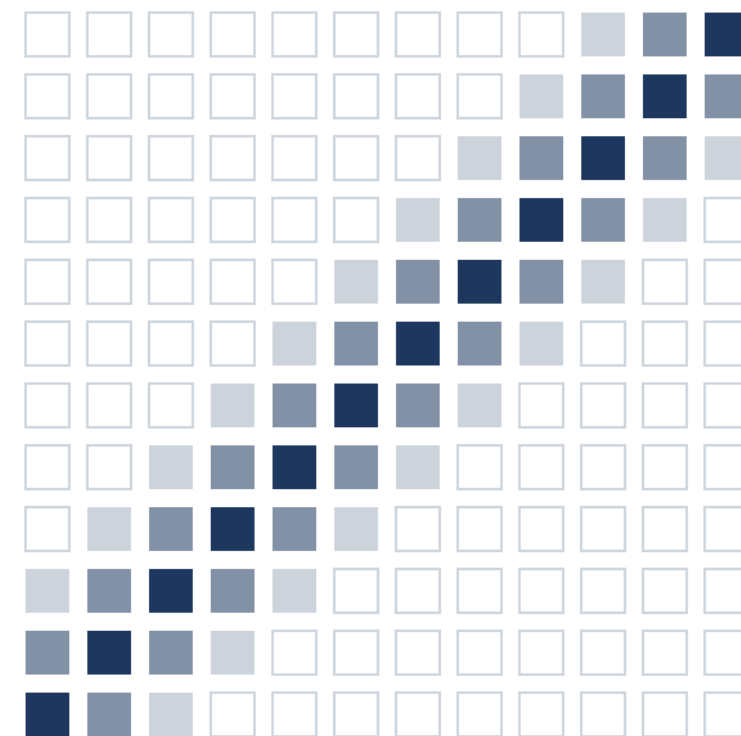


把 datacenter 的 inference 做进一颗芯片

In-Datcenter Performance Analysis of a Tensor Processing Unit

Norman P. Jouppi · Cliff Young · Nishant Patil · David Patterson 等 ·
Google Inc.

ISCA '17 · DOI 10.1145/3079856.3080246 · 自 2015 年起部署于生产
datacenter



256×256 systolic MAC 阵列 · 对角
wavefront

92 TOPS

8-bit 峰值吞吐

65,536

MACs · systolic

28 MiB

on-chip 内存

15–30×

快于同期 CPU·GPU

30–80×

perf/Watt 高于同期

一颗 15 个月落地的定制 ASIC，用生产负载实测

这篇论文做了什么

TPU 是 Google 2015 年部署的 8-bit systolic 矩阵推断 coprocessor；本文把它与同期 Haswell CPU、K80 GPU 放在同一 datacenter，用代表 95% inference 需求的 6 个生产应用，完整测量性能、功耗与设计取舍。

核心结论预览

- 比同期 GPU / CPU 平均快 **15x–30x**，TOPS/Watt 高 **30x–80x**
- 6 个应用中 4 个 memory bound；换 GDDR5 还能再 3x
- 25x 的 MACs、3.5x 的 on-chip 内存，die 却更小、功耗不到一半
- 确定性执行更贴合 99th-percentile 响应时间

解读路线

- 01 背景与挑战
- 02 解决方案：domain-specific ASIC
- 03 具体架构（四页）
- 04 实验验证：workload 与平台
- 05 实验结果（六页）
- 06 讨论与结论

来源：论文 Abstract 与 Section 1 的 highlights · 95% 指 2016-07 部署 TPU 的 inference 占比

NN 复兴走进 datacenter: 跑的是 inference, 不是 training

为什么是「deep」

云端大数据集 + GPU 算力, 让「更多更大的层」成为可能: 语音识别 word error rate -30% (20 年最大涨幅)、图像识别错误率 26% → 3.5%、Go 击败人类冠军。

层数与规模换精度, 也换来了巨大的推理算力需求——这是本文一切的前提。

两个阶段, 两种硬件诉求

Training = 开发期: 浮点、GPU 的主场。**Inference** = 生产期: quantization 到 8-bit 整数通常就够。

8-bit 整数乘比 FP16 乘省 6× 能量、6× 面积; 整数加省 13× 能量、38× 面积。

三类主流 NN

- **MLP**: fully connected, weights 跨 batch 复用
- **CNN**: 空间局部卷积, weights 跨坐标复用
- **LSTM (RNN)**: 状态跨 time step 传递, weights 跨时间复用

每个模型 5M–100M weights, 访问代价高昂 → 用 batch 摊销。

Datacenter inference 的三重约束

① 要的是 response time, 不是 throughput

user-facing 服务有硬的 99th-percentile 响应时间 deadline (如 MLP0 上限 7 ms) ; queueing theory 说长队列能推高吞吐、却拉长尾延迟——于是输入队列通常保持空。

② 成本与功耗墙

2013 年预测: 若每人每天语音搜索 3 分钟, datacenter 算力需求翻倍; 靠「免费」excess capacity 消化的老路走不通了。

③ 通用硬件的错配

CPU/GPU 的 time-varying 优化 (caches、out-of-order、multithreading...) 利于平均吞吐、不利于保证延迟; 实测: K80 做 inference 只比 Haswell 快一点。

来源: 论文 Section 1、2、4 与 Table 4 · MLP0 的 7 ms 上限来自应用开发方

答案：一颗「做得更少」的 domain-specific ASIC

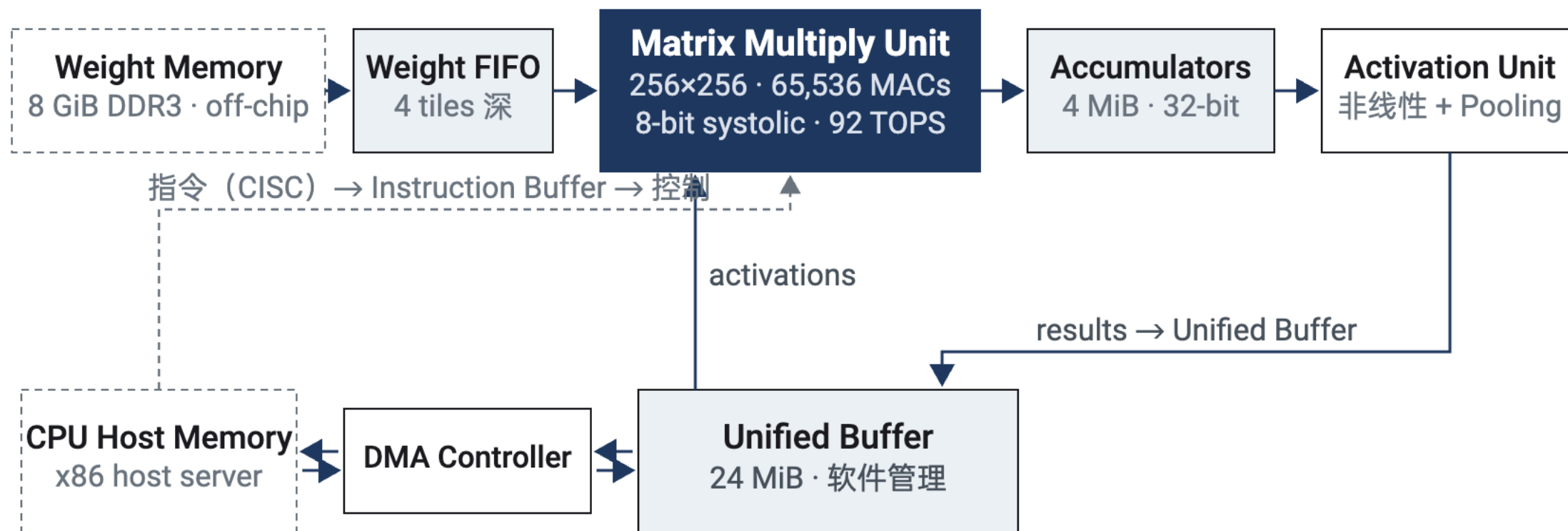
- **PCIe 上的 coprocessor**，不与 CPU 紧集成：像 GPU 一样插进现有 server，不拖慢部署；指令由 host 下发，精神上更接近 FPU 而非 GPU
- **整个 inference model 在片上跑完**，把与 host CPU 的交互压到最少
- **8-bit 整数量化**：乘法省 6× 能量与 6× 面积，把省下的硅面积堆成 MACs
- **激进 minimalism**：no caches、no branch prediction、no out-of-order、no multithreading——「minimalism is a virtue of domain-specific processors」
- **灵活性边界**：要匹配 2015 年及以后的 MLP / CNN / LSTM，而不只是 2013 年的模型

从立项到部署

2006	开始讨论 GPU / FPGA / ASIC； 结论：excess capacity 几乎免费，难上加难
2013	语音搜索预测改变决策：高优先级立项定制 ASIC（训练买现成 GPU）
15 个月	设计、验证、制造、部署进 datacenter 一次完成
目标	cost-performance 比 GPU 提升 10×

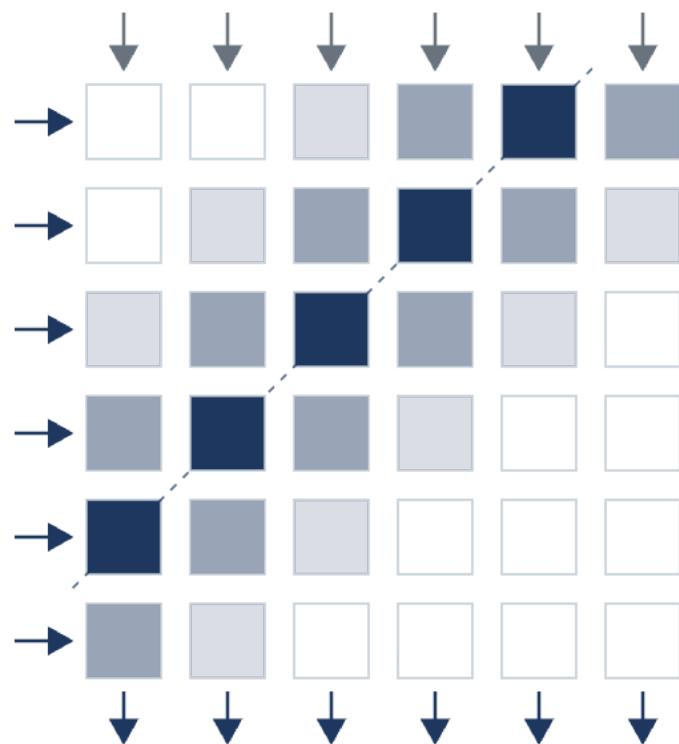
来源：论文 Section 2 · 引文为论文原文

架构总览：整颗 TPU 是围着矩阵单元的数据流



重绘自论文 Figure 1 · 指令由 host 经 PCIe 下发、经 Instruction Buffer 控制各单元 · 内部数据通路 256-byte 宽

Systolic 矩阵单元：用布线换能耗



数据流（重绘自论文 Figure 4）

- **Weights** 自顶部预加载，随 wavefront 推进逐拍生效
- **Activations** 自左侧流入，每 cycle 读写 256 个值
- 乘加沿**对角 wavefront** 推进，每 cycle 自底部输出一个 256-element partial sum → Accumulators

为什么 systolic

读大 SRAM 比算术耗电得多——systolic 让数据在阵列内接力，减少对 Unified Buffer 的读写，从源头省能耗。

正确性上软件对 systolic 无感，但性能上要在意它的 latency；8/16-bit 混合输入半速、16/16 四分之一速；为赶部署砍掉 sparse 支持，只做 dense 矩阵。

重绘自论文 Figure 4 · 示意 6×6 阵列，实际 256×256 = 65,536 MACs · 92 TOPS @ 700 MHz

Die 把 2/3 给了 datapath, control 只有 2%

Die 构成 (论文 Figure 2)



datapath 67% · I/O 10% · control 2% · 其余 21%

Unified Buffer 24 MiB 占 die 近 1/3、矩阵单元 1/4；
control 仅 2%——在 CPU/GPU 里它大得多、也难设计得多。

TPU <331 mm² · 28 nm · 700 MHz；对照 Haswell 662 mm² · 22 nm，K80 561 mm² · 28 nm。

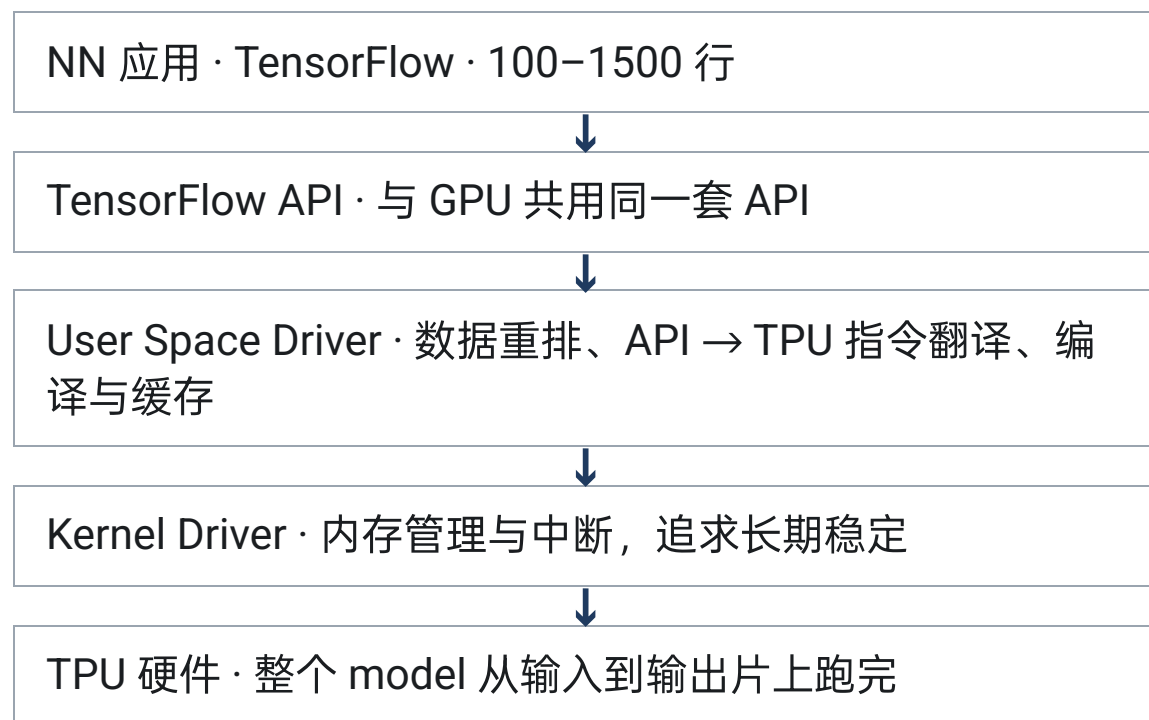
极简 CISC ISA：约一打指令，5 条核心

- **Read_Host_Memory**: host → Unified Buffer
- **Read_Weights**: Weight Memory → Weight FIFO (decoupled access/execute)
- **MatrixMultiply/Convolve**: 12-byte 指令；B×256 输入 × 256×256 权重 → B×256 输出，B 个 pipelined cycles
- **Activate**: 非线性 + pooling；Accumulators → Unified Buffer
- **Write_Host_Memory**: Unified Buffer → host

4-stage pipeline：每条指令独占一级，其余指令藏在 MatrixMultiply 背后执行；PCIe 总线慢 → CISC + repeat 字段，平均 CPI 10–20。

来源：论文 Section 2 与 Figure 2 · die 比例 67% / 10% / 2% 为论文原文

软件栈：编译一次，之后全速运行



关键行为

- 首次评估编译 model、缓存 program image，并把 weight image 写入 Weight Memory；**第二次起全速运行**
- User Space driver 常变、Kernel Driver 求稳——与 GPU 同样的分层哲学
- host 与 TPU 的交互只占 TPU 执行时间的 **5.4%–11.2%** (Table 5)
- overlapped execution 把多数非关键路径操作藏在矩阵单元背后

来源：论文 Section 2 (software stack) 与 Table 5 · 栈分层对照 GPU 的 user/kernel driver 设计

Workload: 6 个生产应用，覆盖 95% inference 需求

App	类型（占比）	LOC	Layers	Weights	Ops/Byte	Batch
MLP0	MLP · 61%	100	5 FC	20M	200	200
MLP1	MLP · 61%	1000	4 FC	5M	168	168
LSTM0	LSTM · 29%	1000	24 FC + 34 vector	52M	64	64
LSTM1	LSTM · 29%	1500	37 FC + 19 vector	34M	96	96
CNN0	CNN · 5%	1000	16 conv	8M	2888	8
CNN1	CNN · 5%	1000	4 FC + 72 conv + 13 pool	100M	1750	32

它们是更大的 user-facing 应用中的一段（host 侧是数千到数百万行 C++）；其中包括 **RankBrain**、GNMT 子集、**Inception**、DeepMind **AlphaGo**。

Ops / Weight Byte = operational intensity，即 roofline 的 x 轴；batch 为延迟约束下实际采用的值。

来源：论文 Table 1 · 占比为 2016 年 7 月已部署 TPU 的 inference 份额

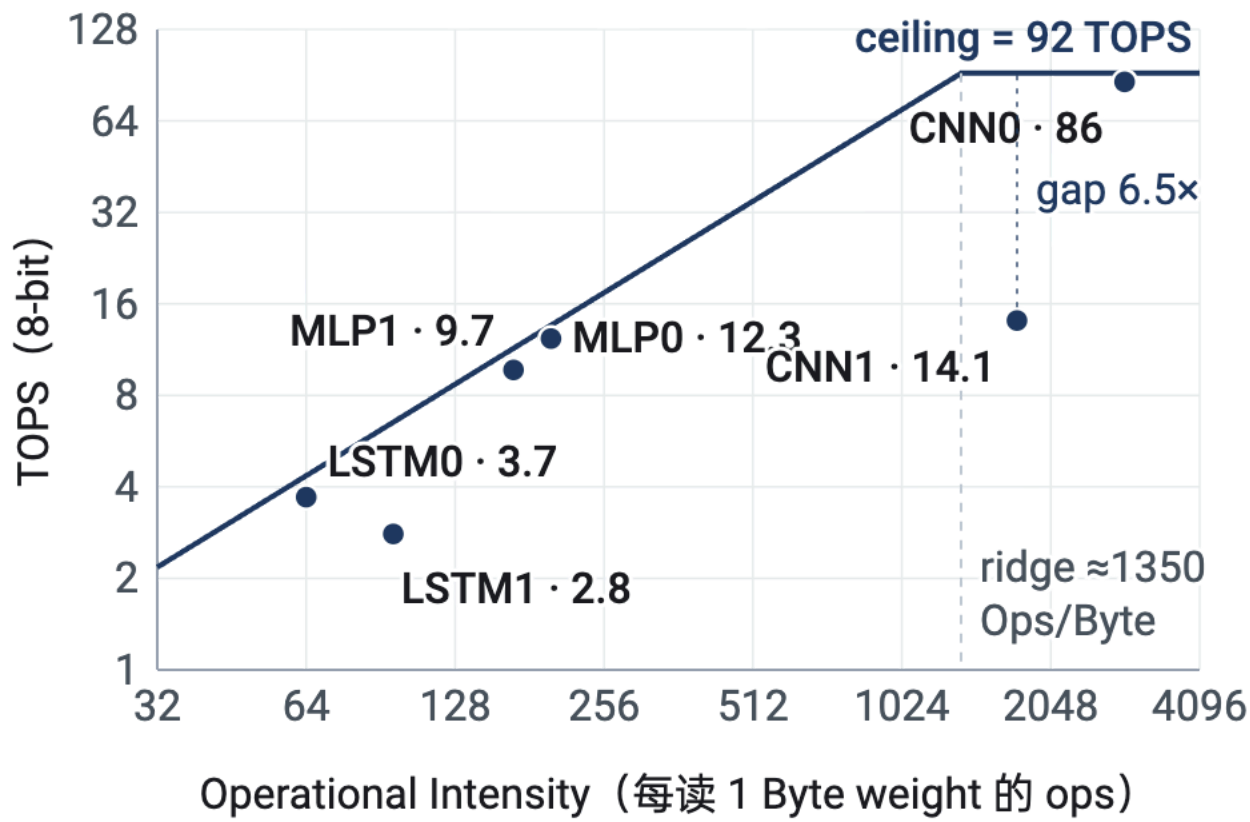
三个同期平台，都是 server-class 的合理配置

指标	Haswell CPU	NVIDIA K80 GPU	TPU
工艺	22 nm	28 nm	28 nm
Die 面积	662 mm²	561 mm²	<331 mm²
时钟	2300 MHz	560 MHz	700 MHz
Die TDP / 实测 busy	145 W / 145 W	150 W / 98 W	75 W / 40 W
峰值 TOPS (8-bit)	1.3	2.8	92
内存带宽	51 GB/s	160 GB/s	34 GB/s
On-chip 内存	51 MiB	8 MiB	28 MiB
每 server dies 数	2	8	4
Server TDP (idle → busy)	504 W (159 → 455)	1838 W (357 → 991)	861 W (290 → 384)

K80 不开 Boost、开 SECCDED：带宽 240 → 160 GB/s、峰值 8.7 → 2.8 TOPS；除注明外结果按 per-die 归一，server 级比较另标。

来源：论文 Table 2 · 三者均为 2015 年部署于同一 datacenter 的同期硬件

Roofline: 5/6 应用顶到 ceiling，只有 CNN1 掉队



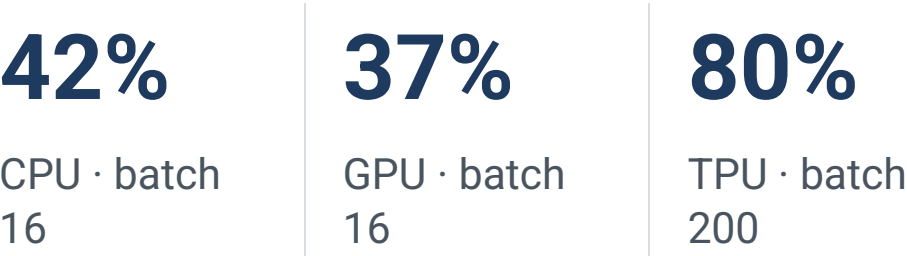
CNN1 解剖 · performance counters

cycle 去向	CNN1	6-app 均值
matrix active cycles	46.2%	28%
其中 useful MACs 占比	22.5%	23%
weight 装载等待	28.1%	43%
RAW hazard stalls	22.8%	11%
实测 TOPS (peak 92)	14.1	21.4

CNN1 的 conv 层 feature depth 浅 → 65,536 个 MACs 近半空转；4 个 fully connected 层 operational intensity 仅 32，weight 供给跟不上。

数据：论文 Figure 5（重绘）与 Table 3 · ceiling 92 TOPS · ridge point ≈ 1350 Ops/Byte

响应时间：确定性胜过峰值性能



MLP0 在 7 ms 99th-percentile 上限内，三者各自能达到最大吞吐的比例。

延迟上限让 CPU/GPU 慢 **2.3×–2.7×**，TPU 只慢 **1.2×**。

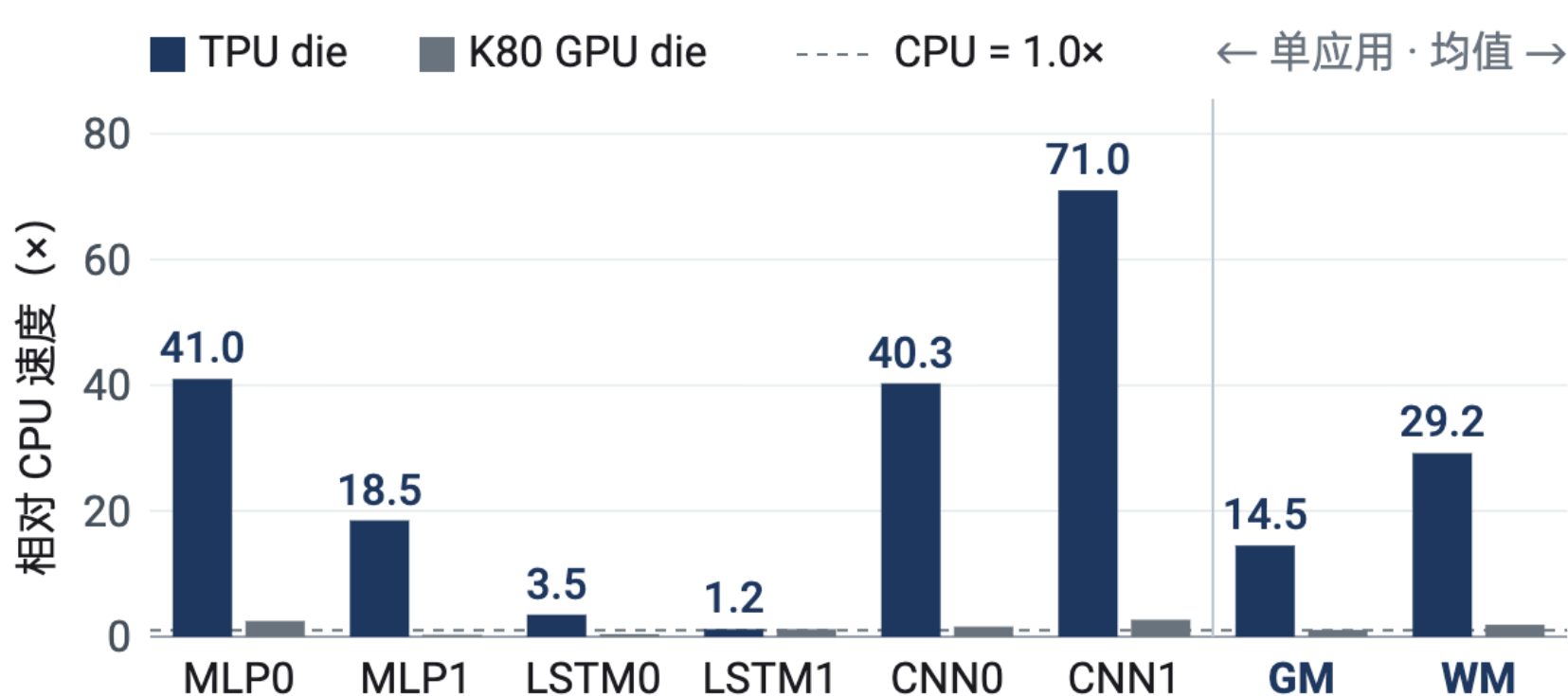
TPU 没有 caches、branch prediction、out-of-order、multithreading——单线程确定性执行让尾延迟可预测。

平台	Batch	99th 响应	IPS / die	% max
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%

batch 越大吞吐越高、但响应超限——CPU/GPU 被迫用小 batch；TPU 的 max IPS 受 host server 开销限制。

来源：论文 Table 4 · MLP0 · 7 ms 上限含 host server 时间

Per-die 加速比: TPU 14.5× (GM) / 29.2× (WM) 于 CPU



GM = geometric mean · WM = weighted mean (按生产负载加权)

来源: 论文 Table 6 (per die、含 host 开销) · 同一数据的 roofline 叠加见论文 Figure 8

GPU ≈ CPU

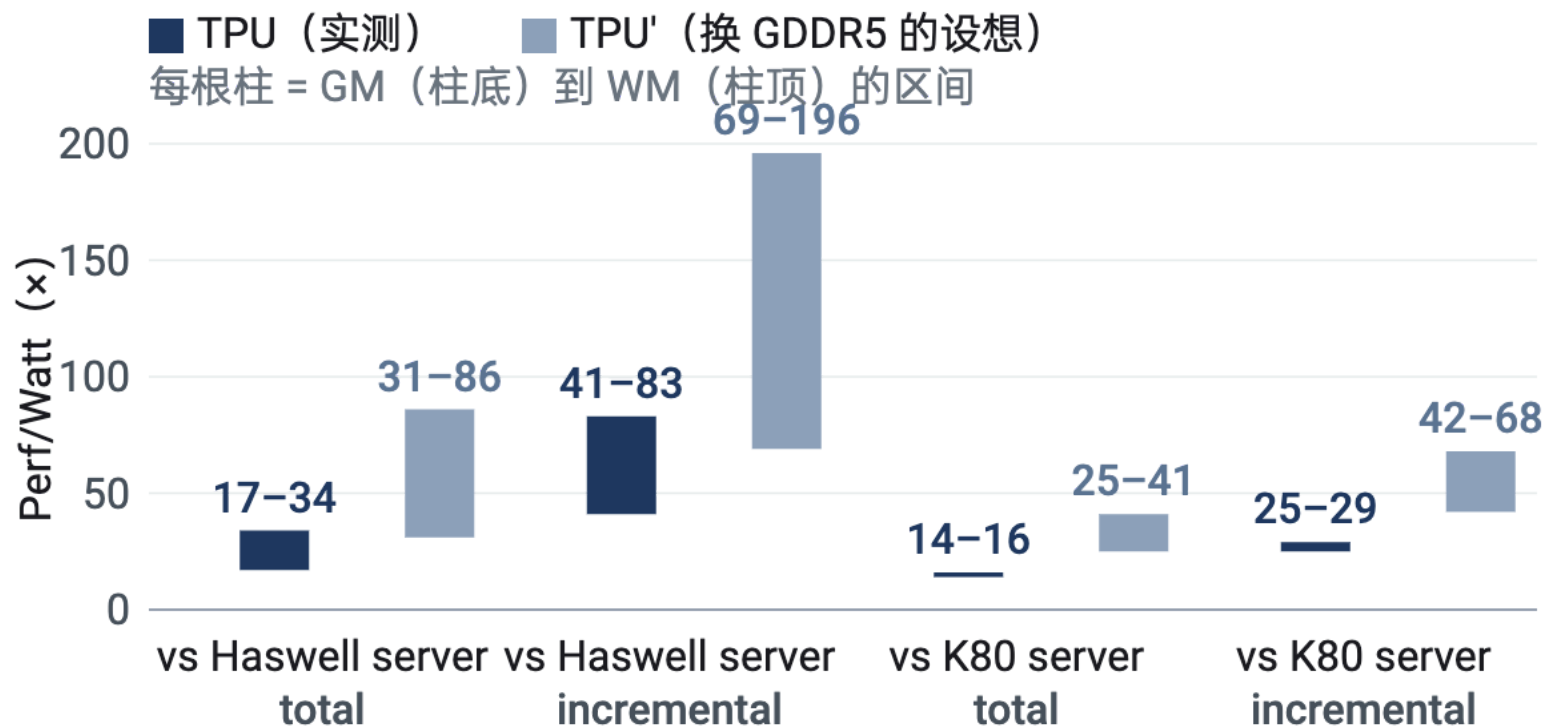
K80 die 仅 Haswell die 的 1.1× / 1.9×——高峰值与高带宽被延迟约束吃掉。

TPU 13.2× / 15.3×

于 GPU die (GM / WM) ; 单应用 3.5×–71×。

数字含 host server 开销; WM 按生产 mix (MLP 61%) 加权。

Perf/Watt: 41×–83×——立项理由的那个数字



incremental 41–83×

vs Haswell server、只计加速卡增量功耗——公司内部批准定制 ASIC 的依据。

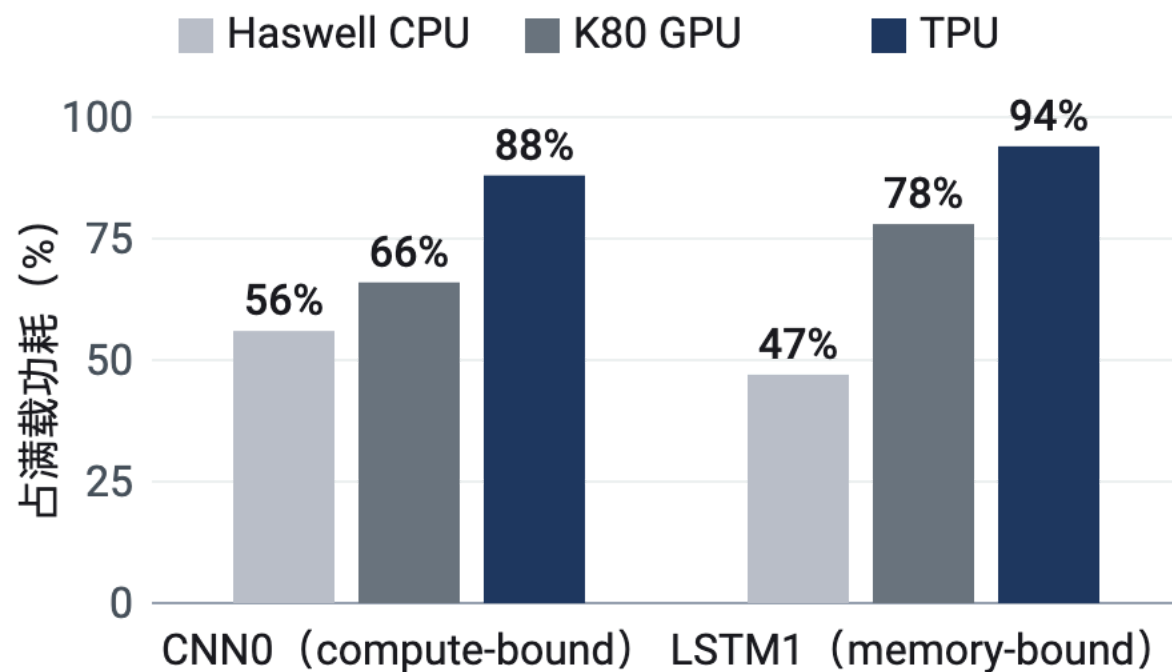
total (含 host 功耗) 17–34×;
TPU server 为 K80 server 的 14–16× (total)、25–29× (incremental)。

K80 server 自身仅 1.2–2.9× 于 Haswell server。

total 含 host 功耗 · incremental 只算加速卡增量 (立项依据)

来源：论文 Figure 9 与 Section 5 · 功耗与 TCO 相关，故以 perf/Watt 作 performance/TCO 的 proxy

能量比例性：功耗最低，也是最诚实的短板



10% 负载 · Watts/die · 越低越接近 energy proportional

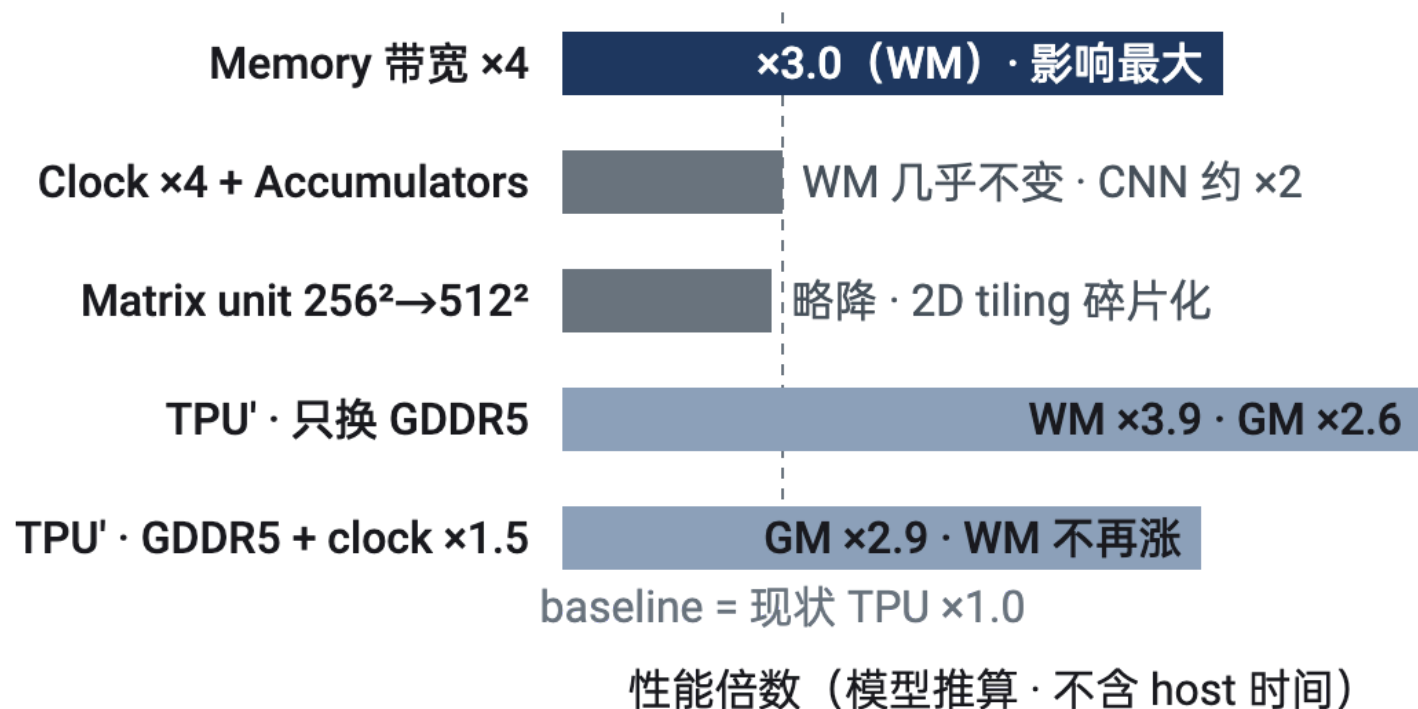
TPU 只有 40 W/die，但 10% 负载时仍烧满载的 **88%** 功耗——15 个月的工期没留给节能特性；Haswell 全班最好（56%），K80 为 66%。

LSTM1 上更明显：10% 负载时 CPU 47% / GPU 78% / TPU 94% 的满载功耗。

但 server 级账算得回来：Haswell + 4×TPU 只增加 **<20%** 功耗，CNN0 却比纯 Haswell server 快 **80×**（4 TPU dies vs 2 CPU dies）。

来源：论文 Section 6 与 Figure 10 · energy proportional computing 引自 Barroso & Hölzle 2007

如果多给 15 个月：内存带宽是唯一主杠杆



- UB 24 MiB 原为 MLP batch 2048 而设；新 allocator 实测 **14 MiB 就够** (Table 8 最大 13.9 MiB)
- TPU' 的 GDDR5 需 4 通道、die +10%；UB 缩到 14 MiB 恰好补回 10%
- server 功耗 861 W $\rightarrow$ 约 900 W

来源：论文 Section 7、Figure 11 与 Table 8 · 条形按原文文字结论绘制，非原图曲线

四个值得记住的 fallacy 与 pitfall

Pitfall · 学界只盯着 CNN

ISCA '16 有 15% 的论文做 NN 加速器、9 篇全面向 CNN；但 CNN 只占生产 inference 负载约 5%——MLP 与 LSTM 值得同样的热情。

Fallacy · Boost mode 会翻盘

在 LSTM1 上实测：clock 1.6× (560 → 875 MHz)、性能 1.4×、功耗 1.3× → perf/W 净收益仅 1.1×。

来源：论文 Section 8 · 引文为论文原文

Pitfall · IPS 不是好的总结指标

同一颗 TPU：MLP1 跑 360,000 IPS、CNN1 只有 4,700 IPS——差 75×；IPS 更多反映模型复杂度，而非硬件好坏。

Pitfall · 无视架构史

systolic arrays、decoupled access/execute、CISC 都是 1980 年代初的思想，在 domain-specific 硬件上复活；「history-aware architects could have a competitive edge」。

数量级差异罕见——TPU 成为 DSA 的 archetype

它为什么成功（论文自陈）

- 矩阵单元大而不太大（256×256）
- 足够大的软件管理 on-chip 内存（28 MiB）
- whole-model 片上执行，减少对 host 的依赖
- 单线程确定性执行，贴合 99th-percentile
- 省略通用特性，换小且低功耗的 die
- 8-bit 量化 + TensorFlow 易移植

Cornucopia Corollary

对 Amdahl's Law 的推论：**巨大而廉价的资源即使利用率低，仍可交付高、且划算的性能。**

6 个应用里 4 个 memory bound、利用率不高；
但 92 TOPS 的基数足够大，「一小份」仍是同期平台
的数量级倍数。

15×

快于 K80 die

29×

perf/W vs K80

29×

快于 Haswell die

83×

perf/W vs Haswell

来源：论文 Section 10 · 四个数字为 weighted mean（per die · perf/W 为 incremental）