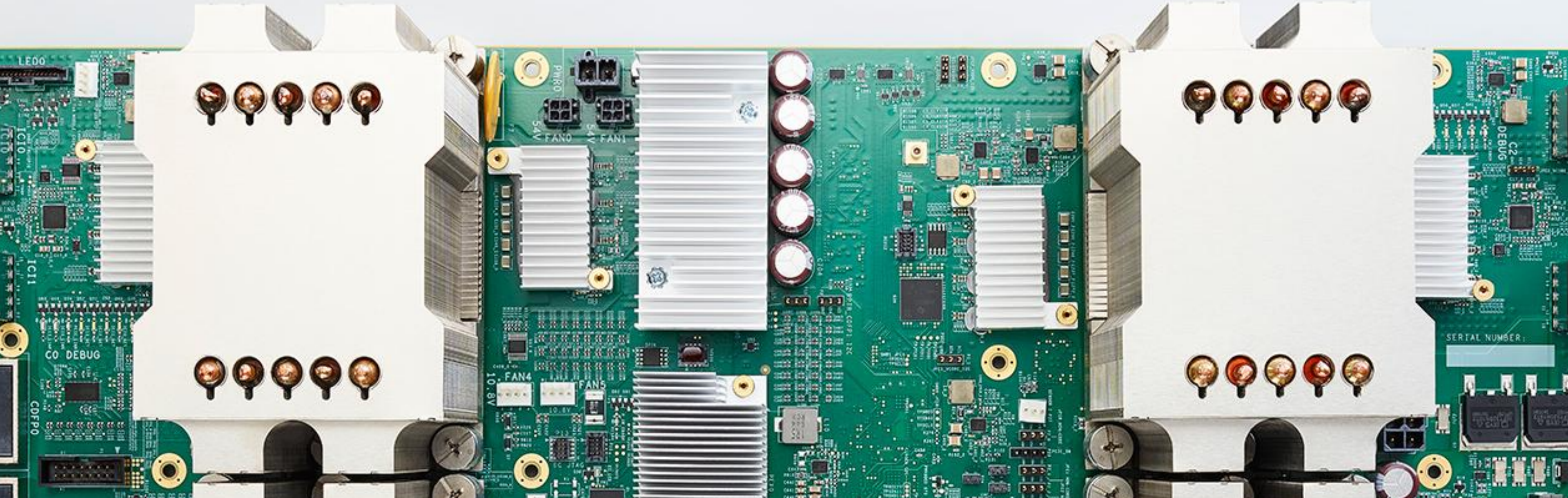
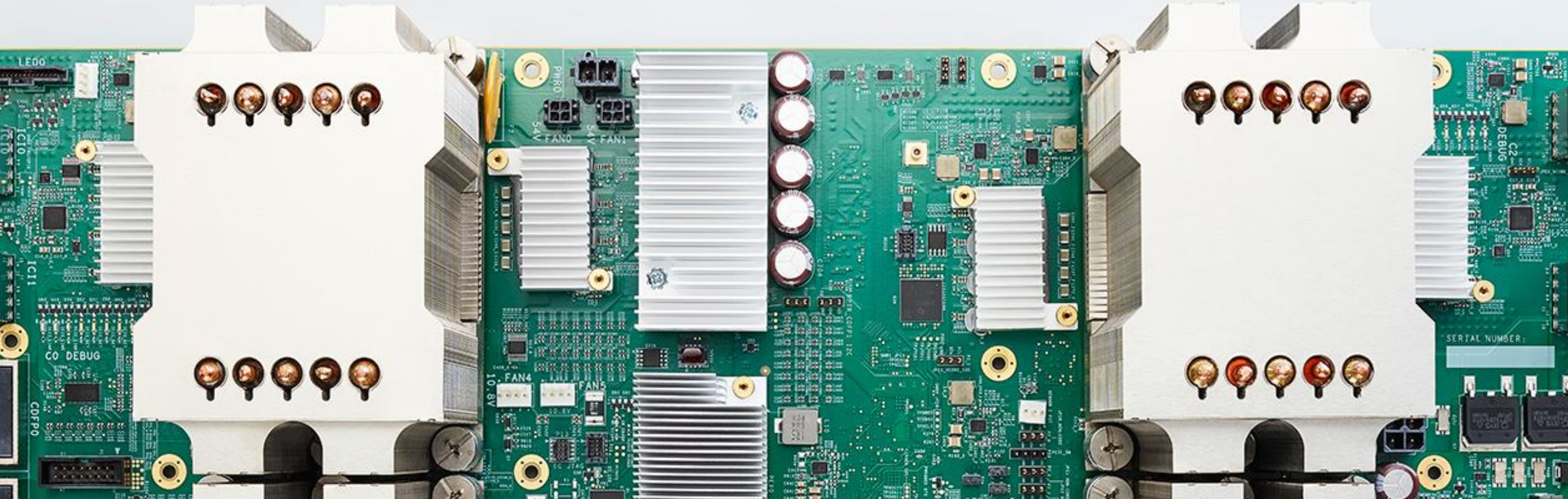
Google's Training Chips Revealed: TPUv2 and TPUv3

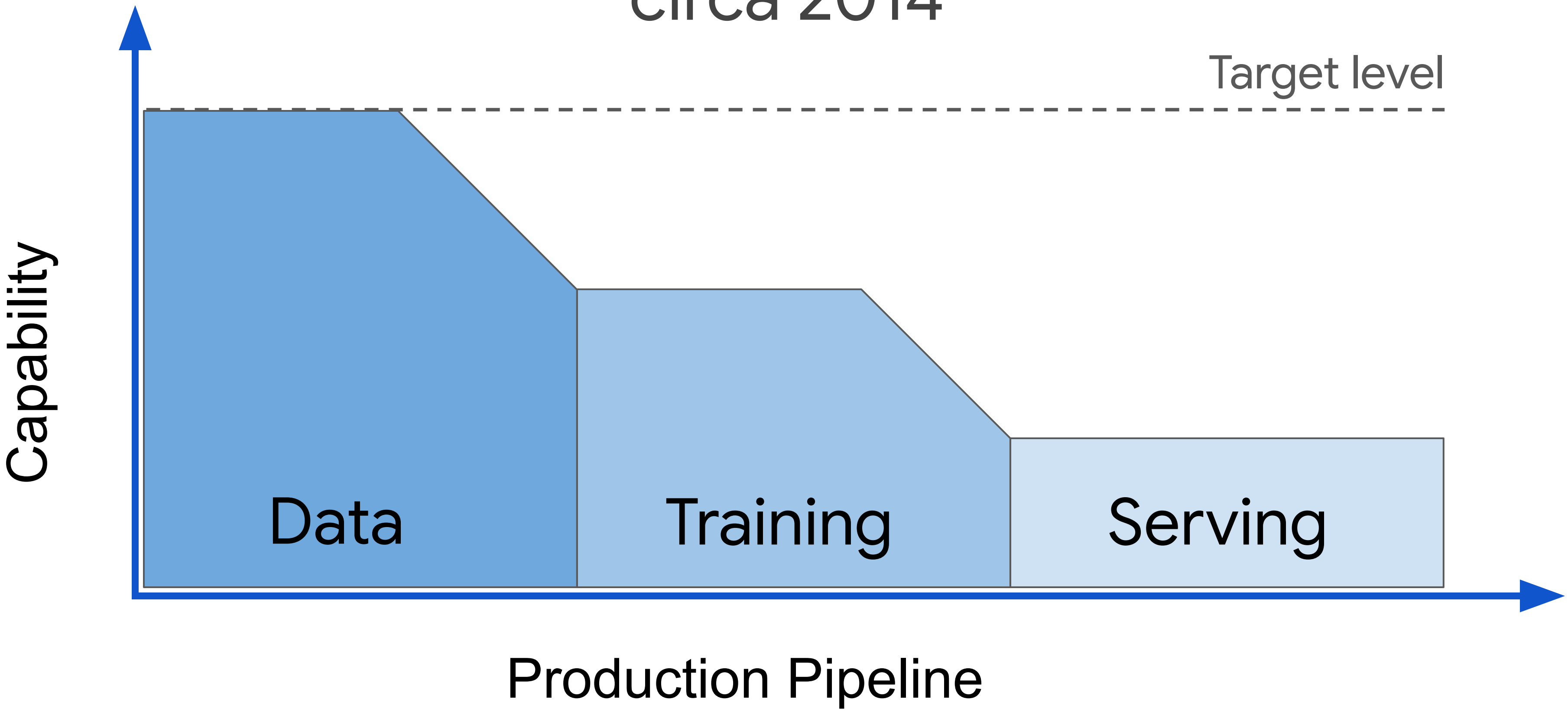
Thomas Norrie, Nishant Patil, Doe Hyun Yoon, George Kurian, Sheng Li, James Laudon, Cliff Young, Norman P. Jouppi, and David Patterson



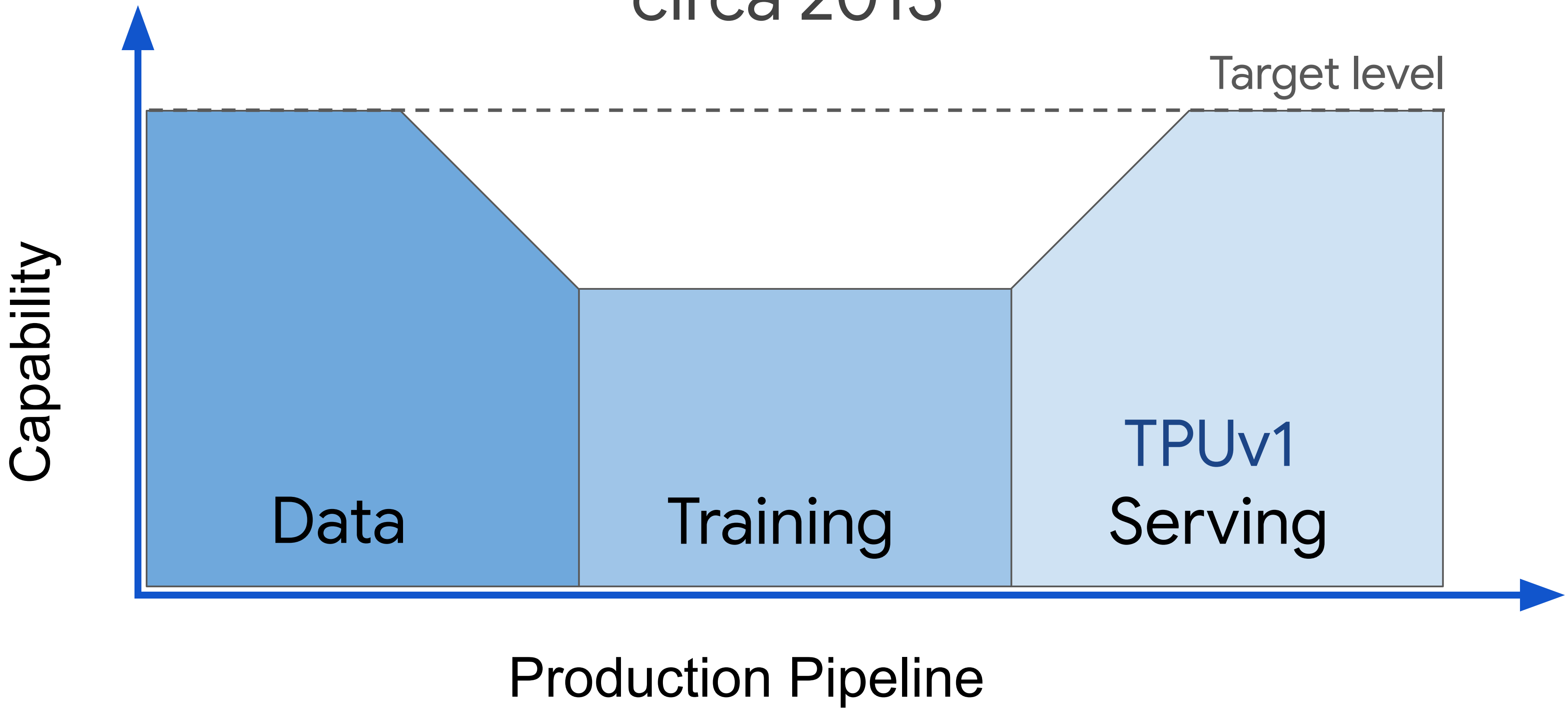
Thanks to the Team



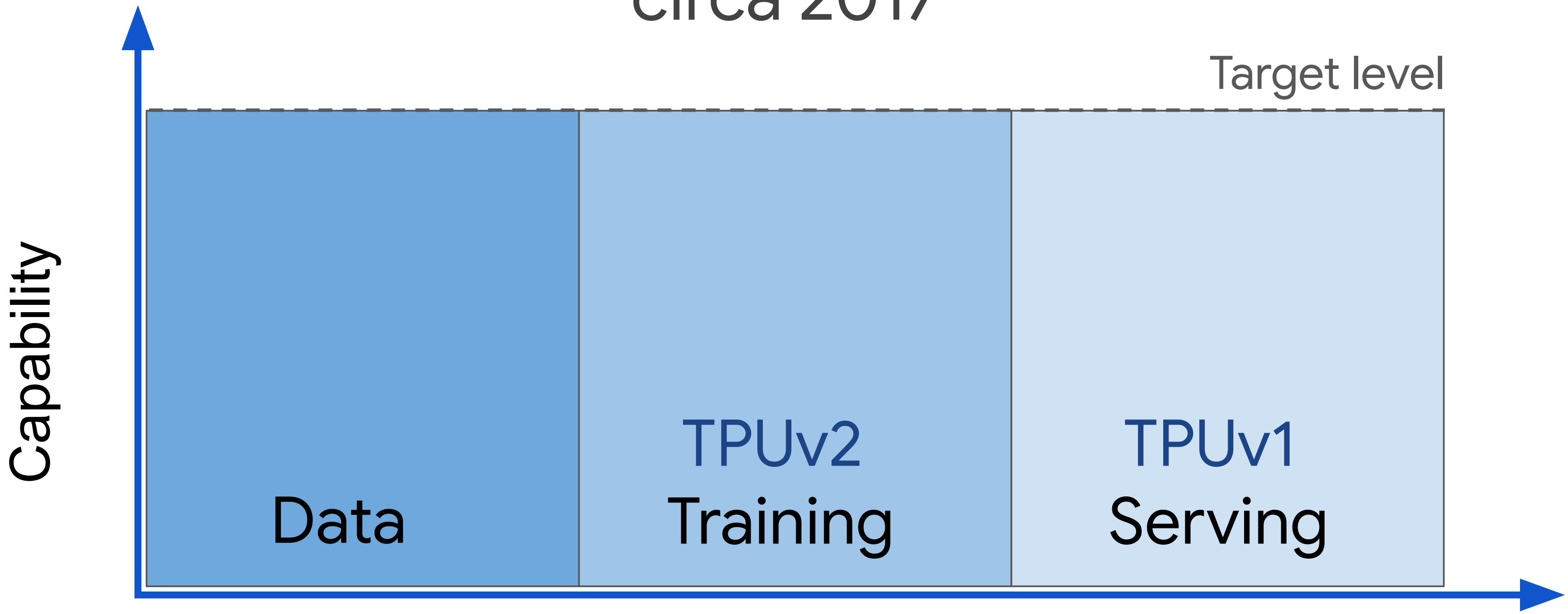
circa 2014


Target level
Capability
Data
Training
Serving
Production Pipeline

circa 2015


Target level
Capability
Data
Training
TPUv1
Serving
Production Pipeline

circa 2017



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
Harder Parallelization	Scale-up instead of scale-out

Our Constraints

Time A development day saved is a deployment day earned.

Staffing We have a lean team.

Key Goals (that is, where will we do a great job?)

Build it quickly

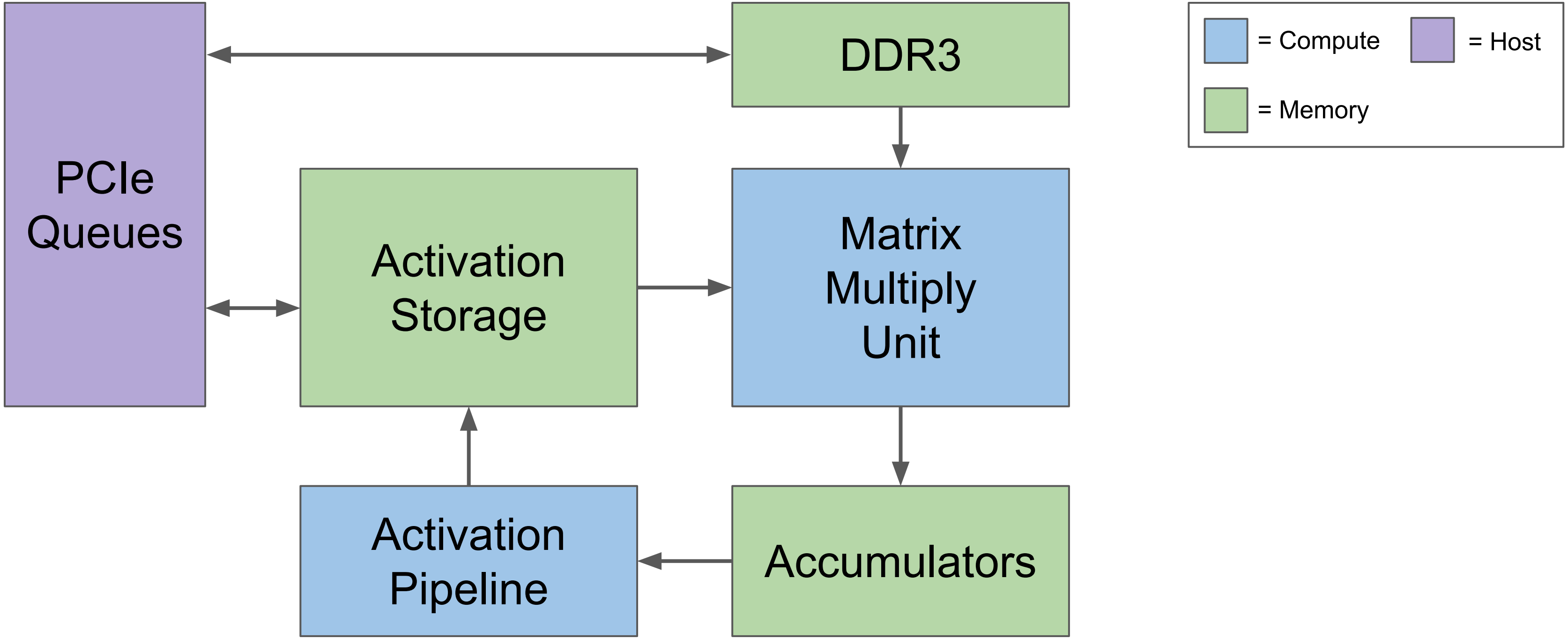
Achieve high performance…

…at scale

…for new workloads out-of-the-box

…all while being cost effective

TPUv1 Recap



PCIe Queues

DDR3

Activation Storage

Matrix Multiply Unit

Activation Pipeline

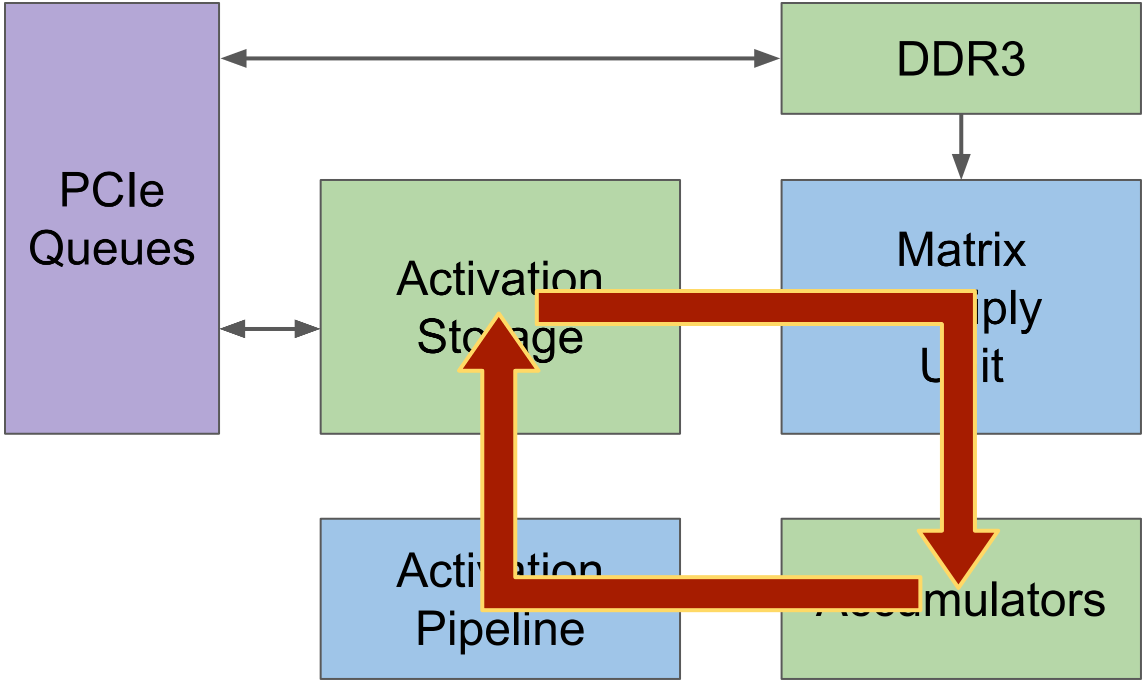
Accumulators

🟦 = Compute

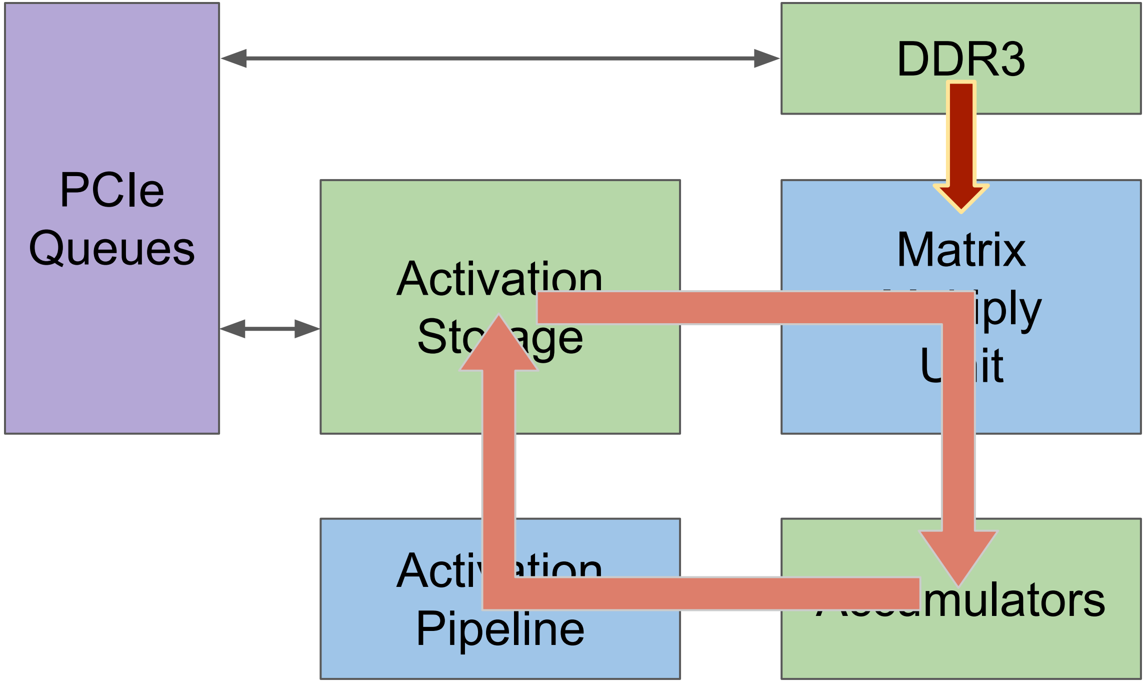
🟪 = Host

🟩 = Memory

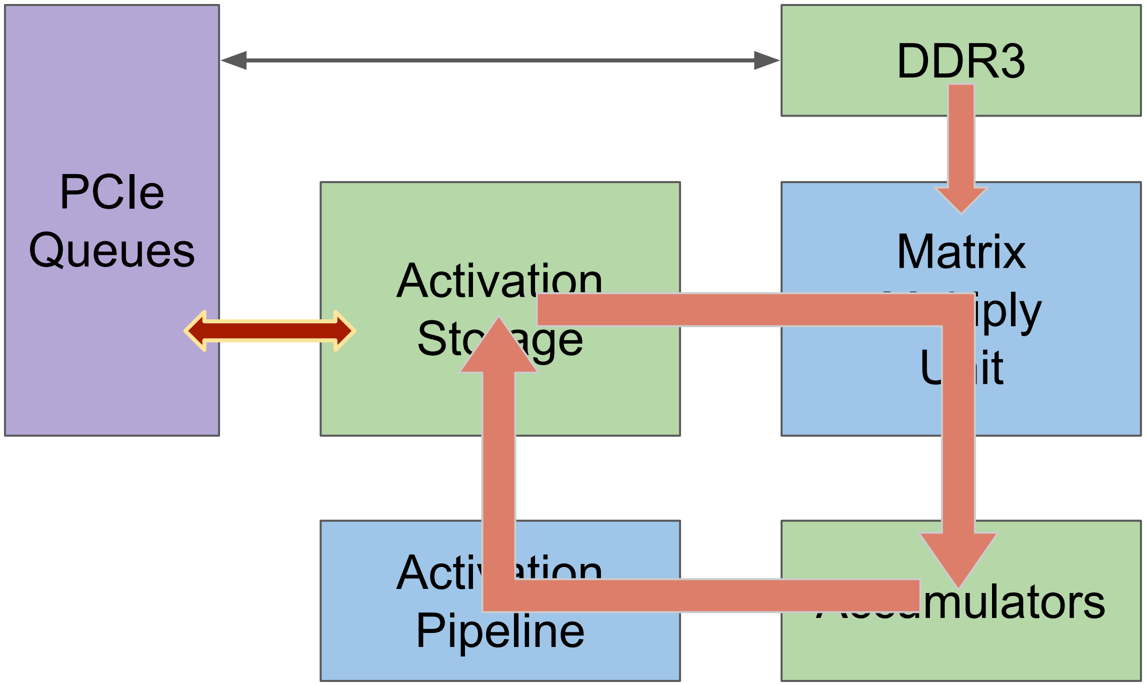
TPUv1 Recap


PCIe Queues
DDR3
Activation Storage
Matrix Multiply Unit
Activation Pipeline
Accumulators

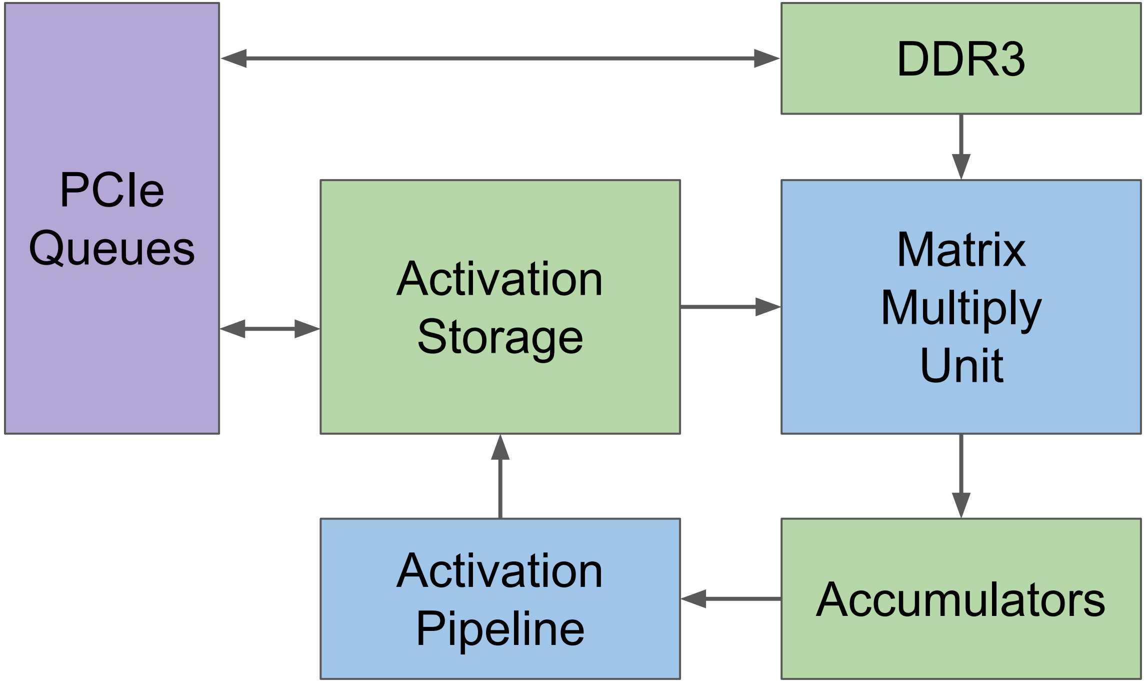
TPUv1 Recap


PCIe Queues
DDR3
Activation Storage
Matrix Multiply Unit
Activation Pipeline
Accumulators

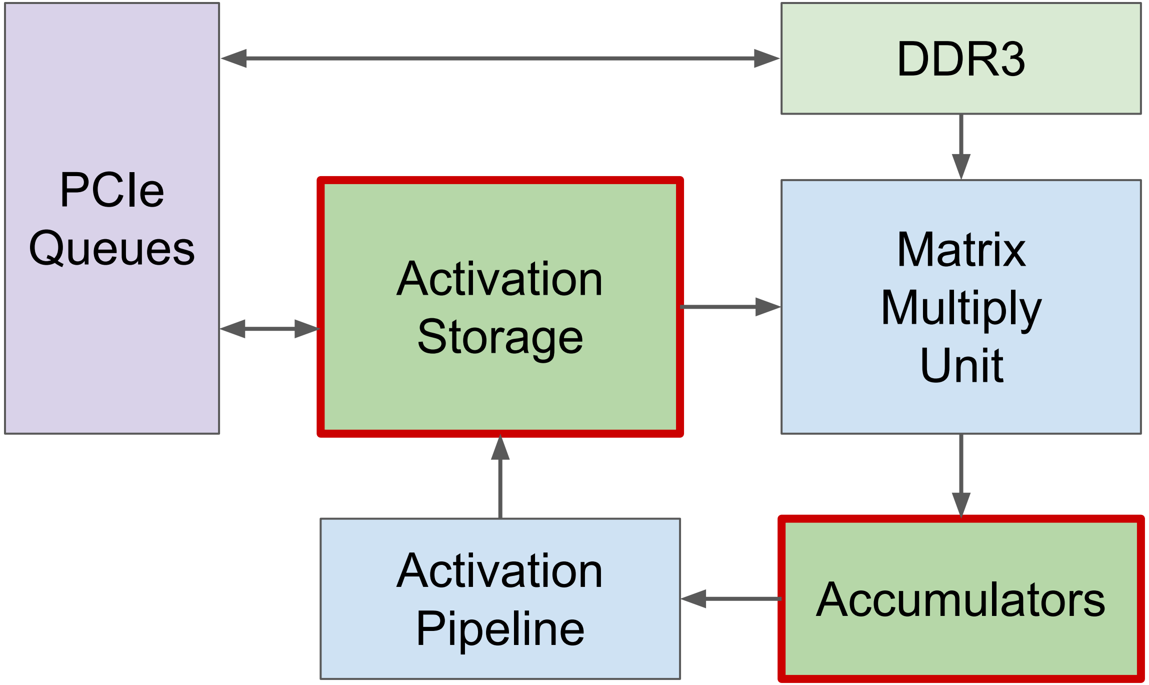
TPUv1 Recap


PCIe Queues
DDR3
Activation Storage
Matrix Multiply Unit
Activation Pipeline
Accumulators

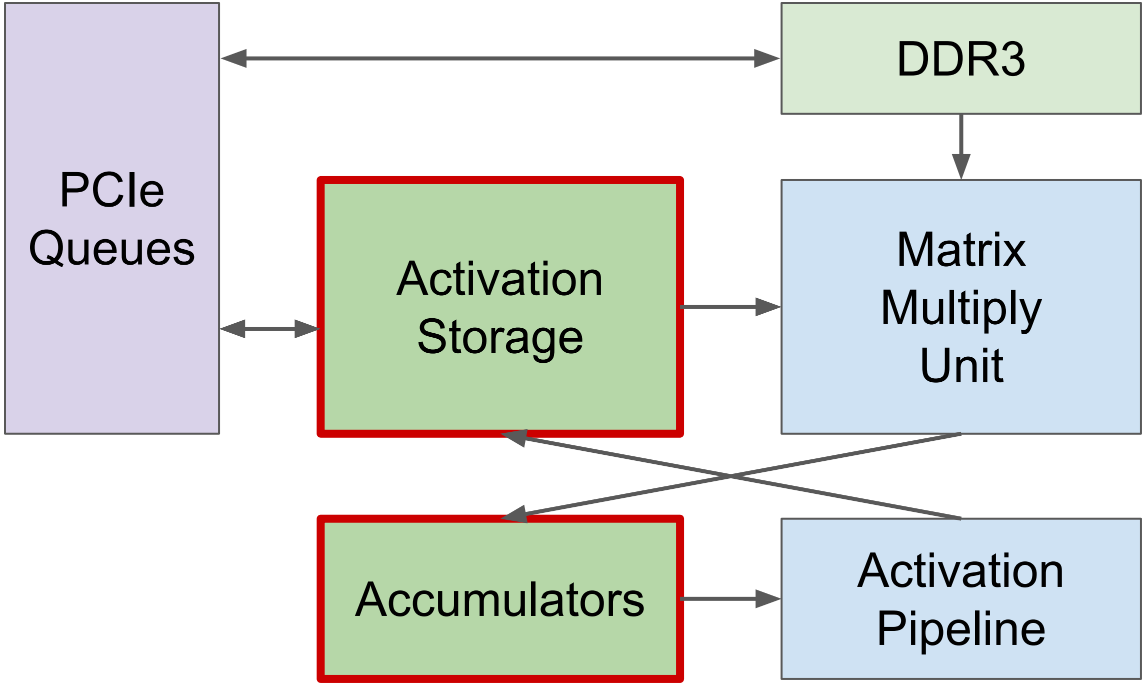
TPUv2 Changes



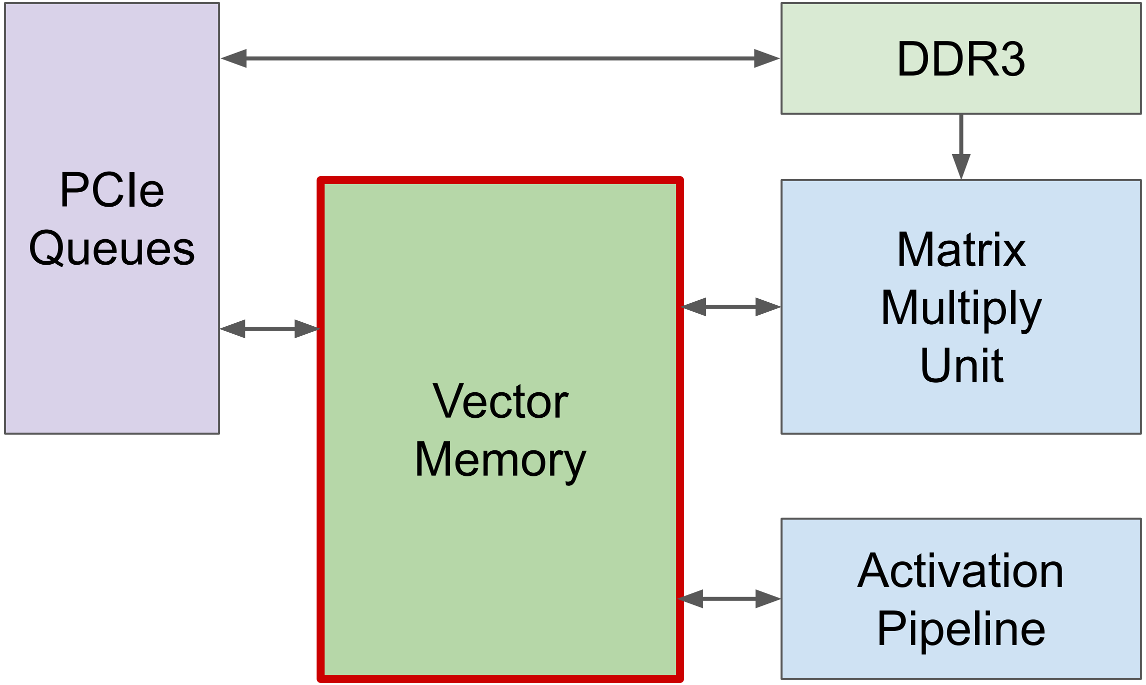
TPUv2 Changes



TPUv2 Changes

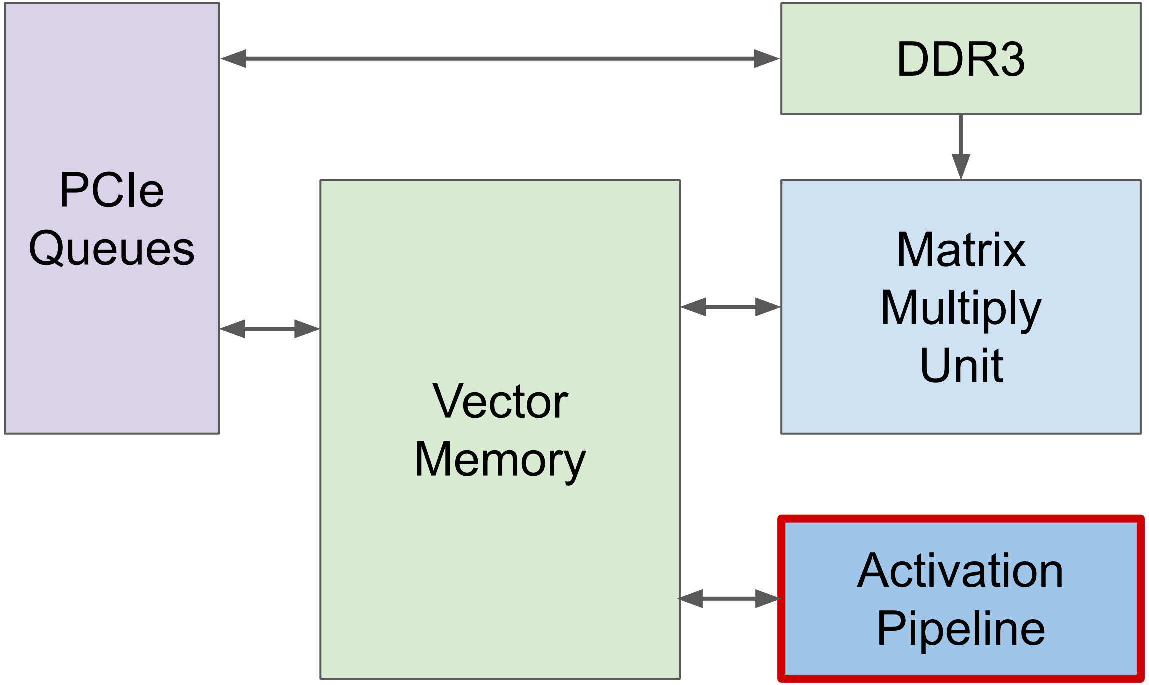


TPUv2 Changes

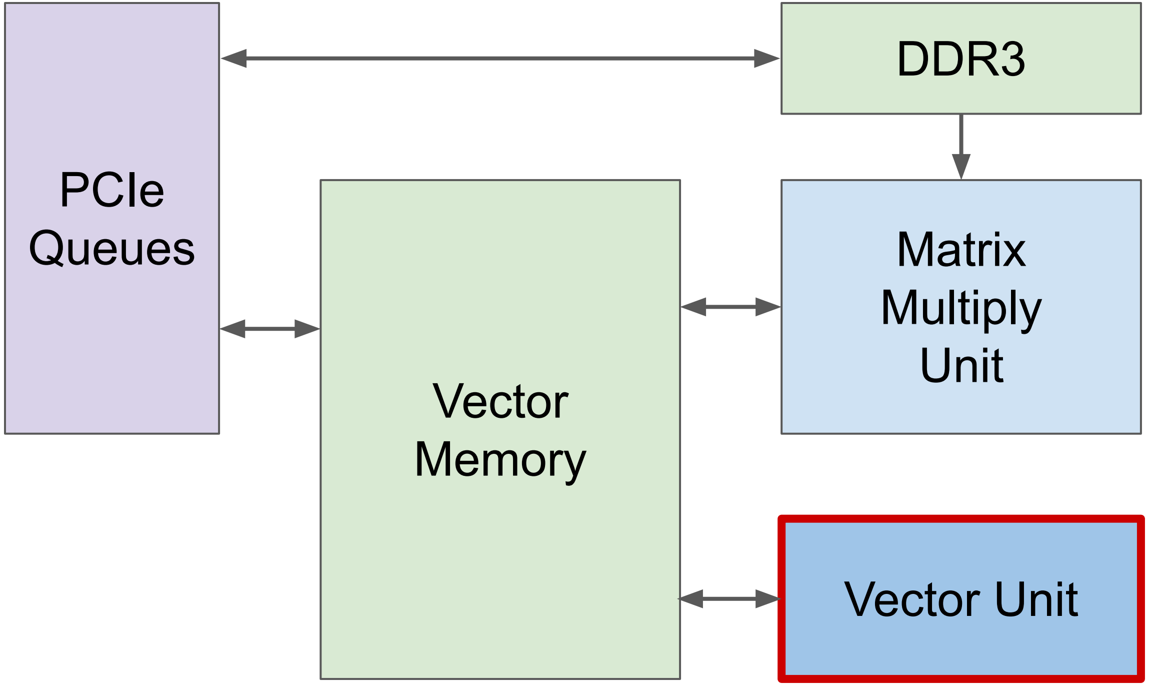


Single vector memory instead of buffers between fixed function units.

TPUv2 Changes

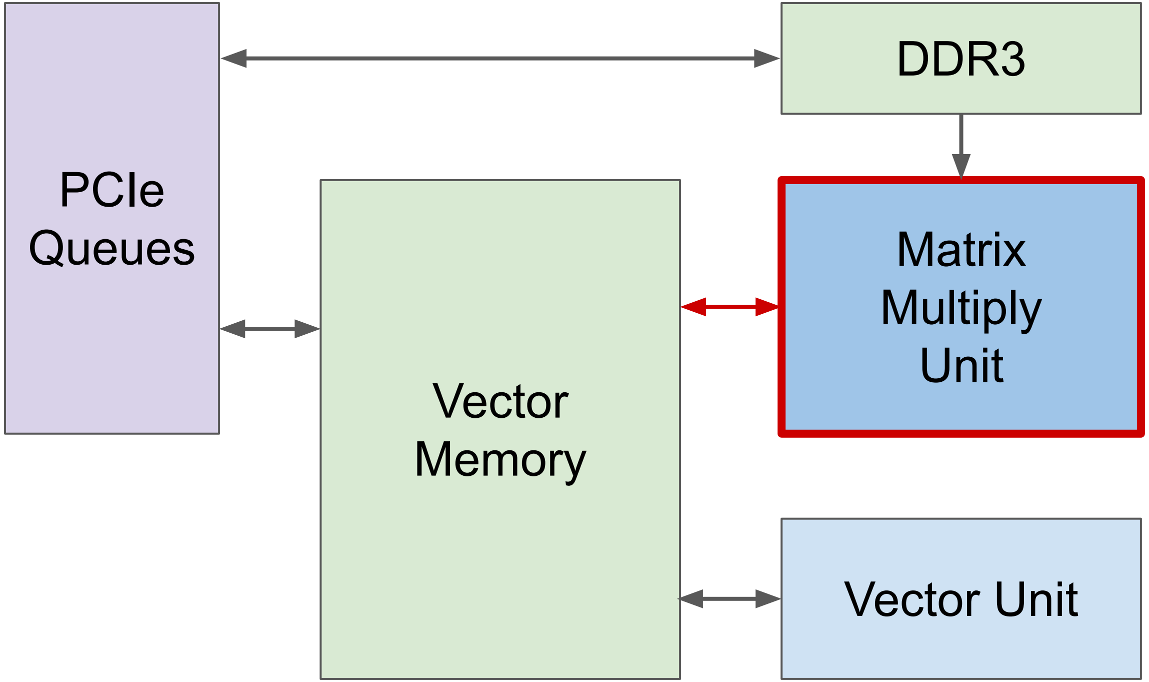


TPUv2 Changes

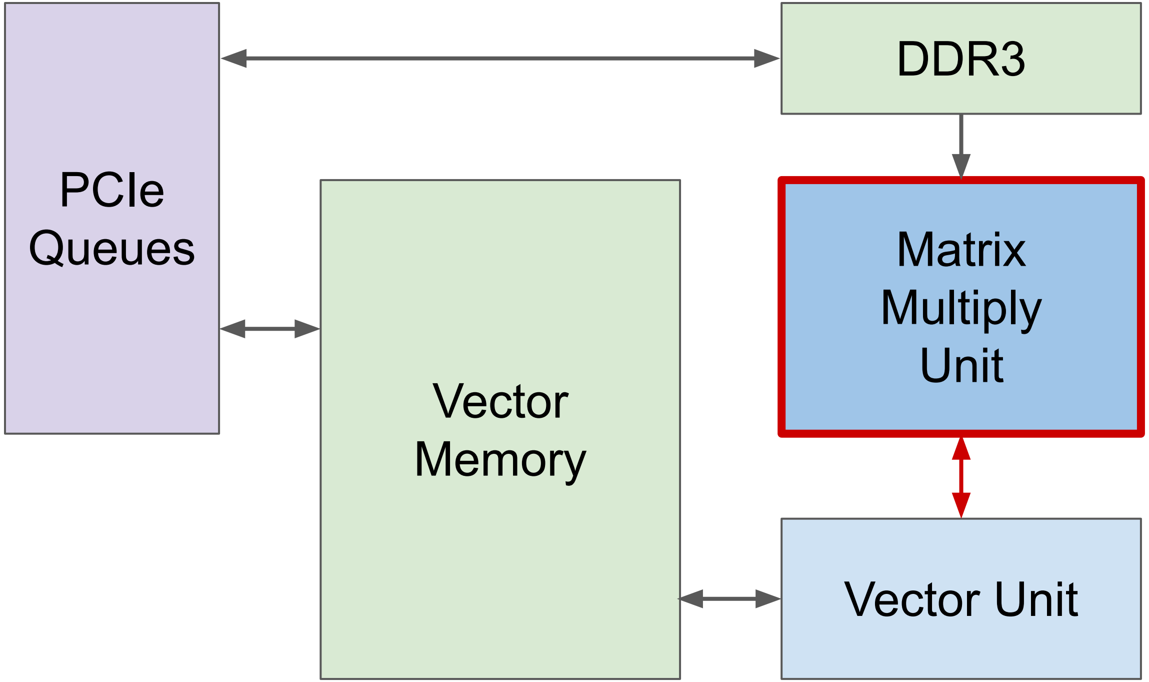


General purpose vector unit instead of a fixed function activation pipeline.

TPUv2 Changes

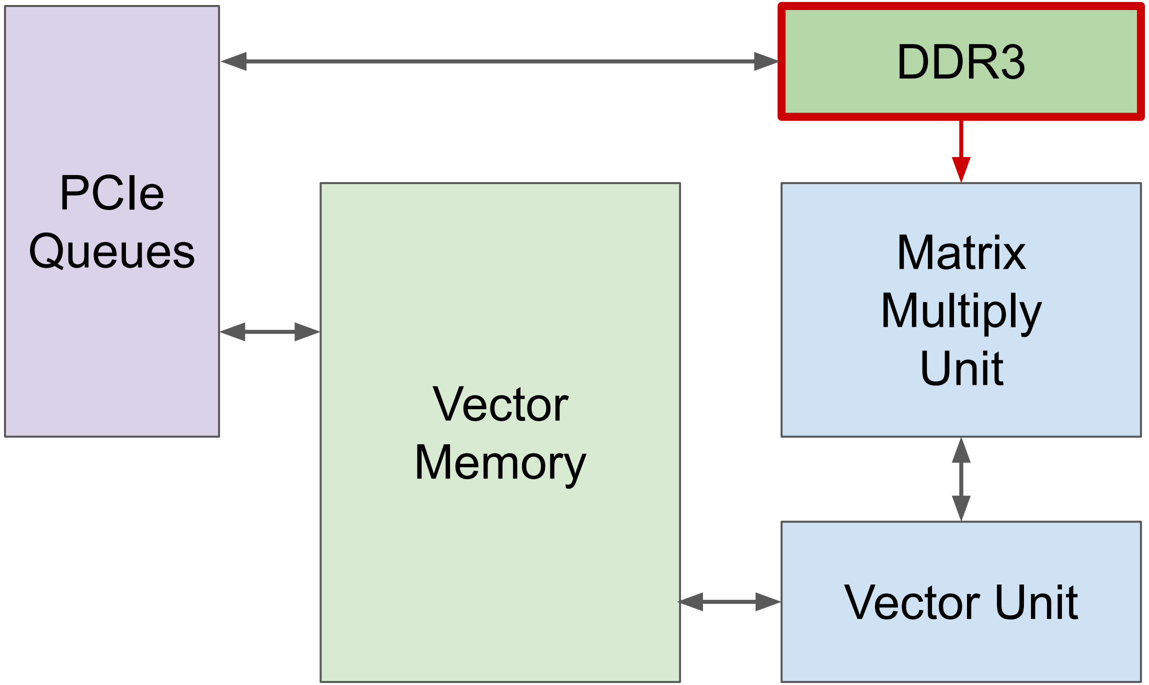


TPUv2 Changes

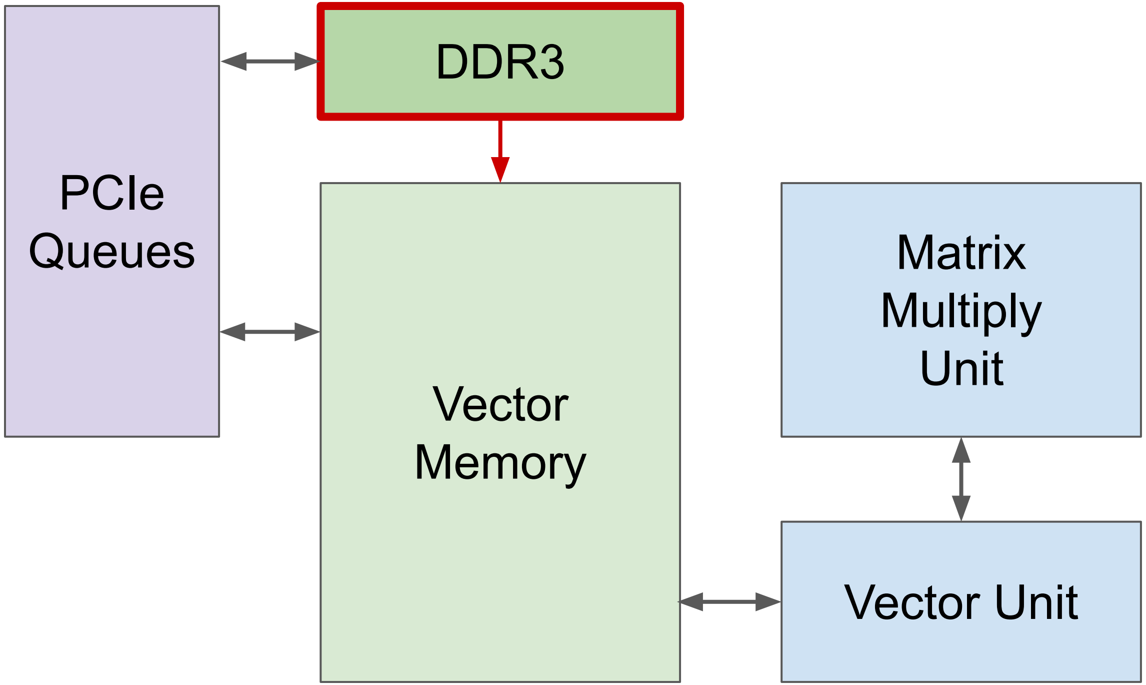


Connect matrix unit as an offload for the vector unit

TPUv2 Changes

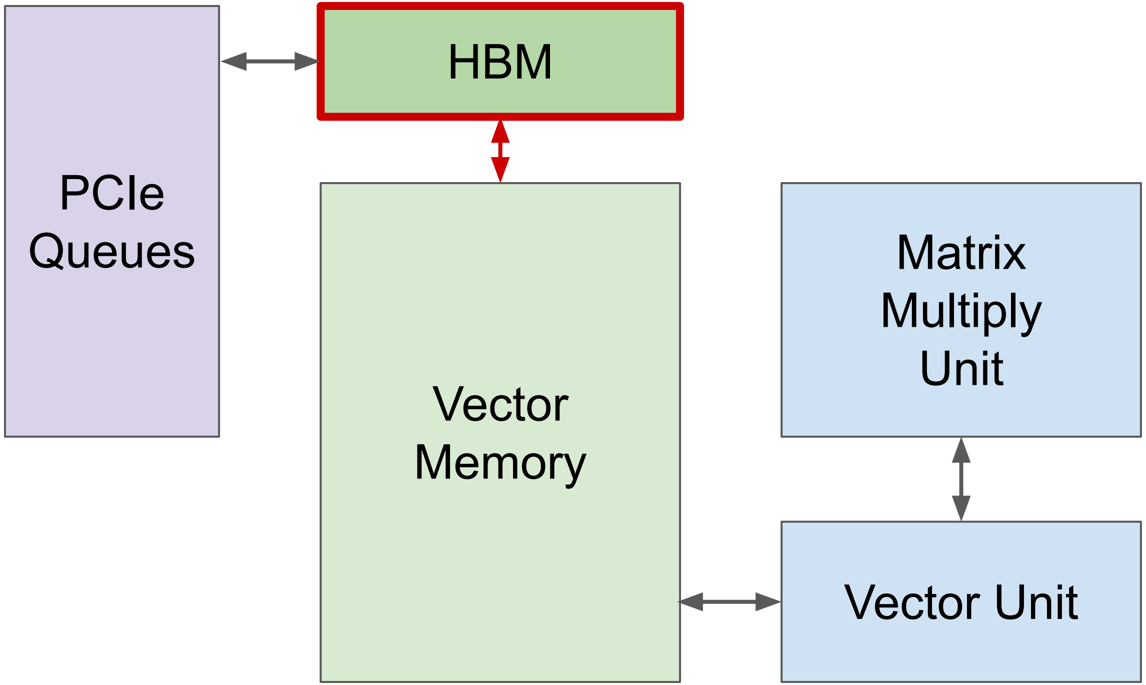


TPUv2 Changes



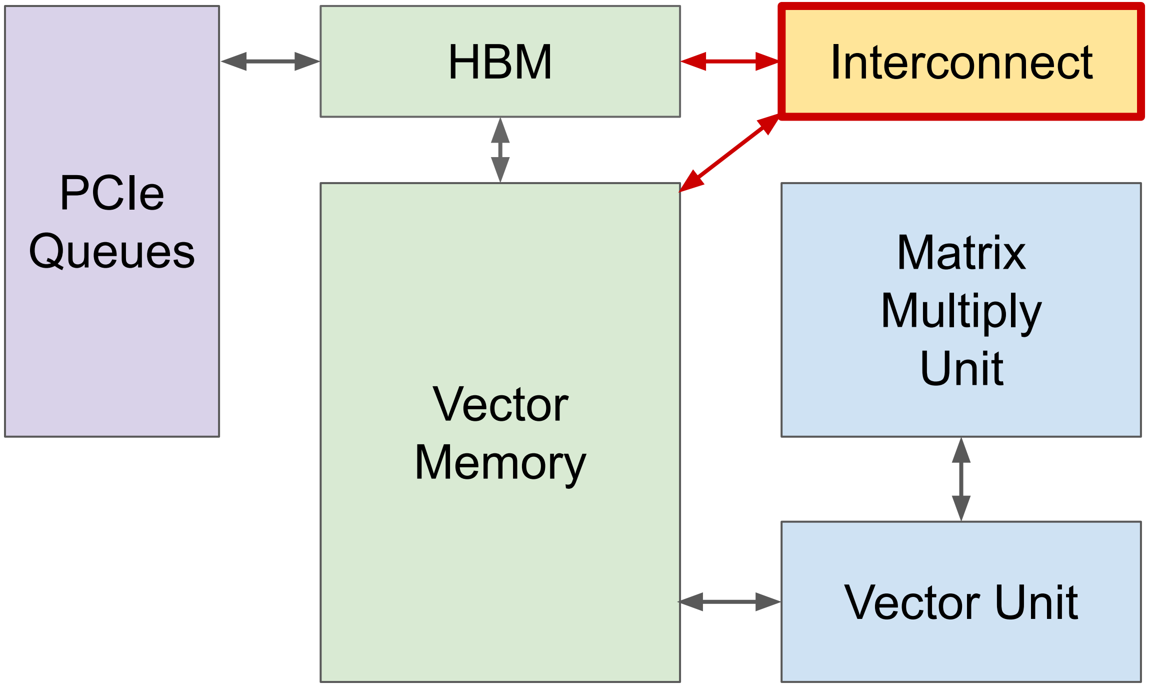
Connect DRAM into the memory system instead of directly into the matrix unit

TPUv2 Changes



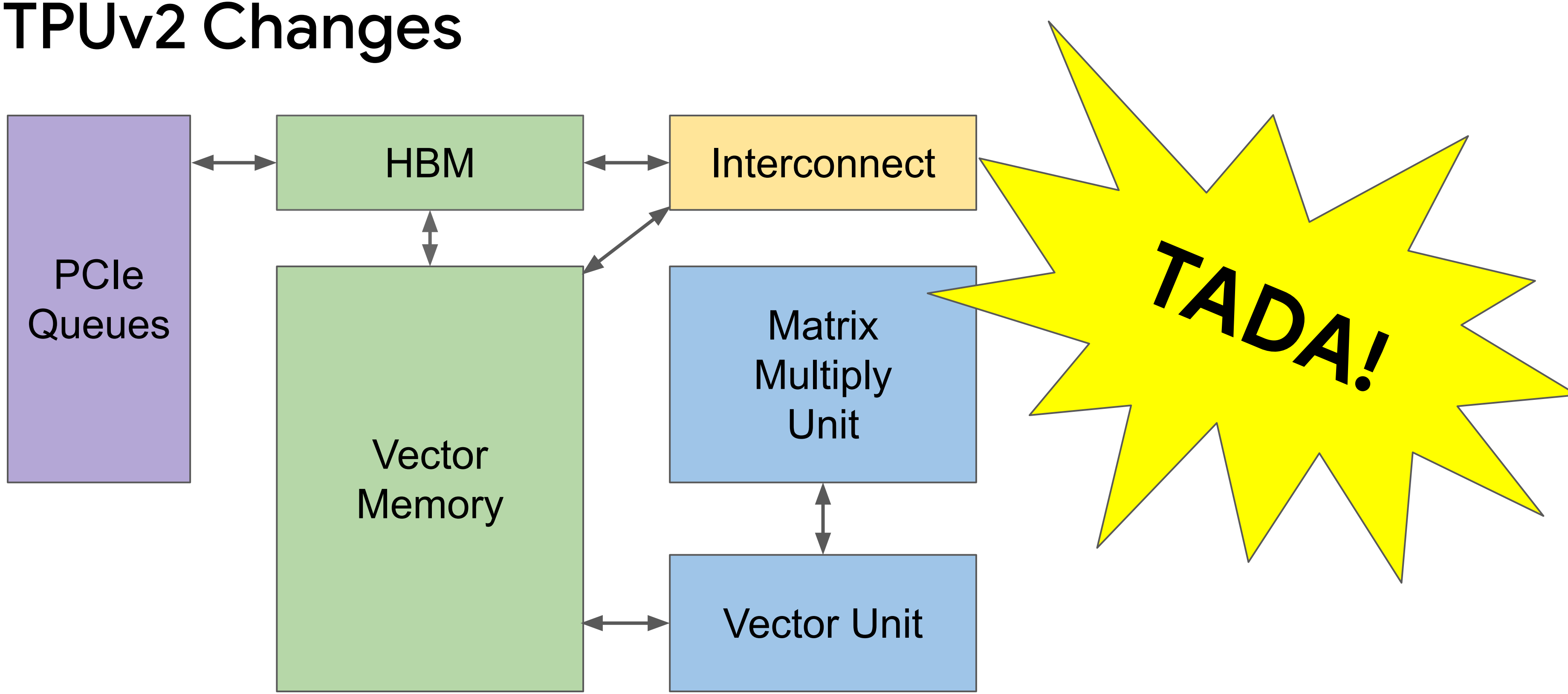
Move to HBM for bandwidth

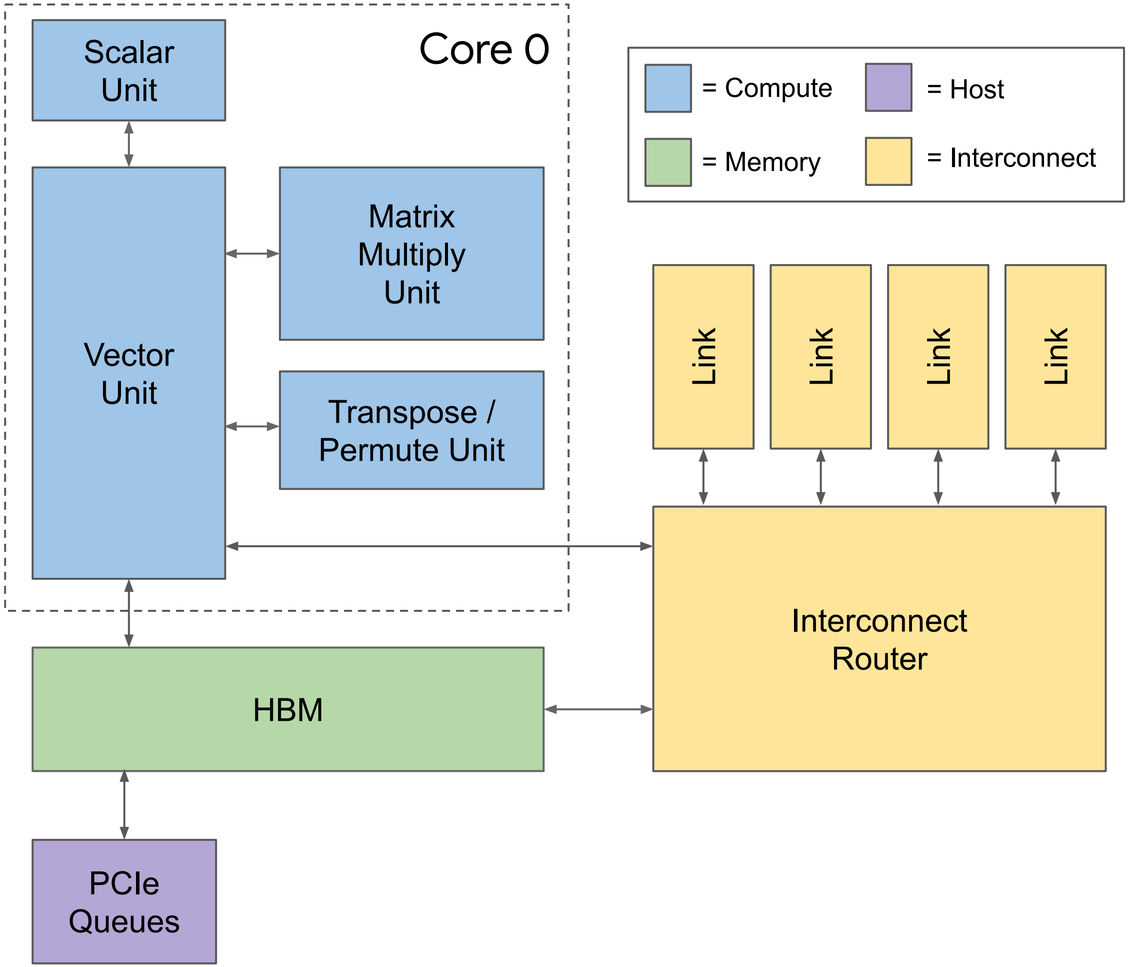
TPUv2 Changes



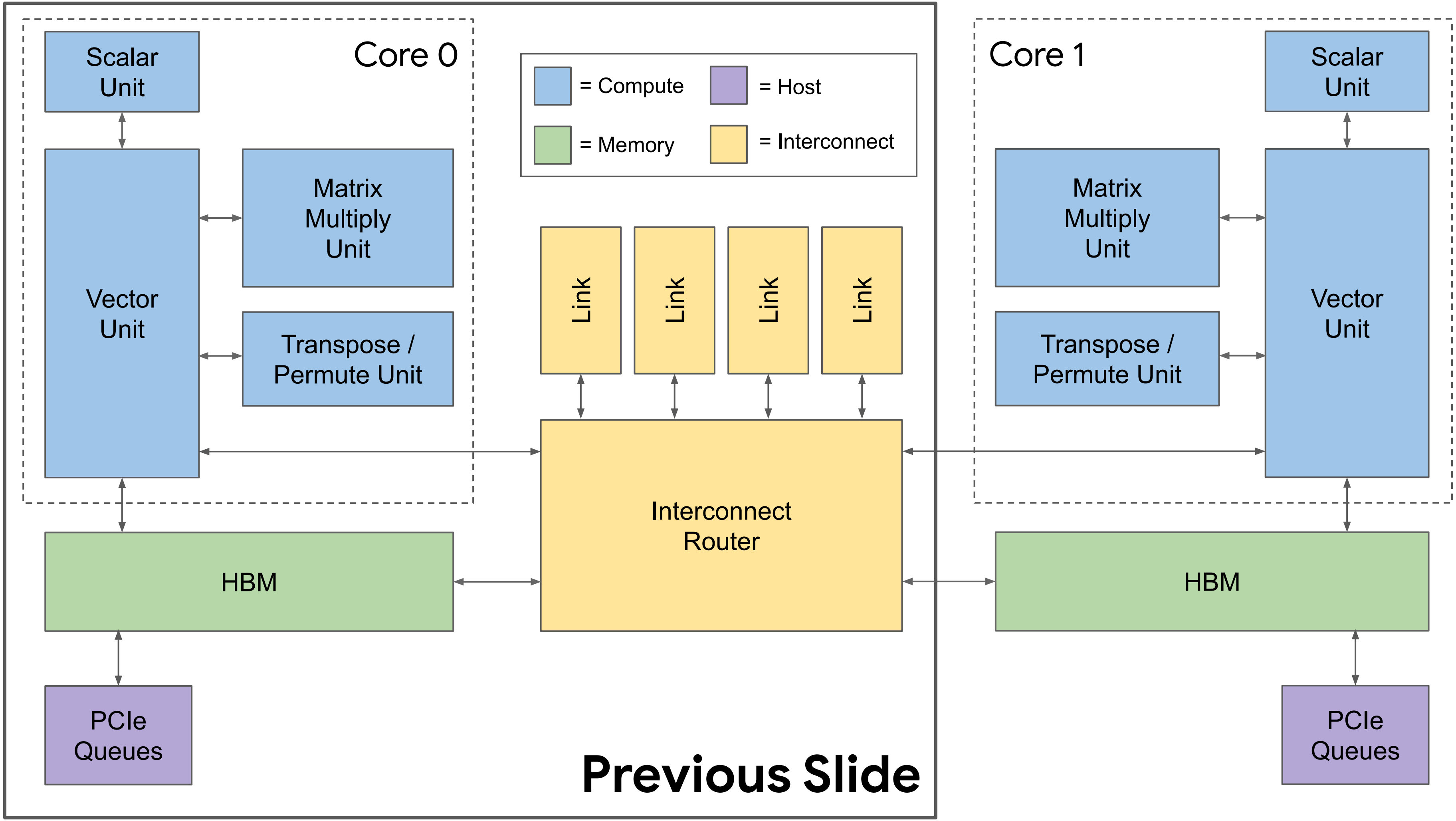
Add interconnect for high-bandwidth scaling

TPUv2 Changes





Redrawn with more detail...



Core 0

Scalar Unit

Vector Unit

Matrix Multiply Unit

Transpose / Permute Unit

HBM

PCIe Queues

🟦 = Compute

🟪 = Host

🟩 = Memory

🟨 = Interconnect

Link

Link

Link

Link

Interconnect Router

Core 1

Scalar Unit

Matrix Multiply Unit

Transpose / Permute Unit

Vector Unit

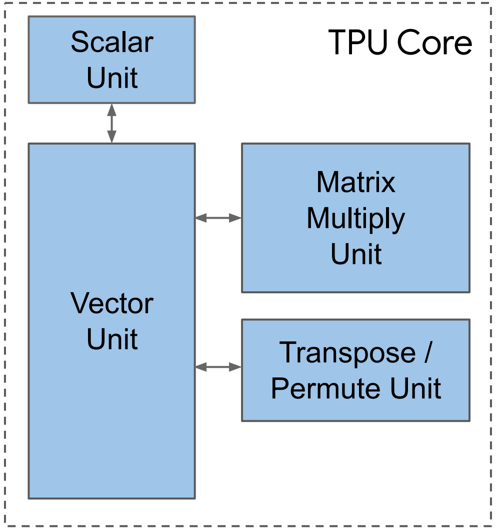
HBM

PCIe Queues

Previous Slide

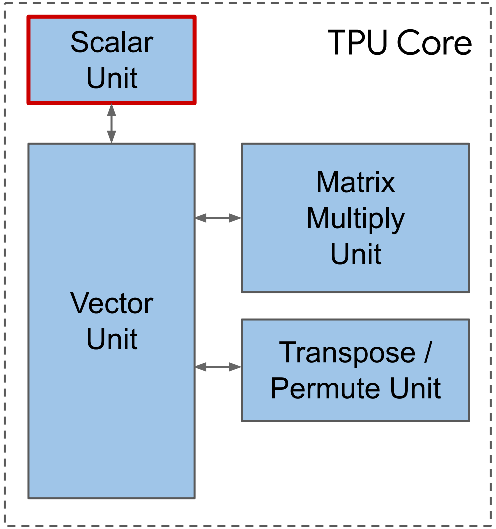
TPU Core

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

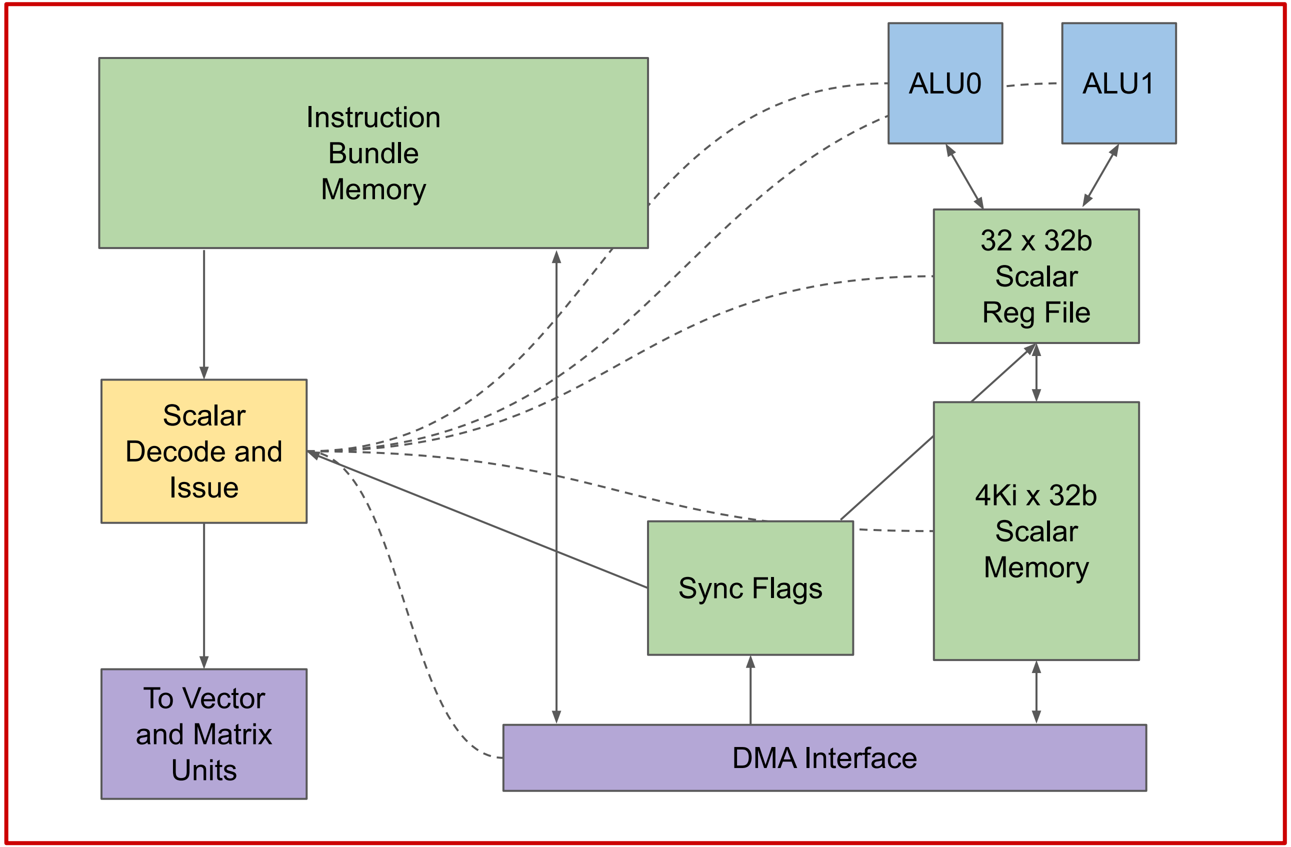


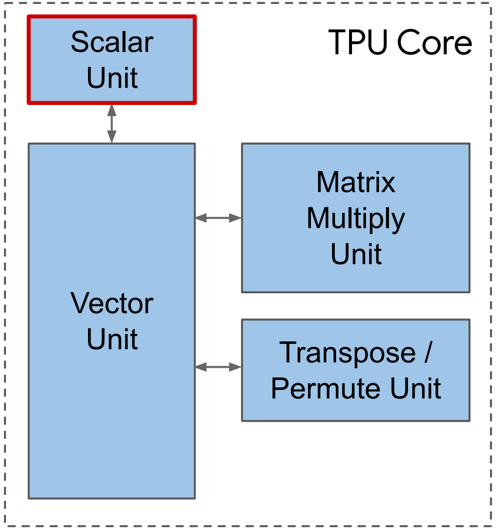
TPU Core: Scalar Unit

- 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

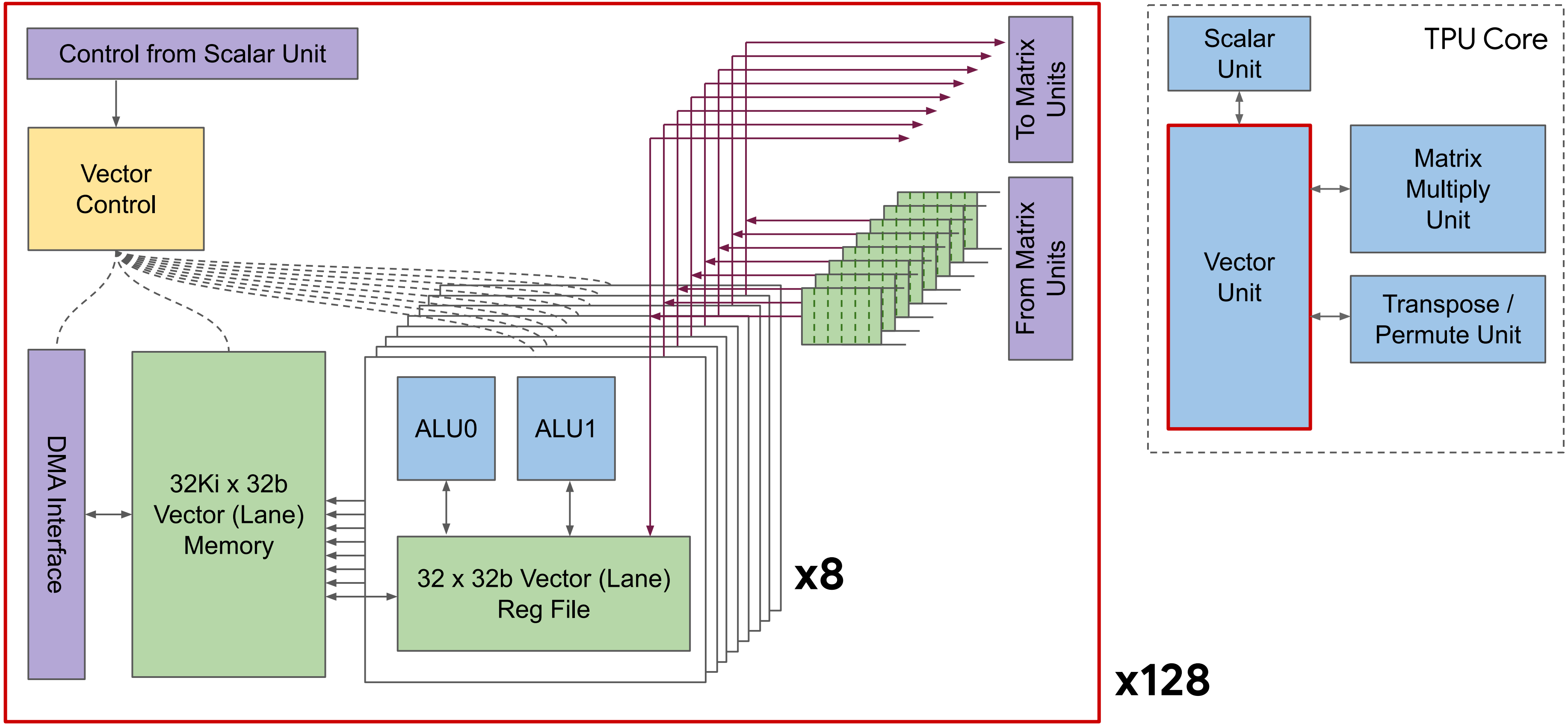


TPU Core: Scalar Unit



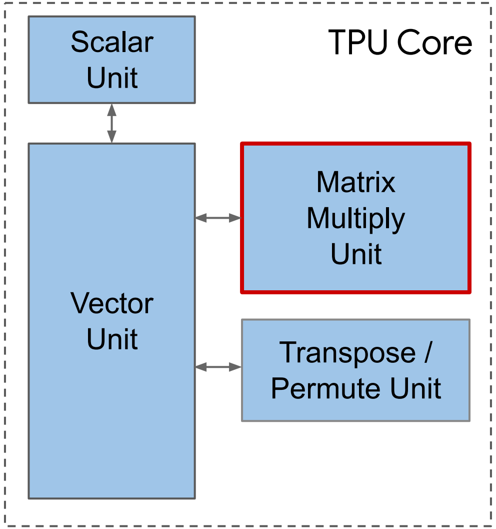


TPU Core: Vector Unit (Lane)

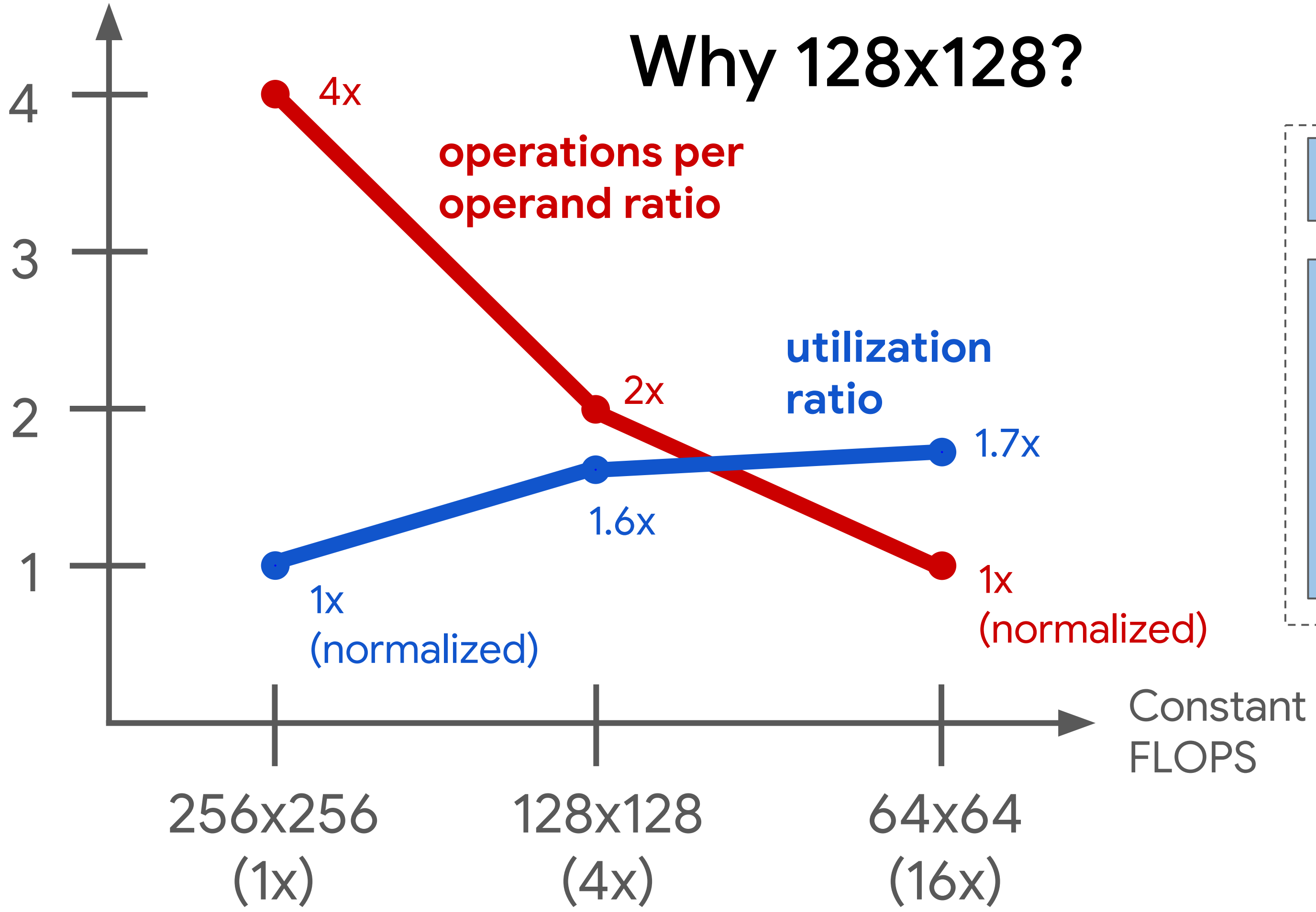


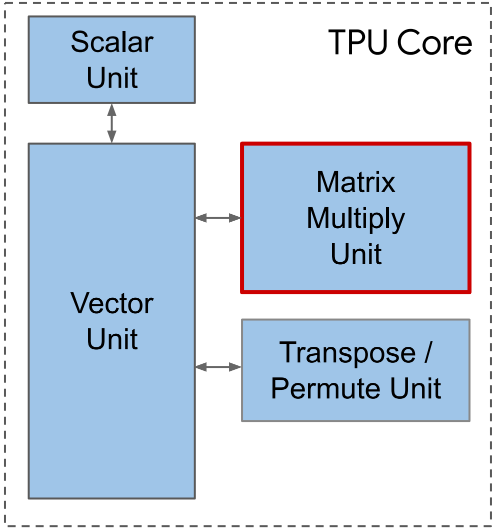
TPU Core: Matrix Multiply Unit

- 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



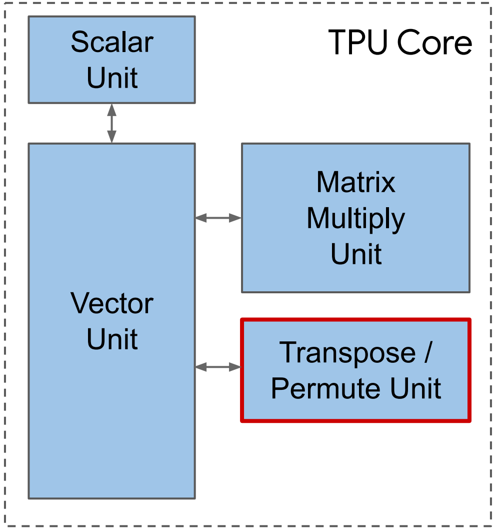
Why 128x128?


4
3
2
1
4x
operations per operand ratio
2x
utilization ratio
1.7x
1.6x
1x (normalized)
1x (normalized)
Constant FLOPS
256x256 (1x)
128x128 (4x)
64x64 (16x)

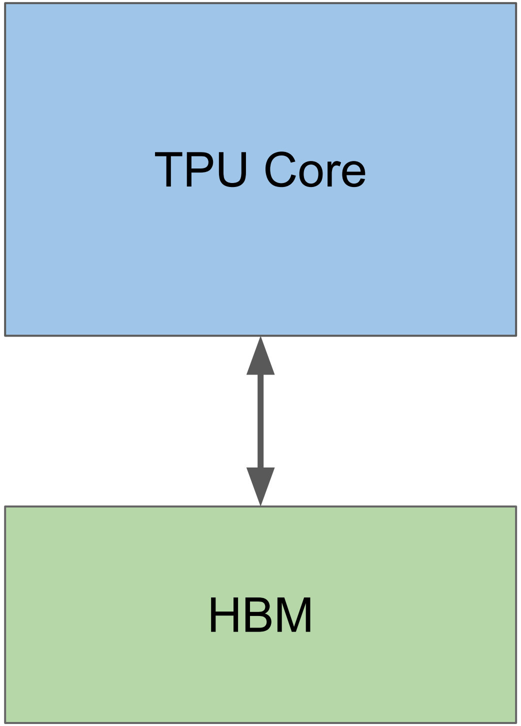

TPU Core
Scalar Unit
Vector Unit
Matrix Multiply Unit
Transpose / Permute Unit

TPU Core: Transpose, Reduction, Permute Unit

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



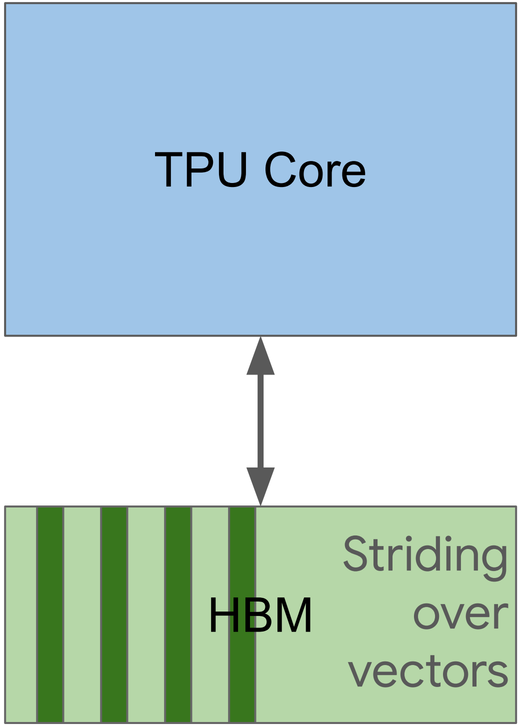
Memory System



- 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

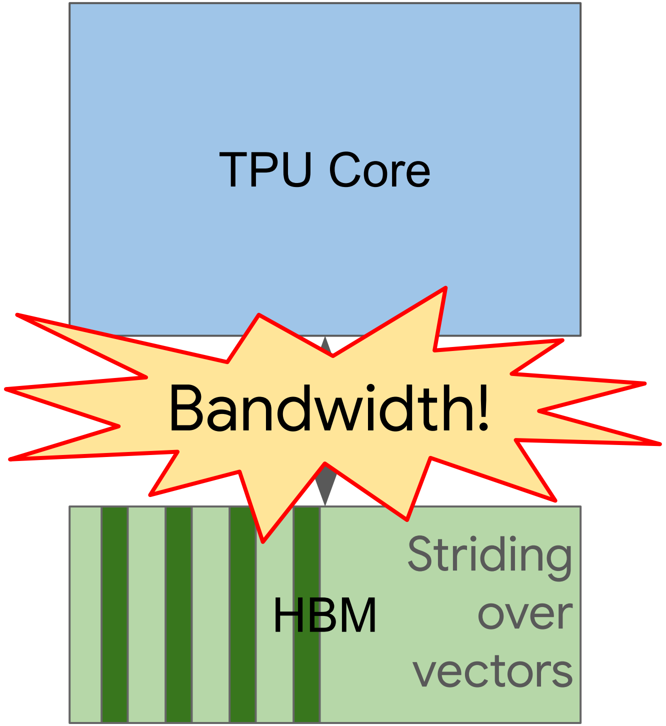
Memory System



- 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

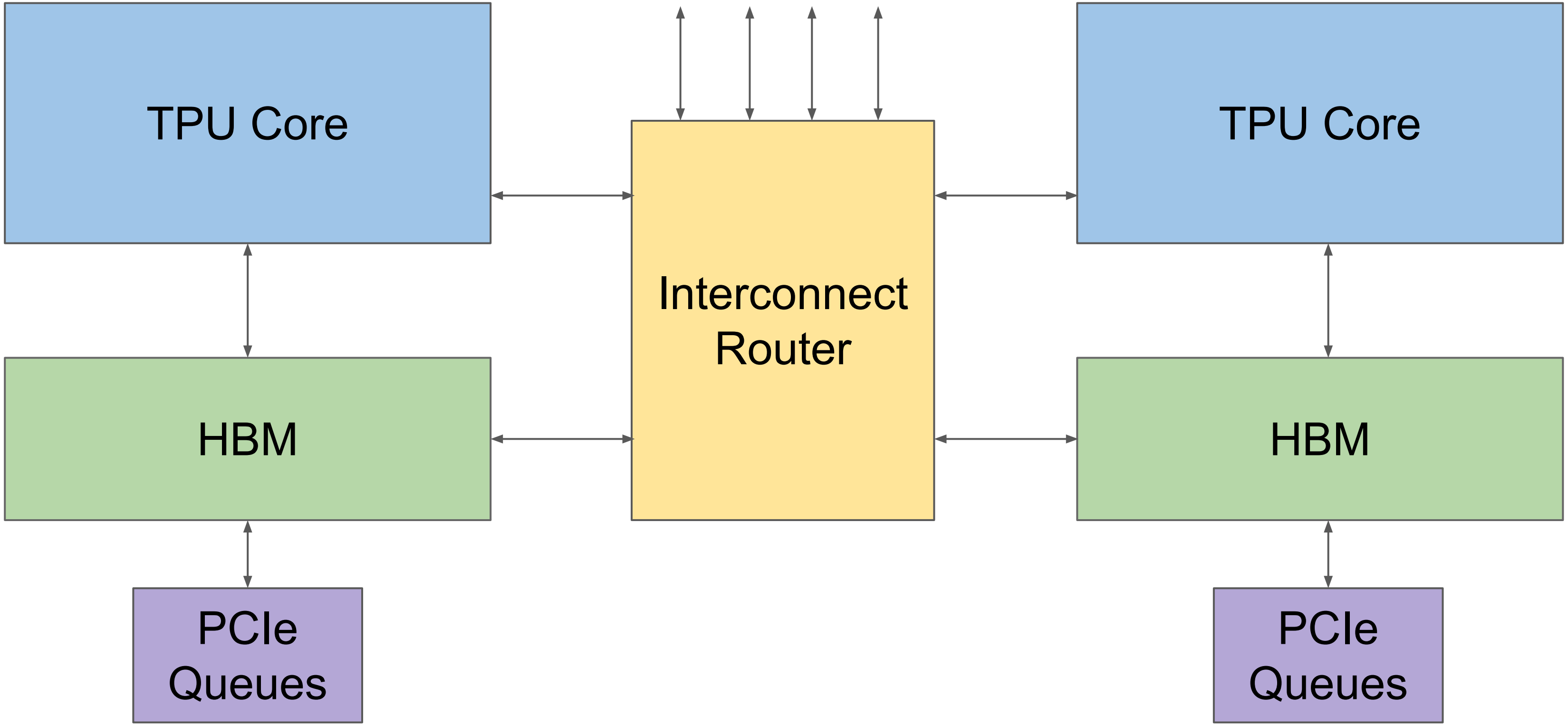
Memory System



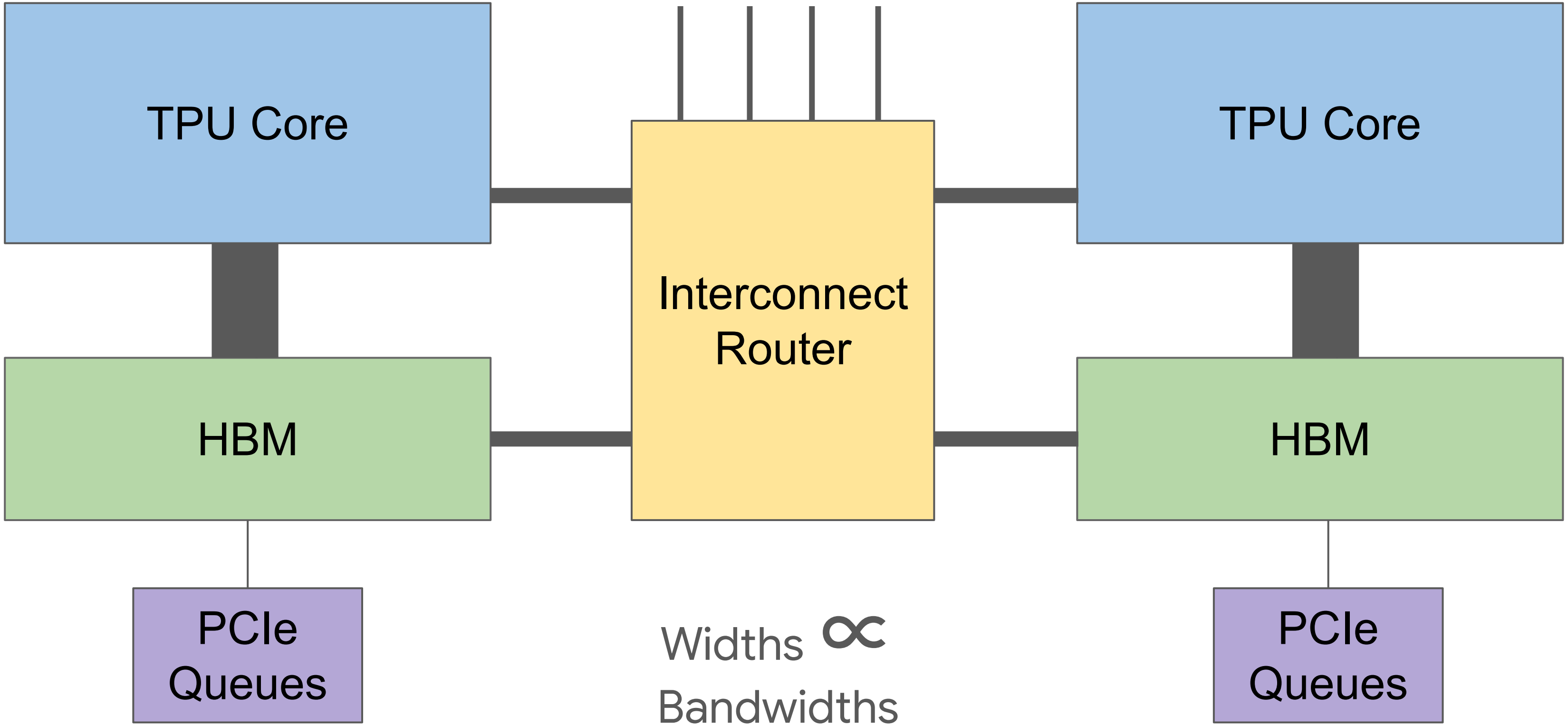
- Loads and stores against SRAM scratchpads
- Provides predictable scheduling within the core
- Can stall on sync flags

- Accessible through asynchronous DMAs
- Indicate completion in sync flags

Memory System

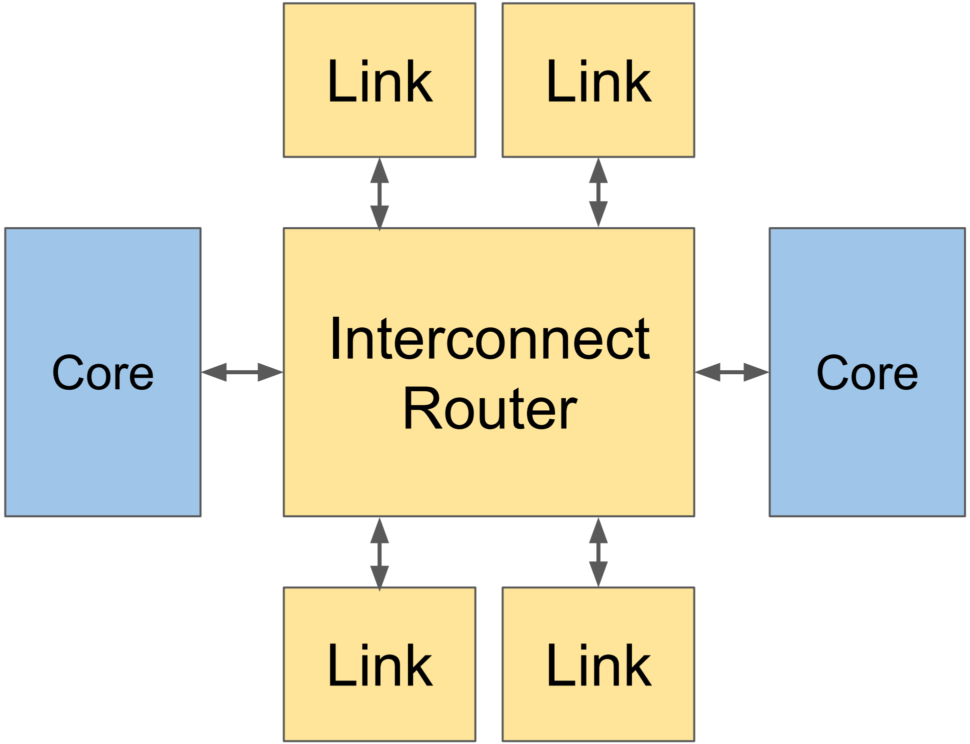


Memory System



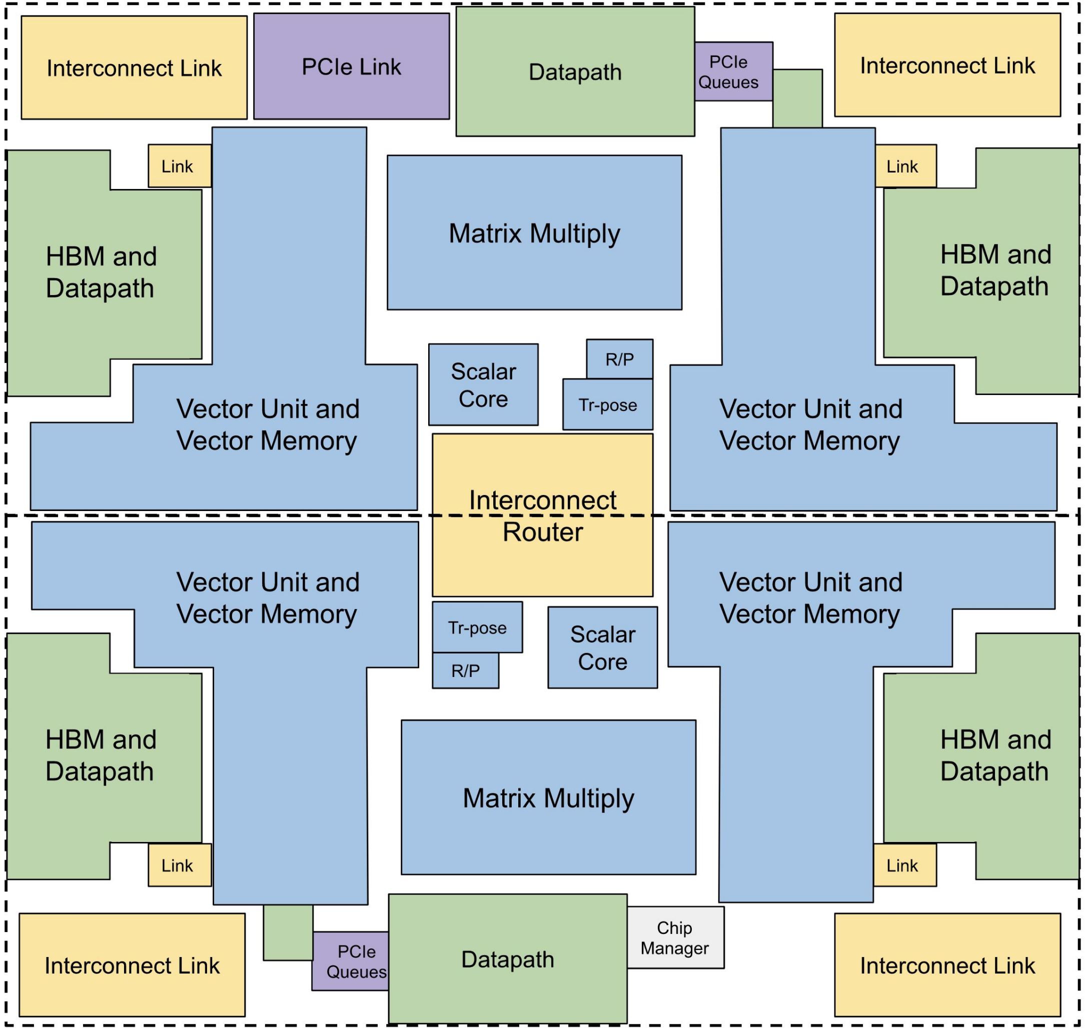
Widths ∝ Bandwidths

Interconnect



- On-die router with 4 links
- 500 Gbps per link
- Assembled into 2D torus
- Software view:
  - Uses DMAs just like HBM
  - Restricted to push DMAs
  - Simply target another chip id

Floorplan



🟦 = Compute

🟩 = Memory

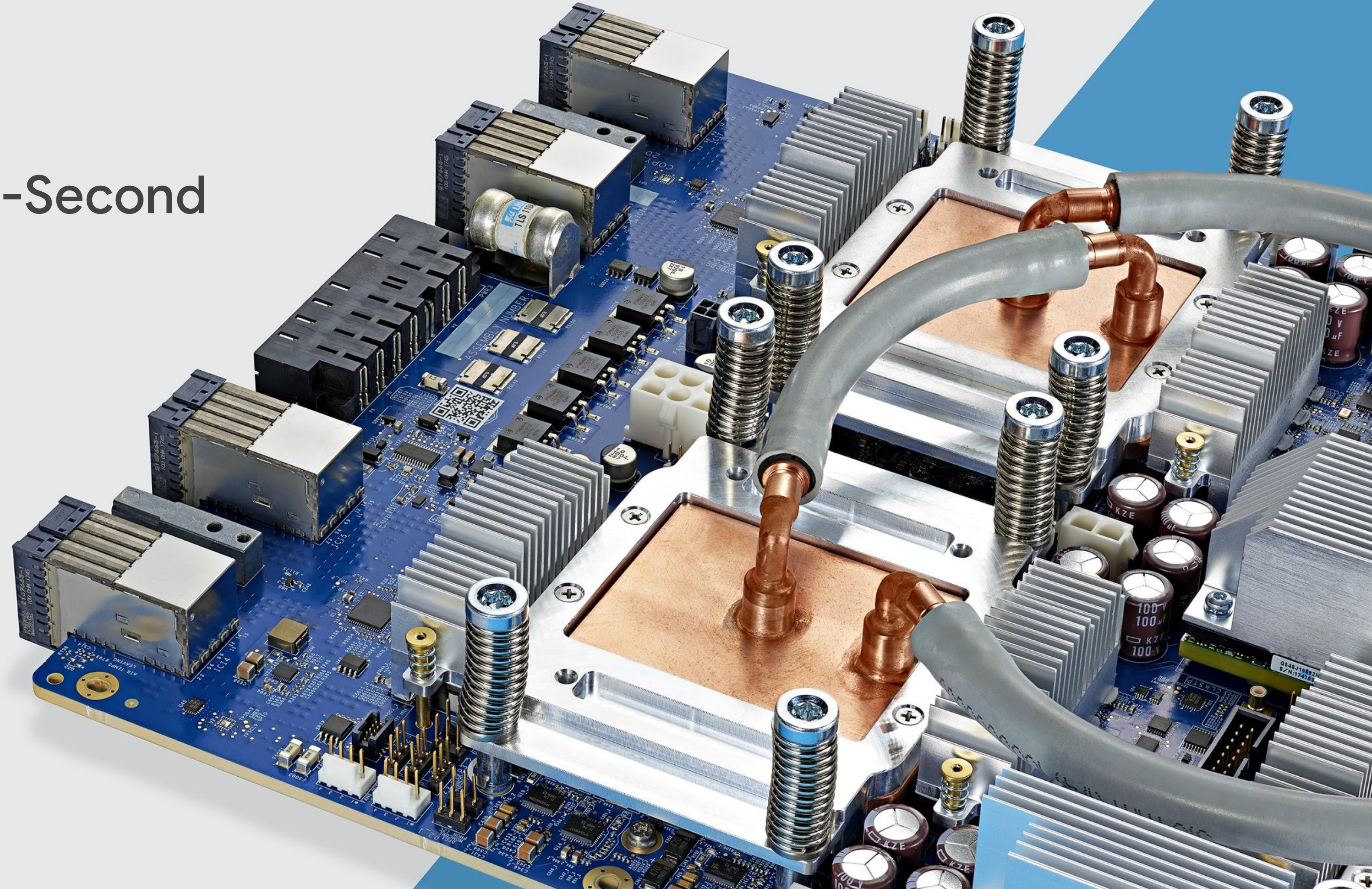
🟨 = Interconnect

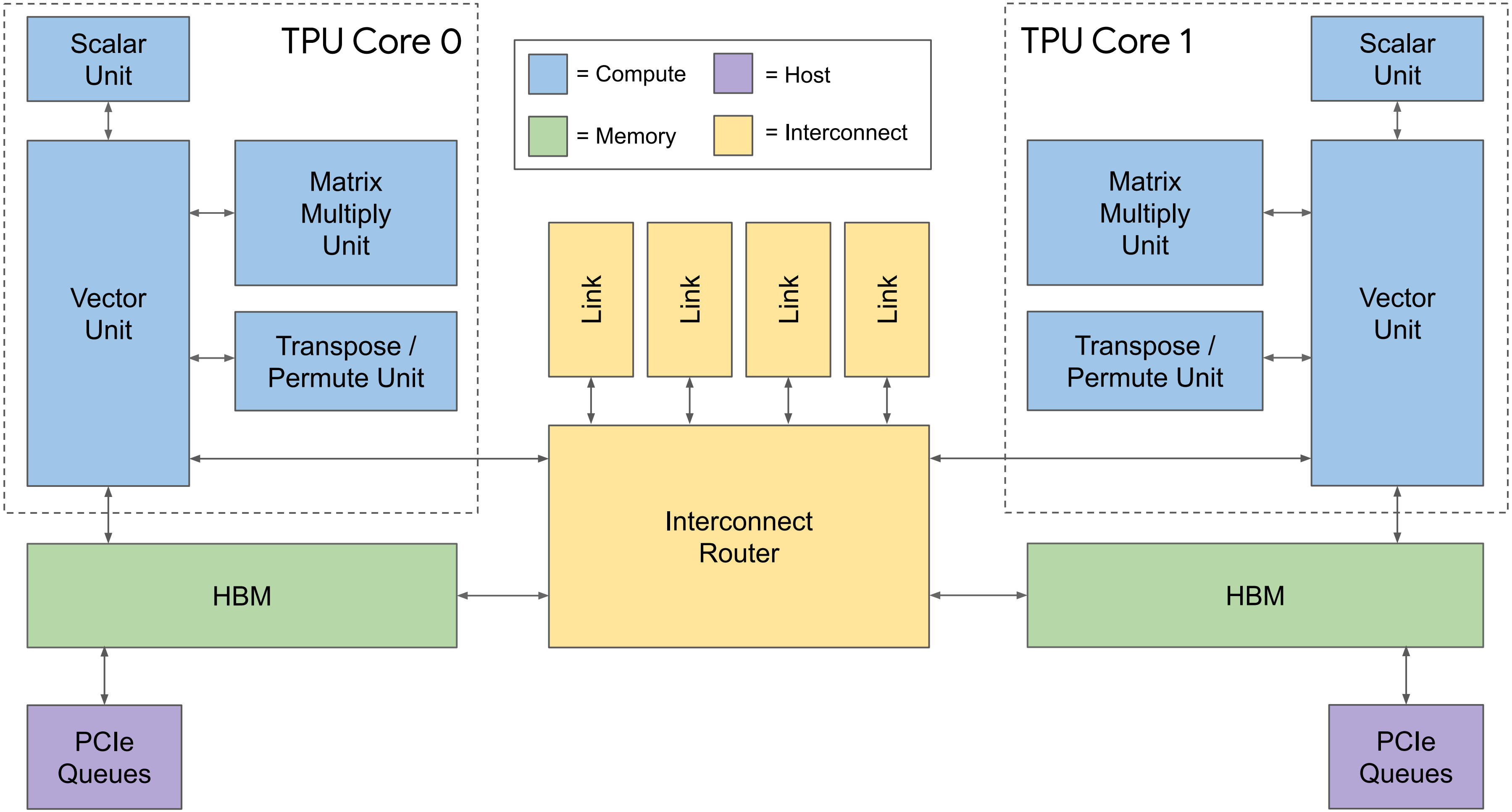
🟪 = Host

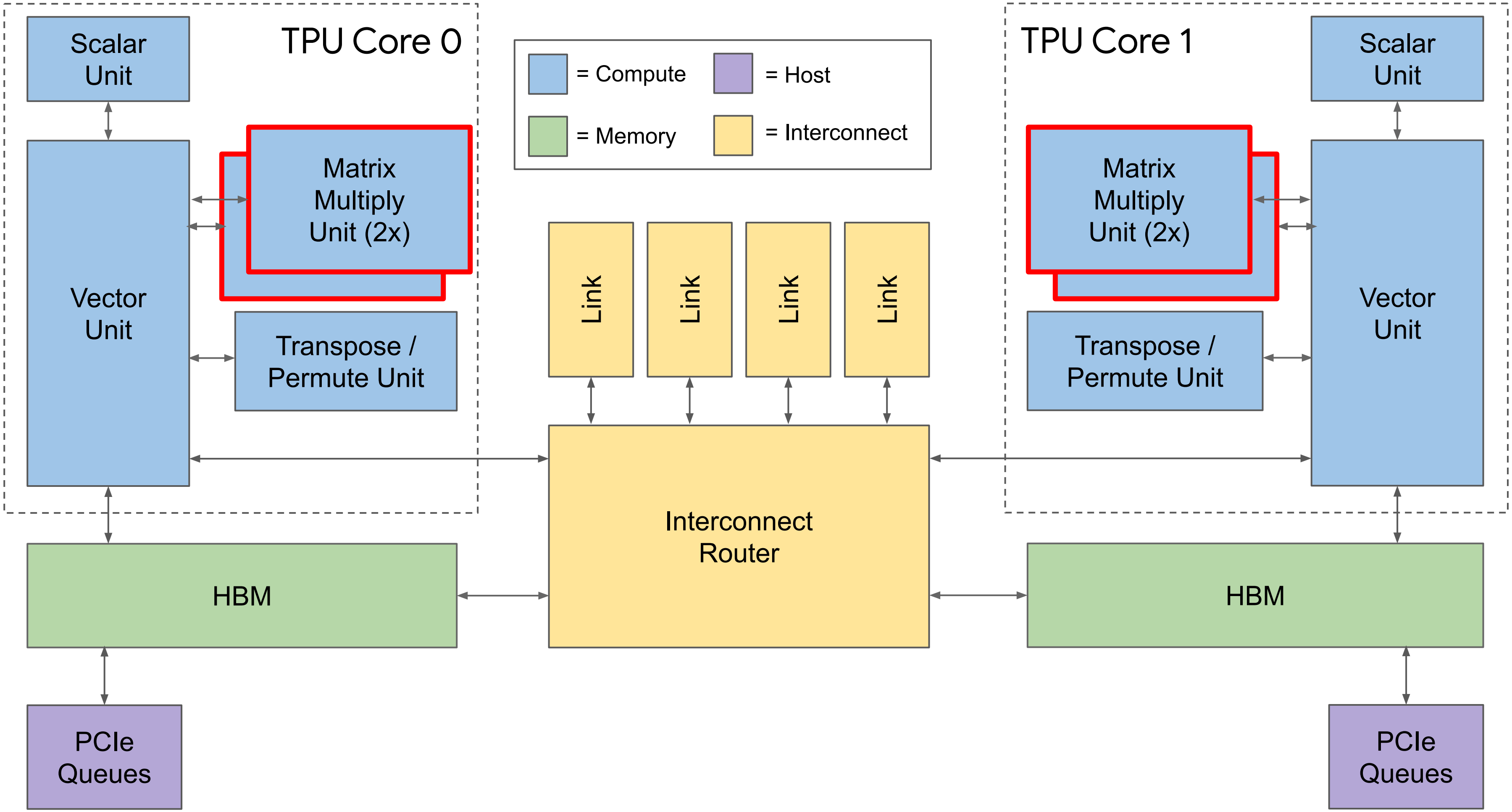
⬜ = Routing

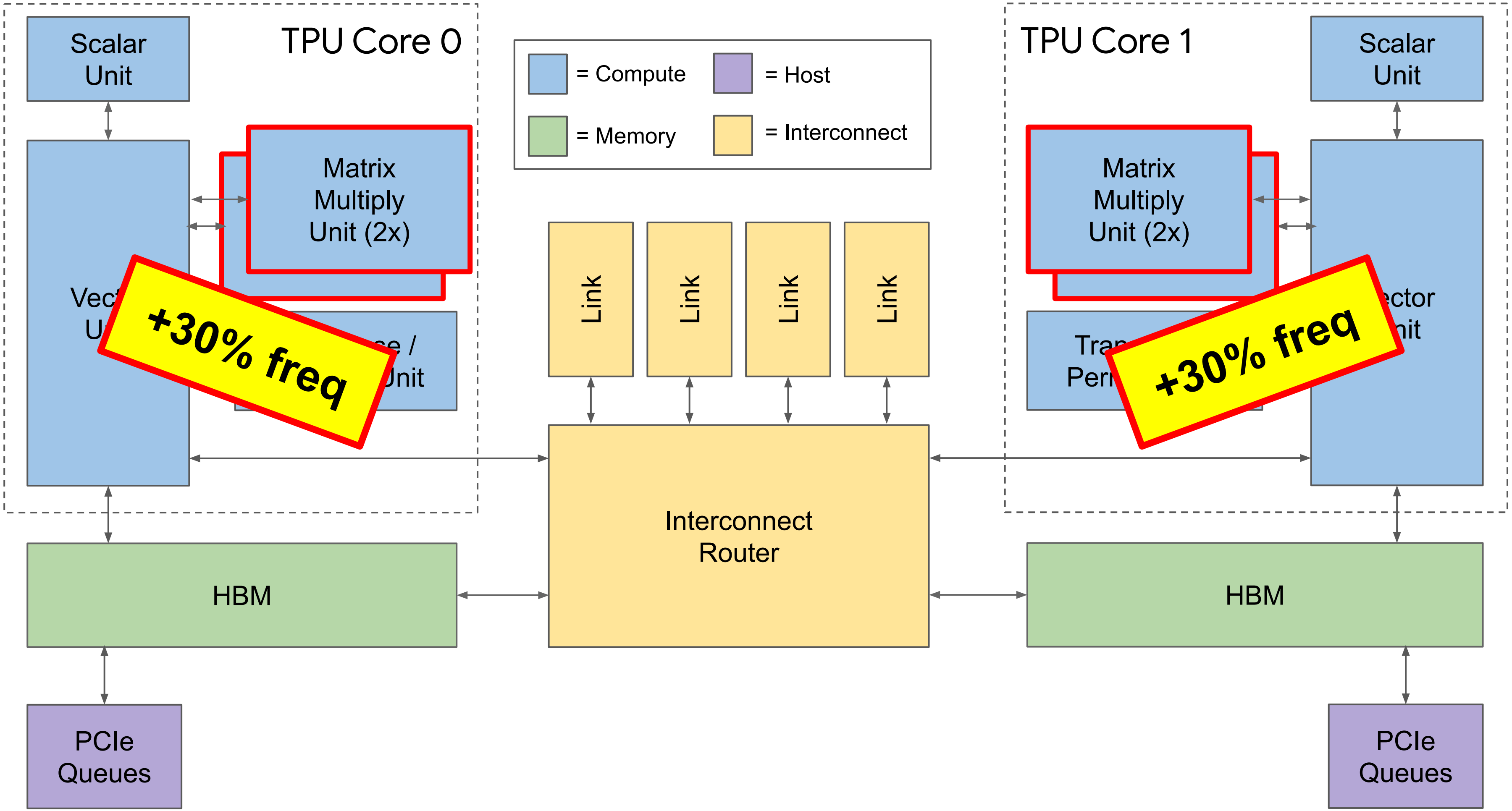
TPUv3

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
TPU Core 1
Scalar Unit
Matrix Multiply Unit
Transpose / Permute Unit
Vector Unit
HBM
HBM
PCIe Queues
PCIe Queues


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



TPU Core 0

Scalar Unit

Vect U

Matrix Multiply Unit (2x)

se / Unit

+30% freq

TPU Core 1

Scalar Unit

ector nit

Matrix Multiply Unit (2x)

Tran Perr

+30% freq

🟦 = Compute

🟪 = Host

🟩 = Memory

🟨 = Interconnect

Link

Link

Link

Link

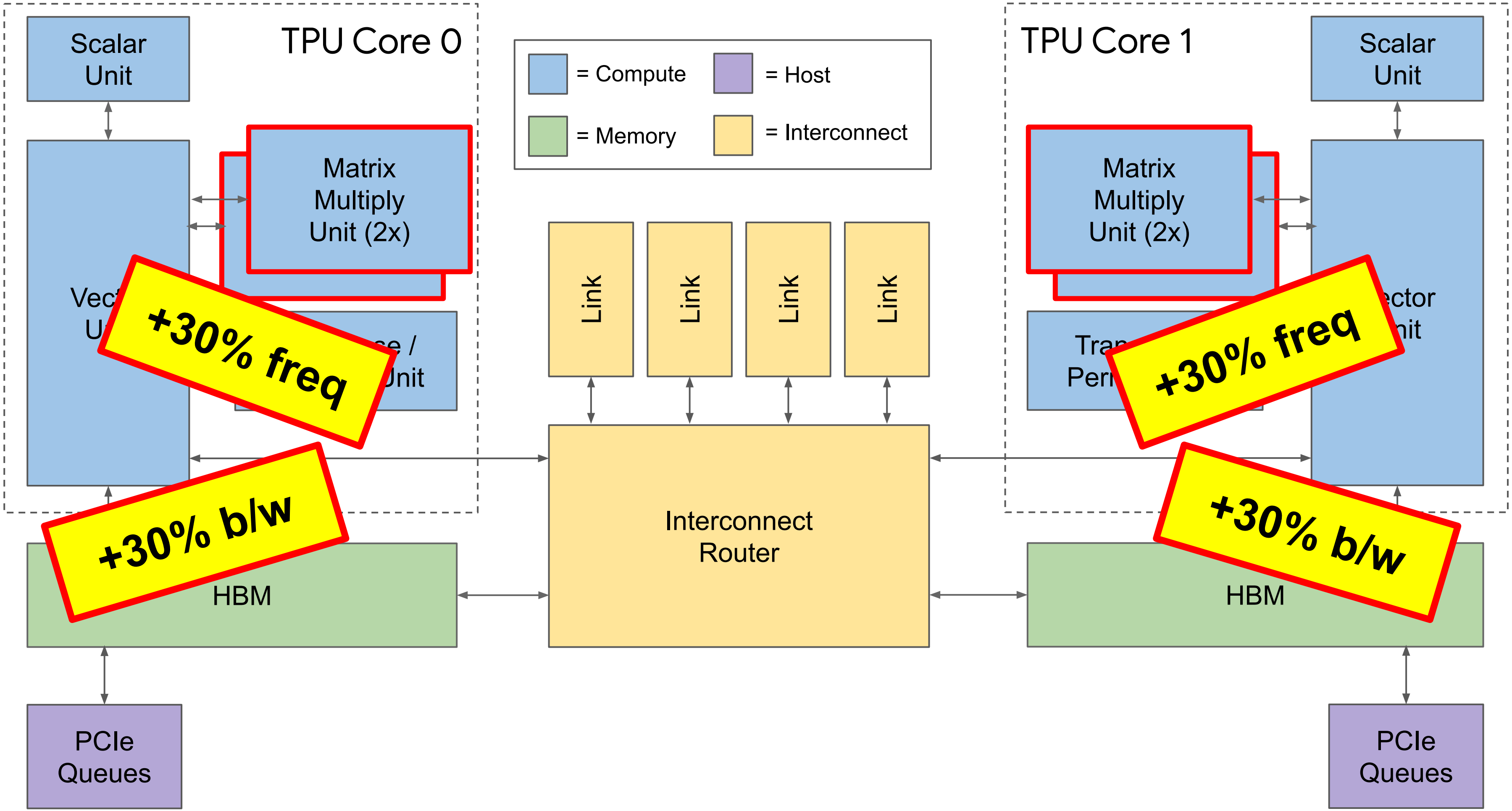
Interconnect Router

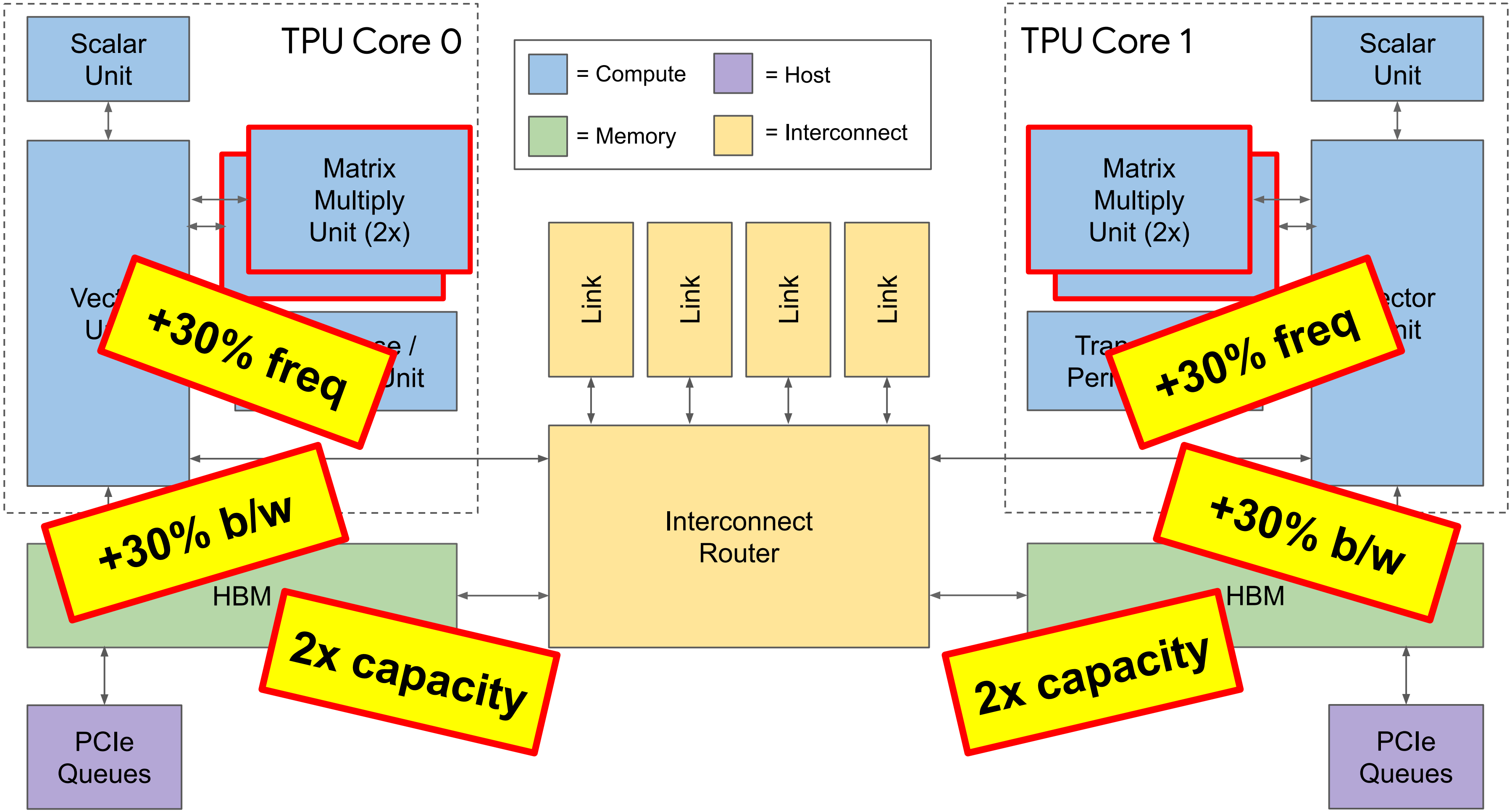
HBM

HBM

PCIe Queues

PCIe Queues


TPU Core 0
Scalar Unit
Matrix Multiply Unit (2x)
Vec
U
se /
Unit
+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)
Tran
Per
ector
nit
+30% freq
+30% b/w
HBM
PCIe Queues



TPU Core 0

Scalar Unit

Matrix Multiply Unit (2x)

Vect
U

se /
Unit

HBM

PCIe Queues

🟦 = Compute

🟪 = Host

🟩 = Memory

🟨 = Interconnect

Link

Link

Link

Link

Interconnect Router

TPU Core 1

Scalar Unit

Matrix Multiply Unit (2x)

ector
nit

Tran
Per

HBM

PCIe Queues

+30% freq

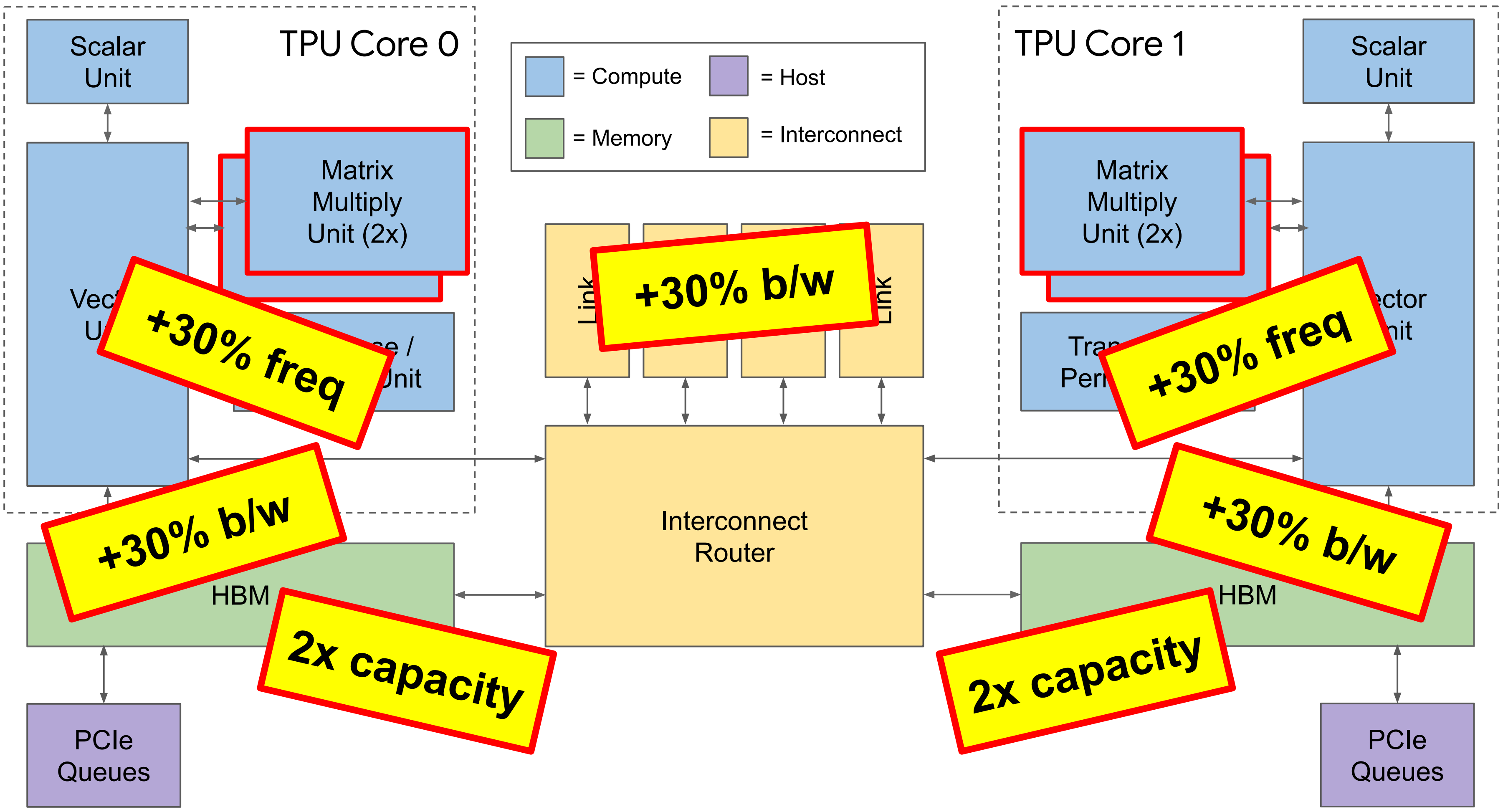
+30% b/w

2x capacity

+30% freq

+30% b/w

2x capacity



TPU Core 0

Scalar Unit

Matrix Multiply Unit (2x)

Vect U

se / Unit

HBM

PCIe Queues

🟦 = Compute

🟪 = Host

🟩 = Memory

🟨 = Interconnect

Link

Link

Interconnect Router

TPU Core 1

Scalar Unit

Matrix Multiply Unit (2x)

Tran Perm

ector nit

HBM

PCIe Queues

+30% freq

+30% b/w

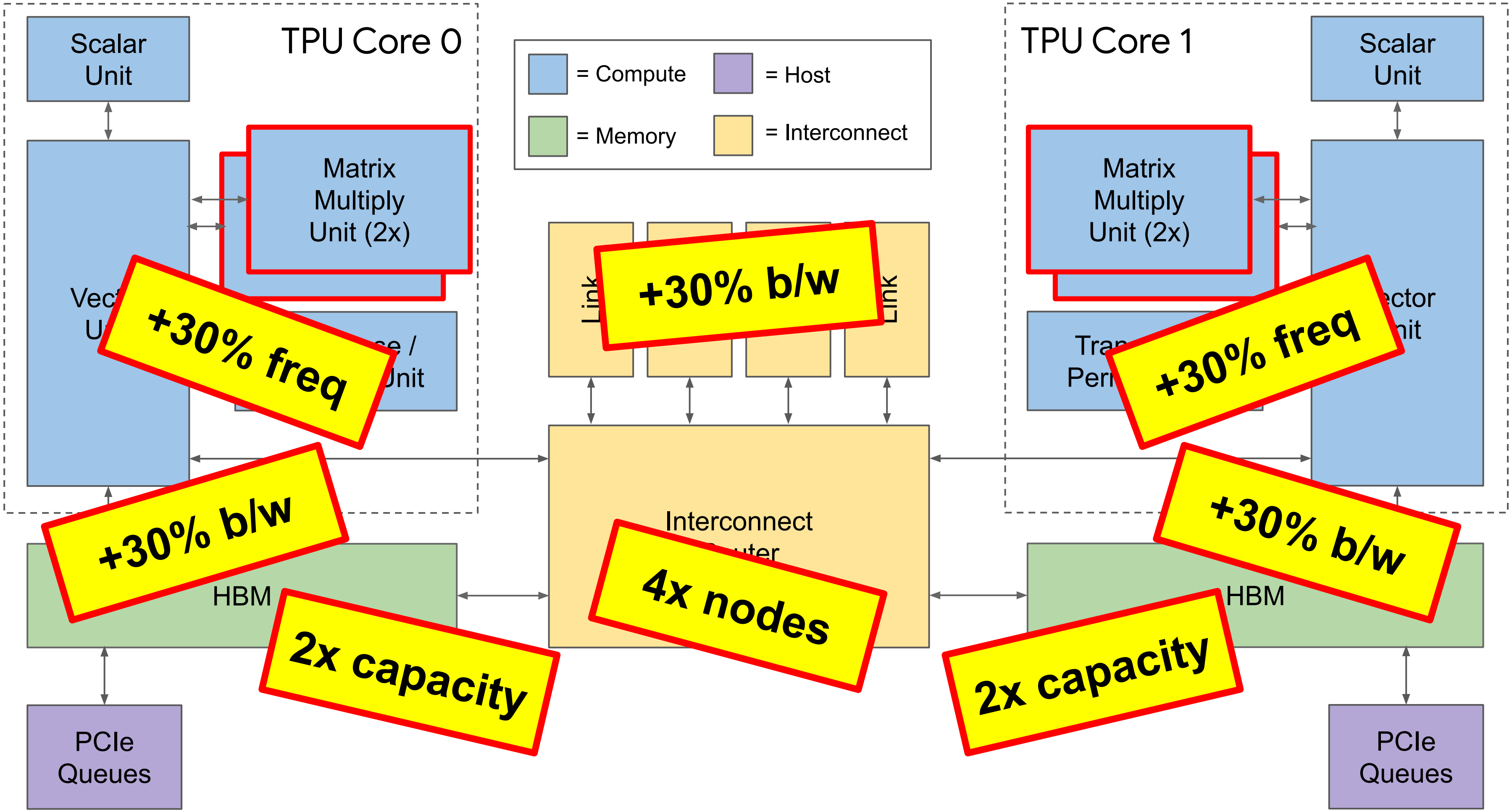
+30% b/w

+30% freq

+30% b/w

2x capacity

2x capacity



+30% freq

+30% b/w

+30% b/w

4x nodes

2x capacity

+30% freq

+30% b/w

2x capacity

Retrospective: Key Goals

Build it quickly

Achieve high performance…

…at scale

…for new workloads
out-of-the-box

…all while being cost effective

Retrospective: Key Goals

Build it quickly

Achieve high performance...

...at scale

...for new workloads out-of-the-box

...all while being cost effective

- Co-design: simplified hardware with software predictability (e.g., VLIW, scratchpads)
- Willingness to make tradeoffs

Retrospective: Key Goals

Build it quickly

Achieve high performance...

...at scale

...for new workloads
out-of-the-box

...all while being cost effective

- Compute density with bfloat16 systolic array
- HBM to feed the compute
- XLA compiler optimizations

Retrospective: Key Goals

Build it quickly

Achieve high performance…

…at scale

…for new workloads out-of-the-box

…all while being cost effective

- System-first approach
- Interconnect with a familiar interface for ease-of-use

Retrospective: Key Goals

Build it quickly

Achieve high performance…

…at scale

…for new workloads out-of-the-box

…all while being cost effective

- Flexible big data cores with principled linear algebra framework
- XLA compiler
- HBM capacity

Retrospective: Key Goals

Build it quickly

Achieve high performance...

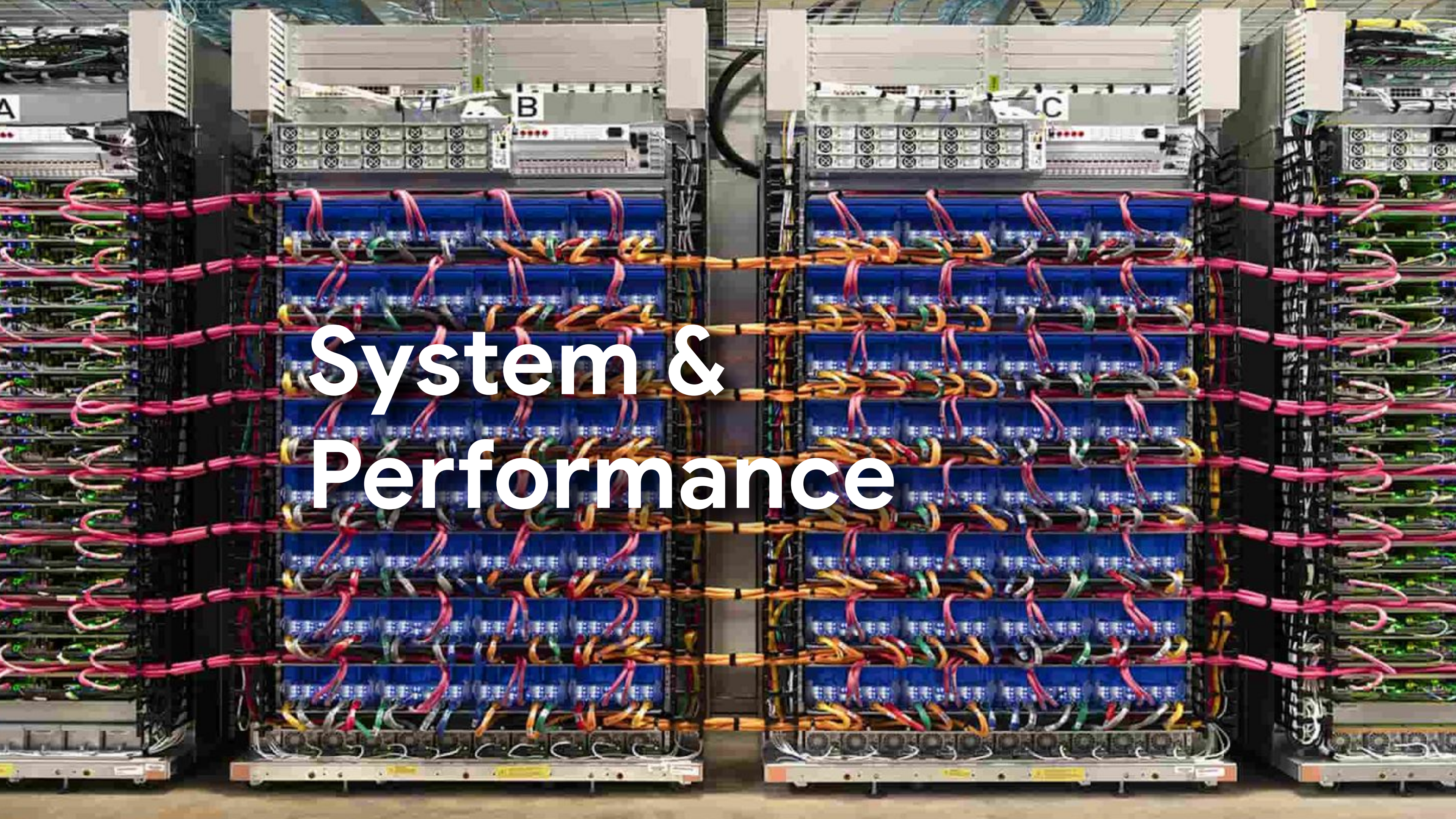
...at scale

...for new workloads out-of-the-box

...all while being cost effective

- Matrix Unit efficiency
- Simplicity
- High performance for good perf/$

System & Performance

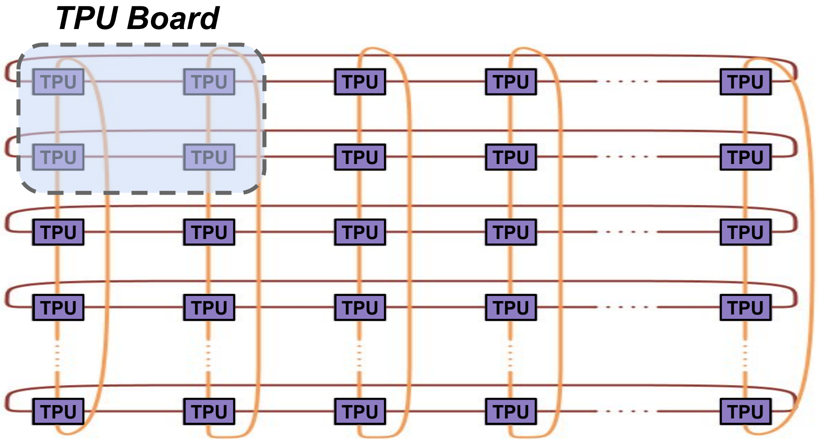


Supercomputer with dedicated interconnect

- 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

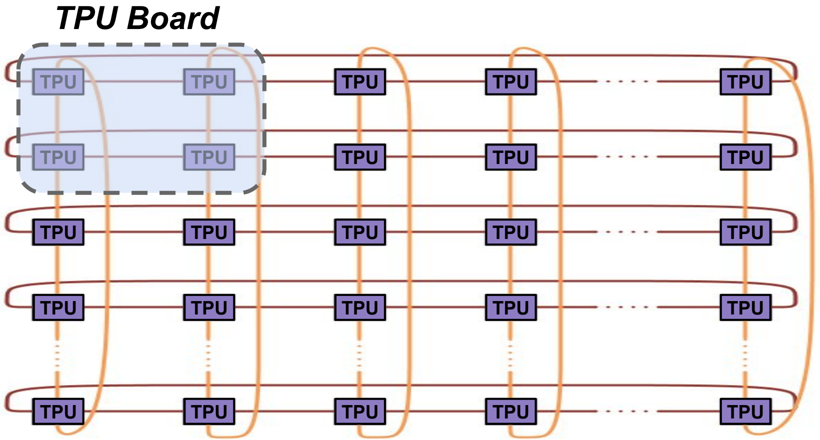
TPU Training Pod Architecture



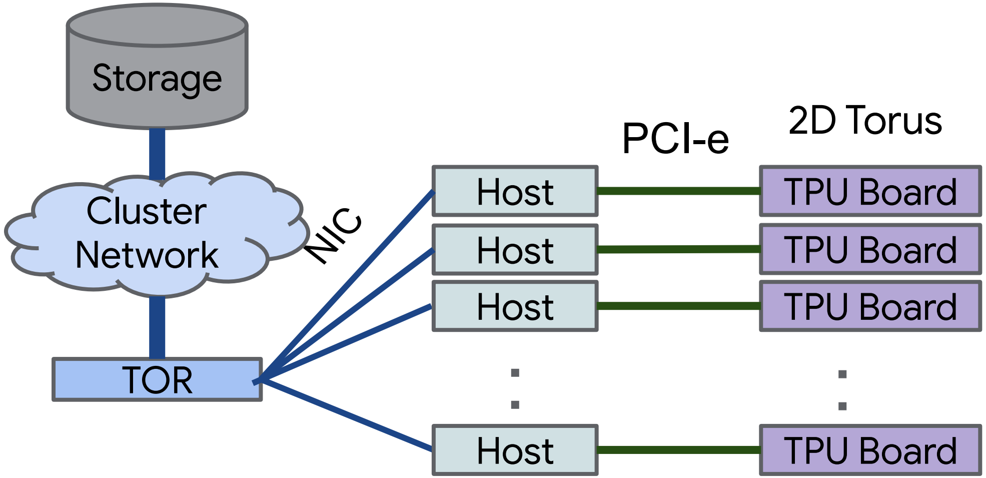
TPUs interconnected in 2D Torus

Dedicated network for **synchronous** parallel training

TPU Training Pod Architecture



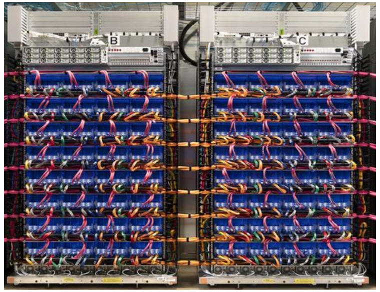
TPUs interconnected in 2D Torus



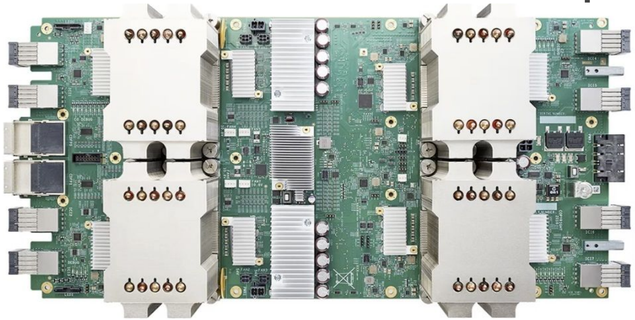
Dedicated network for **synchronous** parallel training

Supercomputer with dedicated interconnect

TPUv2 supercomputer
(256 chips)

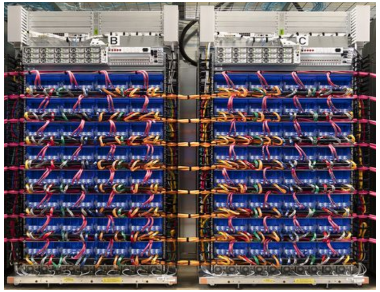


TPUv2 boards = 4 chips

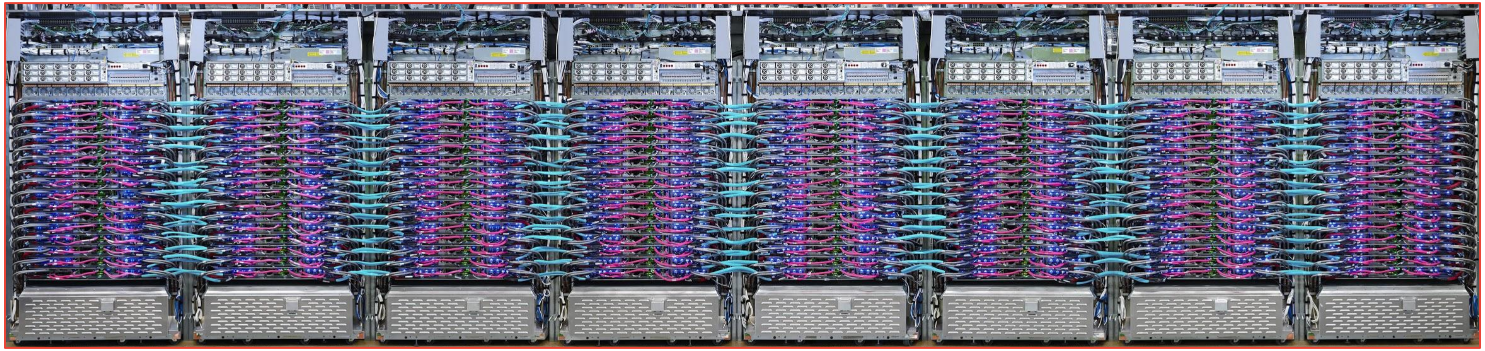


Supercomputer with dedicated interconnect

TPUv2 supercomputer (256 chips)

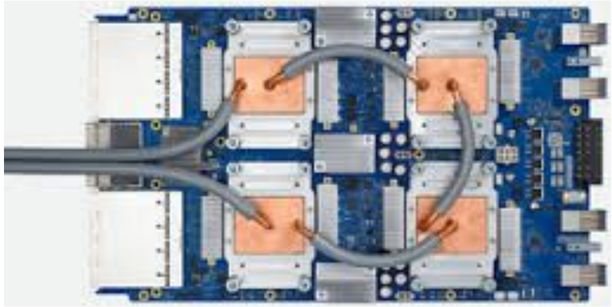


TPUv3 supercomputer (1024 chips)



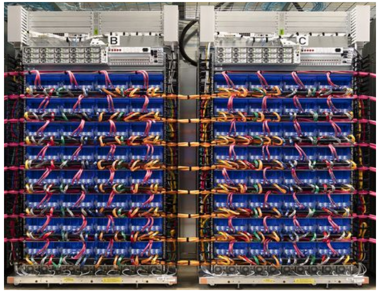
TPUv2 boards = 4 chips

TPUv3 boards = 4 chips

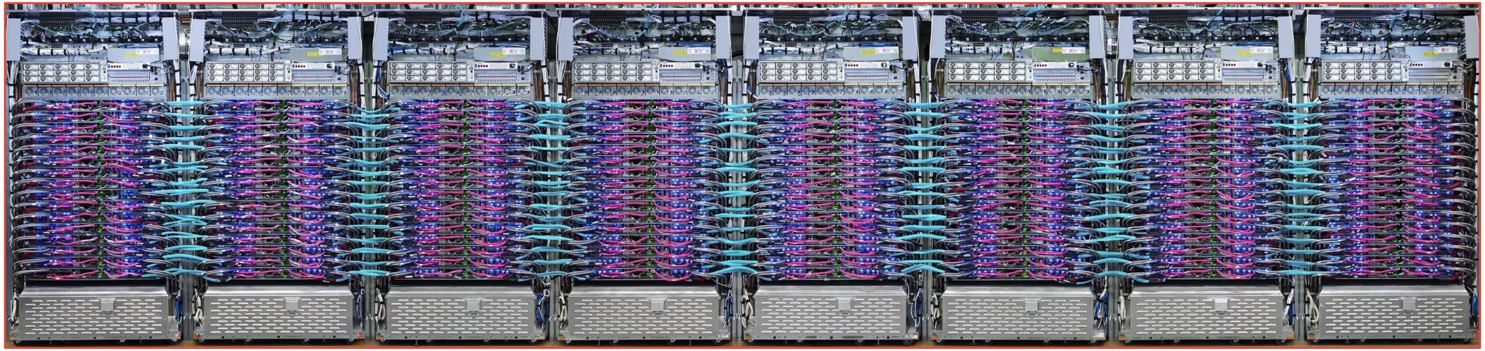


Supercomputer with dedicated interconnect

TPUv2 supercomputer (256 chips)



TPUv3 supercomputer (1024 chips)



11.5 petaflops

4 TB HBM

2-D torus

256 chips

> 100 petaflops

32 TB HBM

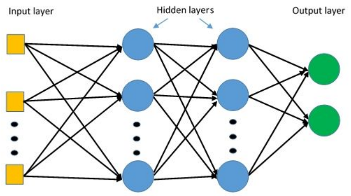
Liquid cooled

New chip + larger-scale system

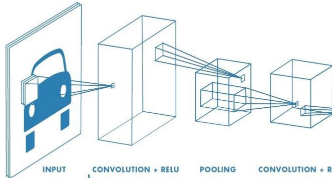
1024 chips

6 Production Applications

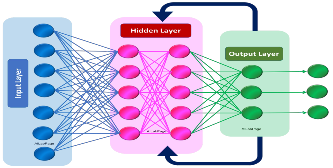
- **MultiLayer Perceptrons (MLP)**
  - MLP0 is unpublished
  - MLP1 is RankBrain [Cla15]



- **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

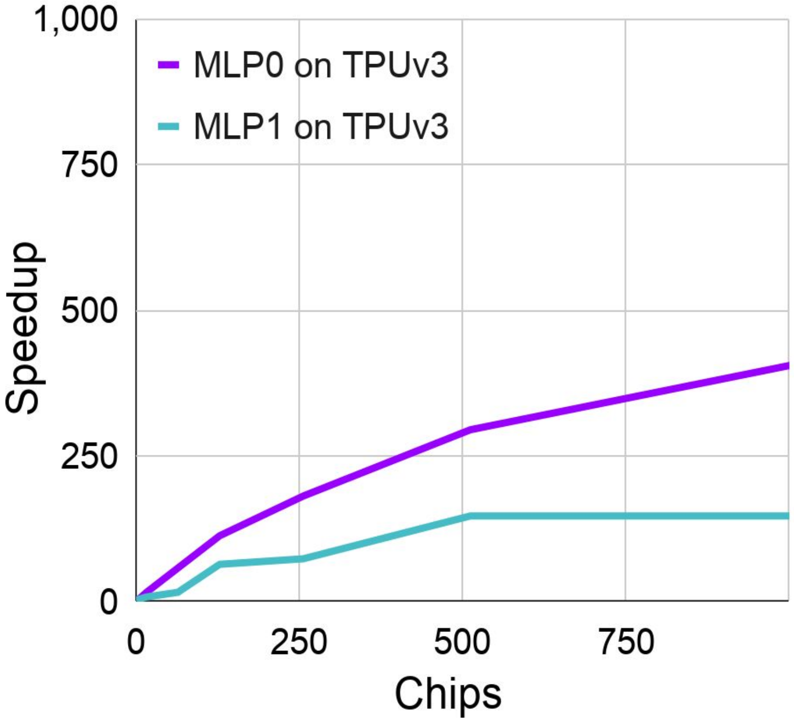


- **Recurrent Neural Networks (RNN)**
  - RNN0 is a Translation model [Che18]
  - RNN1 is a Speech model [Chi18]



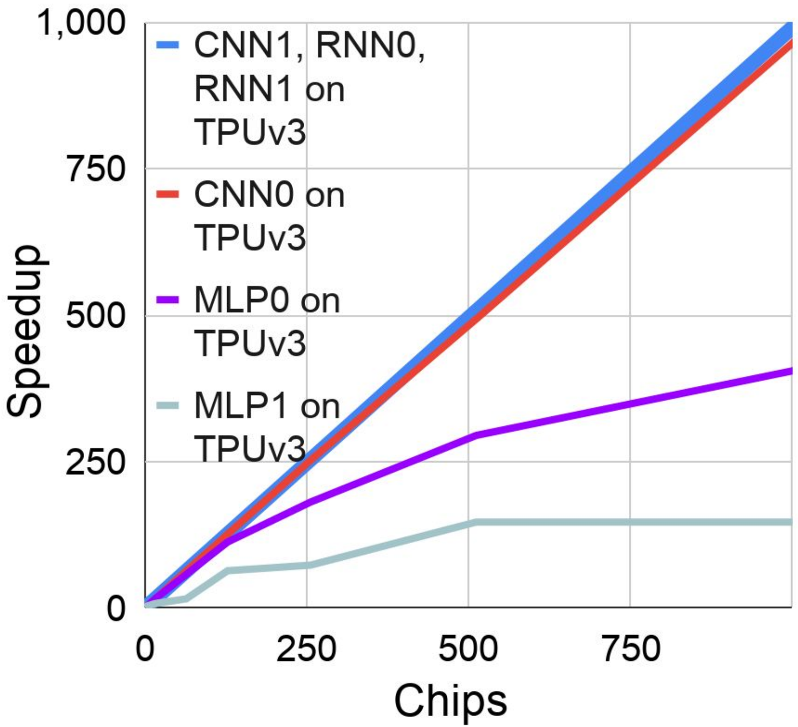
[Cla15] Clark, J. October 26, 2015, Google Turning Its Lucrative Web Search Over to AI Machines. Bloomberg Technology.
[Che18] Chen, M.X. et al, 2018. The best of both worlds: Combining recent advances in neural machine translation. arXiv preprint arXiv:1804.09849.
[Chi18] Chiu, C.C. et al, 2018, April. 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



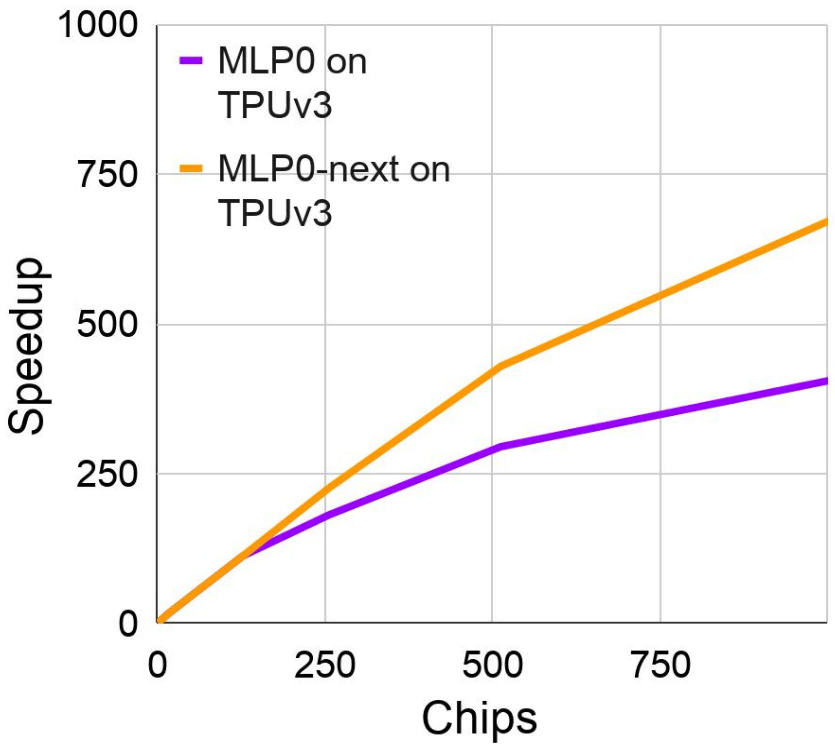
- **MLP0** & **MLP1**
  - 40% & 14% of perfect linear scale at 1024 chip-scale
    - Limited by embeddings

TPUv3 Supercomputer Scaling: 6 Production Apps



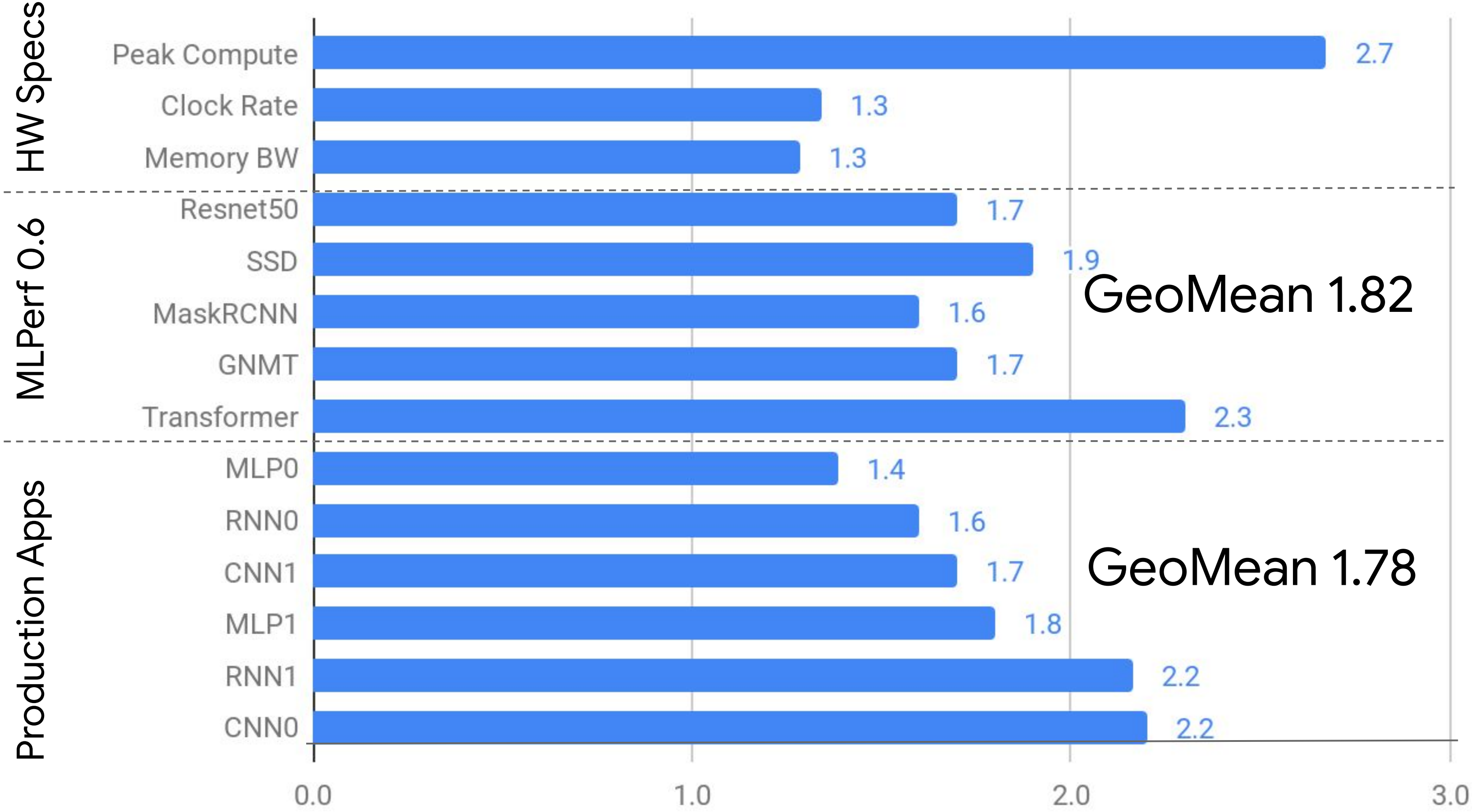
- MLP0 & MLP1
  - 40% & 14% of perfect linear scaling
- CNN0
  - 96% of perfect linear scaling!
- CNN1, RNN0, RNN1
  - **3 production apps** run at **99%** of perfect linear scaling at 1024 chips!

TPUv3 Supercomputer Scaling: MLP0-next vs. MLP0

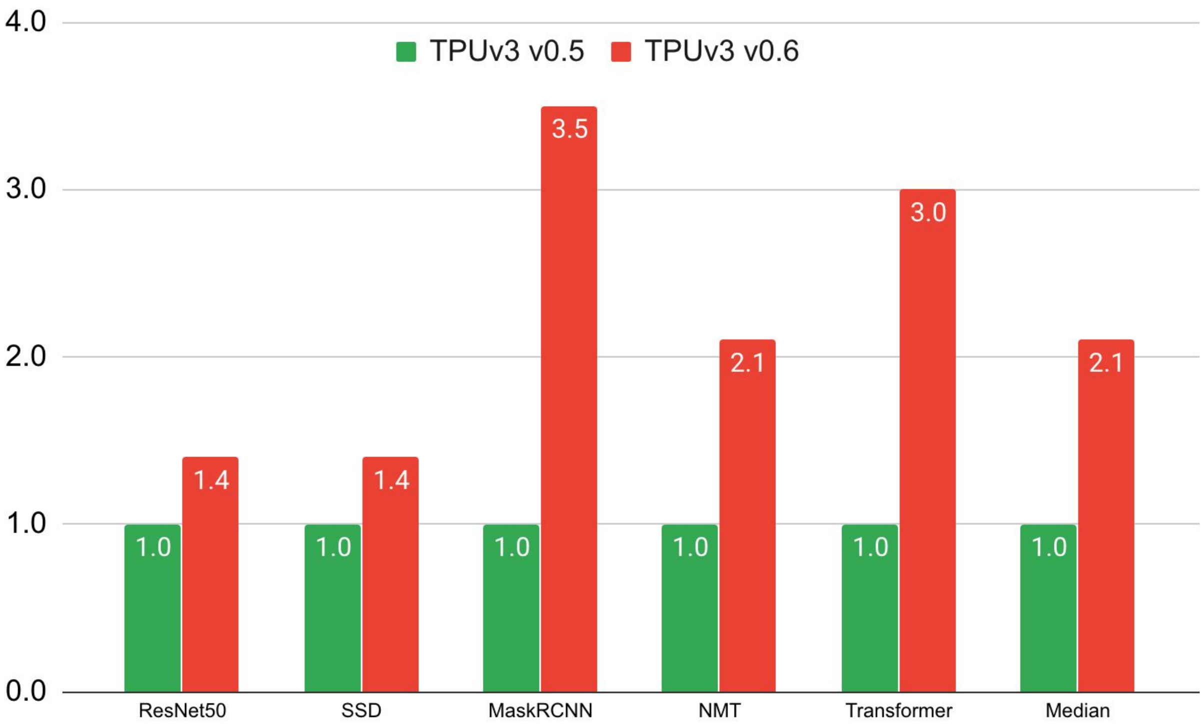


- Improved scaling for newer larger models and SW improvements for better quality
    - **MLP0-next**: 67% of perfect linear scale at 1024 chips
        - Up from 40% from MLP0

TPUv3 vs TPUv2: Memory Bound or 2X MXUs Useful?



Speedup: MLPerf v0.5 (11/2018) - v0.6 (5/2019)



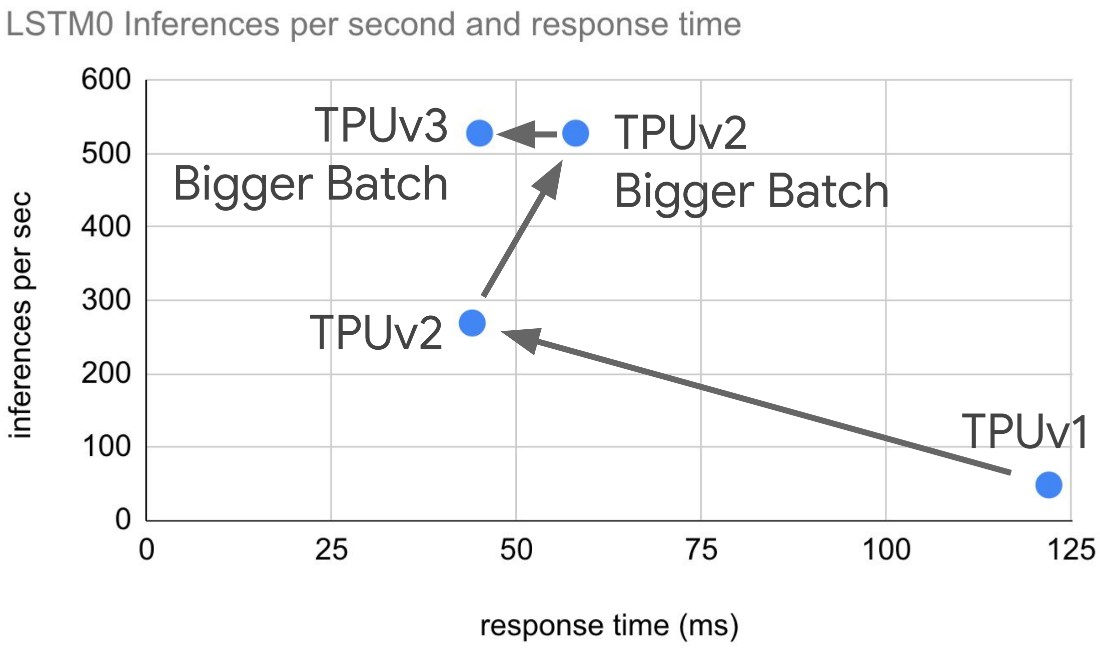
Production apps also sped up:

- CNN0 1.8x (more bfloat16 use)
- MLP0 1.6x (better partitioning and placement of embeddings)

Performance enables larger models for improved accuracy

Inference: TPUv2/v3 vs TPUv1

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

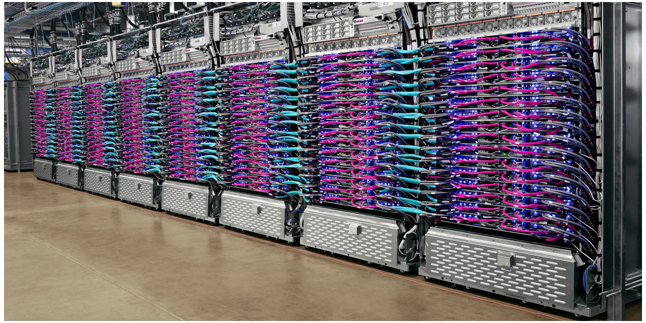


Key Takeaways

- 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