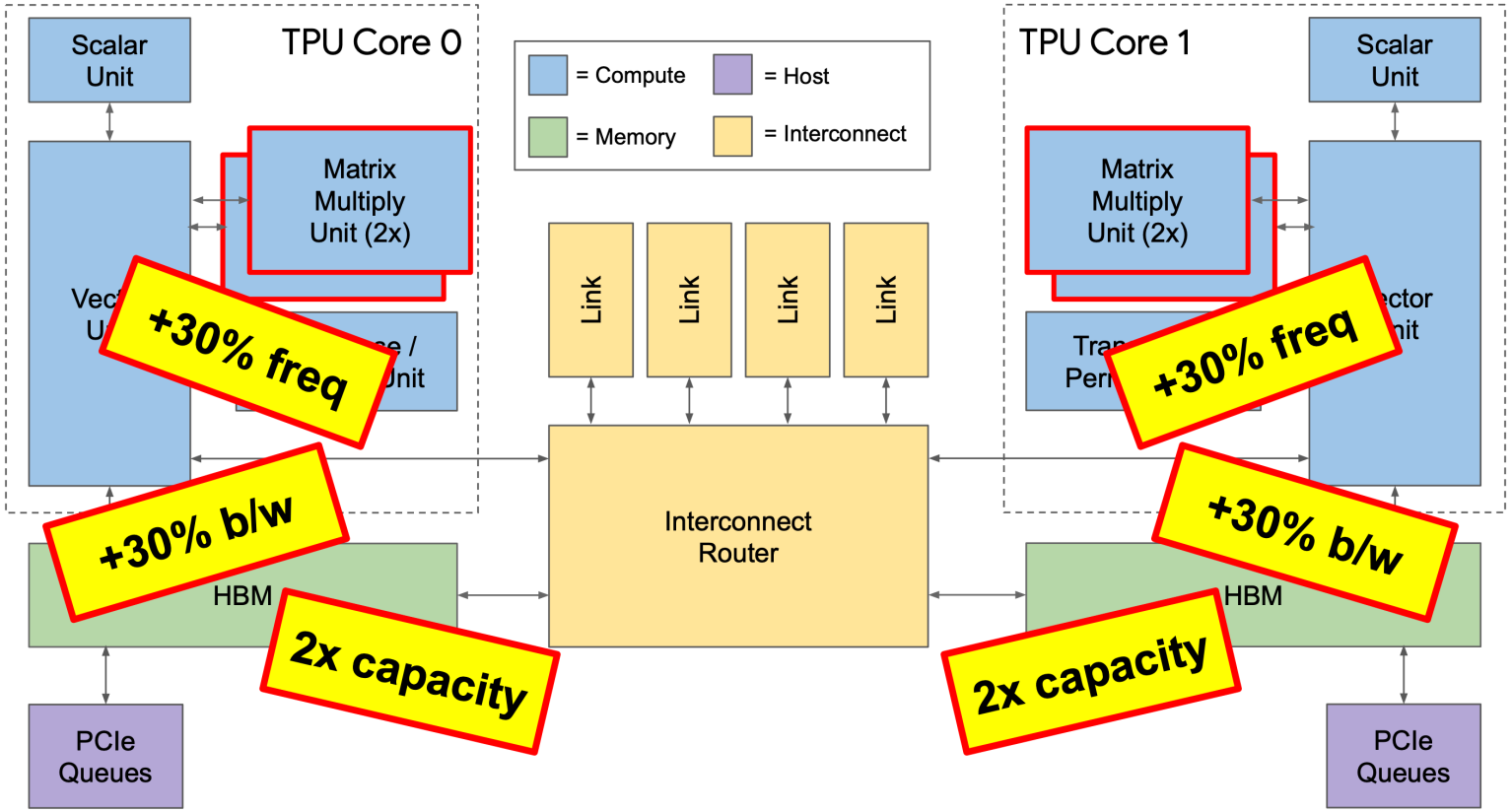


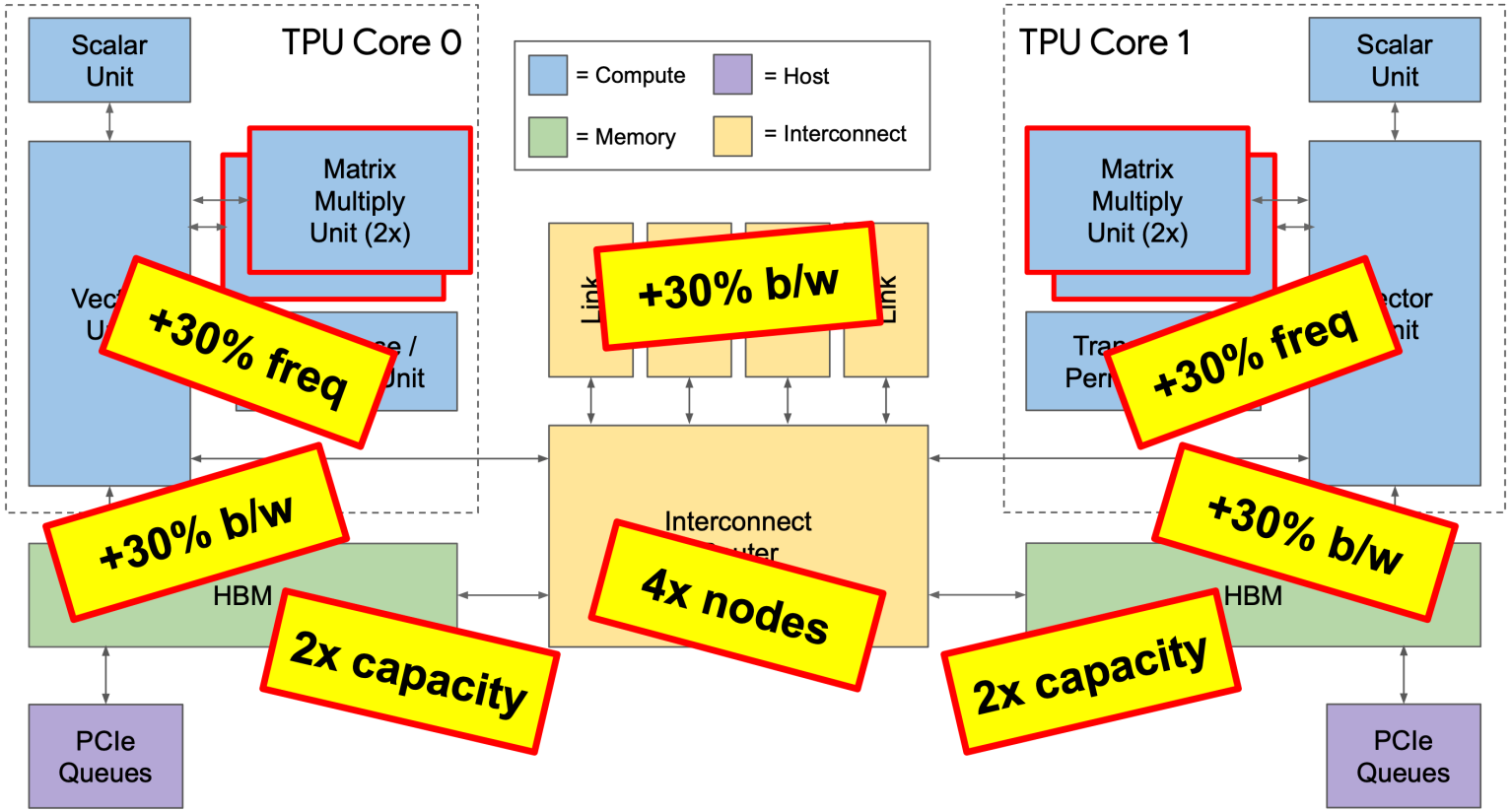

TPU Core 0
Scalar Unit
Vector Unit
Matrix Multiply Unit
Transpose / Permute Unit
= Compute
= Host
= Memory
= Interconnect
Link
Link
Link
Link
Interconnect Router
HBM
PCIe Queues
TPU Core 1
Scalar Unit
Matrix Multiply Unit
Transpose / Permute Unit
Vector Unit
HBM
PCIe Queues

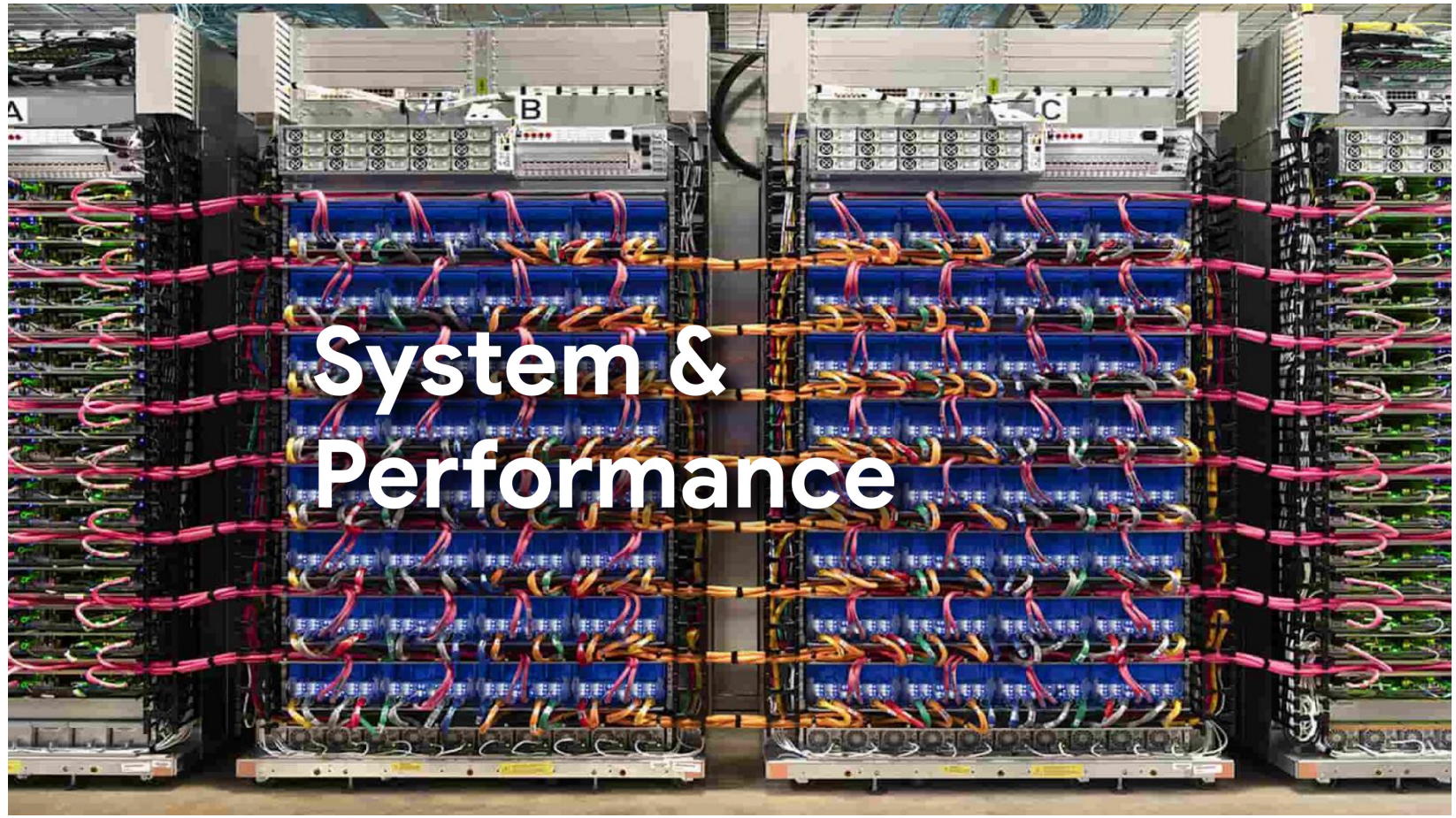

TPU Core 0
Scalar Unit
Vector Unit
Matrix Multiply Unit (2x)
Transpose / Permute Unit
= Compute
= Host
= Memory
= Interconnect
Link
Link
Link
Link
Interconnect Router
HBM
PCIe Queues
TPU Core 1
Scalar Unit
Vector Unit
Matrix Multiply Unit (2x)
Transpose / Permute Unit
HBM
PCIe Queues


TPU Core 0
Scalar Unit
Matrix Multiply Unit (2x)
+30% freq
= Compute
= Host
= Memory
= Interconnect
Link
Link
Link
Link
Interconnect Router
HBM
PCIe Queues
TPU Core 1
Scalar Unit
Matrix Multiply Unit (2x)
+30% freq
HBM
PCIe Queues


TPU Core 0
Scalar Unit
Matrix Multiply Unit (2x)
+30% freq
+30% b/w
HBM
PCIe Queues
= Compute
= Host
= Memory
= Interconnect
Link
Link
Link
Link
Interconnect Router
TPU Core 1
Scalar Unit
Matrix Multiply Unit (2x)
+30% freq
+30% b/w
HBM
PCIe Queues


TPU Core 0
Scalar Unit
Matrix Multiply Unit (2x)
+30% freq
+30% b/w
2x capacity
HBM
PCIe Queues
= Compute
= Host
= Memory
= Interconnect
Link
Link
Link
Link
Interconnect Router
TPU Core 1
Scalar Unit
Matrix Multiply Unit (2x)
+30% freq
+30% b/w
2x capacity
HBM
PCIe Queues


TPU Core 0
Scalar Unit
Matrix Multiply Unit (2x)
HBM
PCIe Queues
= Compute
= Host
= Memory
= Interconnect
Link
Link
Interconnect
+30% b/w
+30% freq
+30% b/w
2x capacity
4x nodes
TPU Core 1
Scalar Unit
Matrix Multiply Unit (2x)
HBM
PCIe Queues
+30% freq
+30% b/w
2x capacity



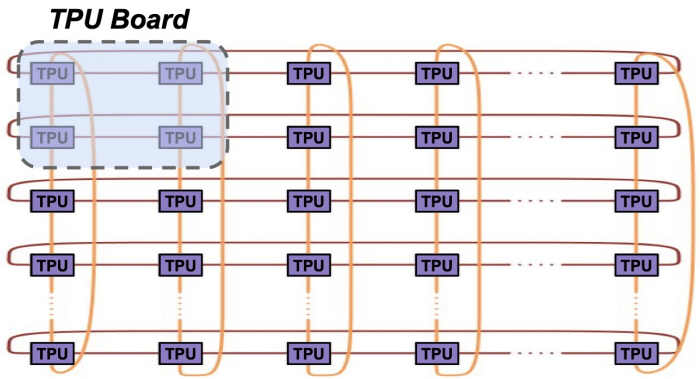
System & Performance

Supercomputer with dedicated interconnect

- TPUv1: single-chip system—built as **coprocessor** to a CPU
  - Works well for inference

- TPUv2, v3: ML **Supercomputer**
  - Multi-chip scaling critical for practical training times
    - Single TPUv2 chip would take 60 - 400 days for production workloads

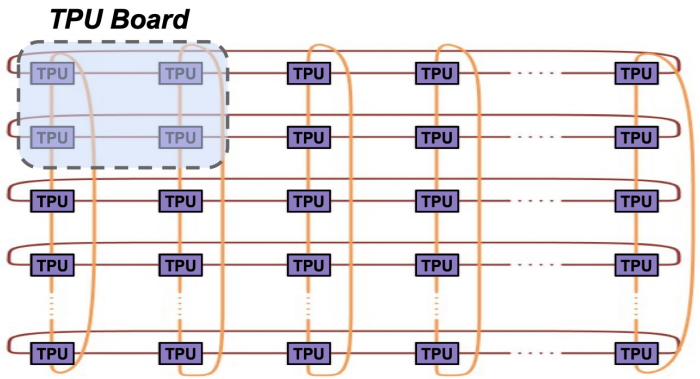
TPU Training Pod Architecture



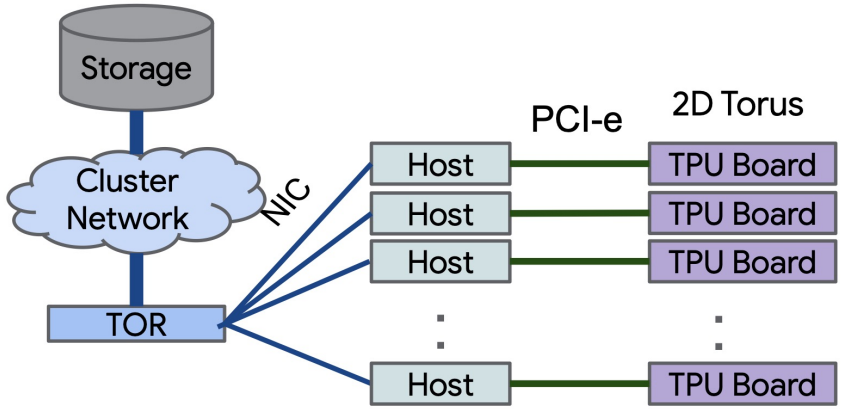
TPUs interconnected in 2D Torus

Dedicated network for
synchronous parallel training

TPU Training Pod Architecture



TPUs interconnected in 2D Torus

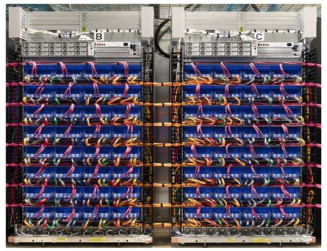


Dedicated network for **synchronous** parallel training

- A "pod" refers to a large-scale cluster of machines that work together as a single system for the purpose of machine learning training.
- TOR refers to a Top-of-Rack (TOR) Switch. It is a high-speed networking switch that connects the TPU boards, hosts, and storage within the TPU pod.

Supercomputer with dedicated interconnect

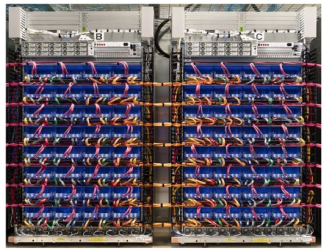
TPUv2 supercomputer
(256 chips)



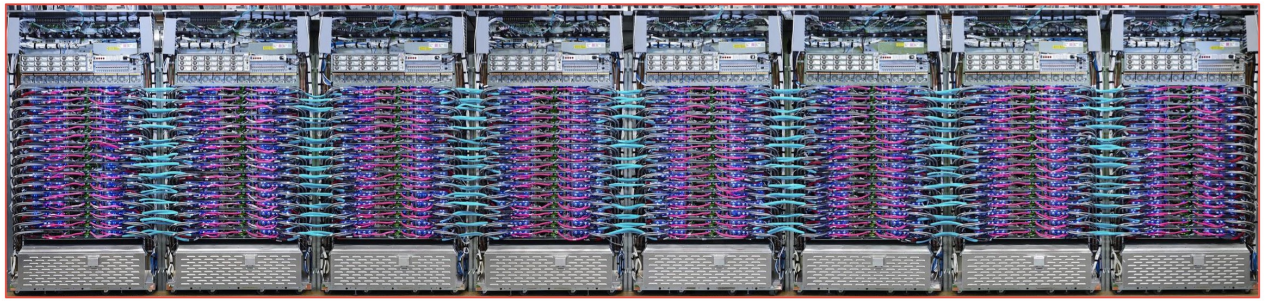
TPUv2 boards = 4 chips

Supercomputer with dedicated interconnect

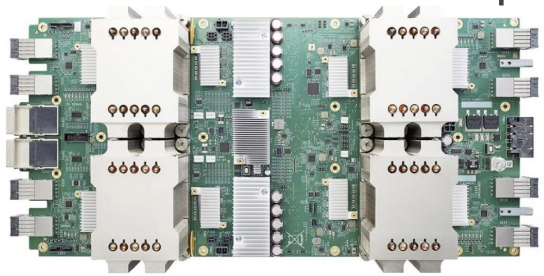
TPUv2 supercomputer (256 chips)



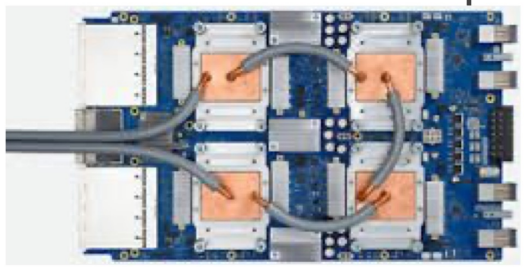
TPUv3 supercomputer (1024 chips)



TPUv2 boards = 4 chips

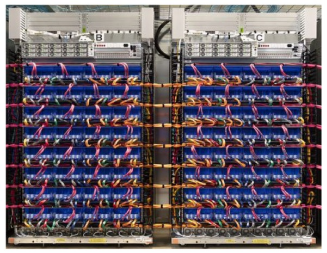


TPUv3 boards = 4 chips



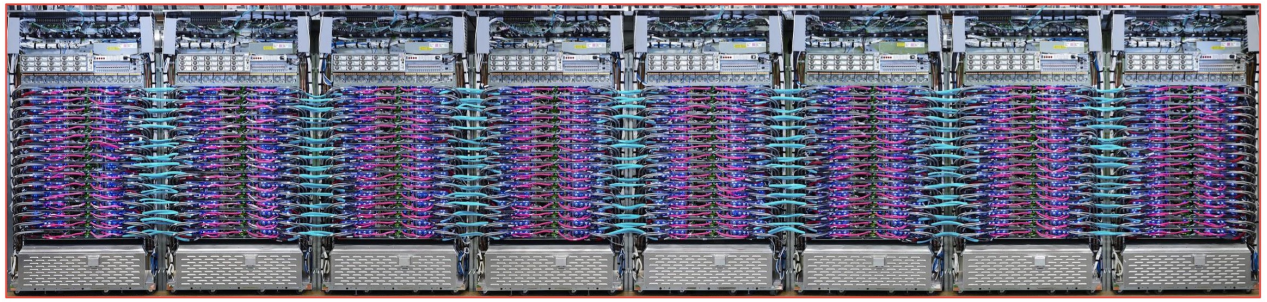
Supercomputer with dedicated interconnect

TPUv2 supercomputer (256 chips)

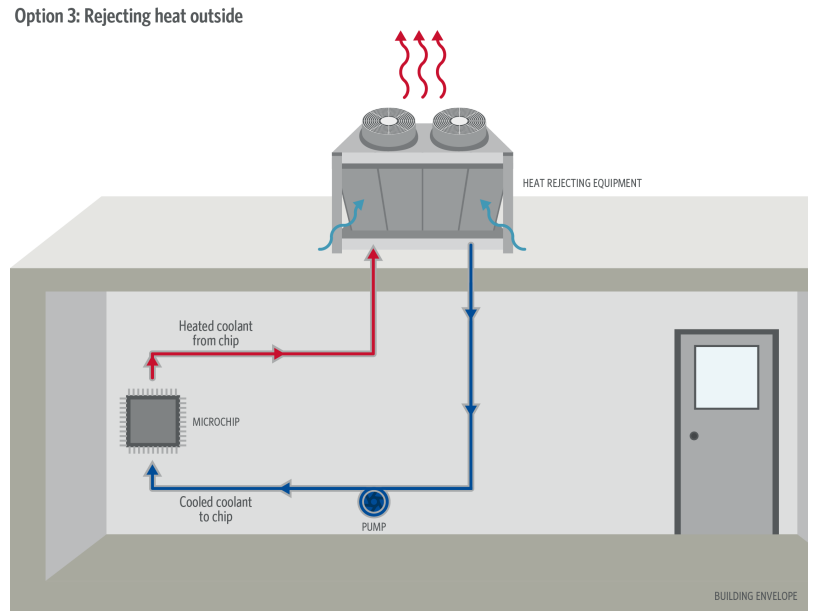


11.5 petaflops
4 TB HBM
2-D torus
256 chips

TPUv3 supercomputer (1024 chips)

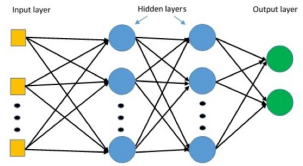


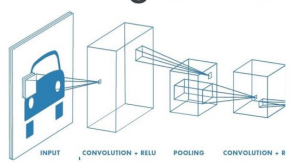
> 100 petaflops
32 TB HBM
Liquid cooled
New chip + larger-scale system
1024 chips

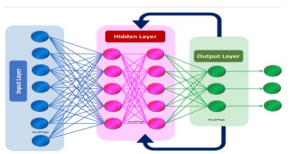


6 Production Applications

- **MultiLayer Perceptrons (MLP)**
  - MLP0 is unpublished
  - MLP1 is RankBrain [Cla15]
- **Convolutional Neural Networks (CNN)**
  - CNN0 is AlphaZero, which mastered the games chess, Go, and shogi [Sil18]
  - CNN1 is an Google-internal model for image recognition
- **Recurrent Neural Networks (RNN)**
  - RNN0 is a Translation model [Che18]
  - RNN1 is a Speech model [Chi18]

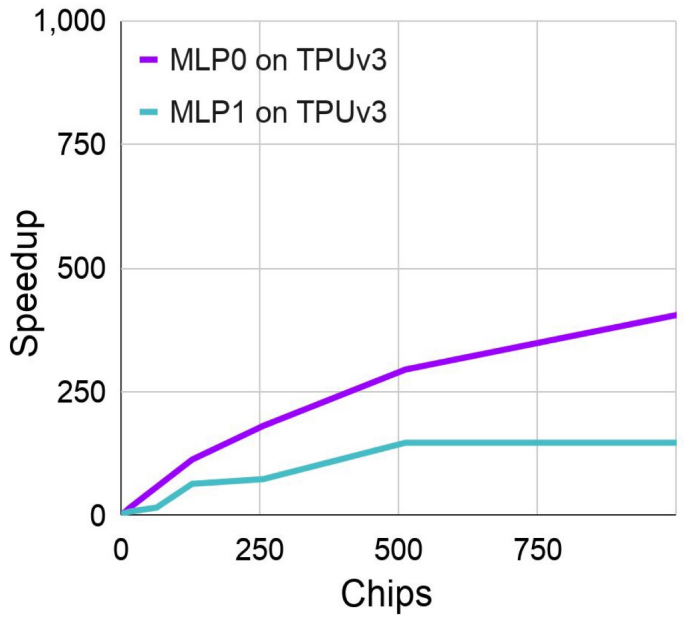






[Cla15] Clark, J. October 26, 2015, Google Turning Its Lucrative Web Search Over to AI Machines. Bloomberg Technology.
[Che18] Chen, M.X. et al, 2018. The best of both worlds: Combining recent advances in neural machine translation. arXiv preprint arXiv:1804.09849.
[Chi18] Chiu, C.C. et al, 2018, April. State-of-the-art speech recognition with sequence-to-sequence models. In IEEE Int'l Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4774-4778.
[Sil18] Silver, D. et al, 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419), pp.1140-1144.

TPUv3 Supercomputer Scaling: 6 Production Apps

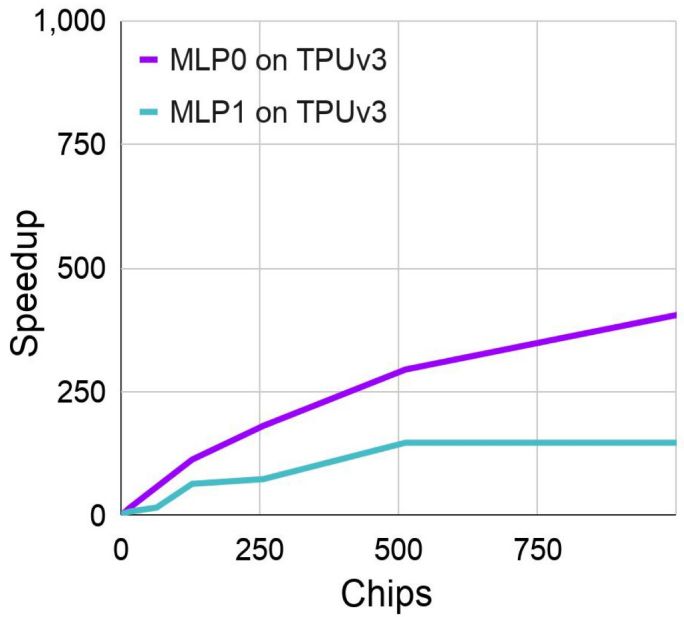


- **MLP0** & **MLP1**
  - 40% & 14% of perfect linear scale at 1024 chip-scale
    - Limited by embeddings

•**MLP0 achieves 40% of perfect linear scaling at 1024 TPU chips.**
•**MLP1 achieves only 14% of perfect scaling.**
•This means that as TPU chips are added:

- **MLP0** sees **somewhat efficient scaling**, but only **40% of the theoretical maximum speedup**.
- **MLP1** scales very **poorly** (only **14% efficiency**) due to **bottlenecks like embeddings**.

TPUv3 Supercomputer Scaling: 6 Production Apps



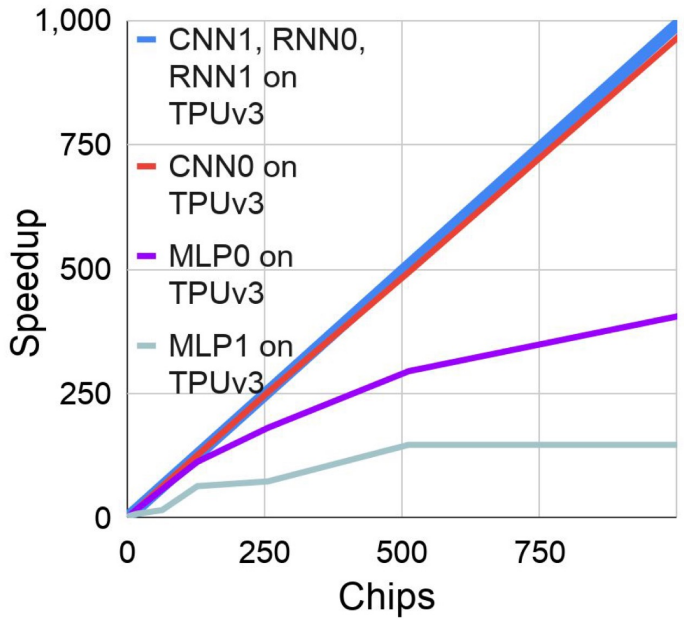
- **MLP0** & **MLP1**
  - 40% & 14% of perfect linear scale at 1024 chip-scale
    - Limited by embeddings

•**MLP0 achieves 40% of perfect linear scaling at 1024 TPU chips.**
•**MLP1 achieves only 14% of perfect scaling.**
•This means that as TPU chips are added:

- **MLP0** sees **somewhat efficient scaling**, but only **40% of the theoretical maximum speedup**.
- **MLP1** scales very **poorly** (only **14% efficiency**) due to **bottlenecks like embeddings**.

Why Doesn't Scaling Reach 100%?
•**Communication Overhead:** More TPUs means **more data transfers**, slowing down computation.
•**Memory Bottlenecks:** Models relying on large embeddings (like in MLP1) **saturate TPU memory bandwidth**.
•**Synchronization Delays:** Some tasks require **waiting for previous computations**, limiting parallel execution.

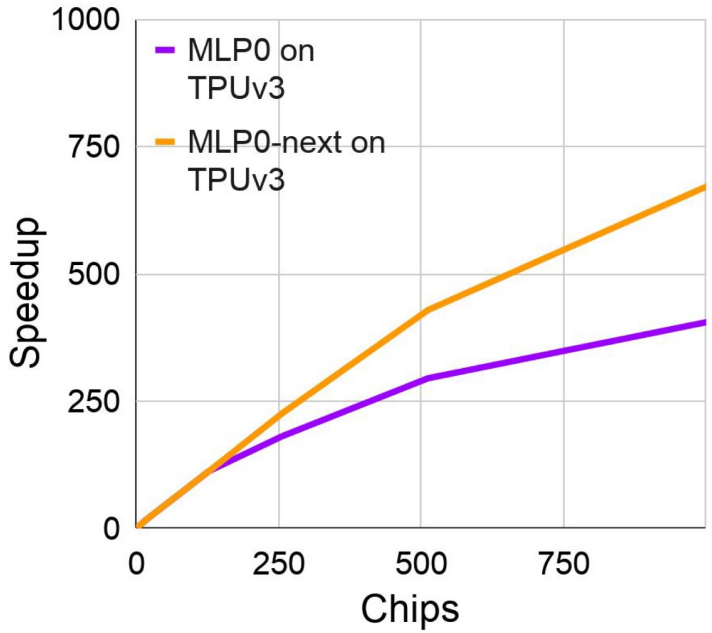
TPUv3 Supercomputer Scaling: 6 Production Apps



- **MLP0** & **MLP1**
  - 40% & 14% of perfect linear scaling
- **CNN0**
  - 96% of perfect linear scaling!
- **CNN1, RNN0, RNN1**
  - **3 production apps** run at **99%** of perfect linear scaling at 1024 chips!

MLPs generally exhibit **worse scaling efficiency** than Convolutional Neural Networks (**CNNs**) on TPUs due to differences in **data movement, memory access patterns, and computational intensity**.

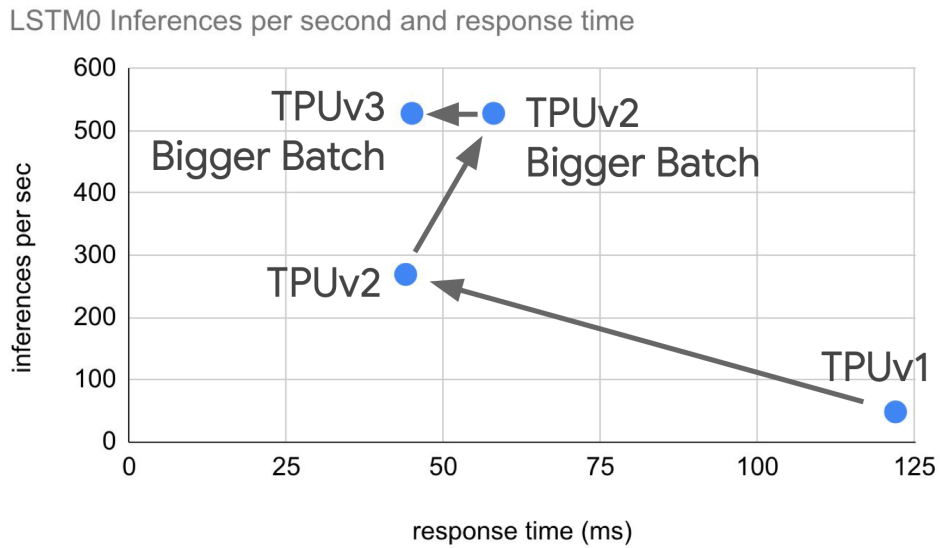
TPUv3 Supercomputer Scaling: MLP0-next vs. MLP0



- Improved scaling for newer larger models and SW improvements for better quality
  - **MLP0-next**: 67% of perfect linear scale at 1024 chips
    - Up from 40% from MLP0

Inference: TPUv2/v3 vs TPUv1

- Training chips can also do inference (looks like forward pass)
- Bfloat16 numerics in TPUv2/v3 vs int8 in TPUv1



Key Takeaways

- TPUv2/v3 supercomputers with 256-1024 chips run production applications at scale, powering many Google products
- TPUs are widely used to accelerate production and research
- Proven results from Model/HW/SW codesign, with more opportunities still available

Used across many products