



中山大學
SUN YAT-SEN UNIVERSITY



国家超级计算广州中心
NATIONAL SUPERCOMPUTER CENTER IN GUANGZHOU

Advanced Computer Architecture

高级计算机体系结构

第14讲：Review

张献伟

xianweiz.github.io

DCS5367, 1/4/2022

期末考试

- 时间地点

- 2022.1.12(周三)、14:30 - 16:30
 - 注：4个班统一时间，不同试题
- B201

- 试题类型

- 选择题(30'): 3' x 10
 - 简答题(20'): 4' x 5
 - 应用题(30'): 15' x 2
 - 综合题(20'): 20' x 1
- Diagram illustrating the total score distribution for the exam types:
- 选择题(30') and 简答题(20') are grouped together with a bracket labeled 50'.
 - 应用题(30') and 综合题(20') are grouped together with a bracket labeled 50'.
 - The two groups of 50' each are further grouped together with a large bracket labeled 100'.

成绩计算

- 课堂参与（**15%**）
 - 出席：6%
 - Review提问点名缺席：-2%
 - Quiz：3% x 3
 - 提交但全错：-3%（即，当次quiz分为0）
 - 未提交：-4%（即，当次quiz分为-1）
 - 代提交：-5%（即，当次quiz分为-2）
- 作业（**35%**）
 - hw1: 10%（理论）
 - hw2: 10%（论文）
 - hw3: 15%（分析）
- 期末考试（**50%**）
 - 闭卷（中文，关键术语英文标注）

课堂参与

- 分值计算（15%）

- 出席：6%

- Review提问点名缺席：-2%

- Quiz：3% x 3

- 提交但全错：当次quiz分为0

- 未提交：当次quiz分为-1

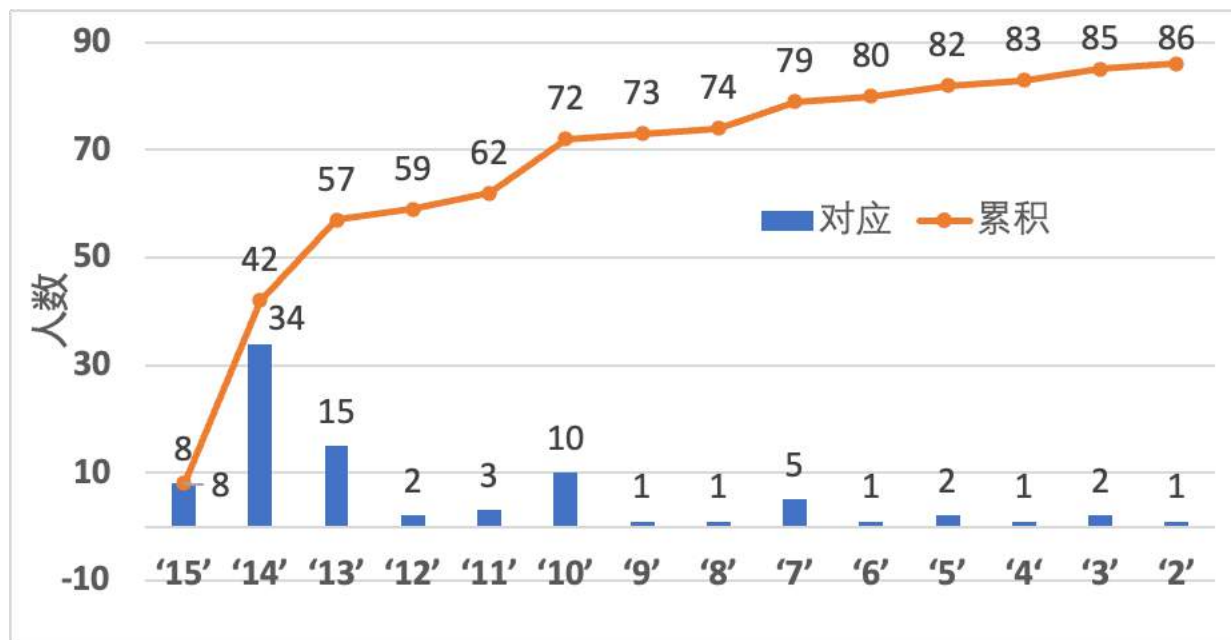
- 代提交：当次quiz分为-2

- 分值分布

- 最低：2

- 最高：15

- 平均：11.65（78%）



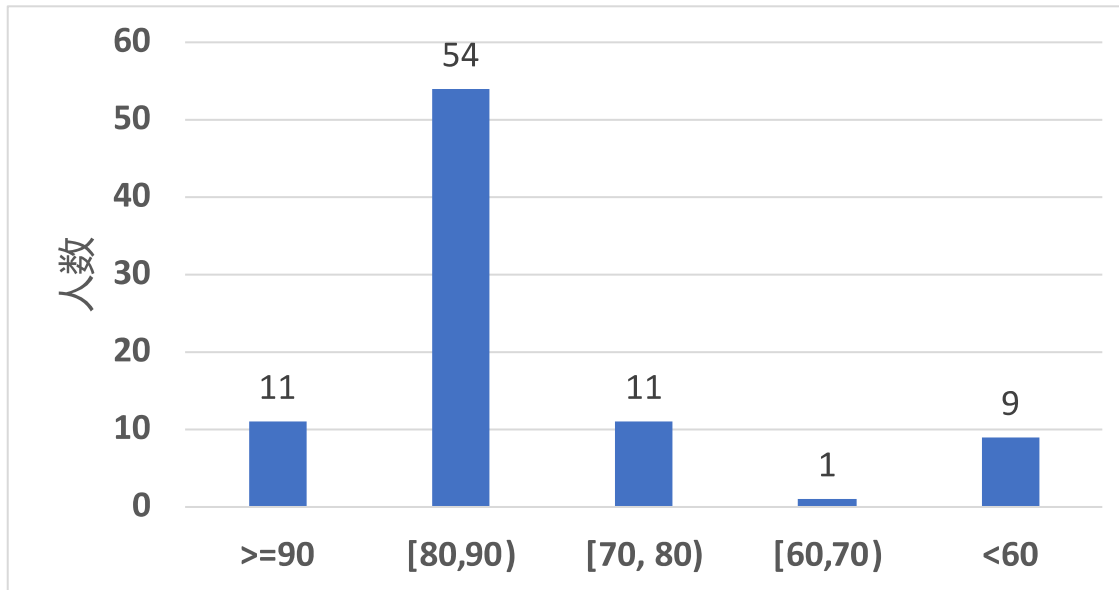
作业

- 作业 (35%)
























- hw1: 10% (理论)
- hw2: 10% (论文)
- hw3: 15% (分析)

- hw1

- 最低: 2 (0)
- 最高: 9.8
- 平均: 7.64



Covered Topics

Week/Date	Topic	Note
wk2: Sep 7	 Overview	
wk3: Sep 14	 ISA and ILP (1)	
wk4: Sep 21	NO CLASS	Holiday
wk5: Sep 28	 ISA and ILP (2)	
wk6: Oct 5	NO CLASS	Holiday
wk7: Oct 12	 ISA and ILP (3)	
wk8: Oct 19	 Memory System (1)	 HW-1 [ sol]
wk9: Oct 26	 Memory System (2)	
wk10: Nov 2	NO CLASS	 HW-2, review-form.txt
wk11: Nov 9	 DLP and GPU (1)	
wk12: Nov 16	 DLP and GPU (2)	
wk13: Nov 23	DLP and GPU (3) -  vgpu,  prof,  cfd	Invited Talks ...
wk14: Nov 30	 TLP (1)	
wk15: Dec 7	 TLP (2)	 HW-3, data_csv.zip
wk16: Dec 14	 TLP (3)	
wk17: Dec 21	 WSC & Interconnect	
wk18: Dec 28	 Domain-specific Architecture	
wk19: Jan 4	Review	
wk20: Jan 12	FINAL EXAM (2022/01/12, Wed, 14:30-16:30, B201)	Closed Book

#0: Quantitative Approaches

- Design goals:
 - Functional, high-performance, reliable
 - Low cost, low power/energy, ...
- Performance evaluation and quantitative principles
 - IPC, CPI, time
 - Parallelism, locality, common-case
- Computing CPU time and improve CPI
 - Instructions, cycles, frequency
- Moore's Law, Amdahl's Law
- Benchmarks and simulation

#1: ISA and ILP

- ISA and instructions
 - RISC vs. CISC (x86 -> ARM -> RISC-V)
 - Pipelining: stages, throughput vs. latency
 - Pipeline hazards and techniques
- Instruction-level parallelism
 - Types of data dependences: true, name (anti-, output)
 - Loop unrolling effects and limits
 - Branch prediction: static vs. dynamic
 - Dynamic scheduling: scoreboard vs. Tomasulo
 - Multiple-issue: superscalar, VLIW

#2: Memory System

- Memory hierarchy, memory wall
- Cache basics
 - block/associativity, addressing, hit/miss, replacement, read/write policy...
 - direct-mapped, set-associative, fully-associative
 - Evaluation metrics: hit ratio, AMAT
- Virtual memory: translation, page table, TLB, page fault ...
- Memory technology
 - SRAM vs. DRAM vs. NVM
 - DRAM variants: DDR, LPDDR, GDDR, HBM, eDRAM, ...
- Advanced cache optimizations
 - #1 -- #10

#3: DLP and GPU

- Flynn's Taxonomy: SISD, SIMD, MISD, MIMD
- SIMD vs. superscalar vs. VLIW
- SIMD vs. SIMT vs. SMT
 - CPU vs. GPU
- GPU architecture
 - Device, SM/CU, core
 - Register, shared memory, cache, memory, ...
 - Register spilling, local memory
- GPU programming
 - Kernel, thread-block, warp, thread
 - Kernel declare and launch
 - Stream and synchronize

#4: Thread-level Parallelism

- Multiprocessor: SMP, DSM
 - UMA vs. NUMA
- Cache coherence
 - Snooping protocol: MSI/MESI, invalidation
 - True sharing vs. false sharing
 - Limits
 - Directory based: home-/requesting-node, P2P messages
 - Storage overhead
- Memory consistency
 - Vs. coherence
 - Types: sequential, TSO, PSO, RC

#5: Warehouse-scale Computing (& ICN)

- Parallelism: instruction, data, thread, request
- Differences with supercomputer and datacenter
- Design goals
 - Some are same with conventional servers
 - Some are unique to WSC
- Architecture
 - Server, rack, array
 - Storage, network
 - Utilization and power usage effectiveness (PUE)
- Interconnect
 - System-area and on-chip
 - Popular topologies

#6: Domain-specific Architecture

- Why DSAs are needed?
- DSA opportunities and challenges
- DSA design guidelines
 - Dedicated memories
 - More arithmetic units
 - Easist parallelism
 - Reduced data size and type
 - Programming and software
- DNN and TPU
 - Roofline performance model

期末考试

- 时间地点

- 2022.1.12(周三)、14:30 - 16:30
 - 注：4个班统一时间，不同试题
- B201

- 试题类型

- 选择题(30'): 3' x 10
 - 简答题(20'): 4' x 5
 - 应用题(30'): 15' x 2
 - 综合题(20'): 20' x 1
- Diagram illustrating the total score distribution for the exam types:
- 选择题(30') and 简答题(20') are grouped together with a bracket labeled 50'.
 - 应用题(30') and 综合题(20') are grouped together with a bracket labeled 50'.
 - The two groups of 50' each are further grouped together with a large bracket labeled 100'.

Example Question

- Q1: 就数据传输带宽而言，下列那种排序符合通常情况？
 - ☒ A. $\text{NVM} < \text{DDR} < \text{HBM}$
 - B. $\text{DDR} < \text{NVM} < \text{HBM}$
 - C. $\text{NVM} < \text{HBM} < \text{DDR}$
 - D. $\text{HBM} < \text{DDR} < \text{NVM}$
- Q2: 关于GPU，单个计算单元（streaming multiprocessor/compute unit）中不包含下列哪种资源？
 - A. FP64 core
 - B. Shared memory
 - C. Register file
 - ☒ D. Local memory

Example Question (cont.)

- Q1: Pipelining理想加速比怎么计算？不能达到理想加速比的原因？
 - n stages
 - stages cannot be perfectly balanced, inherent overhead
- Q2: 内存一致性（memory consistency）SC和TSO的区别与联系。
 - SC: 严格program order, 读必须写可见后进行；
 - TSO: 采用store buffer, 允许读在写后立即开始，相比SC可显著提升性能。
- Q3: 列出领域专用架构（domain specific architecture）的3条设计原则。
 - 专用存储、更多计算、最易并行、低精度数据、领域编程

Example Question (cont.)

- 假设一个GPU配置参数为：时钟频率1.5GHz；包含16个SM，每个又有32个FP64计算核；显存分为8个通道（channel），每个宽度为32b，数据传输频率为10Gb/s。

1) 计算FP64理论峰值性能

2) 计算显存理论带宽

3) 以上显存带宽能否支持计算吞吐？若不能，如何解决？

– 计算： $1.5\text{G} \times 16 \times (32 \times 2) = 1536 \text{ GFLOPS}$

– 带宽： $8 \times 32\text{b} \times 10\text{Gb/s} = 320 \text{ GB/s}$

– 假设计算为： $d = a * b + c$, 3 reads and 1 write $\rightarrow 32\text{B}$, x
 $1536\text{G} = 49.2 \text{ GB/s} \ggg 320\text{GB/s}$; 使用寄存器和片上缓存

Domain Specific Architecture

How TPU Follows the Guidelines

- *Use dedicated memories*
 - 24 MB dedicated buffer, 4 MB accumulator buffers
- *Invest resources in arithmetic units and dedicated memories*
 - 60% of the memory and 250X the arithmetic units of a server-class CPU
- *Use the easiest form of parallelism that matches the domain*
 - Exploits 2D SIMD parallelism
- *Reduce the data size and type needed for the domain*
 - Primarily uses 8-bit integers
- *Use a domain-specific programming language*
 - Uses TensorFlow

TPU Performance[性能]

- Compare using six benchmarks
 - Representing 95% of TPU inference workload in Google data center in 2016
 - Typically written in TensorFlow, pretty short (100-1500 LOCs)
- Chips/servers being compared
 - CPU server: Intel 18-core, dual-socket Haswell; host server for GPUs/TPUs
 - GPU accelerator: Nvidia K80

Inference Datacenter Workload (95%)

Name	LOC	Layers					Nonlinear function	Weights	TPU Ops / Weight Byte	TPU Batch Size	% Deployed
		FC	Conv	Vector	Pool	Total					
MLP0	0.1k	5				5	ReLU	20M	200	200	61%
MLP1	1k	4				4	ReLU	5M	168	168	
LSTM0	1k	24		34		58	sigmoid, tanh	52M	64	64	29%
LSTM1	1.5k	37		19		56	sigmoid, tanh	34M	96	96	
CNN0	1k		16			16	ReLU	8M	2888	8	5%
CNN1	1k	4	72		13	89	ReLU	100M	1750	32	

Roofline Performance Model[屋顶线]

- The roofline model was introduced in 2009
 - Samuel Williams, Andrew Waterman, and David Patterson. 2009. Roofline: an insightful visual performance model for multicore architectures. Commun. ACM
- It provides an easy way to get performance bounds for **compute and memory bandwidth bound** computations
- It relies on the concept of **Computational Intensity (CI)**
 - Sometimes also called Arithmetic or Operational Intensity
- The model provides a relatively simple way for **performance estimates** based on the computational kernel and hardware characteristics
 - Performance [GF/s] = function (hardware and software characteristics)

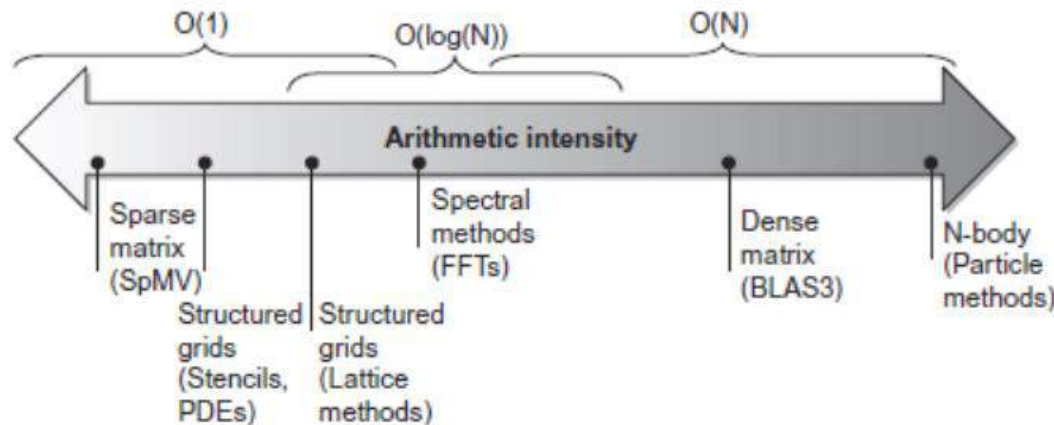
Roofline Performance Model(cont.)

- Basic idea

- Plot peak FP throughput as a function of arithmetic intensity
- Ties together FP performance and memory performance for a target machine

- Arithmetic intensity[运算密度]

- Ratio of FP operations per byte of memory accessed
 - (total #FP operations for a program) / (total data bytes transferred to main memory during program execution)



Arithmetic Intensity

- $A.I. = \frac{W}{Q}$ (FLOP/Byte)

- W: amount of work / i.e floating point operations required
- Q: memory transfer / i.e access from DRAM to lowest level cache

- Examples

```
for (i = 0; i < N; ++i)  
    z[i] = x[i]+y[i]
```



1 ADD
2 (8 byte) loads
1 (8 byte) write
 $AI = 1 / (2*8 + 8) = 1/24$

```
for (i = 0; i < N; ++i)  
    z[i] = x[i]+y[i]*x[i]
```



1 ADD
1 MUL
2 (8 byte) loads
1 (8 byte) write
 $AI = 2 / (2*8 + 8) = 1/12$

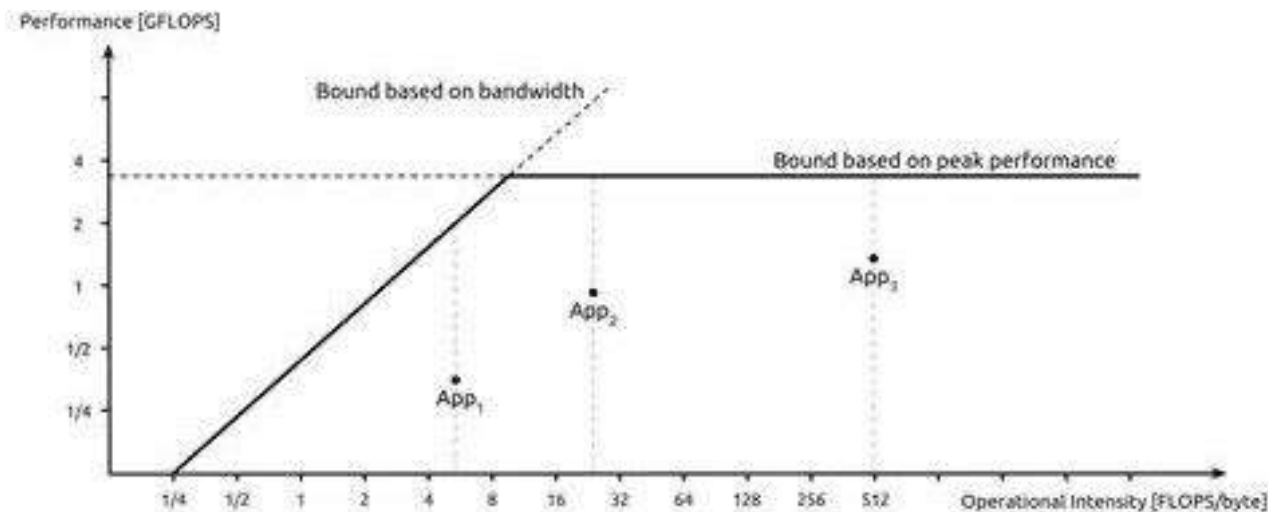
Example

```
float in[N], out[N];  
for (int i=1; i<N-1; i++)  
    out[i] = in[i-1]-2*in[i]+in[i+1];
```

- Amount of FLOPS: $3(N-2)$
 - For every i : $\text{out}[i] = \text{in}[i-1] - 2 \cdot \text{in}[i] + \text{in}[i+1] \rightarrow 3 \text{ flop}$
 - Loop over: $\text{for}(\text{int } i=1; i < N-1; i++) \rightarrow (N-2) \text{ repetitions}$
- Memory accesses Q : depends on cache size
 - No cache (read directly from slow memory) \rightarrow every data accessed is counted
 - $4 \text{ accesses} \times (N-2) \text{ repetitions} \times 4 \text{ bytes} \rightarrow \text{A.I.} = 3/16$
 - Perfect cache (infinite sized cache) \rightarrow data is read & written only once
 - $2 \text{ accesses} \times (N-2) \text{ repetitions} \times 4 \text{ bytes} \rightarrow \text{A.I.} = 3/8$

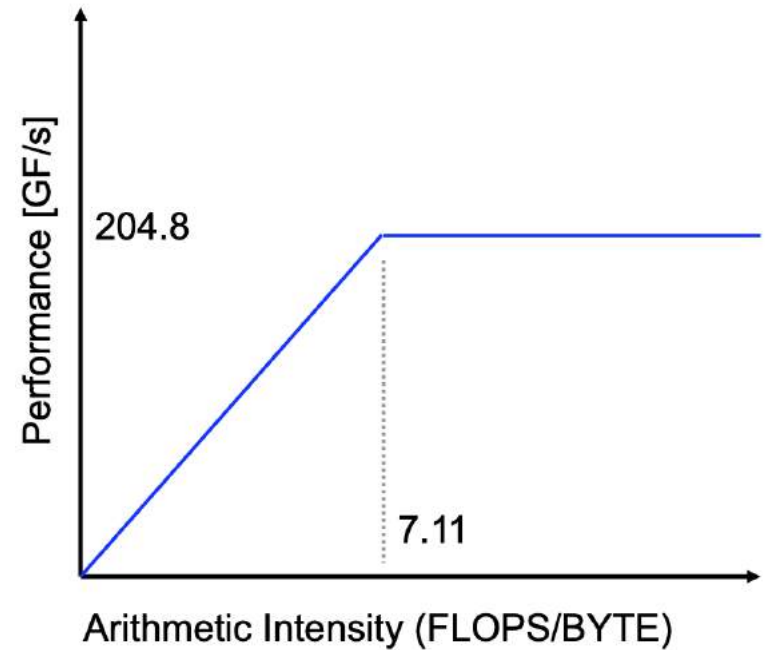
Roofline Analysis

- **“Roofline”** sets an upper bound on perf of a kernel depending on its arithmetic intensity
 - Think of arithmetic intensity as a pole that hits the roof
 - Hits the flat part: perf is computationally limited
 - Hits the slanted part: perf is ultimately limited by memory bandwidth
- **Ridge point:** the diagonal and horizontal roofs meet
 - Far to right: only very intensive kernels can achieve max perf
 - Far to left: almost any kernel can potentially hit max perf



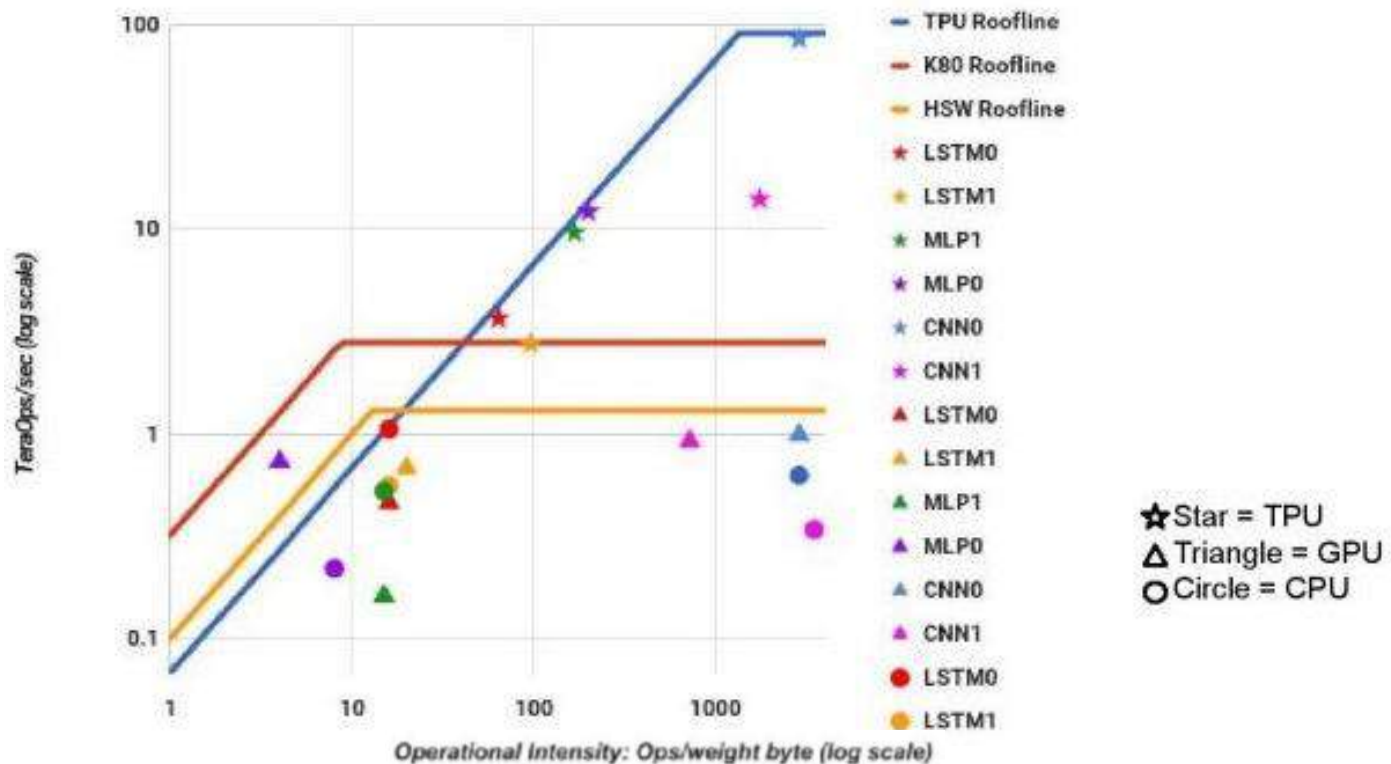
Example

- Consider: for ($i = 0$; $i < N$; $++i$) $y[i] = a * x[i] + y[i]$
 - For each “ i ” :
 - 1 addition, 1 multiplication
 - 2 loads of 8 bytes each
 - 1 store
- Execution on BlueGene/Q
 - Peak 204.8 GFLOP/node
- Performance estimates:
 - $AI = 2 / (3 * 8) = 1 / 12$ $1/12 < 7.11 \rightarrow$ limited area on the Roofline plot
 - $7.11 / (1/12) = 85.32$
 - $204.8 / 85.32 = 2.4$ GF/s



TPU Roofline Performance

- TPU: its ridge point is far to the right at 1350
 - CNN1 is much further below its Roofline than the other DNNs
 - Waiting for weights to be loaded into the matrix unit
 - Ridge point comparison:
 - CPU: 13, GPU: 9 → better balanced, but perf a lot lower



Cost-Performance

- Cost metric: performance per watt
 - “Total”: includes the power consumed by the host CPU server when calculating perf/watt for the GPU and TPU
 - “Incremental”: subtracts the host CPU power from the total
- Total: GPU is 2.1x CPU, TPU is 34x CPU
- Incremental: TPU is 83x CPU, 29x GPU

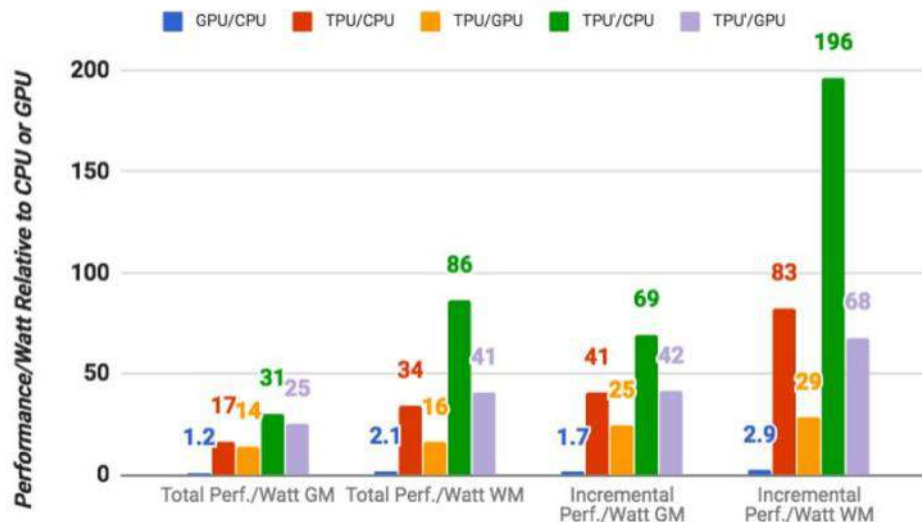
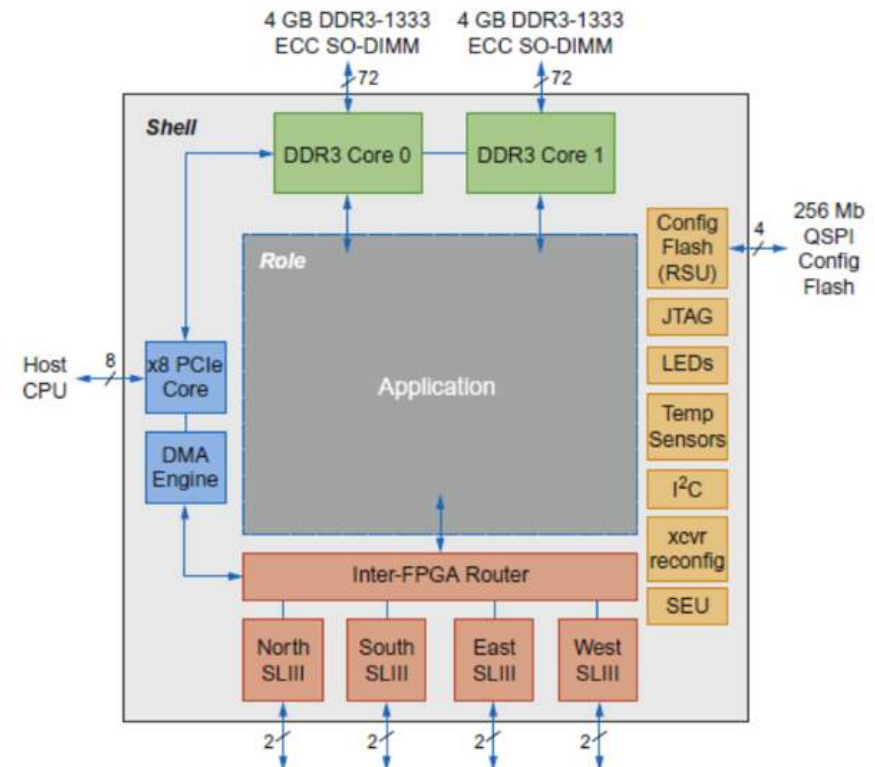


Figure 9. Relative performance/Watt (TDP) of GPU server (blue bar) and TPU server (red bar) to CPU server, and TPU server to GPU server (orange bar). TPU* is an improved TPU (Sec. 7). The green bar shows its ratio to the CPU server and the lavender bar shows its relation to the GPU server. Total includes host server power, but incremental doesn't. GM and WM are the geometric and weighted means.

<https://www.extremetech.com/computing/247199-googles-dedicated-tensorflow-processor-tpu-makes-hash-intel-nvidia-inference-workloads>

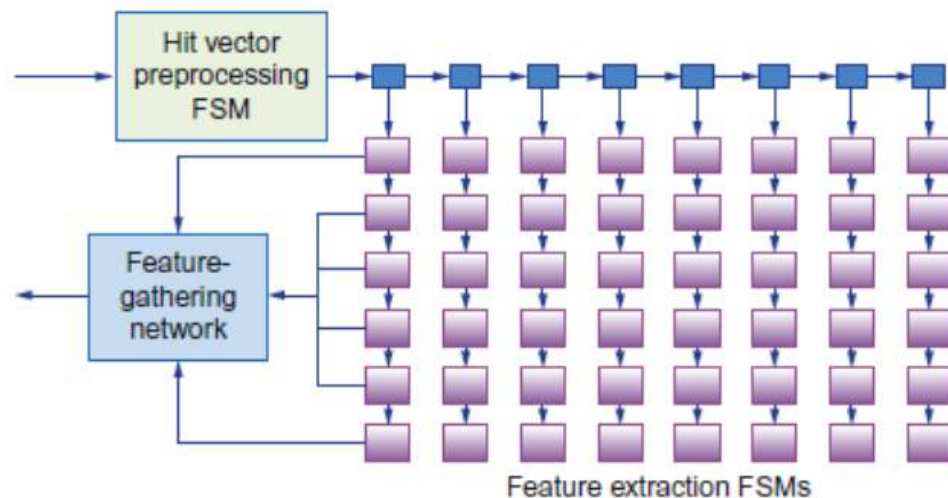
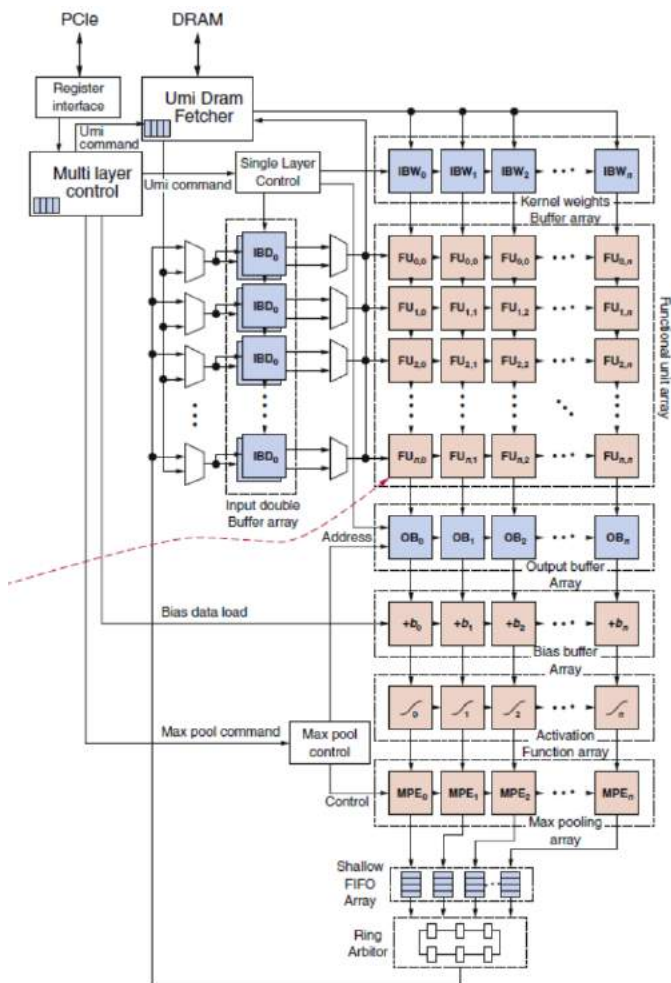
Microsoft Catapult

- Needed to be general purpose and power efficient
 - Uses FPGA PCIe board with dedicated 20 Gbps network in 6 x 8 torus
 - Each of the 48 servers in half the rack has a Catapult board
 - Limited to 25 watts
 - 32 MB Flash memory
 - Two banks of DDR3-1600 (11 GB/s) and 8 GB DRAM
 - FPGA (unconfigured) has 3962 18-bit ALUs and 5 MB of on-chip memory
 - Programmed in Verilog RTL
 - Shell is 23% of the FPGA



Catapult Applications

- The processing element (PE) of the CNN Accelerator for Catapult
- The architecture of FPGA implementation of the Feature Extraction stage in search acceleration



How Catapult Follows the Guidelines

- *Use dedicated memories*
 - 5 MB dedicated memory
- *Invest resources in arithmetic units and dedicated memories*
 - 3926 ALUs
- *Use the easiest form of parallelism that matches the domain*
 - 2D SIMD for CNN, MISD parallelism for search scoring
- *Reduce the data size and type needed for the domain*
 - Uses mixture of 8-bit integers and 64-bit floating-point
- *Use a domain-specific programming language*
 - Uses Verilog RTL; **Microsoft did not follow this guideline**