In-Datacenter Performance Analysis of a Tensor Processing Unit

Original Paper by: Norman P. Jouppi et al. (Google, ISCA `2017)

Presented by Jongyun Hur

Introduction

Introduction

Why build the TPU?

2006: Rejected Custom HW → **2013: TPU Project Started** → **2015: Datacenter Deployment**

FPGAs/ASICs Considered
(Only niche apps needed custom HW)

↓

Excess Capacity
(Abundant idle CPU resources)

↓

"Free" Compute
(Spare cycles cost almost nothing)

↓

Conclusion: CPU Suffices
(No custom HW needed)

DNN Accuracy Explosion
(Voice Search: +3 min/day?)

↓

Compute Demand Crisis
(Datacenter demand would **Double**)

↓

CPU Scaling Limits
(Cost prohibitive to scale with CPUs)

↓

Solution: Build TPU
(10x Perf/Watt vs GPU,
15-month turnaround)

↓

2016: Data Collection

↓

2017: Paper Publication

Introduction

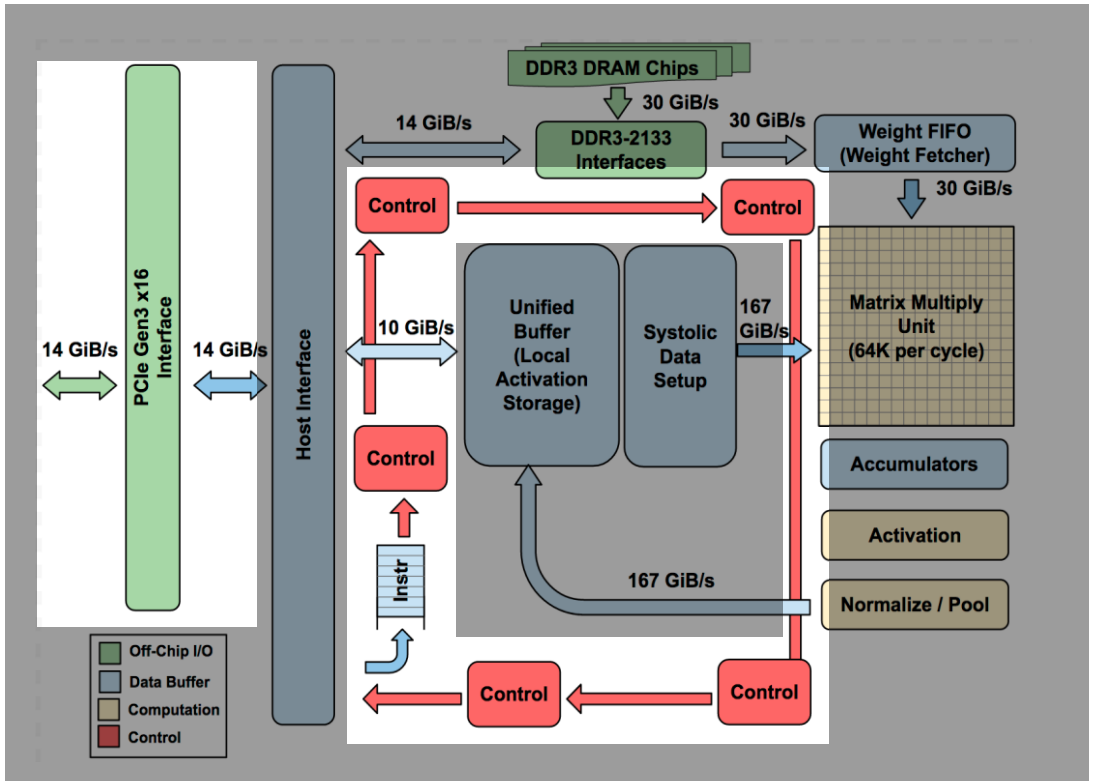
The 15-month mandate: Design trade-offs

Design Area	The Trade-off Decision	Reason for Speed
1. Connectivity	CPU Integration → **PCIe Coprocessor**	To plug into existing servers and avoid delaying deployment.
2. Control	Self-fetching Instructions → **Host-Driven (FPU-like)**	To **simplify hardware design** and debugging.
3. Memory	High Bandwidth (GDDR5) → **Conservative DDR3**	To use a proven interface and minimize circuit design risks.
4. Features	Power Gating → **Features Omitted**	Explicitly omitted to meet the strict **"time-to-deployment"** goal.

Architecture

Architecture

TPU: System control & Instruction flow



CPU

- Host-Driven Control
- Sends instructions to execute

PCIe

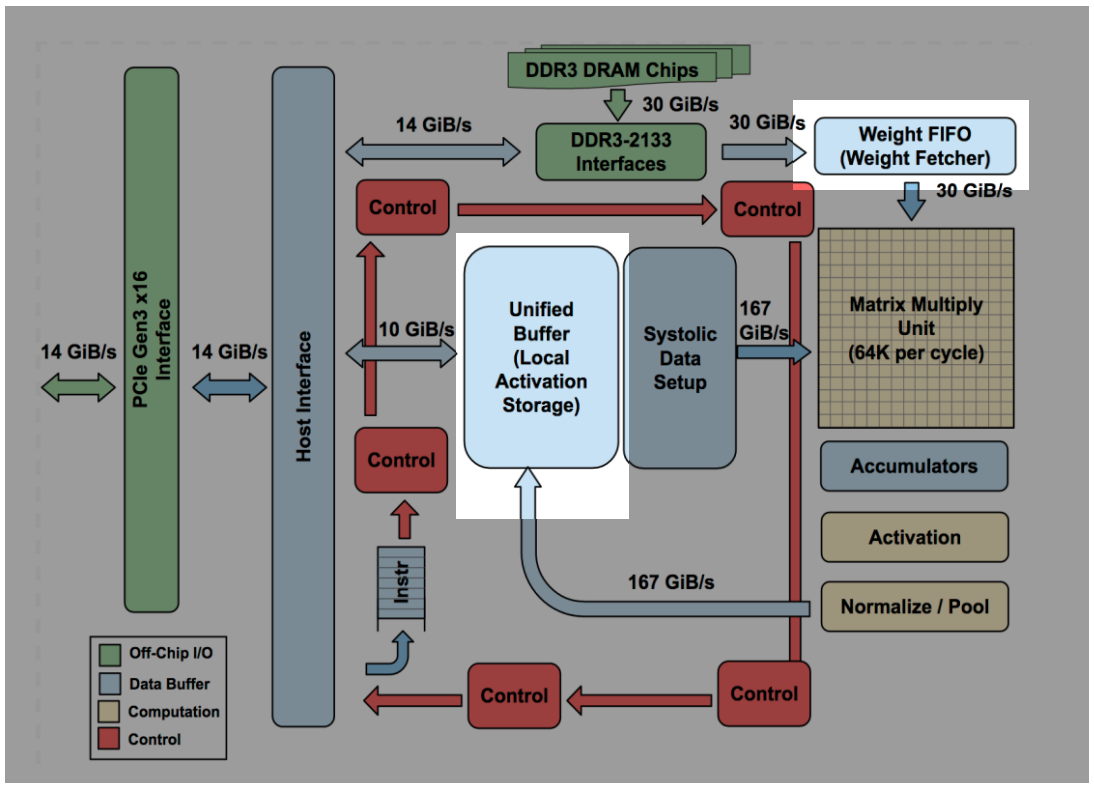
- Instruction Push
- "Instructions are sent over PCIe Bus"

TPU

- Passive Execution
- "Closer to FPU coprocessor spirit"

Architecture

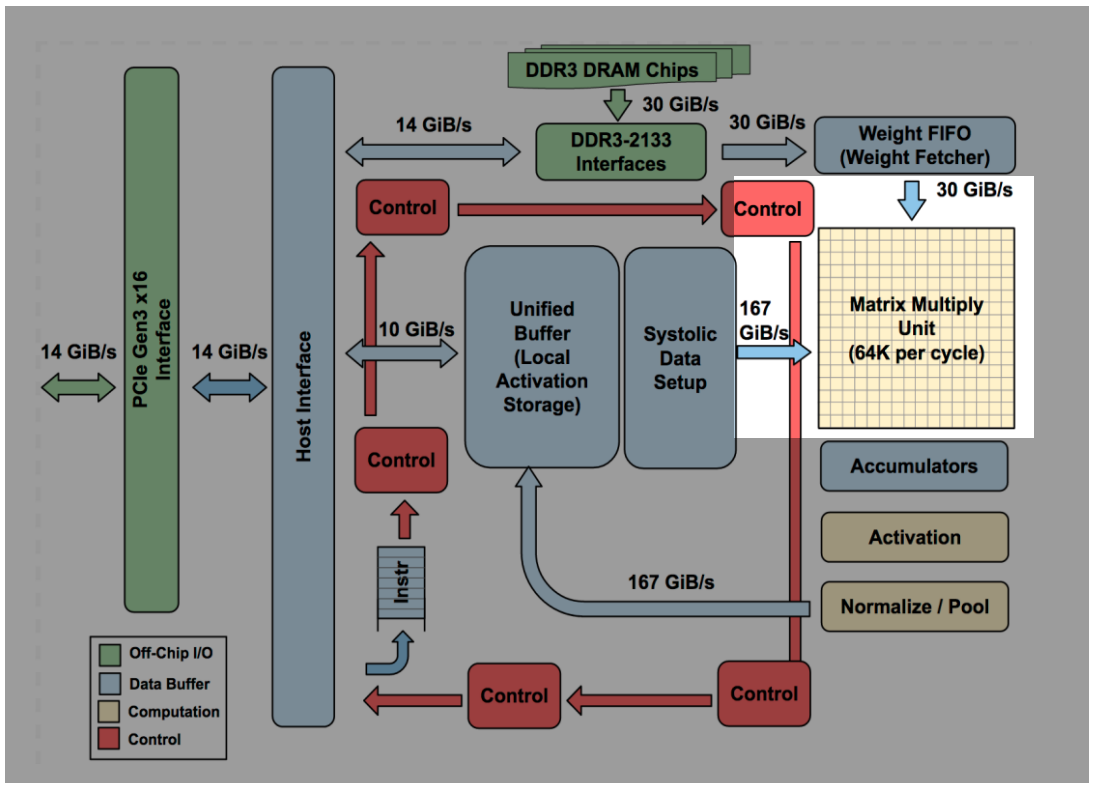
TPU: Unified Buffer & Weight FIFO



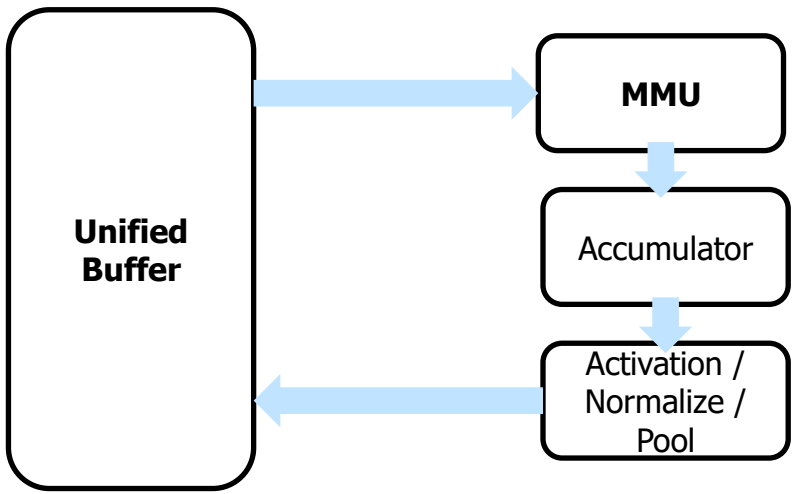
Category	Unified Buffer	Weight FIFO
Data Type	Activations	Weights
Source	Host CPU (via PCIe Bus)	Off-chip DRAM (Weight Memory)
Key Feature	Software-Managed	Prefetching (Hides Latency)

Architecture

TPU: Deterministic matrix multiplication



Matrix Multiply Unit (MMU)
• **65,536 MACs** (256 × 256 Array)
• **Systolic Data Flow**
• **8-bit Integer Math** (Optimization for Inference)



Architecture

Matrix Multiply Unit: 256x256 Systolic Array

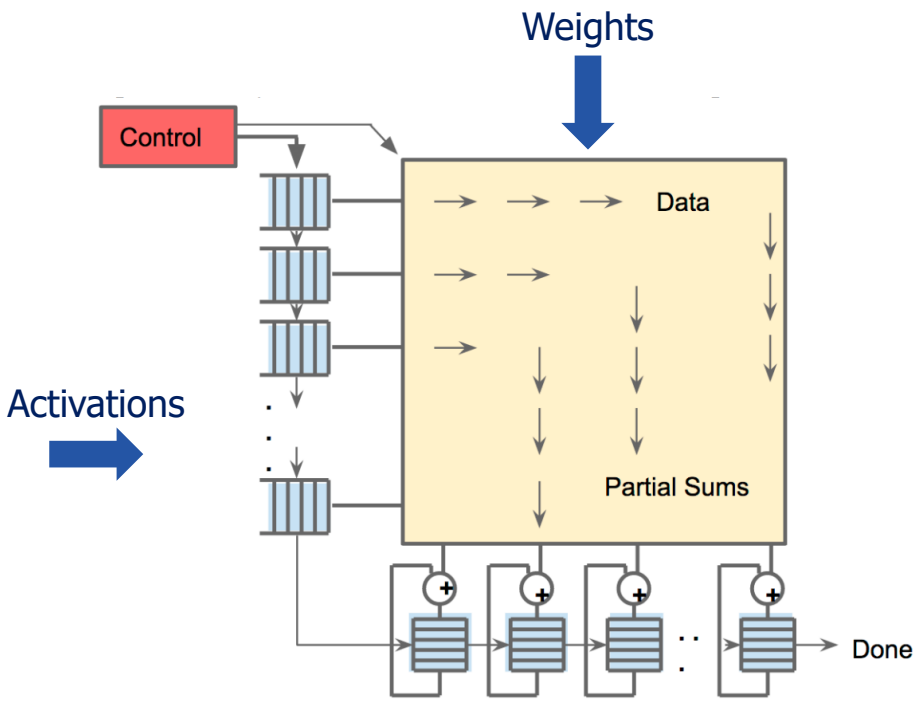


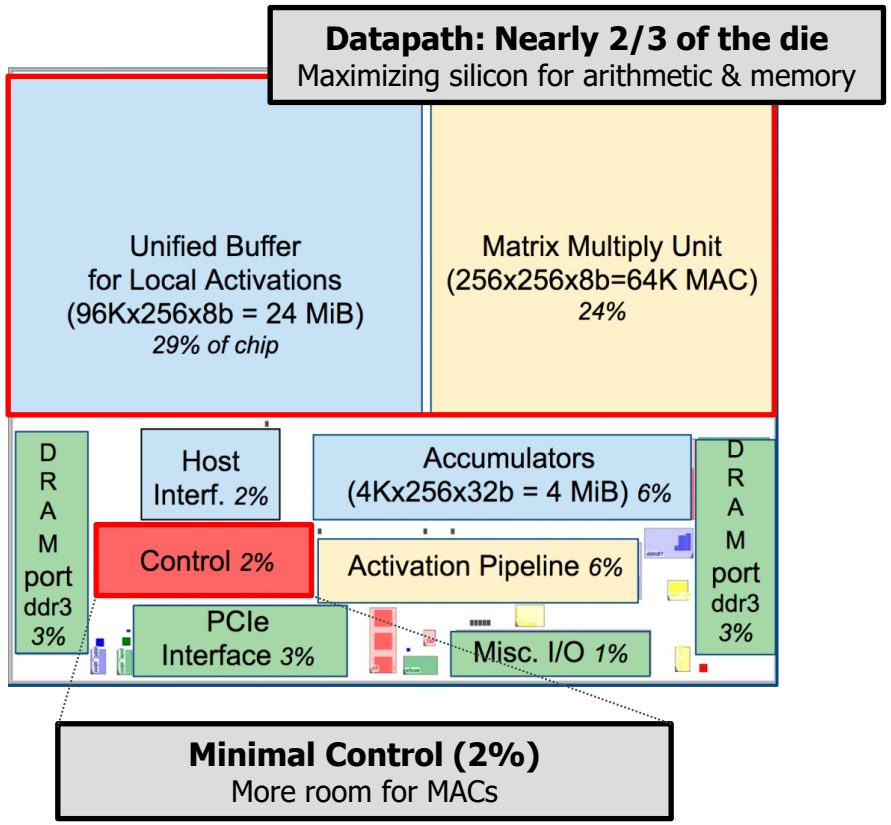
Figure 4. Systolic data flow of the Matrix Multiply Unit.

MXU: 256 × 256 Systolic Array of 8-bit MACs

- **Mechanism:** Activations flows in from the left, Weights from the top (Diagonal Wavefront).
- **Efficiency: "Read Once, Use Many"** strategy reduces expensive 'United Buffer' accesses.

Architecture

Floorplan of TPU die

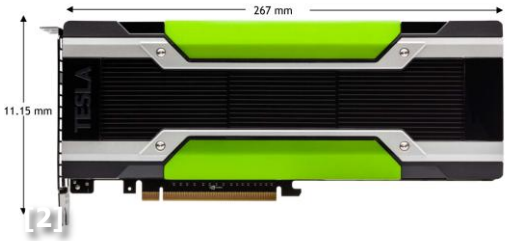


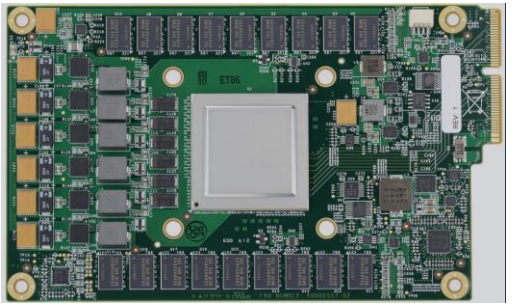
Component	Die Area	Key Role & Characteristics
Unified Buffer	**29%**	**Data Reuse**
Matrix Multiply Unit	**24%**	**Compute Power**
Control Logic	**2%**	**Minimalism** • No caches or branch prediction • Saves space for more arithmetic units

Methodology

Methodology

Comparing TPU with contemporary datacenter hardware





Metric	CPU (Baseline)	GPU (Accelerator)	TPU (Custom ASIC)
Product	Intel Haswell E5-2699 v3	Nvidia K80 (Tesla)	Google TPU v1
Technology	22 nm	28 nm	28 nm
Die Size	662 mm²	561 mm² (per die)	**< 331 mm²**
TDP (Power)	145 W	150 W (per die)	**75 W**
Memory	51 MiB Cache	16 MiB + 12 GB GDDR5	28 MiB + 8 GB DDR3
Peak Ops/s	2.6 TOPS (8b)	2.8 TOPS (FP)	**92 TOPS (8b)**

[1] Intel, "Intel Haswell E5-2699 v3"
[2] NVIDIA, "Tesla K80 GPU Accelerator Board Specification"

Methodology

Datacenter workloads

엄격한 응답시간 제한이라는 조건 하라는 걸 먼저 소개

엄격한 응답시간 제한이라는 조건 하에서 tpu가 세팅한 결과값

Constrained by 7ms latency limit

	App Name	% of Deployed TPUs in July 2016	Total Layers	TPU Ops / Weight Byte	TPU Batch Size
FC	**MLP0**	61%	5	200	**200**
FC	**MLP1**		4	168	168
FC	**LSTM0**	29%	24	64	64
FC	**LSTM1**		37	96	96
Conv	**CNN0**	5%	16	2,888	8
Conv	**CNN1**		89	1,750	32

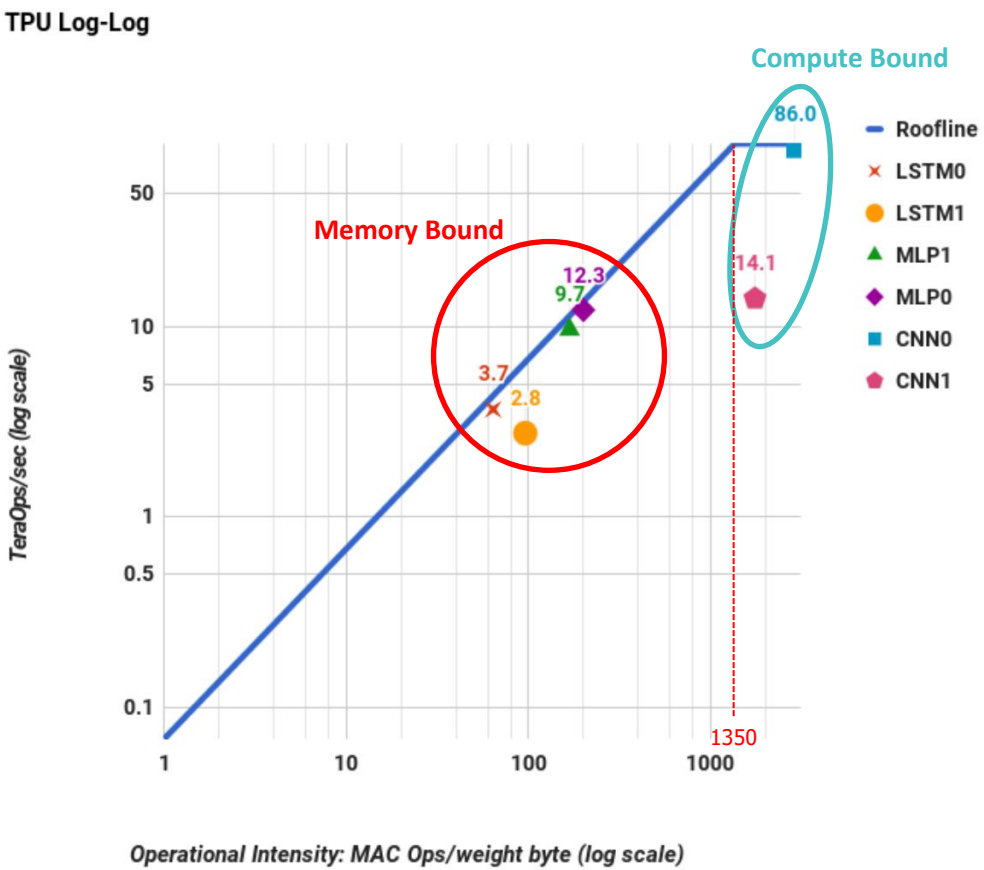
Type	*Batch*	*99th% Response*	*Inf/s (IPS)*	*% Max IPS*
CPU	16	7.2 ms	5,482	42%
CPU	64	21.3 ms	13,194	100%
GPU	16	6.7 ms	13,461	37%
GPU	64	8.3 ms	36,465	100%
TPU	200	7.0 ms	225,000	80%
TPU	250	10.0 ms	280,000	100%

Table 4: MLP0 Performance: Throughput vs. Latency Constraints

Evaluation

Evaluation

TPU Performance analysis: The roofline model



Region	Memory Bound	Compute Bound
Criteria (Intensity)	**Intensity < 1,350 Ops/Byte**	**Intensity > 1,350 Ops/Byte**
Models	LSTM0, LSTM1, MLP0, MLP1	CNN0, CNN1

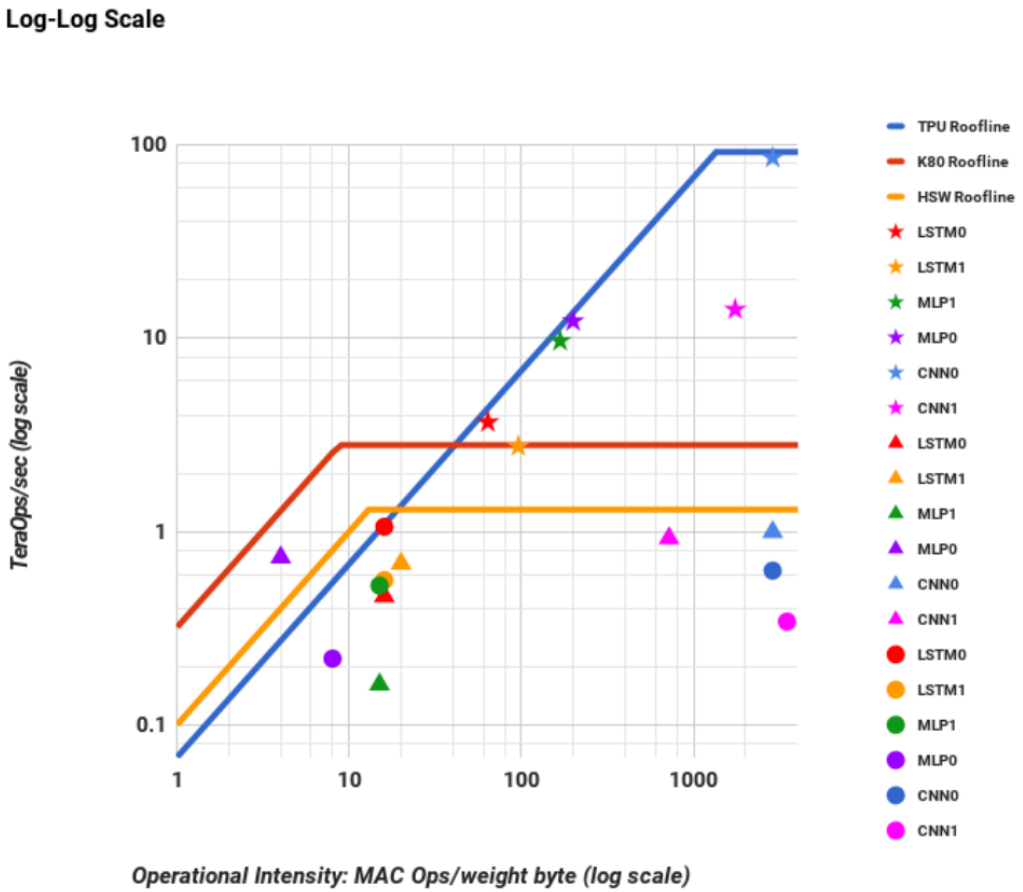
Metric	**MLP0**	**LSTM1**	**CNN0** (Inception)	**CNN1** (AlphaGO)
TeraOps/ sec	12.3	2.8	86.0	14.1
Useful MACs	12.5%	6.3%	78.2%	**22.5%**
Weight Stall	**53.9%**	**62.1%**	0.0%	28.1%

Performance Gap Analysis (CNN0 vs CNN1)

- **Vertical Depth: CNN1 (89 layers) ≫ CNN0 (16 layers)**
- **Horizontal Depth: CNN1 is "Shallow" → Low MMU Utilization**

Evaluation

Roofline comparison (TPU vs. Others)

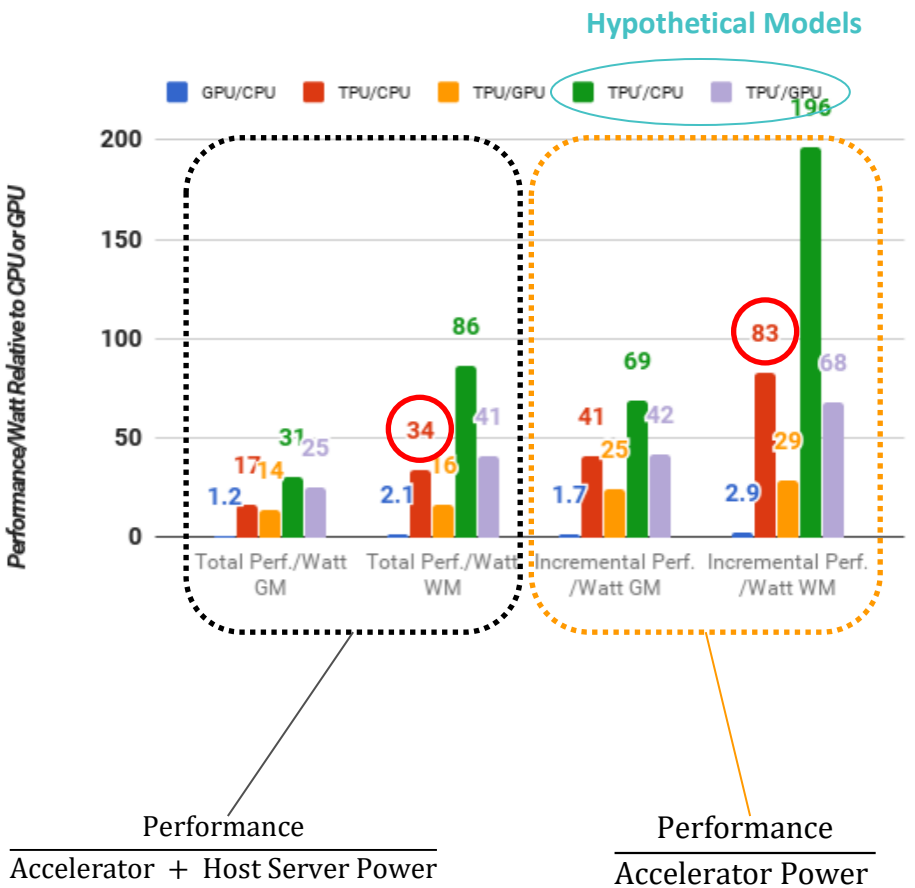


Deployed TPUs	61%		29%		5%			
Type	*DNN*		*LSTM*		*CNN*		*GM*	*WM*
	0	*1*	*0*	*1*	*0*	*1*		
GPU	2.5	0.3	0.4	1.2	1.6	2.7	1.1	1.9
TPU	41.0	18.5	3.5	1.2	40.3	71.0	14.5	29.2
Ratio	16.7	60.0	8.0	1.0	25.4	26.3	13.2	15.3

Feature	**Geometric Mean (GM)**	**Weighted Mean (WM)**
Definition	Equal weight (1/6) per app.	Weighted by actual usage (%).
Key Driver	Rare apps (CNN) count equally.	Top apps (MLP0 61%) dominate.
TPU Speedup (vs. CPU)	**14.5x**.	**29.2x**.
Conclusion	Undervalues performance.	Shows **real-world impact**.

Evaluation

Relative TDP of each server (CPU vs. GPU vs. TPU)



Feature	**Total Perf/Watt**	**Incremental Perf/Watt**
Formula	Accelerator + **Host Server Power**	Accelerator Power **Only**
Focus	Real datacenter electric bill (TCO).	True efficiency of the TPU chip.
TPU Impact (WM) (vs. Haswell CPU)	**~30x** Better (Diluted by host power).	**~83x** Better (Justifies Custom ASIC).

The Bottleneck

- Current TPU uses standard DDR3 memory
- Result: Key workloads (MLP, LSTM) are Memory Bound

The Experiment

- Model a revised design (**TPU'**) using high-bandwidth **GDDR5** (similar to K80 GPU)

The Result

- **Performance:** Triples the achieved TOPS on average
- **Efficiency: ~196x Better** (vs. Haswell CPU)

Evaluation

Energy proportionality of workload utilization

The Observation

- **CPU (Haswell):** Power drops significantly when idle (56% power at 10% load).
- **TPU:** Power stays high even when idle (**88% power** at 10% load)

The Reason

- Short development schedule (**15 months**) prevented including complex energy-saving features.

The Implication

- **TPU – "Poor Energy Proportionality"**
- To maximize ROI, the TPU workload queues must be kept full (100% utilization).

Evaluation

Performance sensitivity & Prospectivity of TPU

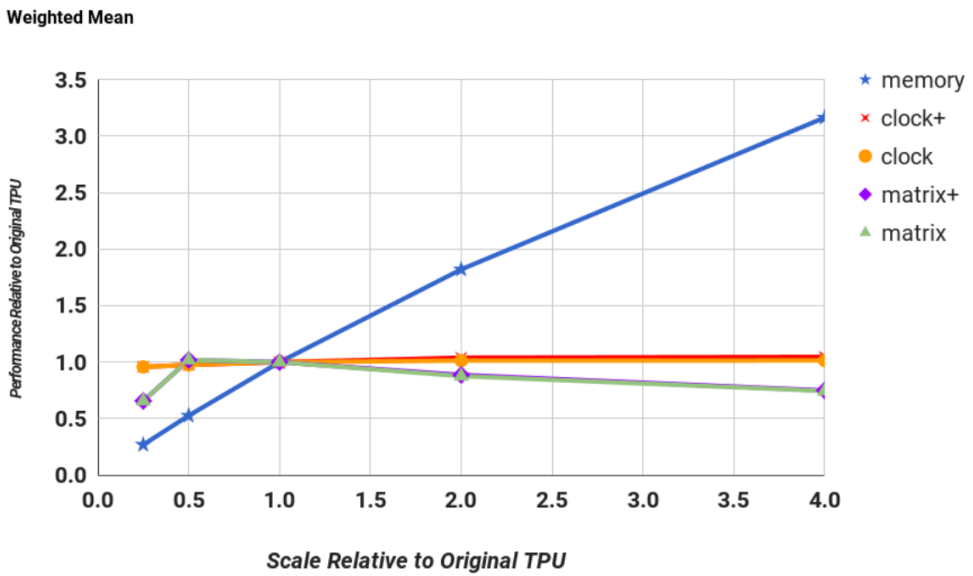
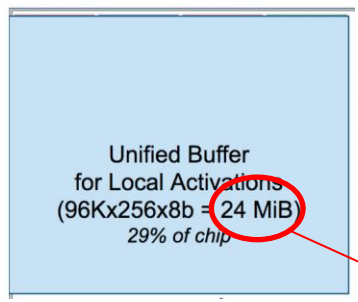


Figure 11. Weighted mean TPU performance as metrics scale

"Simulation Error rate: < 8%"



MLP0	*MLP1*	*LSTM0*	*LSTM1*	*CNN0*	*CNN1*
11.0	2.3	4.8	4.5	1.5	13.9

Table 8. Maximum MiB of the 24 MiB Unified Buffer used per NN application. A 14 MiB Unified Buffer is sufficient for them now, due to improvements in the new software allocator.

Underutilized

Reduce Buffer size to 14 MiB.

Free up **10% silicon area.**

Use space for **GDDR5 Interface.**

Achieve **Triple Performance.**

Conclusion

Conclusion

Summary and Key Takeaways

Category	Key Factors	Metrics / Evidence
1. Quantitative Impact	**Speedup**	• vs CPU/GPU: **Avg 15x ~ 30x** Faster
	Efficiency	• TOPS/Watt: **30x ~ 80x** Higher
	Cost / TCO	• **Order-of-Magnitude** Better Cost-Performance
2. Success Factors	**Minimalism**	• No Caches, No Branch Predict, No Out-of-Order
	Massive Compute	• **65,536 (64K)** 8-bit MAC Matrix Unit
	On-Chip Memory	• **28 MiB** Software-Managed Memory
	Determinism	• Meets **99th-percentile Latency** Target
3. Key Point	**Memory Bound**	• 4 of 6 NN Workloads (MLP, LSTM) are **Memory-Bound**
	TPU' Potential	• **3x Performance Boost** with GDDR5 Memory