Ten Lessons From Three Generations Shaped Google's TPUv4i

Frances, Krish, Soumil

Carnegie Mellon University

Author Backgrounds

Norm Jouppi: VP, Engineering Fellow at Google.

Doe Hyun Yoon: Software Engineer at Facebook

Matt Ashcraft: Principal Engineer at Google

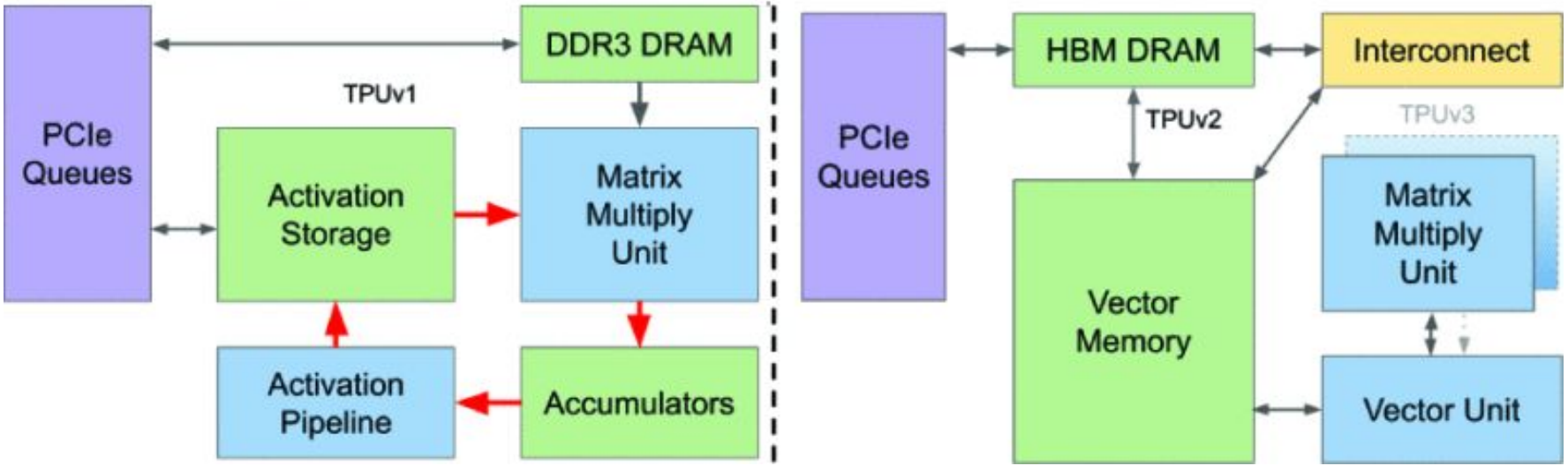
Paper Motivation + Background

- Problem: Deep Neural Networks (DNNs) have been increasing in demand, but Moore's Law has been slowing
- This paper reflects on the lessons learned from the three generations of Google's TPUs in order to design their TPUv4i
- Not only do benchmarks matter, but other factors such as multi-tenancy, cooling, and total cost impact future design decisions.

TPUv1 vs. TPUv2/v3

Feature	TPUv1 (2015)	TPUv2 (2017)	TPUv3 (2018)
Primary Focus	Inference Only	Training & Inference	Training & Inference
Arithmetic	8-bit Integer (quantized)	bfloat16 Floating Point	bfloat16 Floating Point
Memory	DDR3 DRAM	HBM (High Bandwidth)	HBM (2x capacity of v2)
Cores/Chip	1	2	2
Connectivity	Hosted via PCIe	Custom Interconnect (ICI)	Enhanced ICI (Large pods)

TPUv1 vs. TPUv2/v3



Lesson 1 - Hardware Scales Unequally

- Logic is “free”: transistor density and logic speeds improve greatly with smaller process nodes
- Wires and SRAM are the bottlenecks, SRAM is not scaling as fast as it has previously.

How does this relate to future designs?

Use multiple modular components to account for lack of wire scaling. Future designs can have more logic units.

Operation		*Picojoules per Operation*		
		45 nm	*7 nm*	*45 / 7*
+	Int 8	0.03	0.007	4.3
	Int 32	0.1	0.03	3.3
	BFloat 16	--	0.11	--
	IEEE FP 16	0.4	0.16	2.5
	IEEE FP 32	0.9	0.38	2.4
×	Int 8	0.2	0.07	2.9
	Int 32	3.1	1.48	2.1
	BFloat 16	--	0.21	--
	IEEE FP 16	1.1	0.34	3.2
	IEEE FP 32	3.7	1.31	2.8
SRAM	8 KB SRAM	10	7.5	1.3
	32 KB SRAM	20	8.5	2.4
	1 MB SRAM[1]	100	14	7.1
GeoMean[1]		--	--	2.6
DRAM		Circa 45 nm	Circa 7 nm	
	DDR3/4	1300[2]	1300[2]	1.0
	HBM2	--	250-450[2]	--
	GDDR6	--	350-480[2]	--

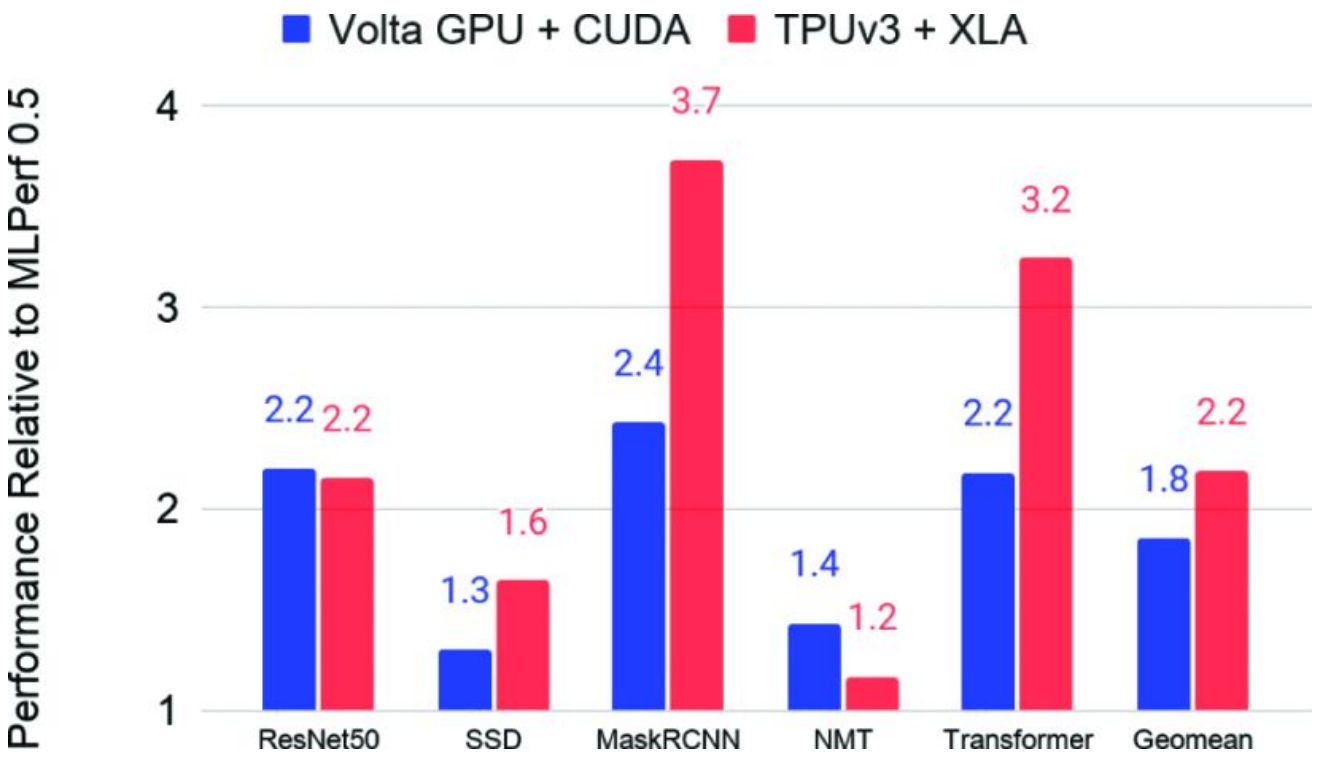
Lesson 2 - Leverage Prior Compilers

- Compiler compatibility is more important than binary compatibility. DSAs prioritize the compiler since it uses VLIW
- Designing for the XLA (Accelerated Linear Algebra) compiler allowed for large software optimizations to be reused in future TPU designs.

  How does this relate to future designs?

  Future designs should leverage existing software optimizations and use the XLA compiler.

Lesson 2 - Leverage Prior Compilers



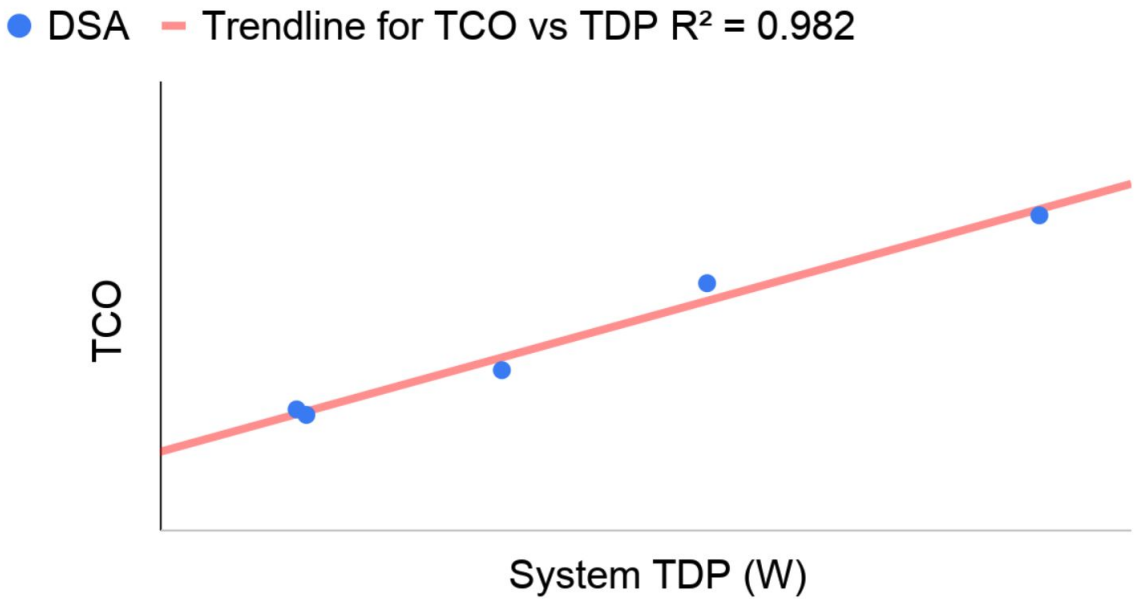
Lesson 3 - Architects should design for TCO

- Companies such as Google care about the perf/TCO (Total Cost of Ownership) where **TCO = CapEx + 3*OpEx**
- CPU and GPU designers often try to maximize their designs for benchmarks, but overlook power and operational costs
- There exists a very high correlation between TCO and TDP (Thermal Design Power)
- Good designs have ample headroom

How does this relate to future designs?

Future designs should prioritize reducing TDP and have ample headroom

Lesson 3 - Architects should design for TCO



Lesson 4 - Backwards ML Compatibility

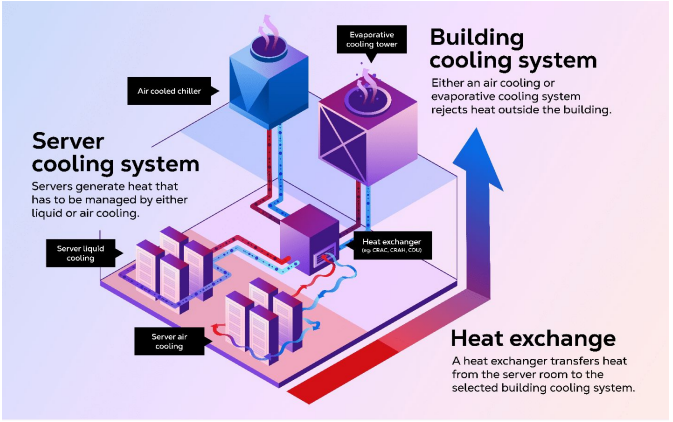
- New TPU generations should produce identical results across versions
- Requires support for same operations and numerics (bfloat16, FP32)
- Floating-point reordering can change model outputs
- Use consistent compiler behavior across hardware generations
- Ensures models trained earlier run immediately on new TPUs

Lesson 5 - Inference DSAs need air cooling for global scale

- Liquid cooling requires multiple adjacent racks and special infrastructure
- Hard to deploy in space-constrained datacenters
- Training clusters:

  few large deployments ➔ liquid cooling acceptable
- Inference systems:

  global deployment ➔ flexible placement required



Lesson 6 - Inference apps need floating point arithmetic

- Quantization aims to retain inference-time model quality using integers
- Pros: Reduces power consumption and memory usage
- Cons: Reduces model accuracy and delayed deployment
- Some applications require floating point precision to maintain quality

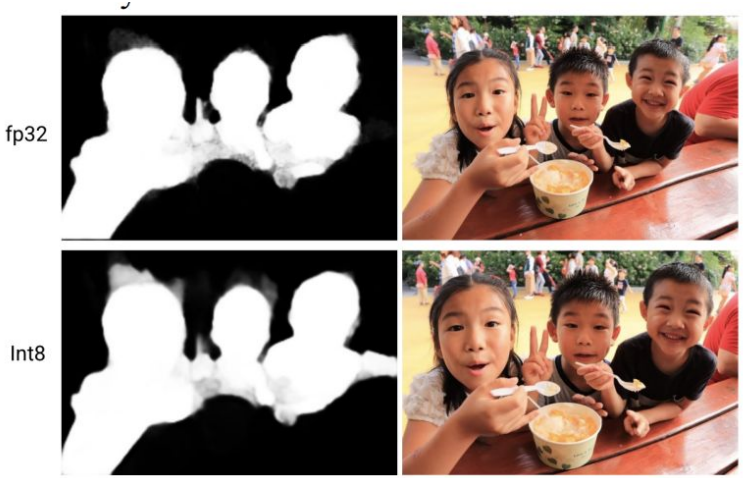


Figure 4. Quantization error for segmentation [40].

Lesson 6 - Inference apps need floating point arithmetic

- Early developers said 1% accuracy loss was acceptable, but later rejected it
  - Loss of 1% added to a 40% error is relatively small but loss of 1% added to a 12% error was relatively large
- Recovering accuracy after quantization can take months
- Inference hardware should support both INT and FP

Lesson 7 - Production Inference needs multi-tenancy

- Datacenter inference often runs many models on the same accelerator
- Sharing hardware lowers cost and improves utilization
- Applications require fast switching between models
- Loading weights from CPU over PCIe is too slow and weights can't fit in SRAM hence accelerators need local fast DRAM

Name	*Avg. Size (MB)*	*Max Size (MB)*	*Multi-tenancy?*	*Avg. Number of Programs (StdDev), Range*	*% Use 2016/ 2020*
MLP0	580	2500	Yes	27 (±17), 1-93	61%-25%
MLP1	90	N.A.	Yes	5 (±0.3), 1-5	
CNN0	60	454	No	1	5%-18%
CNN1	120	680	Yes	6 (±10), 1-34	

Table 3. The 2020 average and maximum size includes multi-tenancy.

Lesson 8 - DNNs grow ~1.5x/year in memory and compute

- Production ML models continuously grow in size and complexity
- Memory footprint and compute requirements increase ~1.5× per year
- This growth rate is similar to Moore's Law
- To not become obsolete quickly accelerators should provide headroom

Model	*Annual Memory Increase*	*Annual FLOPS Increase*
CNN1	0.97	1.46
MLP1	1.26	1.26
CNN0	1.63	1.63
MLP0	2.16	2.16

Table 4. Annual increase in production applications of 2016.

Lesson 9 - DNN workloads evolve with DNN breakthroughs

- Production workloads change as new DNN architectures emerge
- New models are adopted very quickly in production
- Accelerators must be programmable and flexible

Name	*Avg. Size (MB)*	*Max Size (MB)*	*Multi-tenancy?*	*Avg. Number of Programs (StdDev), Range*	*% Use 2016/ 2020*
MLP0	580	2500	Yes	27 (±17), 1-93	61%-25%
MLP1	90	N.A.	Yes	5 (±0.3), 1-5	
CNN0	60	454	No	1	5%-18%
CNN1	120	680	Yes	6 (±10), 1-34	
RNN0	1300	1300	Yes	13 (±3), 1-29	0%-29%
RNN1	120	400	No	1	
BERT0	3000	3000	Yes	9 (±2), 1-14	0%-28%
BERT1	90	N.A.	Yes	5 (±0.3), 1-5	

Table 3. The 2020 average and maximum size includes multi-tenancy.

Lesson 10 - Inference SLO limit is latency, not batch size

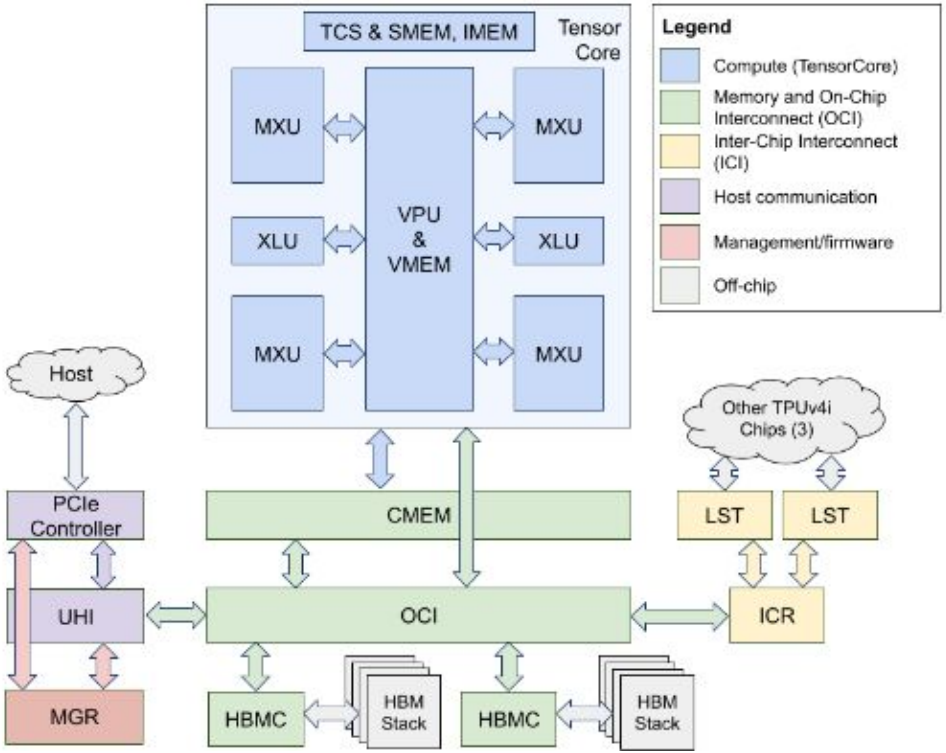
- Production systems optimize for P99 latency (tail latency)
- Batch size is not fixed, it depends on latency constraints
- Real systems use larger batches while meeting latency SLOs
- Google workloads:
  - ~7× stricter latency requirements yet achieve ~9× larger batch sizes

Production						*MLPerf 0.7*		
DNN	*ms*	*batch*	*DNN*	*ms*	*batch*	*DNN*	*ms*	*batch*
MLP0	7	200	RNN0	60	8	Resnet50	15	16
MLP1	20	168	RNN1	10	32	SSD	100	4
CNN0	10	8	BERT0	5	128	GNMT	250	16
CNN1	32	32	BERT1	10	64			

Table 5. Latency limit in ms and batch size picked for TPUv4i.

TPUv4i Architecture

Main Changes Summarized

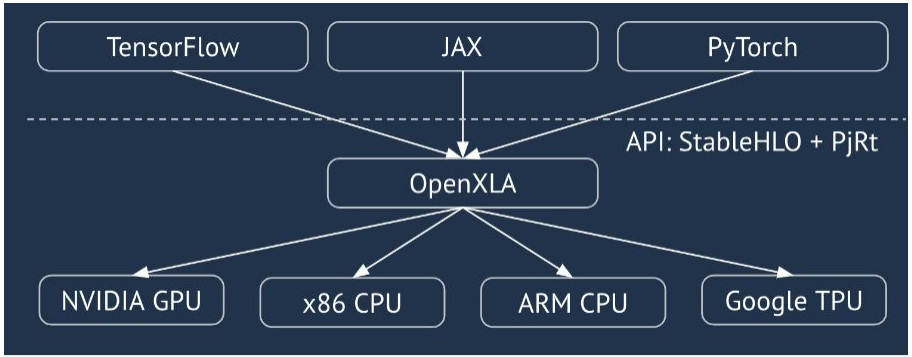


- One core architecture for separate training and inference chips
- Compiler Compatibility
- Introduction of CMEM (Common Memory)
- 4D Tensor DMA & Compartmentalization
- New On-Chip Interconnect
- New Adders for Systolic Arrays & Increased number of MXUs
- Lower TDP for Air Cooling in Datacenter
- ICI-Based Scaling

Compiler, Not Binary Compatibility

➔ **Lesson 2: Leverage Existing Compiler Optimizations**

- Due to increased VLIW length, cannot use previous binaries
- XLA (Accelerated Linear Algebra ML Compiler for CPUs/GPUs/etc.)



Addition of Common Memory (CMEM)

➔ **Lesson 3: Design for perf/TCO instead of perf/CapEx**
➔ **Lesson 7: Production inference normally needs multi-tenancy**
➔ **Lesson 8: DNNs grow ~1.5x annually in memory and compute**
➔ **Lesson 9: DNN workloads evolve with DNN breakthroughs**

- Reduce cost logic is to use DDR memory, but HBM & SRAM have lower power envelope and can scale with DNN memory & performance demands

4D Tensor Direct Memory Access (DMA)

➔ **Lesson 2: Leverage Existing Compiler Optimizations**
➔ **Lesson 8: DNNs grow ~1.5x annually in memory and compute**
➔ **Lesson 9: DNN workloads evolve with DNN breakthroughs**

- Hide memory access latency with separate DMA coprocessor
- Take away memory-access work from TensorCore
- Unified on-chip, chip-to-chip, host-to-chip DMA for application scaling

Custom On-Chip Interconnect (OCI)

➔ **Lesson 8: DNNs grow ~1.5x annually in memory and compute**
➔ **Lesson 9: DNN workloads evolve with DNN breakthroughs**

- Interconnect instead of point-to-point connections for scaling
- OCI split into 4 smaller networks between HBM, CMEM, VMEM, each 128B-wide for 512B-wide total cache lines ➔ Total serving 614 GB/s memory bandwidth

TensorCore Arithmetic Improvements

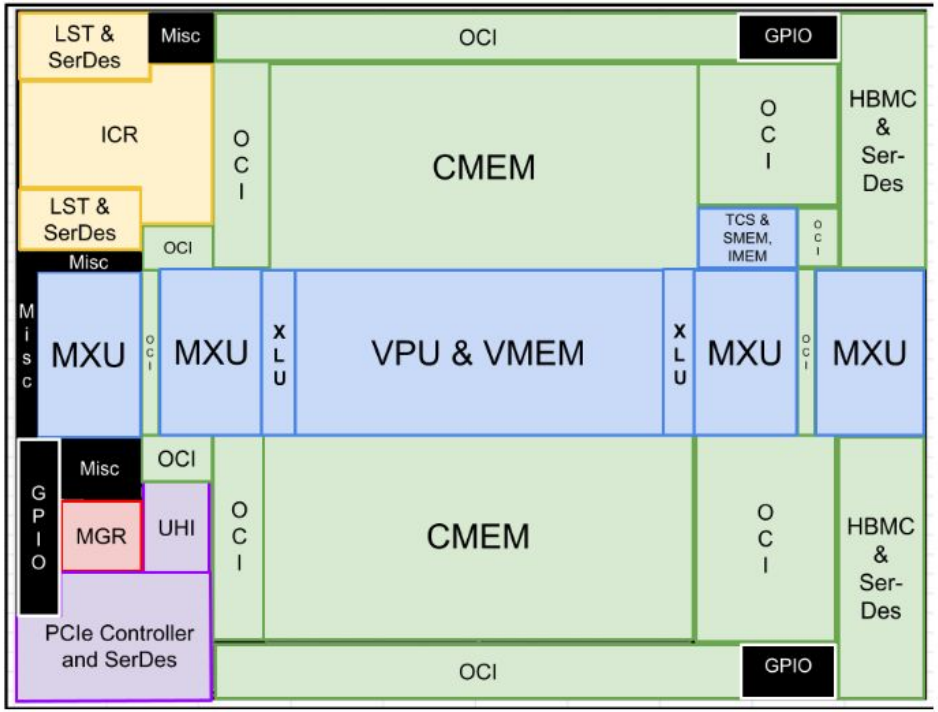
- ➔ **Lesson 4: Backwards ML compatibility enables rapid deployment of trained DNNs**
- ➔ **Lesson 5: Inference DSAs need air cooling for global scale**
- ➔ **Lesson 6: Some inference apps need floating point arithmetic**
- ➔ **Lesson 8: DNNs grow ~1.5x annually in memory and compute**
- ➔ **Lesson 9: DNN workloads evolve with DNN breakthroughs**

- Support for bf16 and fp32 for backwards compatibility
- Modified systolic array adder reduces area and power
- More MXUs (Matrix Multiply Units)

More Improvements (Summarized)

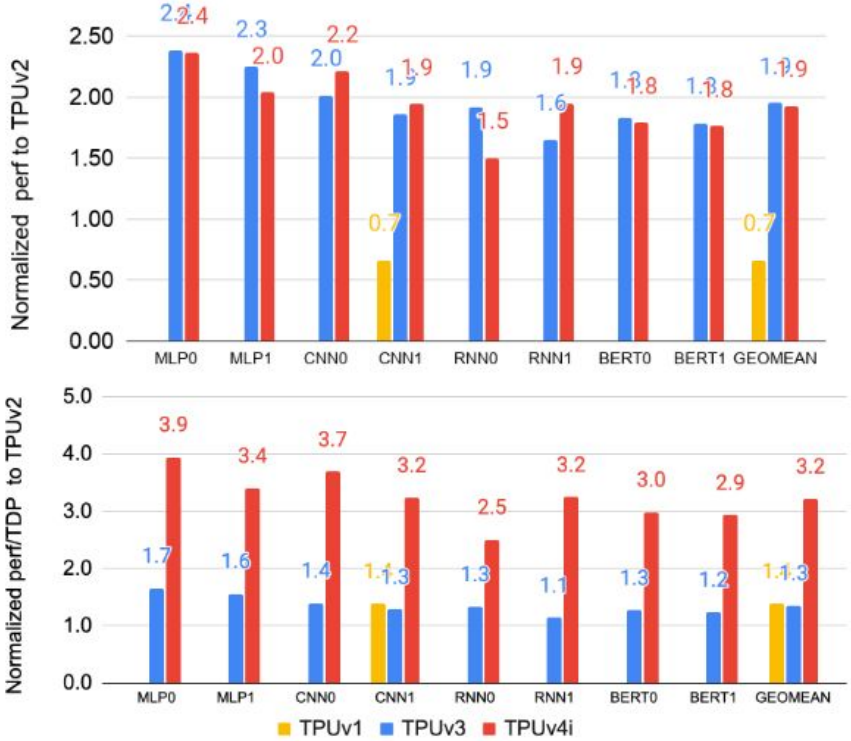
- **Clock Rate & TDP → Lesson 3 (Design for perf/TCO) and Lesson 5 (Need Air Cooling for Scalability)**
  - Higher Freq (1.0 GHz) but much lower TDP (175 W)
- **ICI Scaling → Lesson 8 (DNN Scaling)**
  - Add Inter-Chip Interconnect so that multiple TPUv4i's can most efficiency work togethe ron same workload
- **Workload Analysis Features → Lesson 3 (Design for perf/TCO) and Lessons 7/8/9 (DNN Scaling)**
  - Add more performance counters to give more insight for future performance improvements to maximize TPU lifetime as DNNs scale

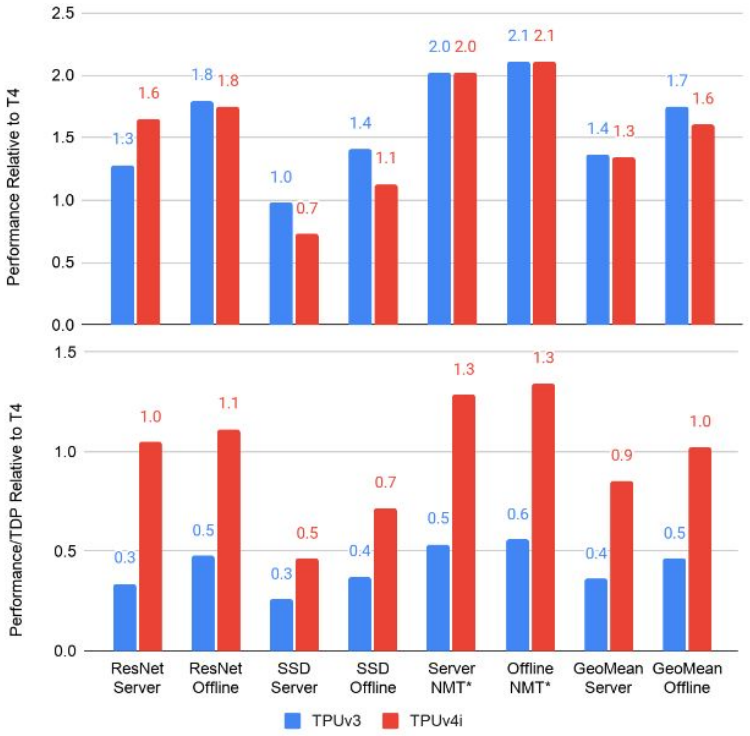
Bottom-Up Perspective



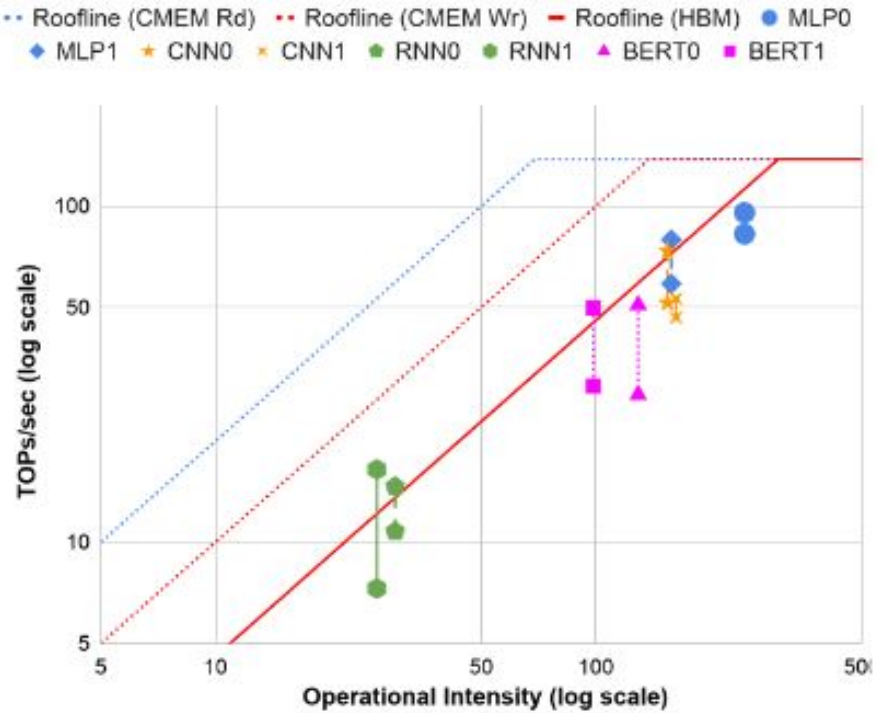
- CMEM Size & Location
- Reduction in Global Wire Lengths
- On-Chip Interconnect Efficiency
- Organized for low-power & air-cooling
- 4-Way Split for new OCI
- Low Logic Footprint
- Power & Heat Spread Out (MXUs surrounded by lower-energy components)

TPUv4i Compiled Results





TPUv4i Compiled Results



- Perf/TDP is 2.3x TPUv3
  - 1.1x FLOPs
  - 4.5x SRAM Size
  - 0.7x DRAM Bandwidth
  - 0.4x TDP
  - 2x MXU Count
  - Addition of CMEM
  - 7 nm process
  - 1.6x # transistors
- Breaks “Accelerator Wall”
  - Perf/TDP did not improve logarithmically with # transistors

Questions ?