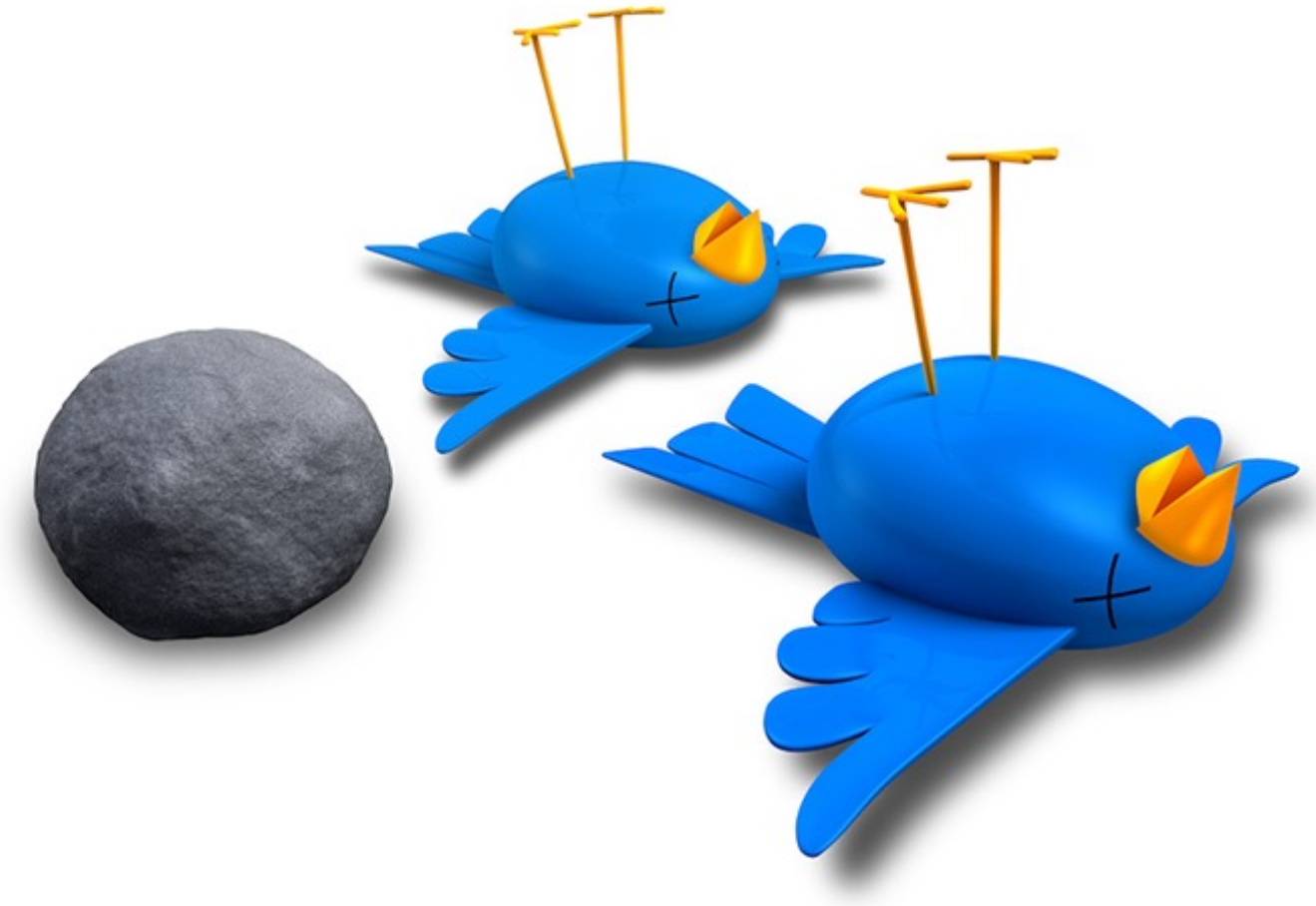
EPFL CS723

Ten Lessons From Three Generations Shaped Google's TPUv4i

Vladimir Somers – 18/04/2023

Elevator speach



What is the problem? How important is it?

- **Growing demand** for AI services with ML workloads that become more **compute intensive**.
- ML workloads are **expensive** (infrastucture + energy), especially at **inference** to serve large-scale AI applications
- Need for **specialized hardware** for ML that is:
  - **Cost effective** and **low-latency**
  - **Flexible** and **scalable**
- Important problem since specialized hardware is a key to:
  - Reduce **infrastructure** and **operational costs**
  - Reduce **environmental** footprint
  - Making AI more **accessible**

(1) What are the insights?

- **Insights** from 5+ years building DSAs for DDN at Google, summarized into 10 **lessons**.
- Lessons split into **3 categories**:

1. Lessons applying to **any DSA**
2. Lessons focusing on **DNN DSAs**
3. Lessons about **DNN applications**

(2) Insights: 3 lessons applying to any DSA

#	Insight	Solution
1	**Logic** improves much faster than wires and SRAM	⟹ DSAs architecture should mitigate the impact of **interconnect latency** and **wire scaling challenges**
2	DSA **architectures** and **compilers co-evolve**, and developing optimized compilers is expensive and time-consuming	⟹ New DSA should **leverage existing compiler optimizations**.
3	Different from benchmarking tools, big companies care about the **overall lifetime efficiency** of DSAs	⟹ Optimize for **perf/TCO** is a better target when designing DSA than perf/CapEx.

TCO = CapEx + 3 X OpEx

(3) Insights: 3 lessons focusing on DNN DSAs

#	Insight	Solution
4	**Backwards** ML **compatibility** enables rapid deployment of trained DNNs	⟹ New DSA version should look like previous version from **compiler's perspective** and support the same precision.
5	Inference DSA requires **global deployment** to reduce latency, but **liquid cooling** might not be possible everywhere	⟹ Inference DSAs should support **air cooling**.
6	Some inference apps need **floating point** arithmetic	⟹ DSA should support **multiple precision** (bf16 and int8), i.e. **quantization** must be **optional**.

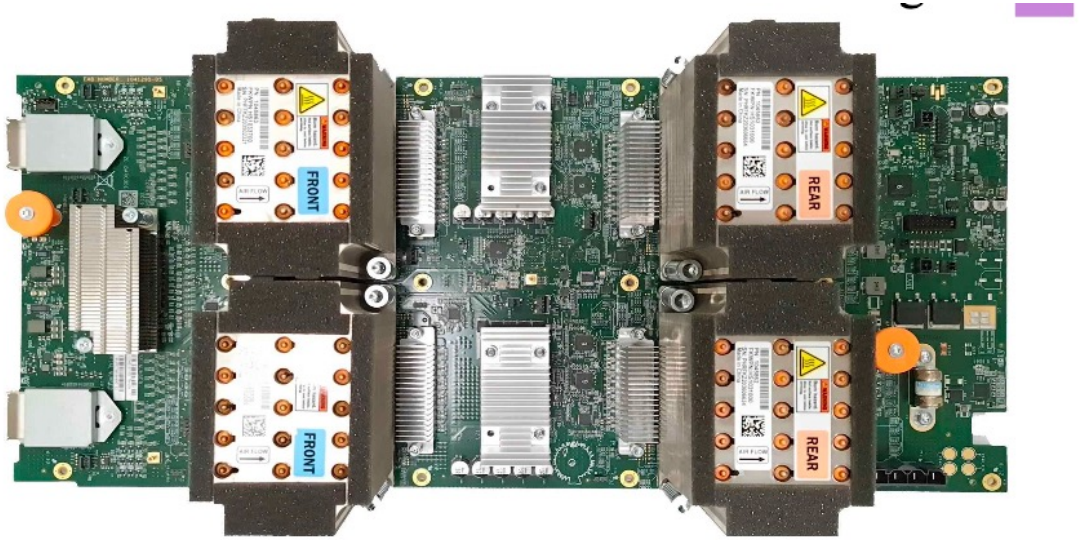
(4) Insights: 4 lessons about DNN applications

#	Insight	Solution
7	Production inference normally needs **multi-tenancy** (for lower costs, reduced latency and good software engineering practices)	⟹ DSAs need **local memory** (i.e. DRAM , since all weights can't fit in SRAM) for fast **switching** time between models.
8	DNNs **grow ~1.5x** annually in memory and compute	⟹ DSAs architects should provide **headroom** to support DNN growth.
9	DNN workloads **evolve** with DNN breakthroughs	⟹ **Programmability** and **flexibility** are important factors for inference DSAs to track DNN progress.
10	The inference **SLO** is **P99 latency**, not batch size	⟹ DSAs should take advantage of **larger batch sizes**.

What is the solution? Is it feasible?

These 10 lessons helped shape the new TPUv4i's design:

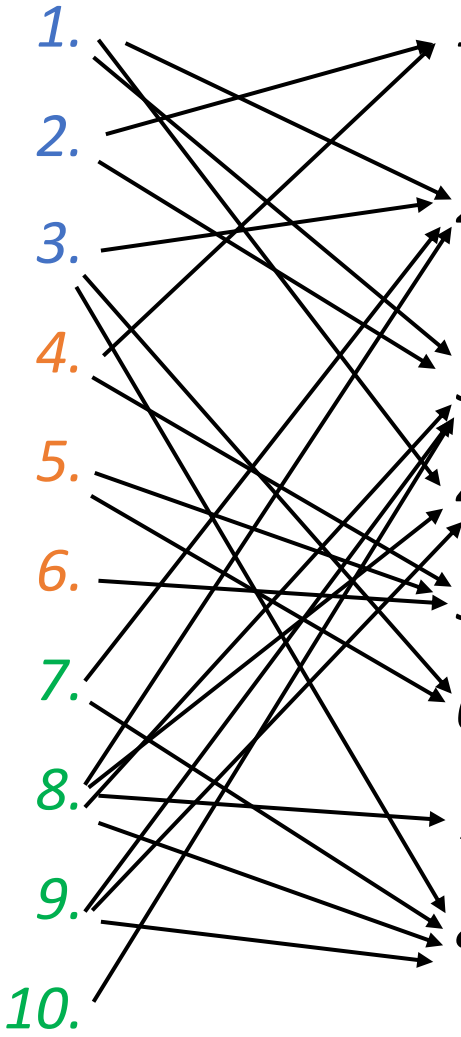
1. Based on **TPUv3**
2. One core chip for **inference** (two core TPUv4 for training)



gure 7. TPUv4i board with 4 chips that are connected by ICI.

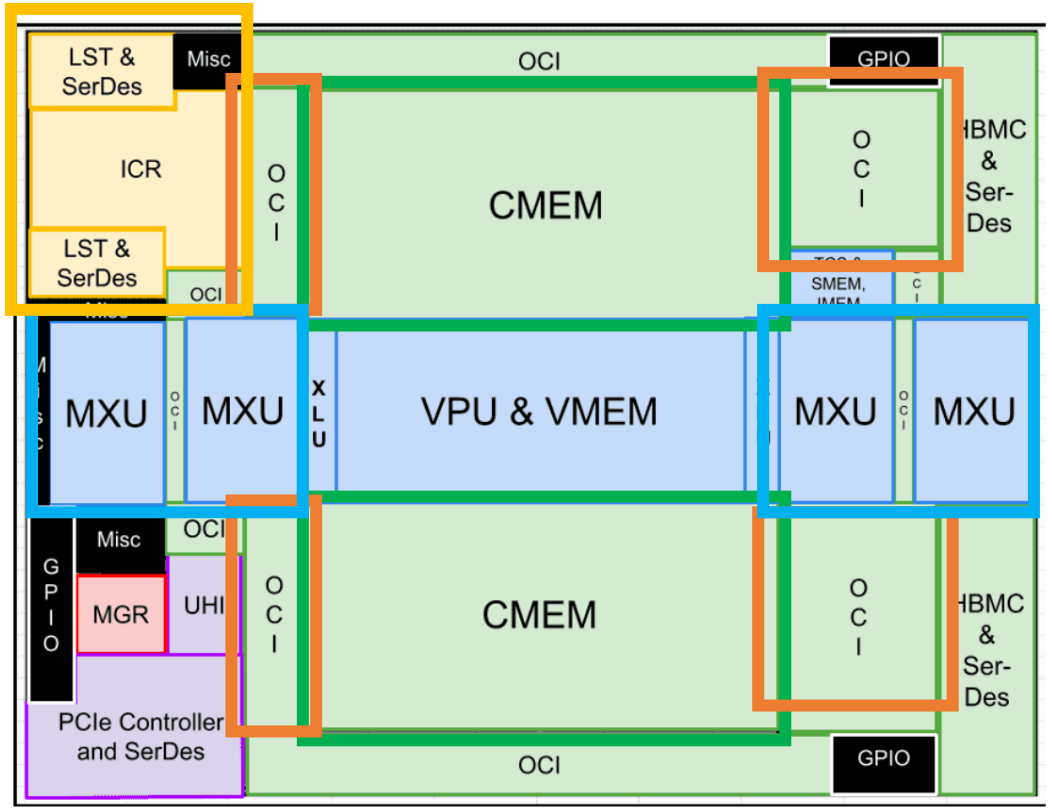
What is the solution? Is it feasible?

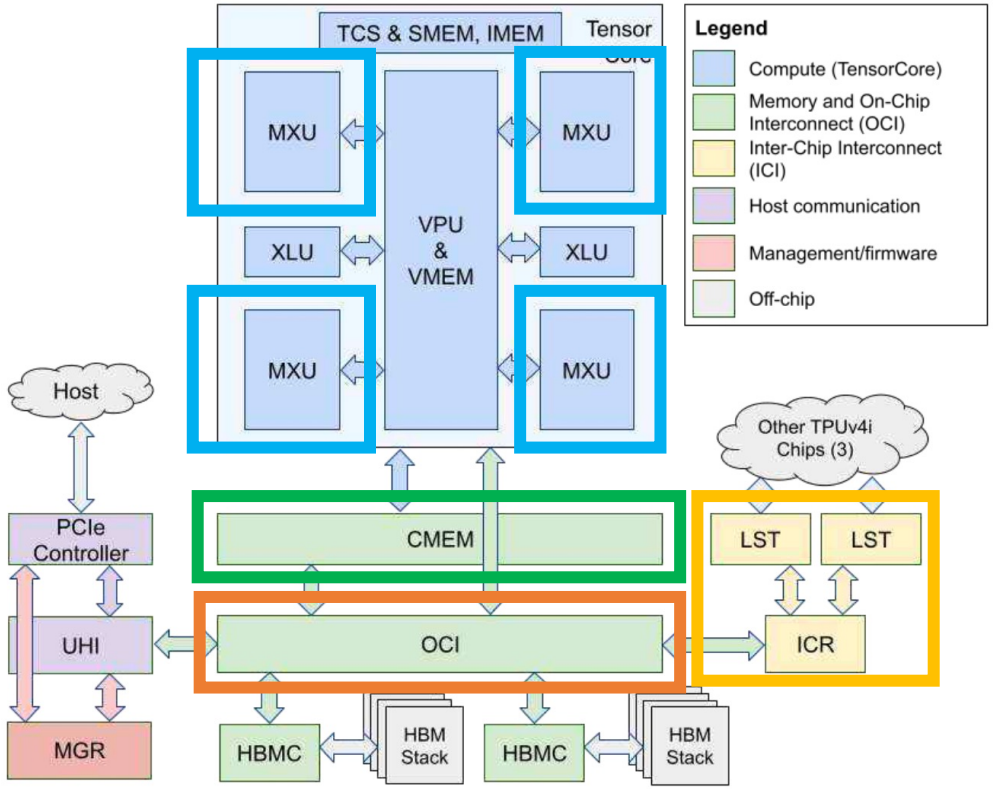
Unequal hardware improvements 1.
Leverage compiler optimizations 2.
Design for perf/TCO 3.
Backwards ML 4.
Air cooling 5.
Floating point support 6.
Multi-tenancy 7.
Memory/compute 1.5x grow 8.
DNN workloads evolve 9.
Inference SLO is P99 latency 10.



1. *Compiler compatibility, not binary compatibility*
2. *Increased on-chip SRAM storage with CMEM*
3. *4D tensor DMA*
4. *Custom on-chip interconnect (OCI)*
5. *Arithmetic improvements*
6. *Clock Rate, TDP*
7. *Inter-Chip Interconnect scaling*
8. *Workload analysis features*

What is the solution? Is it feasible?





2. Increased on-chip SRAM storage with common memory (CMEM)

5. Arithmetic improvements with four MXUs

4. Custom on-chip interconnect (OCI)

7. Inter-chip Interconnect (ICI) scaling

3. 4D tensor Direct Memory Access (DMA)

What is the solution? Is it feasible?

1. Compiler compatibility, not binary compatibility:

XLA compiler compatible with multiple TPUs

$\Longrightarrow$ only LLO should be optimized per TPU

8. Clock Rate, TDP (Thermal Design Power):

Clock rate of 1.05 GHz + chip TDP of 175W

$\Longrightarrow$ Support air cooling and reduce TCO

6. Workload analysis features:

Tracing and performance counter hardware features

$\Longrightarrow$ Analyze system-level bottlenecks and support continuous system-level performance improvements

What is the takeaway message?

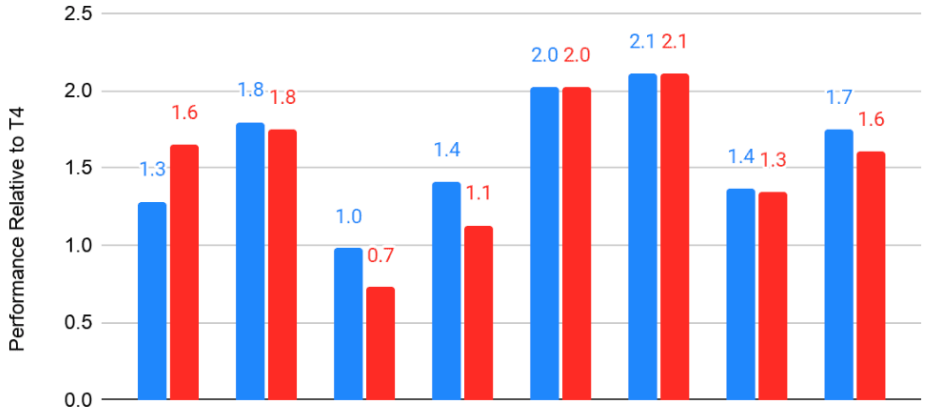
Key takeaway messages in designing DSA for DNN:

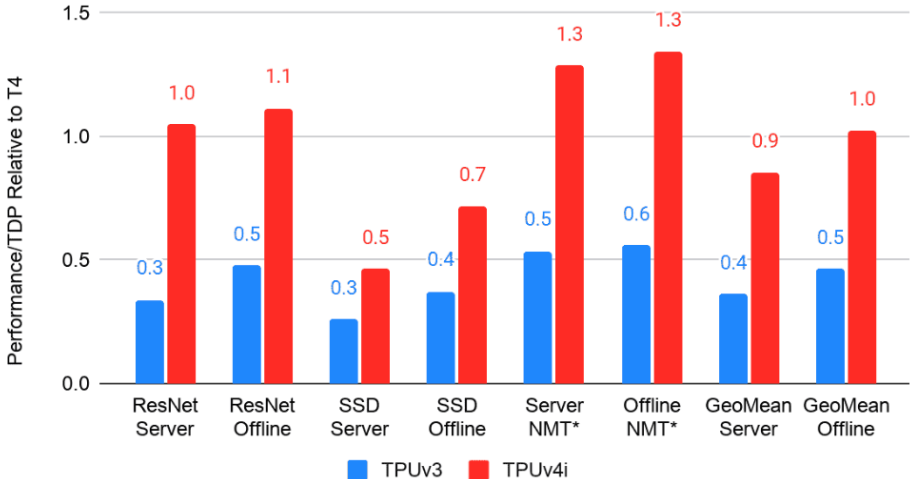
1. **Design** should be driven by the specific needs of **real-world workloads**.
2. **Iterative improvement** is the key to keeping up with the fast-evolving demand.
3. DSA should be **programmable** and **flexible** to cover previous/current/future needs.
4. **Moore's Law** is diminishing and **Dennard scaling** is dead $\Longrightarrow$ **hardware/software/DNN co-design** is the best chance for DNN DSAs to continue making progress.

What is the takeaway message?

Takeaway messages regarding the TPUv4i chip itself:

1. TPUv4i has **similar performance** to TPUv3 (1.9x TPUv2), but is much **more energy efficient** (2.3x perf/TDP).
2. ”**Killing 2 birds with 1 stone**": two separate chips for inference/training is key to achieve **better perf/TCO**.
3. **Thermal Design Power** offers a good proxy for DSA **Total cost of ownership**





Will this paper win the test of time award?

NO

1. **Architectural choices** made for the TPUv4i **obsolete** in 15+ years?
2. New **computing paradigms** are emerging.

   $\Longrightarrow$ DNN DSAs may be very different in 15+ years.
3. Most insights are related to ML **challenges from 2015-2020**

   $\Longrightarrow$ ML community's challenges will be different in 15+.

Name one reason why this paper should have not appeared in MLSYS, NeurIPS, ICML, OSDI, ASPLOS, etc.?

The paper was presented to ISCA21 (103 citations on Semantic Scholar)

1. Valuable for the hardware designer community.
2. Scope limited to Google infrastructure?
3. Limited comparison to existing (similar) DSA for DNN?
4. Architectural description broad, with few details
5. Concerns regarding writing:
    1. Paper is very dense, with a lot of jargon/abbreviations, hard to read.
    2. Some motivations are obscured by referencing internal feedback from other design teams, without providing further insights.

Preliminary

Feature	*TPUv1*	*TPUv2*	*TPUv3*	*TPUv4i*	*NVIDIA T4*
Peak TFLOPS / Chip	92 (8b int)	46 (bf16)	123 (bf16)	138 (bf16/8b int)	65 (ieee fp16)/130 (8b int)
First deployed (GA date)	Q2 2015	Q3 2017	Q4 2018	Q1 2020	Q4 2018
DNN Target	Inference only	Training & Inf.	Training & Inf.	Inference only	Inference only
Network links x Gbits/s / Chip	--	4 x 496	4 x 656	2 x 400	--
Max chips / supercomputer	--	256	1024	--	--
Chip Clock Rate (MHz)	700	700	940	1050	585 / (Turbo 1590)
Idle Power (Watts) Chip	28	53	84	55	36
TDP (Watts) Chip / System	75 / 220	280 / 460	450 / 660	175 / 275	70 / 175
Die Size (mm^2)	< 330	< 625	< 700	< 400	545
Transistors (B)	3	9	10	16	14
Chip Technology	28 nm	16 nm	16 nm	7 nm	12 nm
Memory size (on-/off-chip)	28MB / 8GB	32MB / 16GB	32MB / 32GB	144MB / 8GB	18MB / 16GB
Memory GB/s / Chip	34	700	900	614	320 (if ECC is disabled)
MXU Size / Core	1 256x256	1 128x128	2 128x128	4 128x128	8 8x8
Cores / Chip	1	2	2	1	40
Chips / CPUHost	4	4	4	8	8

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]	--

Table 2. Energy per Operation: 45 nm [16] vs 7 nm. Memory is pJ per 64-bit access.

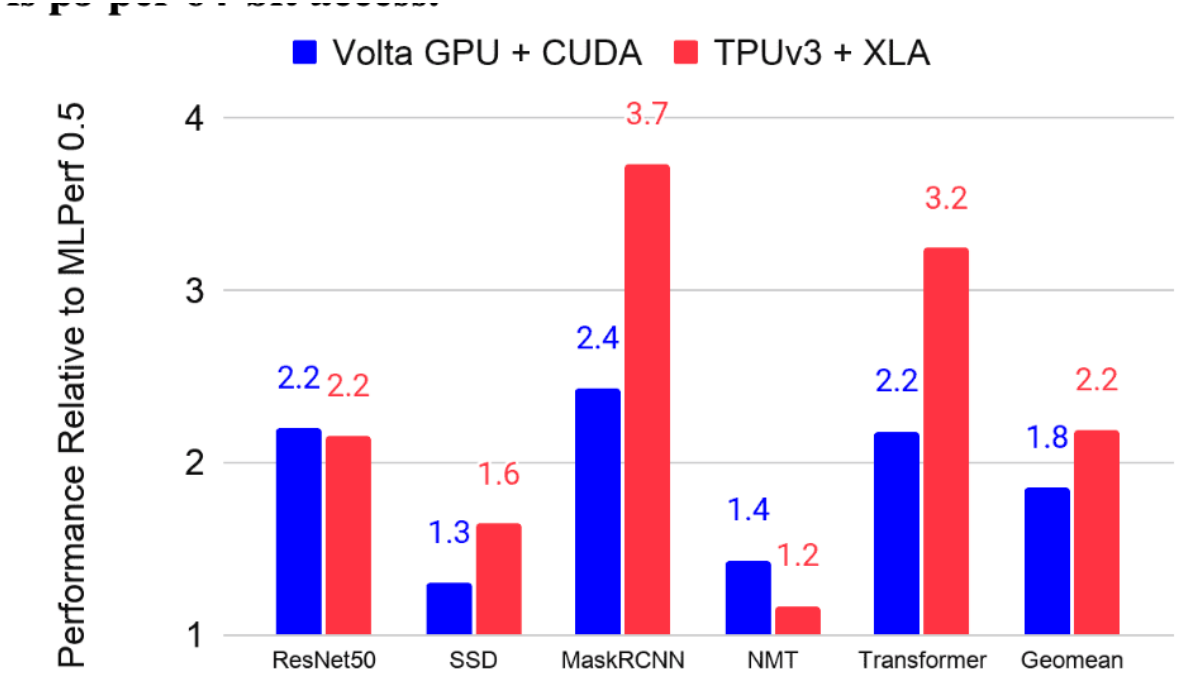
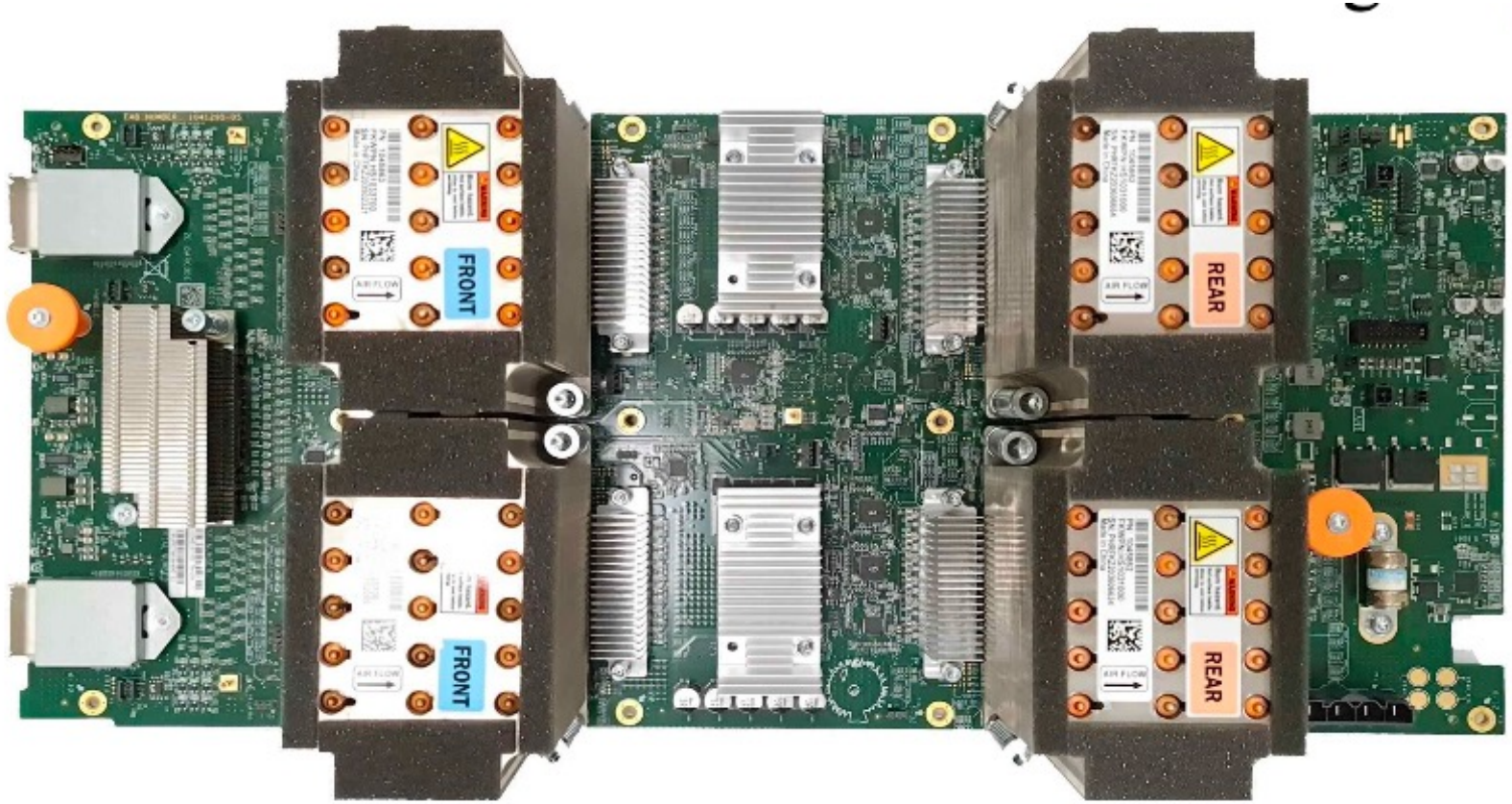


Figure 2. DSA gains per chip for MLPerf Training 0.5 to 0.7 over 20 months for the same compilers. The unverified TPUv3 MLPerf 0.5 scores for Mask R-CNN and Transformer are from [23]; all other results are from [28].



gure 7. TPUv4i board with 4 chips that are connected by ICI.

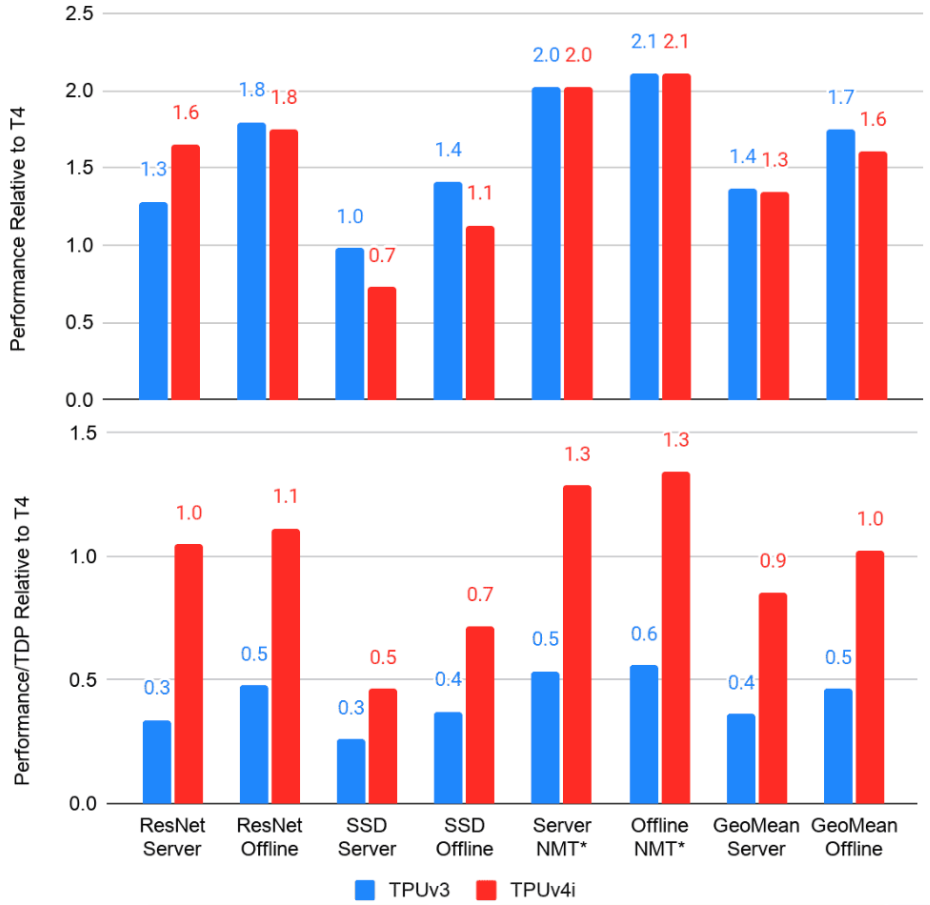


Figure 9. Performance (top) and performance/system TDP ③ relative to T4 for TPUv3/v4i in our datacenter (§7.B). Note that the T4 uses int8 for ResNet and SSD and fp16 for NMT. The TPUs use bf16 for all three to maintain backwards ML compatibility ④ with TPUv3/v4. MLPerf Inference 0.7 omits NMT, so we use MLPerf Inference 0.5 code for it and 0.7 code for ResNet and SSD in Figures 9, 10, and 13. (These results are unofficial, as they have not been verified by MLPerf.)

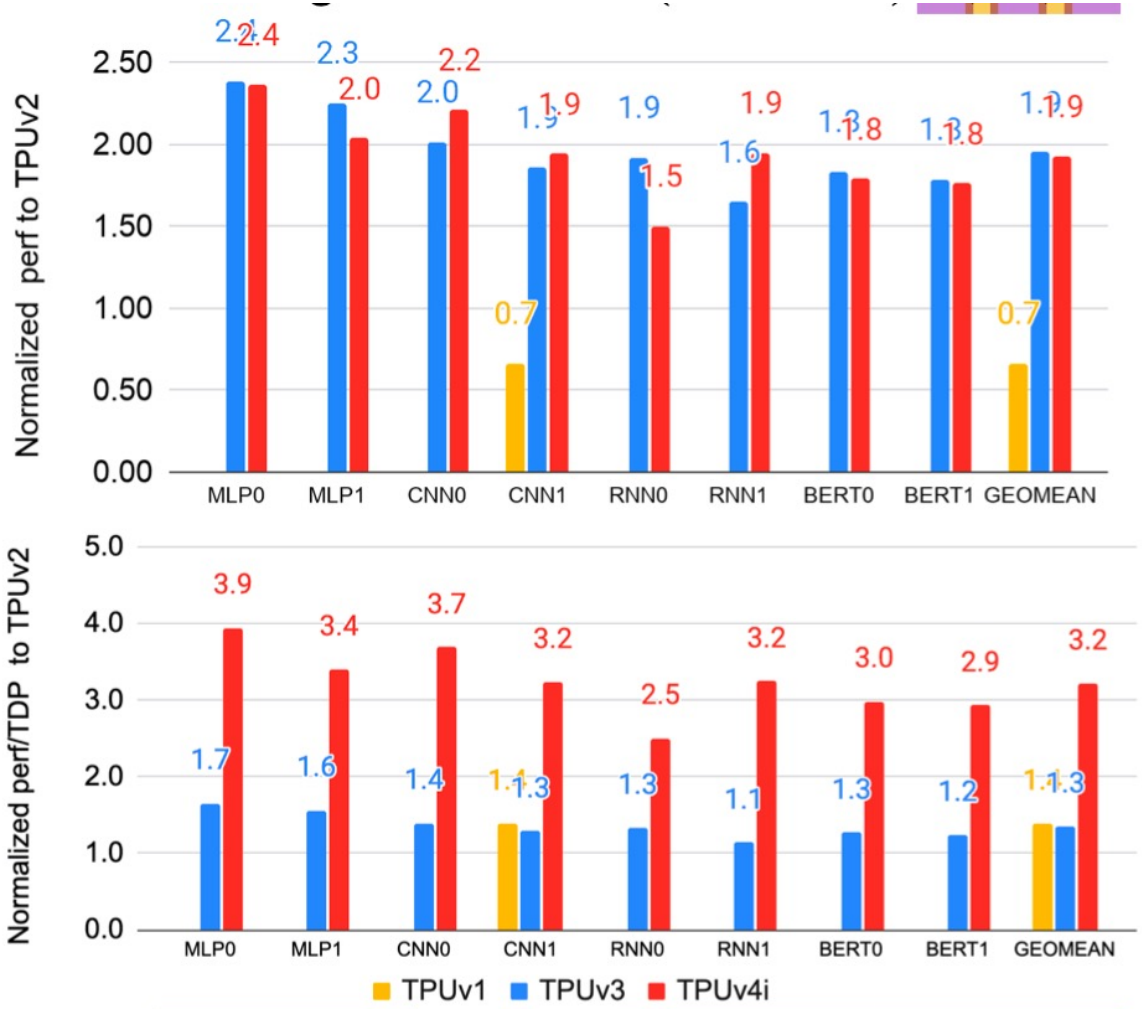


Figure 8. Performance (top) and performance/system Watt ③ for production apps ⑨ relative to TPUv2 for the other TPUs.

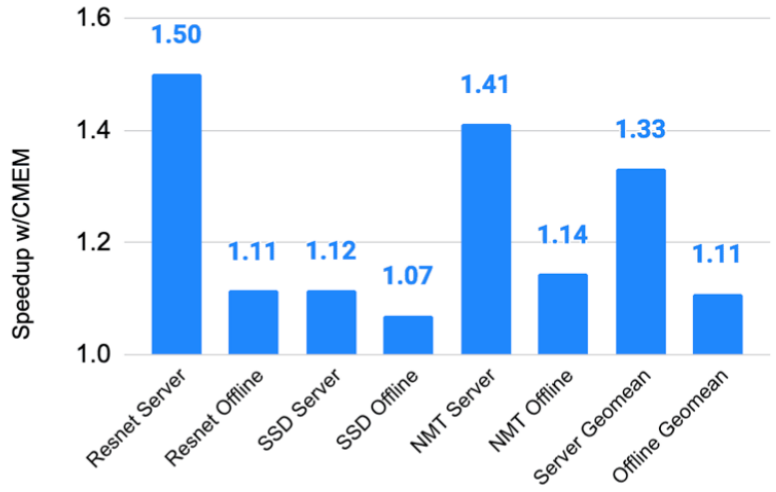


Figure 10. Unverified MLPerf Inference impact of turning on CMEM vs no CMEM.

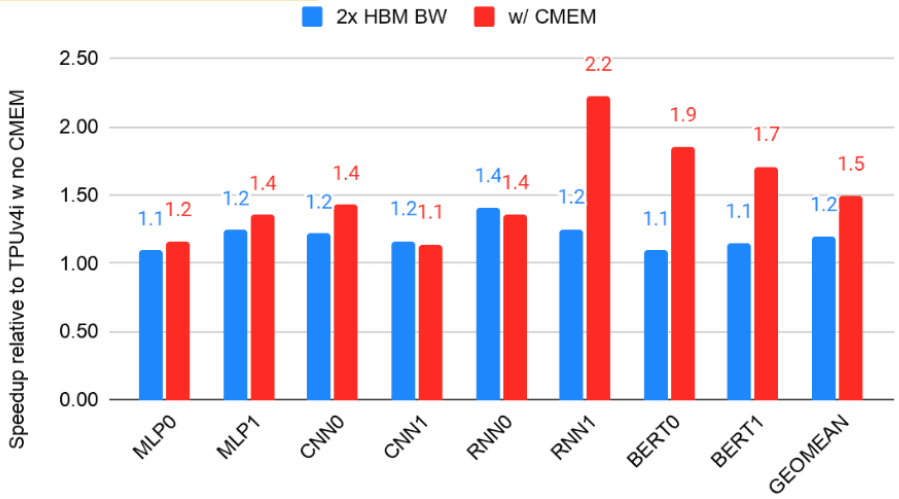


Figure 11. Performance of 2X HBM bandwidth with no CMEM relative to CMEM with standard HBM bandwidth.

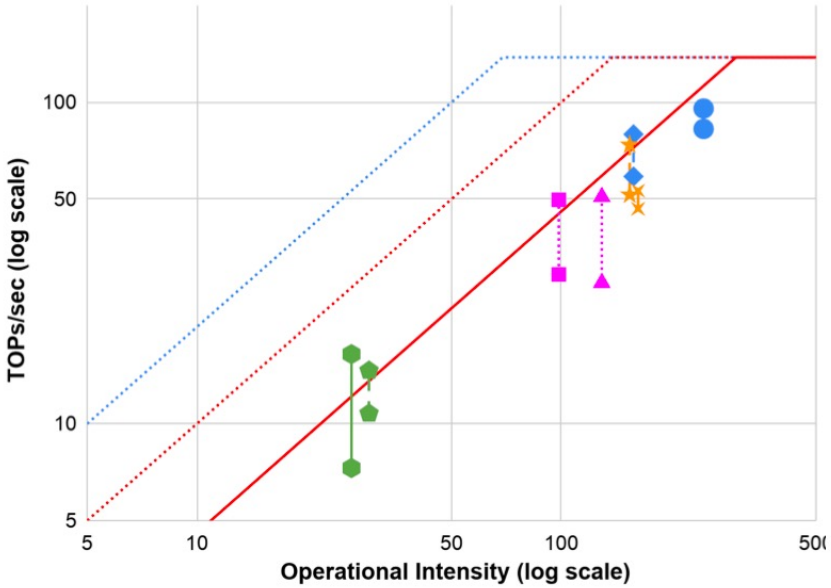


Figure 12. Roofline model showing apps without CMEM (low point) vs with CMEM (high point). Operational intensity (OI) here is operations divided by memory accesses to HBM *or* to CMEM. If OI were relative to HBM only, CMEM would increase OI and move the points to the right as well as up.

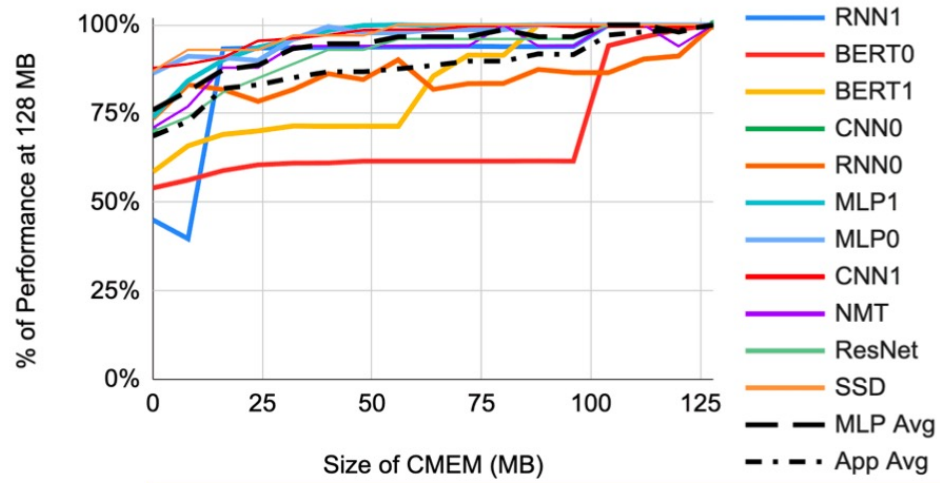


Figure 13. Percent of 128 MB speed as CMEM varies 0–128 MB for the apps and MLPerf Inference 0.5-0.7 server code.

	perf/TDP vs T4	*TDP (W)*	Cores	*GHz*	*MiB SRAM*	*Type DRAM*
T4	1.00	70	40	0.6	20	GDDR6
Goya	0.46	200	8	2.1	≈32	DDR4
Nervana	0.69	100	24	1.1	60	LPDDR4
Zebra	0.25	225	--	--	54	DDR4
HanGuang	2.17	280	4	0.7	192	none

Table 6. Five DSAs that ran the MLPerf Inference 0.5 (Goya, Nervana, HanGuang) or 0.7 (T4, Zebra) server scenarios. Goya ran only SSD, and the last three ran only ResNet. Performance is relative to T4 for MLPerf Inference 0.5. TDP in this table is per chip rather than per system, since the latter was unavailable. All have 16–32GB of DRAM except HanGuang, which has none. All results are from [29].

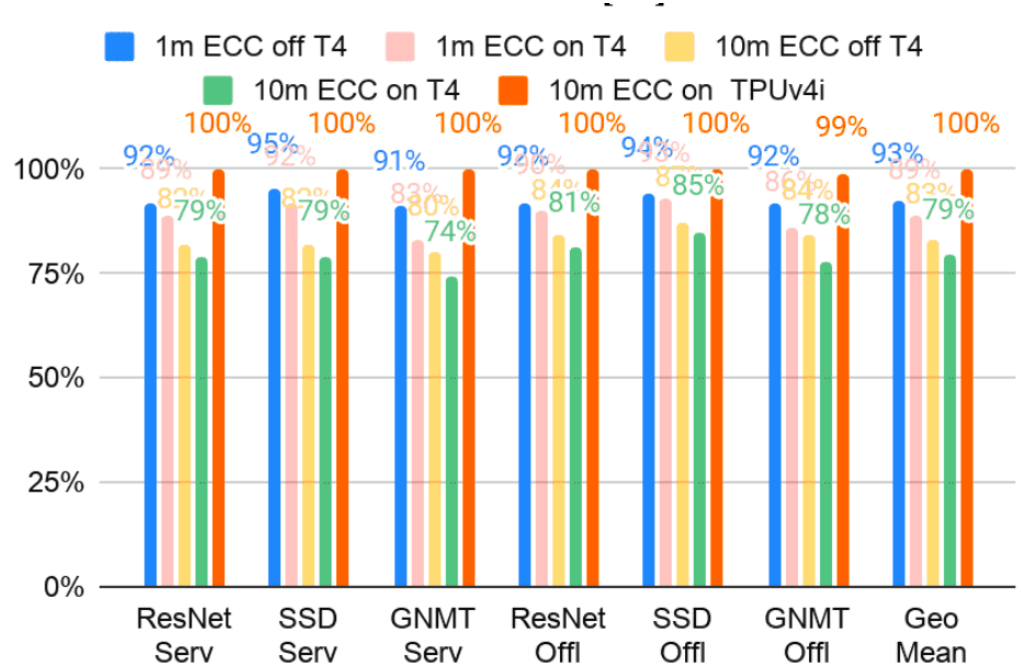


Figure 14. T4 and TPUv4 running unverified MLPerf Inference benchmarks 0.5/0.7 at Google with memory ECC off and on. NVIDIA's MLPerf score is 100% for T4 and 100% for TPUv4i is unverified MLPerf in our datacenter. For the 1 minute case, the T4 was idle initially. It then ran MLPerf with ECC off and on for 1 minute at the fastest clock rate. (T4 offers inline ECC, which uses memory bandwidth.) For the 10 minute case, the machines are not idle beforehand. TPUv4i's speed is unchanged whether ECC is on or off or how long it runs.

TPUv4i FLOPS/s would drop to 22% and roofline to 67%.

TPU	TPUv1	TPUv2	TPUv3	TPUv4i
MXUs/Chip	1 256 x256	2 128x128	4 128x128	4 128x128
MXUs % Die Area	24%	8%	11%	11%
FLOPS/s Utilization	20%	51%	38%	33%
HBM Roofline Util.	20%	66%	63%	99%

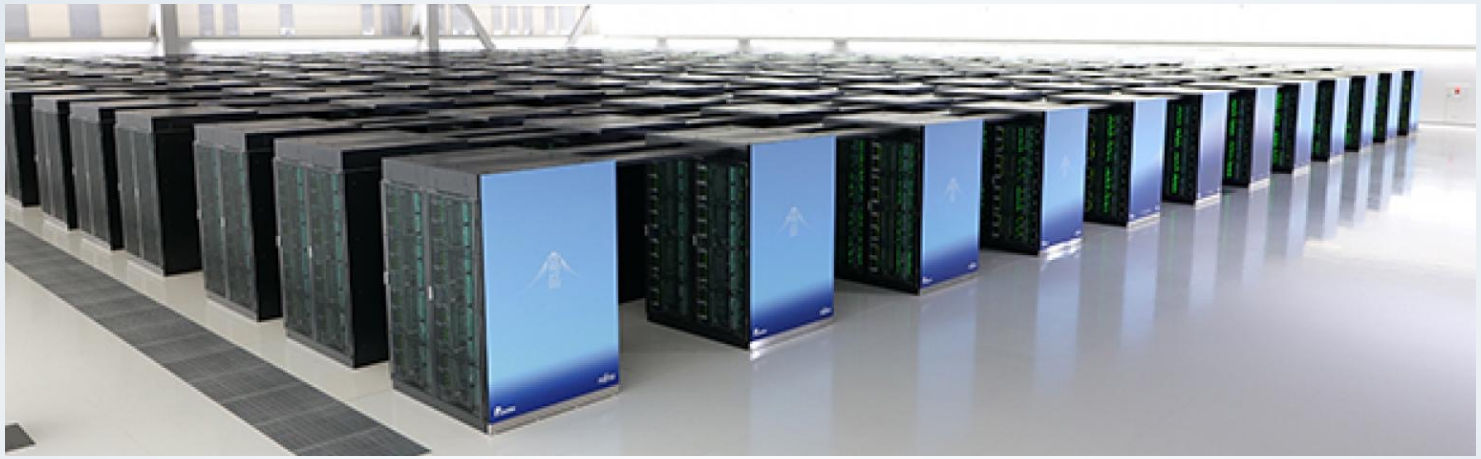
Table 7. Average utilization of peak performance and of roofline for our eight production applications.

RaPiD:

AI Accelerator for Ultra-low Precision Training and Inference

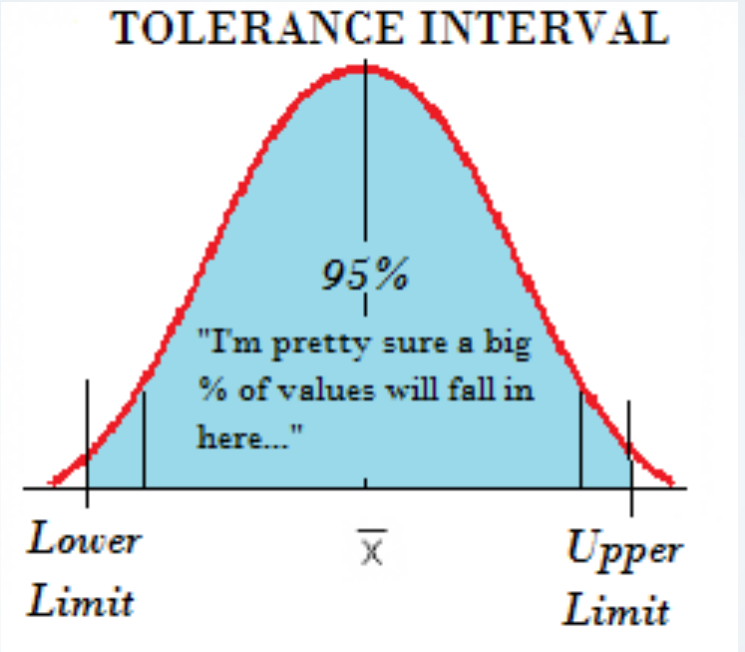
Elevator pitch

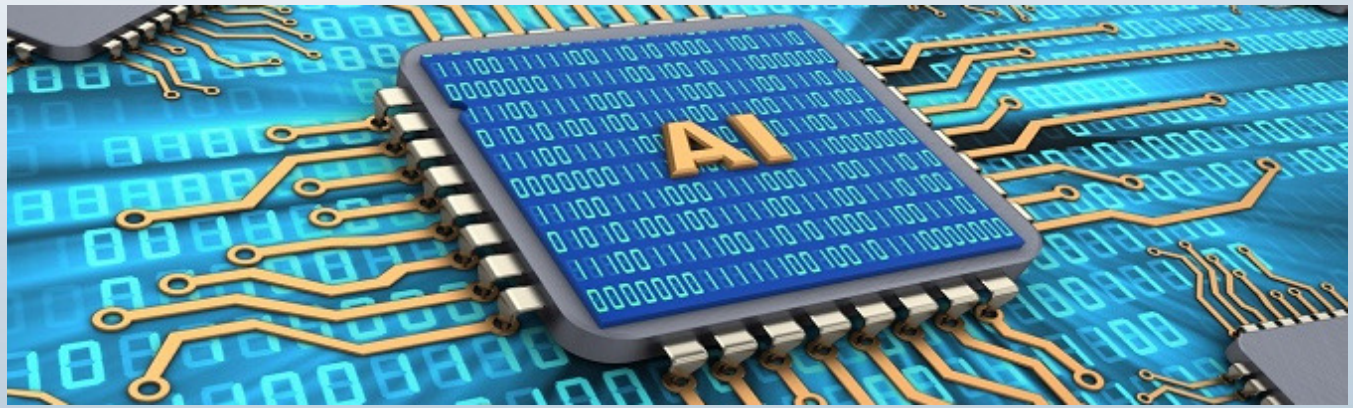
AI =



or

Move forward?





Then, hopefully…

Write-up questions

x

1. What is the problem?

- AI/DNN have different paradigm of computation workloads
- the high accuracy of these tasks have a high computational cost.

- => they severely stress the capabilities of traditional computing platforms.

2. What are the insights?

- AI workloads are static dataflow graphs
- the computations only need a small number of primitives
- specialized hardware accelerators can improve system performance
- AI-workloads are error-resilient, so precision scaling can be beneficial.

3. What is the solution?

- varying level of precision is designed and supported;
- both performance and energy efficiently should be improved;
- several operations, like data-shuffle and polling, by special function units;
- zero-gating logic to boost the performance of sparse AI models;
- memory neighbor interface to incorporate core-to-core and core-to-memory communication and synchronization.

3. solution – cont'd

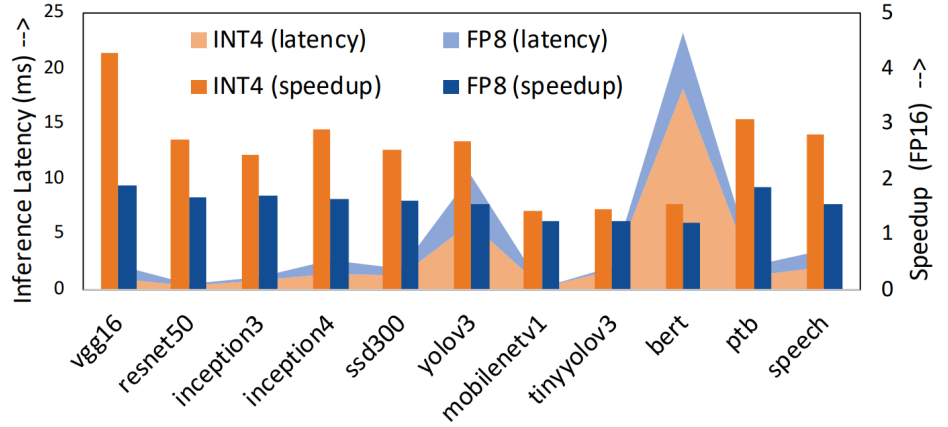


Figure 13: Classifications per second using 4-core RAPID chip

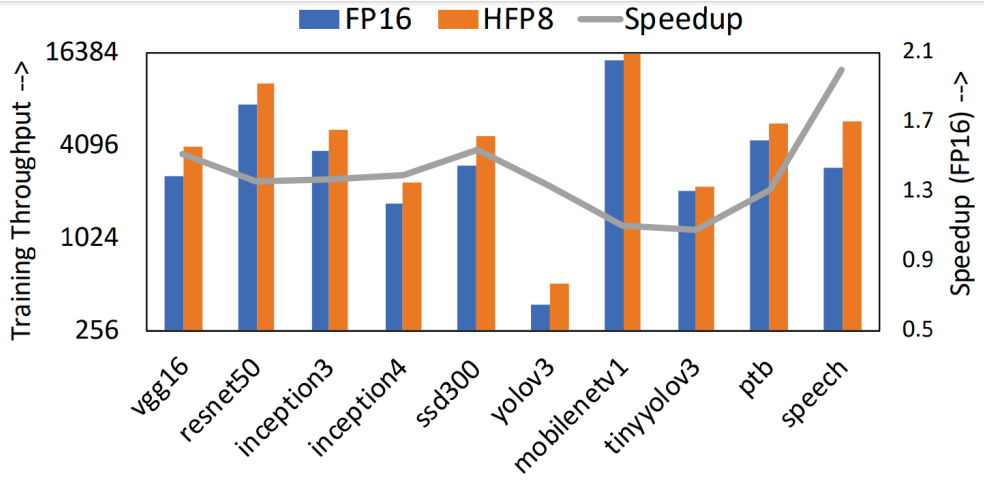


Figure 15: Throughput with 4-chip RAPID training system

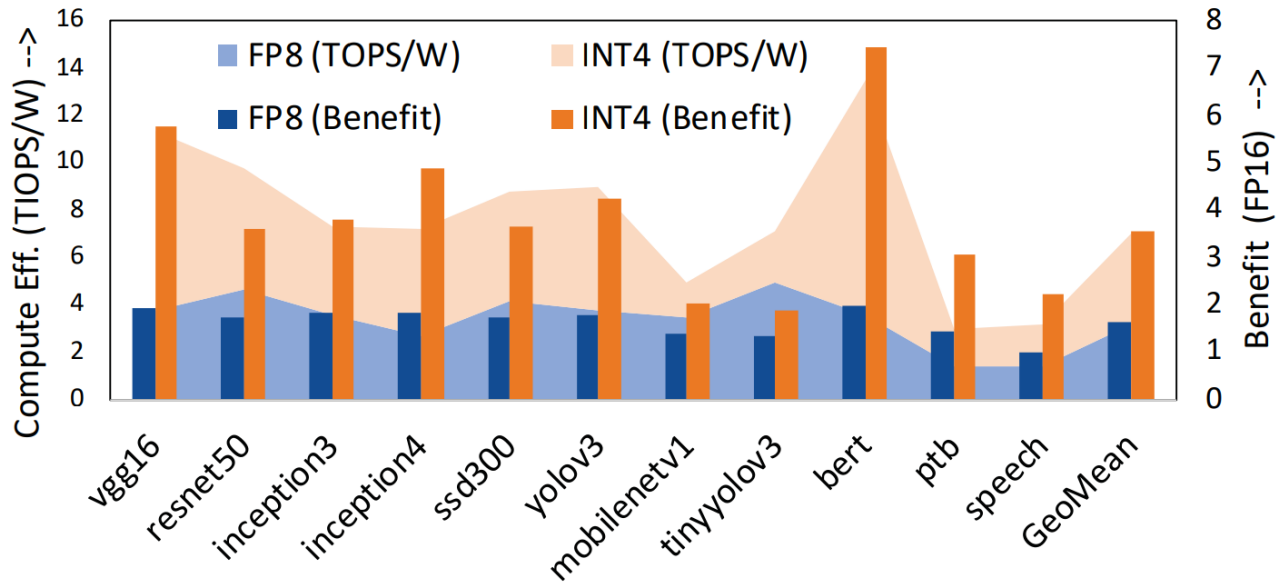


Figure 14: Sustained TOPS/W on 4-core RAPID chip

3. solution – cont'd

4. What is the takeaway message?

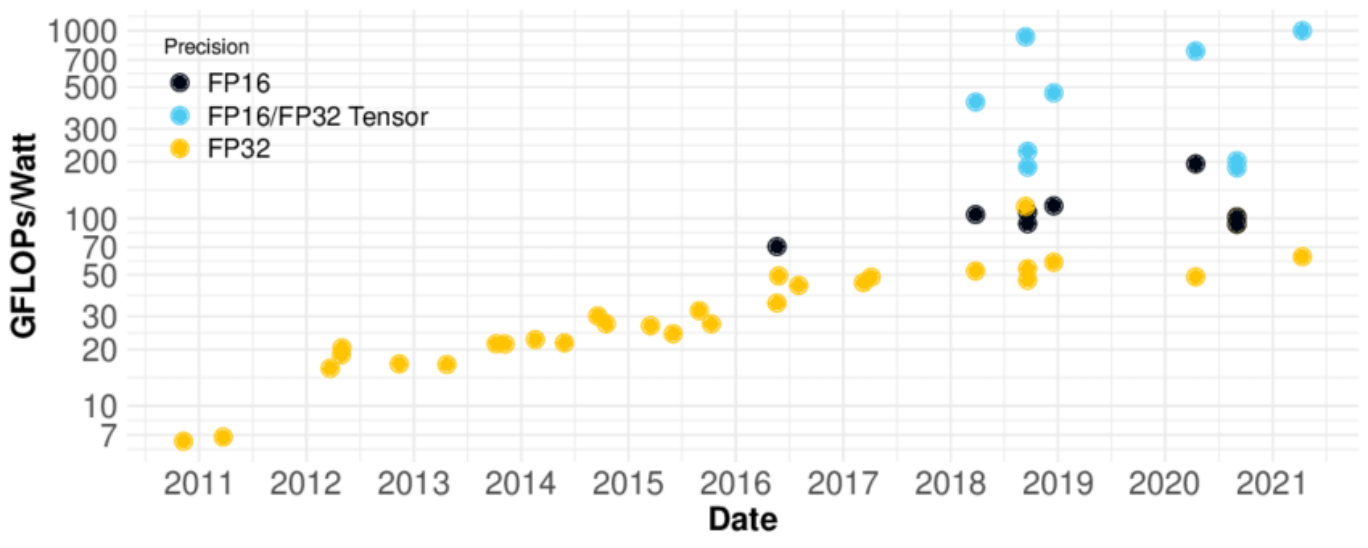
- Precision scaling is an effective foundation for designing hardware accelerators for AI.

5. Will this paper win the test of time award?

- Possible, from the perspective that the chip could be used for edge or battery-operated devices for AI applications with less energy consumption.
- However, in terms of generic AI accelerators, I don't think the design has achieved a great leap.

6. Reason why it isn't top conference quality?

- No critical reasons to reject this paper.
- However, how much better this accelerator performs with/compared-to different CPU/GPU/TPUs? Below btw.



Theoretically, for Nvidia GPU

Thank you

Q&A