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0讲：课程介绍、概述

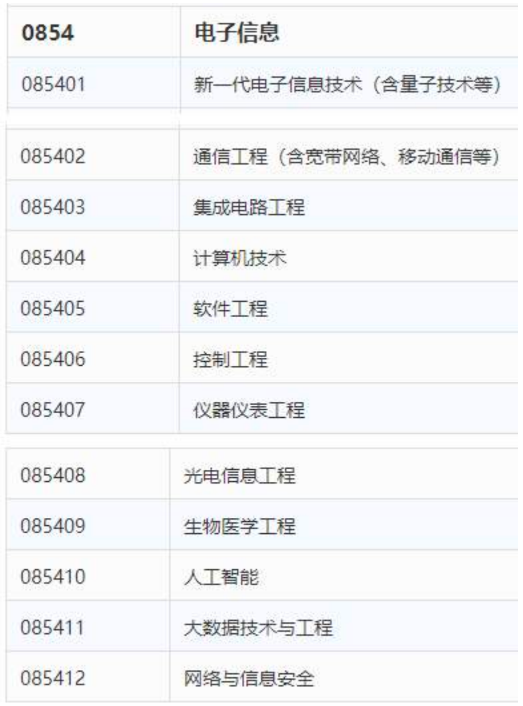
张献伟

xianweiz.github.io

DCS5367, 9/7/2021

关于课程

- 年级专业
  - 21级专硕（必修？）
    - 共4个班 （ 119/120人？ ）
- 先修课程
  - 计算机组成原理
  - 程序设计、数据结构、操作系统等



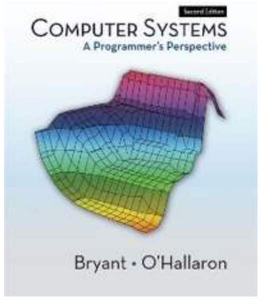
0854	电子信息
085401	新一代电子信息技术（含量子技术等）
085402	通信工程（含宽带网络、移动通信等）
085403	集成电路工程
085404	计算机技术
085405	软件工程
085406	控制工程
085407	仪器仪表工程
085408	光电信息工程
085409	生物医学工程
085410	人工智能
085411	大数据技术与工程
085412	网络与信息安全

- 计算机体系结构
  - 如何设计一台符合系统设计目标的计算设备？
  - 使用量化分析方法，对计算机瓶颈问题定位、描述、分析、评估
    - 包括指令流水线、并行、存储层次结构，多核结构等

课程教材

- 主要教材
  - John L. Hennessy and David A. Patterson，计算机体系结构：量化研究方法（英文版·原书第6版）
  - 非必须购买

- 参考资料
  - ECE 447, Onur Mutlu (Carnegie Mellon U.)
  - CIS 501, Milo Martin (U Penn)
  - Computer Organization and Design RISC-V Edition: The Hardware Software Interface (2nd Edition), Hennessy and Patterson
  - Computer Systems: A Programmer’s Perspective (CSAPP), Bryant and O’Hallaron



任课教师

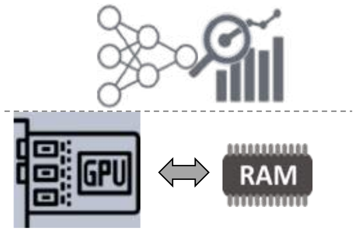
博士，2011 – 2017，University of Pittsburgh

学士，2007 – 2011，西北工业大学

副教授，2020.10 – 今

工程师/研究员，2017.08 – 2020.09

实习研究员，2016.05 – 2016.08



计算机体系结构

高性能及智能计算

软硬件设计及优化

学院个人主页：http://cse.sysu.edu.cn/content/5592

课件及答疑

- 课件
  - https://xianweiz.github.io/teach/dcs5637/f2021.html
  - 课后上传更新
- **课程QQ群：962501738**

Advance CA fall...

- 教师
  - 张献伟 （超算中心）
    - Email: zhangxw79@mail.sysu.edu.cn
    - 课前课间答疑，其他时间需预约
- 助教
  - 葛天傲 (getao3@mail2.sysu.edu.cn)
  - 莫泽威 (mozw5@mail2.sysu.edu.cn)

课时及考核

- 课时（3学分，54学时）
  - 排课：2-9，11-19周
  - 每次授课包括3个课时
    - 周二，2-4节（8:55-11:40）
  - 地点：教学大楼 D304
- 考核
  - 课堂参与（15%）- 点名、提问、测试
  - 作业（35%）- 课下
    - 习题、实践、paper review。。。
  - 期末考试 （50%）- 闭卷

- 课堂
  - 随机点名
    - 缺席优先
  - 随机提问
    - 后排优先
  - 随机测试
    - 不定时间
- 实验/作业
  - 个人完成
    - 杜绝抄袭
  - 按时提交
    - 超算习堂

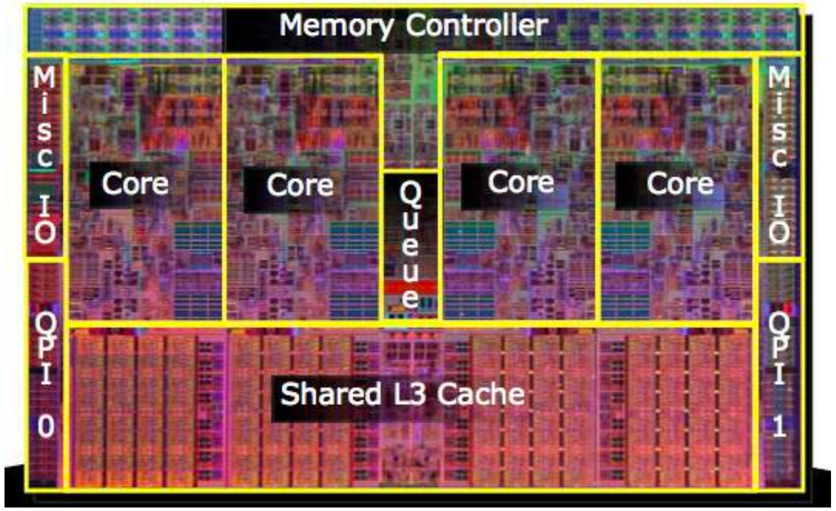
上课安排

Week/Date	Topic	Note
wk2: Sep 7	Overview	
wk3: Sep 14		
wk4: Sep 21	NO CLASS	Holiday
wk5: Sep 28		
wk6: Oct 5	NO CLASS	Holiday
wk7: Oct 12		
wk8: Oct 19		
wk9: Oct 26		
wk10: Nov 2	NO CLASS	Midterm Exercise
wk11: Nov 9		
wk12: Nov 16		
wk13: Nov 23		
wk14: Nov 30		
wk15: Dec 7		
wk16: Dec 14		
wk17: Dec 21		
wk18: Dec 28		
wk19: Jan 4		
wk20: Jan 11	NO CLASS	FINAL EXAM

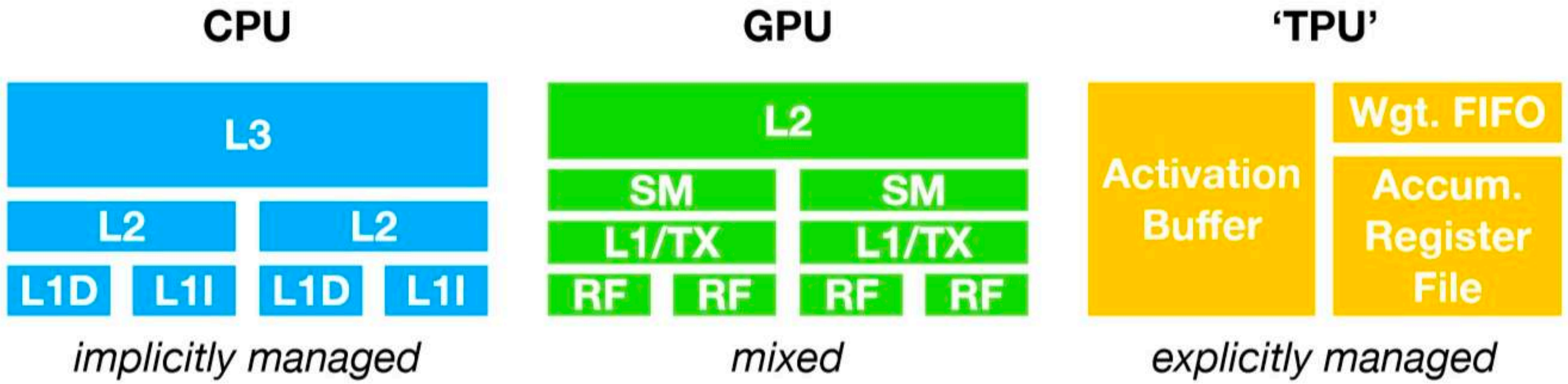
内容安排

- Overview and Fundamentals[概览与基础]
- Instruction Set Architecture[指令集架构]
  - Review
- Instruction Level Parallelism[指令级并行]
  - Pipelining, Branch Prediction, Instruction Scheduling
- Memory Hierarchy[存储层级]
  - Memory, Cache, Virtual Memory
- Data/Thread-Level Parallelism[数据/线程级并行]
  - SIMD, GPU
- Domain-specific Architectures[领域专用架构]
- And more …

Examples



Memory Controller
Misc IO
Core
Core
Queue
Core
Core
Misc IO
QPI 0
Shared L3 Cache
QPI 1



CPU
L3
L2
L2
L1D
L1I
L1D
L1I
implicitly managed
GPU
L2
SM
SM
L1/TX
L1/TX
RF
RF
RF
RF
mixed
'TPU'
Activation Buffer
Wgt. FIFO
Accum. Register File
explicitly managed

课程目标

- This course covers HW/SW, and the interface[内容覆盖]
  - We will focus on performance analysis and design tradeoffs
- Two key goals of this course are[主要目标]
  - To understand how hardware components works with the software layer and how decisions made in hardware affect the software/programmer
  - To enable you to be comfortable in making design and optimization decisions that cross the boundaries of different layers and system components
- Two other goals of this course[额外目标]
  - Enable you to think critically
  - Enable you to think broadly

一些问题？？？

“摩尔定律”真的要终结吗？量子计算是未来吗？

GPU怎么就从图形专用到了计算通用？

NPU, TPU, 加速器？异构计算？

为什么Nvidia过去几年市值增长了40倍？没有竞争对手的吗？

Intel挤牙膏？AMD YES！

DRAM、DDR、HBM、LPDDR，什么关系？

Meltdown, Spectre漏洞为什么会发生？

华为被制裁，卡脖子卡的是什么？x86, ARM, RISC-V？

太阳位置会影响计算机稳定？

为什么要有cuBLAS基础库？

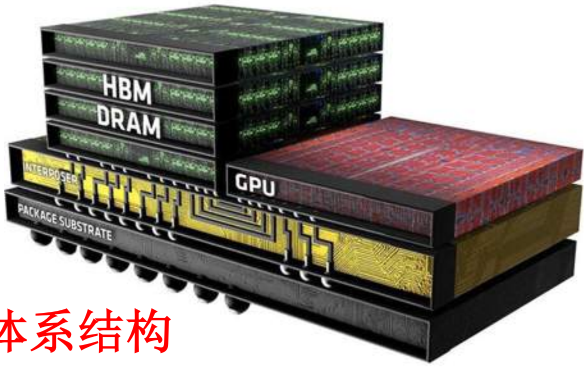
GPU任务怎么被执行的？profiling怎么实现的？

数据中心、超算、云，什么关系？

“冯·诺伊曼”架构要被淘汰了？

什么是体系结构?

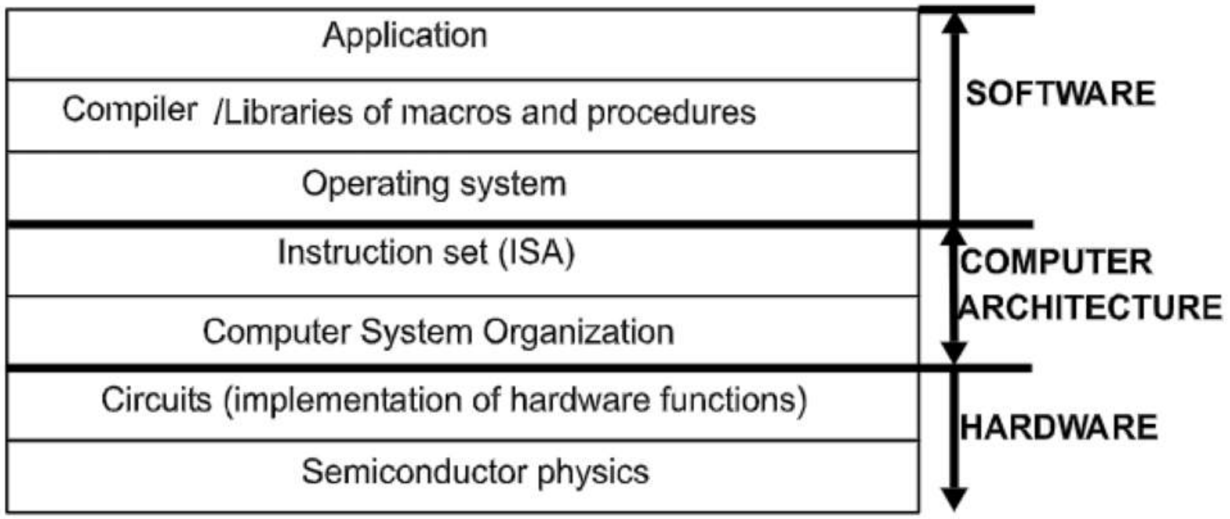
- Computer Architecture
  - The science and art of designing, selecting, and interconnecting hardware components and designing the hardware/software interface to create a computing system that meets functional, performance, energy consumption, cost, and other specific goals[硬件设计 – 软硬件接口 – 系统目标]
- Computer architecture is the glue that binds SW and HW
  - Inter-disciplinary in nature[连接软硬件、跨学科]



建筑 == 体系结构

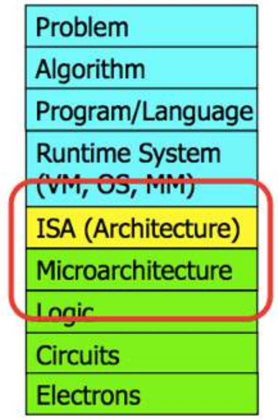
什么是体系结构？ (cont.)

- Computer organization[组成]
  - How to build computers?
- Computer architecture[架构]
  - How to design computer system?
- Layered view of computer systems[分层角度]


Application
Compiler /Libraries of macros and procedures
Operating system
Instruction set (ISA)
Computer System Organization
Circuits (implementation of hardware functions)
Semiconductor physics
SOFTWARE
COMPUTER ARCHITECTURE
HARDWARE

计算机分层与抽象

- Layering to deal with complex systems[分层]
  - Computers are complex, built in layers
    - Software: applications, compiler, operating system
    - Hardware: logic, circuits, electrons
- Layering brings abstraction[抽象]
  - A higher level only needs to know about the interface to the lower level, not how the lower level is implemented
    - 99% of users don't know hardware layers implementation
    - 90% of users don't know implementation of any layer
    - That's okay, world still works just fine
- Then, **why** would you want to know what goes on underneath or above?

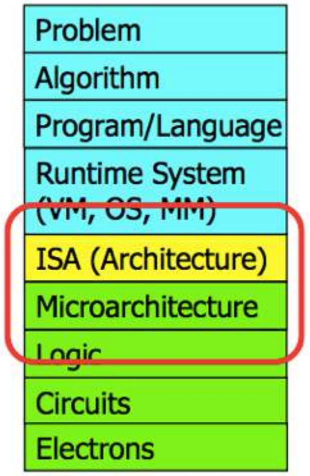


计算机分层与抽象 (cont.)

- As long as everything goes well, not knowing what happens in the underlying level (or above) is not a problem
- What if
  - The program you wrote is running slow?[慢]
  - The program you wrote does not run correctly?[错误]
  - The program you wrote consumes too much energy?[高能耗]
- What if
  - The hardware you designed is too hard to program?[难用]
  - The hardware you designed is too slow because it does not provide the right primitives to the software?[性能差]
- What if
  - You want to design a much more efficient and higher performance system?[效率更高、性能更高]

计算机分层与抽象 (cont.)

- How would you solve the problem?[怎样解决]
- What is the right place to solve the problem? [哪里解决]
  - Programmer?
  - System software? Compiler?
  - Hardware? Circuits?



- Breaking the abstraction layers and knowing what is underneath enables you to solve problems and design better future systems[打破抽象]
- Cooperation between multiple components and layers enable more effective solutions and systems[协同考虑]

架构和架构师

- Computer architect[计算机架构师]
  - To make design trade-offs across the hw/sw interface to meet functional, performance and cost requirements
- Being an architect is not easy[并不容易]
  - You need to consider many things in designing a new system + have good intuition/insight into ideas/tradeoffs
- But, it is fun and can be very technically rewarding[会很有意思]
- And, enables a great future[影响未来]
  - Advancement of computer architecture is vital to all other areas of computing
    - E.g., IoT, Embedded, Mobile, Data centers, HPC

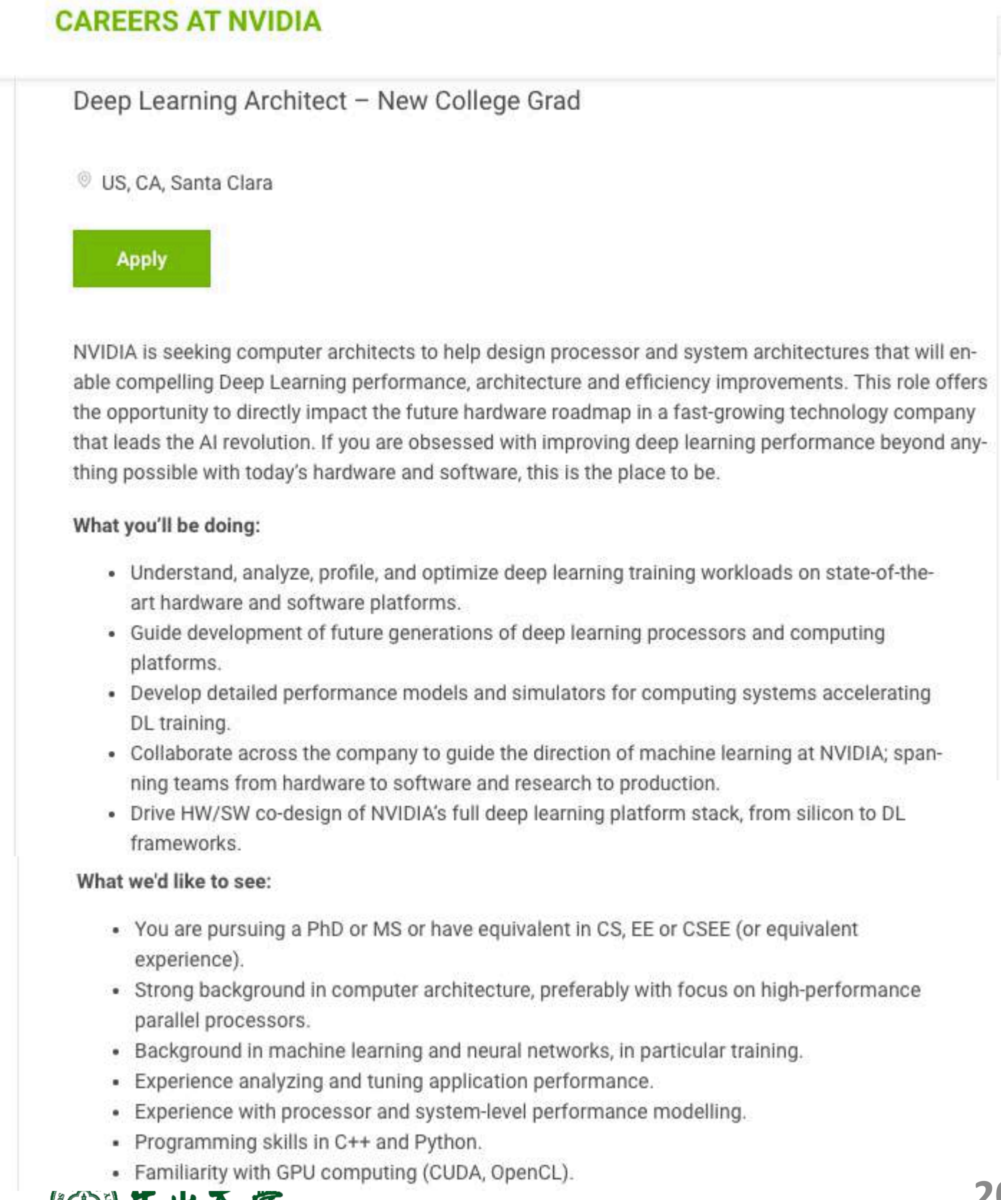
Role of [Computer] Architect[职责]

- Look backward (to the past)
  - Understand tradeoffs and designs, upsides/downsides, past workloads. Analyze and evaluate the past.
- Look forward (to the future)
  - Be the dreamer and create new designs. Listen to dreamers.
  - Push the state of the art. Evaluate new design choices.
- Look up (towards problems in the computing stack)
  - Understand important problems and their nature.
  - Develop architectures and ideas to solve important problems.
- Look down (towards device/circuit technology)
  - Understand the capabilities of the underlying technology.
  - Predict and adapt to the future of technology (you are designing for *N* years ahead). Enable the future technology.

为什么要学习体系结构?

- Understand why computers work the way they do
  - Source code --> instructions --> executions
  - Processors --> cache/memory --> storage
- Understand where computers are going
  - Future capabilities drive the (computing) world
  - Real-world impact: no computer architecture -> no computers!
- Understand high-level design concepts and performance
  - The best architects understand all the levels (hw, OS, apps, alg)
  - Need to understand hardware to write fast software
- Job?
  - Your MS/PhD research may involve CA
  - Your job positions (design or research, hw or sw) may need CA
  - You may manage a team working on systems

工作职位？？？



CAREERS AT NVIDIA

Deep Learning Architect – New College Grad

US, CA, Santa Clara

Apply

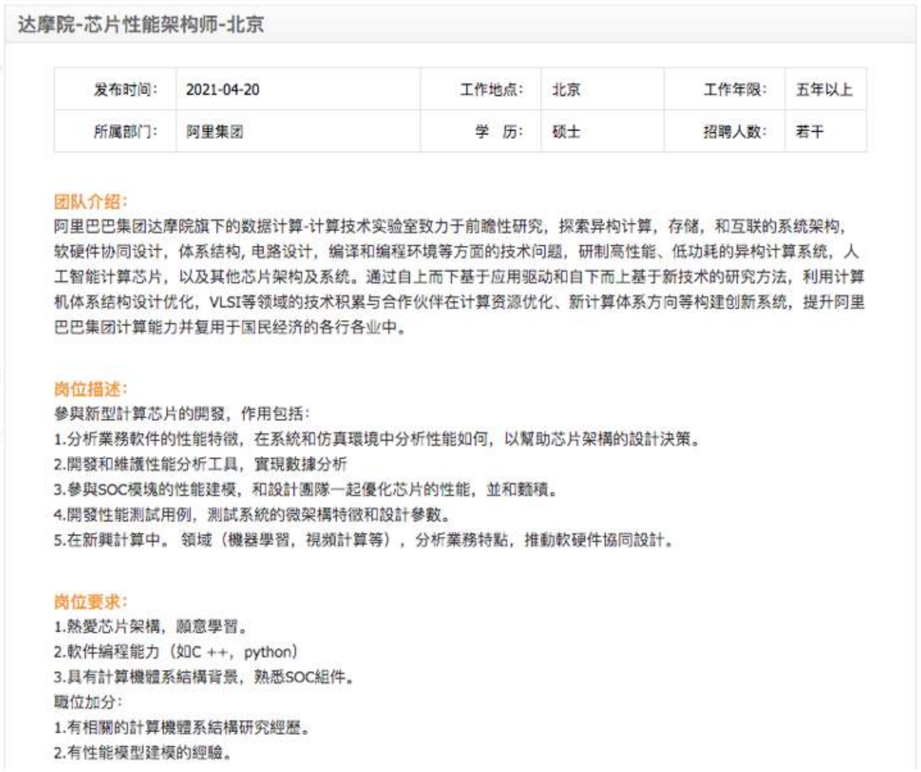
NVIDIA is seeking computer architects to help design processor and system architectures that will enable compelling Deep Learning performance, architecture and efficiency improvements. This role offers the opportunity to directly impact the future hardware roadmap in a fast-growing technology company that leads the AI revolution. If you are obsessed with improving deep learning performance beyond anything possible with today's hardware and software, this is the place to be.

What you'll be doing:

- Understand, analyze, profile, and optimize deep learning training workloads on state-of-the-art hardware and software platforms.
- Guide development of future generations of deep learning processors and computing platforms.
- Develop detailed performance models and simulators for computing systems accelerating DL training.
- Collaborate across the company to guide the direction of machine learning at NVIDIA; spanning teams from hardware to software and research to production.
- Drive HW/SW co-design of NVIDIA's full deep learning platform stack, from silicon to DL frameworks.

What we'd like to see:

- You are pursuing a PhD or MS or have equivalent in CS, EE or CSEE (or equivalent experience).
- Strong background in computer architecture, preferably with focus on high-performance parallel processors.
- Background in machine learning and neural networks, in particular training.
- Experience analyzing and tuning application performance.
- Experience with processor and system-level performance modelling.
- Programming skills in C++ and Python.
- Familiarity with GPU computing (CUDA, OpenCL).



达摩院-芯片性能架构师-北京

发布时间：	2021-04-20	工作地点：	北京	工作年限：	五年以上
所属部门：	阿里集团	学　历：	硕士	招聘人数：	若干

团队介绍：
阿里巴巴集团达摩院旗下的数据计算-计算技术实验室致力于前瞻性研究，探索异构计算，存储，和互联的系统架构，软硬件协同设计，体系结构，电路设计，编译和编程环境等方面的技术问题，研制高性能、低功耗的异构计算系统，人工智能计算芯片，以及其他芯片架构及系统。通过自上而下基于应用驱动和自下而上基于新技术的研究方法，利用计算机体系结构设计优化，VLSI等领域的技术积累与合作伙伴在计算资源优化、新计算体系方向等构建创新系统，提升阿里巴巴集团计算能力并复用于国民经济的各行各业中。

岗位描述：
參與新型計算芯片的開發，作用包括：
1.分析業務軟件的性能特徵，在系統和仿真環境中分析性能如何，以幫助芯片架構的設計決策。
2.開發和維護性能分析工具，實現數據分析
3.參與SOC模塊的性能建模，和設計團隊一起優化芯片的性能，並和麵積。
4.開發性能測試用例，測試系統的微架構特徵和設計參數。
5.在新興計算中。 領域（機器學習，視頻計算等），分析業務特點，推動軟硬件協同設計。

岗位要求：
1.熱愛芯片架構，願意學習。
2.軟件編程能力（如C ++，python）
3.具有計算機體系結構背景，熟悉SOC組件。
職位加分：
1.有相關的計算機體系結構研究經歷。
2.有性能模型建模的經驗。

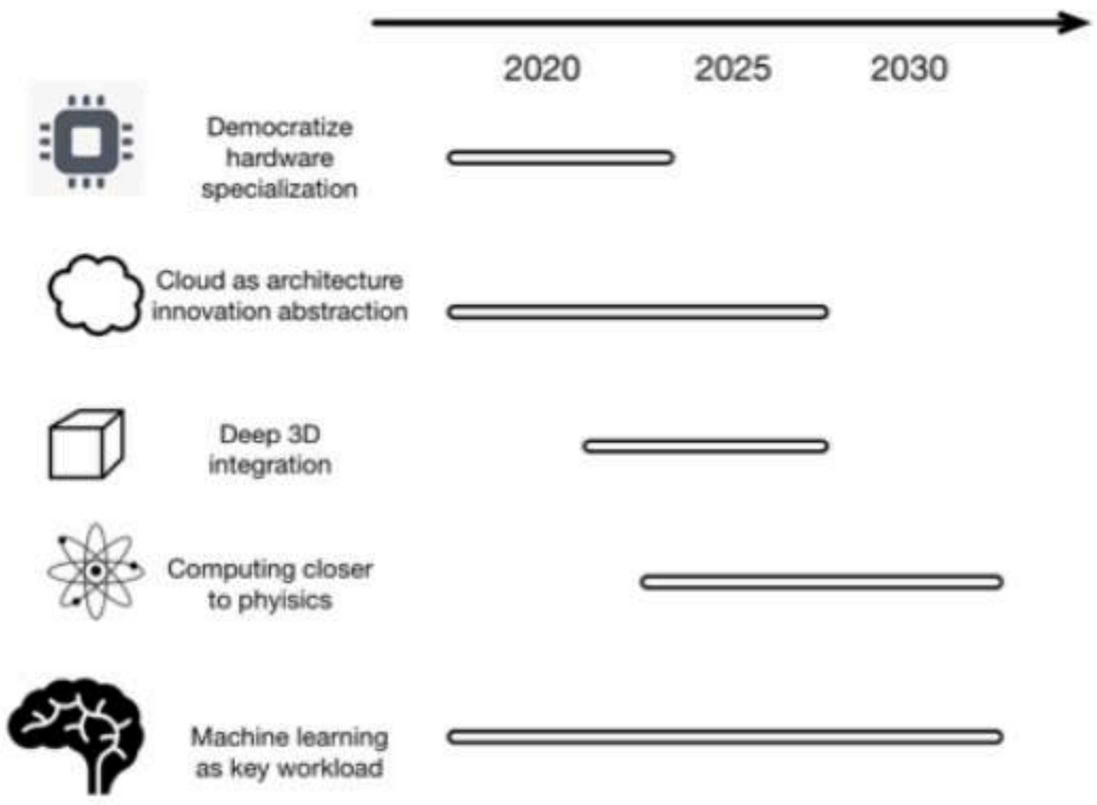
Golden Age for CA

- Today is a very exciting time to study architecture
  - Many new demands from the top
  - Fast changing demands and personalities of users
  - Many new issues at the bottom
- Computing landscape is very different from 10-20 years ago
  - Every component and its interfaces, as well as entire system designs are being re-examined
  - You can revolutionize the way computers are built, if you understand both the hardware and the software (and change each accordingly)
- **No clear, definitive answers to these problems[有问题，缺方案]**

学术界

Architecture 2030 Workshop @ ISCA 2016

John L. Hennessy, David A. Patterson



[1] Arch2030, https://arxiv.org/pdf/1612.03182.pdf (2016)
[2] A New Golden Age for Computer Architecture (2019)

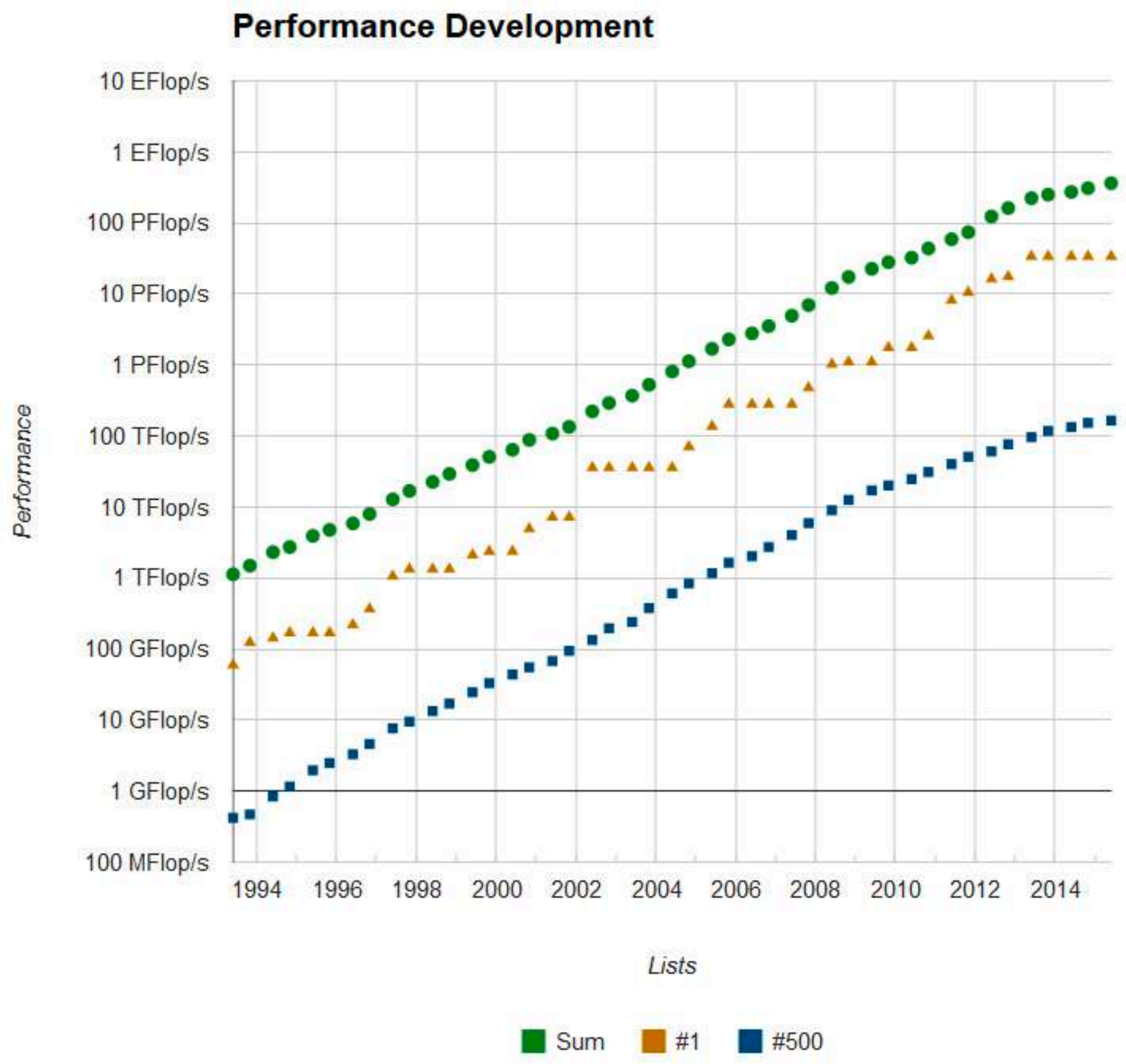
- Current challenges[问题]
  - End of Moore's Law and Dennard Scaling
  - Overlooked security
- Future opportunities in computer architecture[机遇]
  - Domain-specific architectures
  - Domain-specific languages
  - Open architectures
  - Agile hardware development

工业界

- Nvidia, 04/2021: Grace, ARM-based data-center CPU[1]
- Apple, 11/2020: M1, ARM-based SoC[2]
- AMD, 10/2020: Acquire Xilinx[3]
- Intel, 09/2020: Xe GPU[4]
- Samsung, 11/2019: Cease CPU development[6]
- Amazon, 11/2018: AWS Graviton[5]
- Intel/IBM/ARM, 01/2018: Meltdown and Spectre
- Micron, 03/2021: Cease 3D-XPoint, invest CXL[7]
- AI Chips: Graphcore, Habana Labs, Cerebras, Cambricon…

[1] https://www.nvidia.com/en-us/data-center/grace-cpu/
[2] https://pdf.wondershare.com/macos/everything-about-apple-m1-chip.html
[3] https://www.amd.com/en/corporate/xilinx-acquisition
[4] https://www.intel.com/content/www/us/en/products/discrete-gpus/iris-xe-aic.html
[5] https://aws.amazon.com/ec2/graviton/
[6] https://www.anandtech.com/show/15061/samsung-to-cease-custom-cpu-development
[7] https://investors.micron.com/news-releases/news-release-details/micron-updates-data-center-portfolio-strategy-address-growing

Exascale Supercomputing[E级超算]



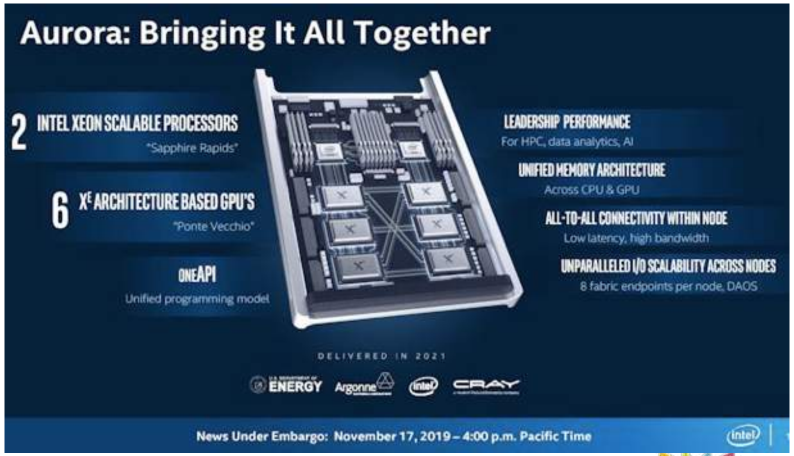
- ECP Director Paul Messina*:
  - Four key challenges: parallelism, memory and storage, reliability and energy consumption.
  - Advances in computer hardware and architecture will contribute to meeting all four challenges.

* https://www.exascaleproject.org/path-nations-first-exascale-supercomputers-pathforward/

中山大學 SUN YAT-SEN UNIVERSITY

SC: Aurora

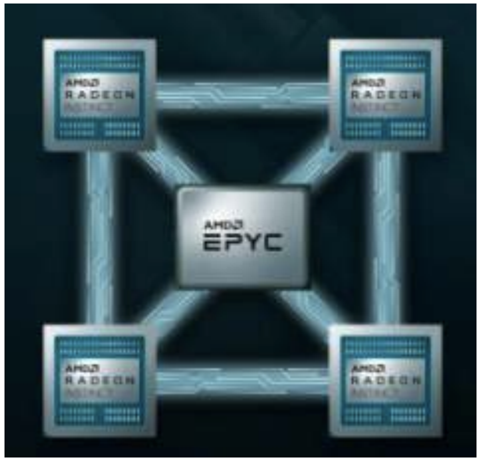
- Aurora @ Argonne National Laboratory*
  - 1 exaFLOPS, ≤ 60 MW, 2018-2021-2022
  - Compute node:
    - 2 Intel Xeon Sapphire Rapids CPUs, 6 Xe GPUs
      - First enterprise CPUs to support CXL standard
      - GPUs communicates directly to each other via CXL
    - Unified memory architecture
  - Interconnect
    - CPU-GPU: PCIe, GPU-GPU: Xe Link
    - System: HPE Slingshot 11; Dragonfly topology with adaptive routing
  - Programming models
    - Intel oneAPI, MPI, OpenMP, C/C++, Fortran, SYCL/DPC++
  - Applications: climate change, cancer, new materials


Aurora: Bringing It All Together
2 INTEL XEON SCALABLE PROCESSORS
"Sapphire Rapids"
6 Xe ARCHITECTURE BASED GPU'S
"Ponte Vecchio"
ONEAPI
Unified programming model
LEADERSHIP PERFORMANCE
For HPC, data analytics, AI
UNIFIED MEMORY ARCHITECTURE
Across CPU & GPU
ALL-TO-ALL CONNECTIVITY WITHIN NODE
Low latency, high bandwidth
UNPARALLELED I/O SCALABILITY ACROSS NODES
8 fabric endpoints per node, DAOS
DELIVERED IN 2021
ENERGY
Argonne
intel
CRAY
News Under Embargo: November 17, 2019 – 4:00 p.m. Pacific Time

* https://www.alcf.anl.gov/aurora

SC: Frontier

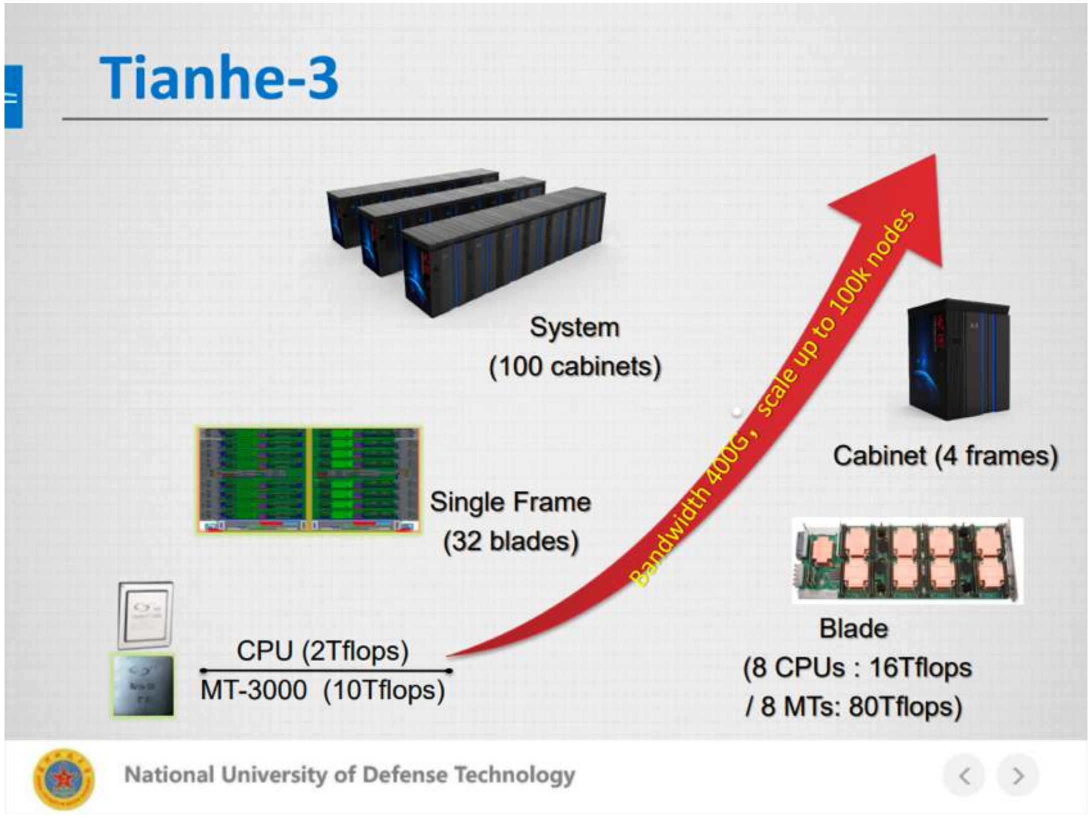
- Frontier @ Oak Ridge National Laboratory*
  - 1.5 exaFLOPS, 30 MW, 2021
  - Compute node
    - 1 AMD EPYC CPU (Zen 3) + 4 AMD Radeon Instinct GPU (MI 200)
  - Interconnect
    - Node: AMD Infinity Fabric, coherent memory across the node
    - System:
      - Multiple Slingshot NICs providing 100 GB/s network bandwidth.
      - Slingshot dragonfly network w/adaptive routing
  - Programming models:
    - AMD ROCm, MPI, OpenMP, HIP C/C++, Fortran
  - Applications: modeling and simulation, data analytics, AI



* https://www.olcf.ornl.gov/wp-content/uploads/2019/05/frontier_specsheet.pdf

SC: Tianhe-3

- Tianhe-3 @ Tianjin
  - China's native manycore Armv8-based Phytium 2000+ (FTP)
  - Matrix 3000 (MTP) Accelerator



Advanced Computer Architecture
高级计算机体系结构

第1讲：量化设计分析基础

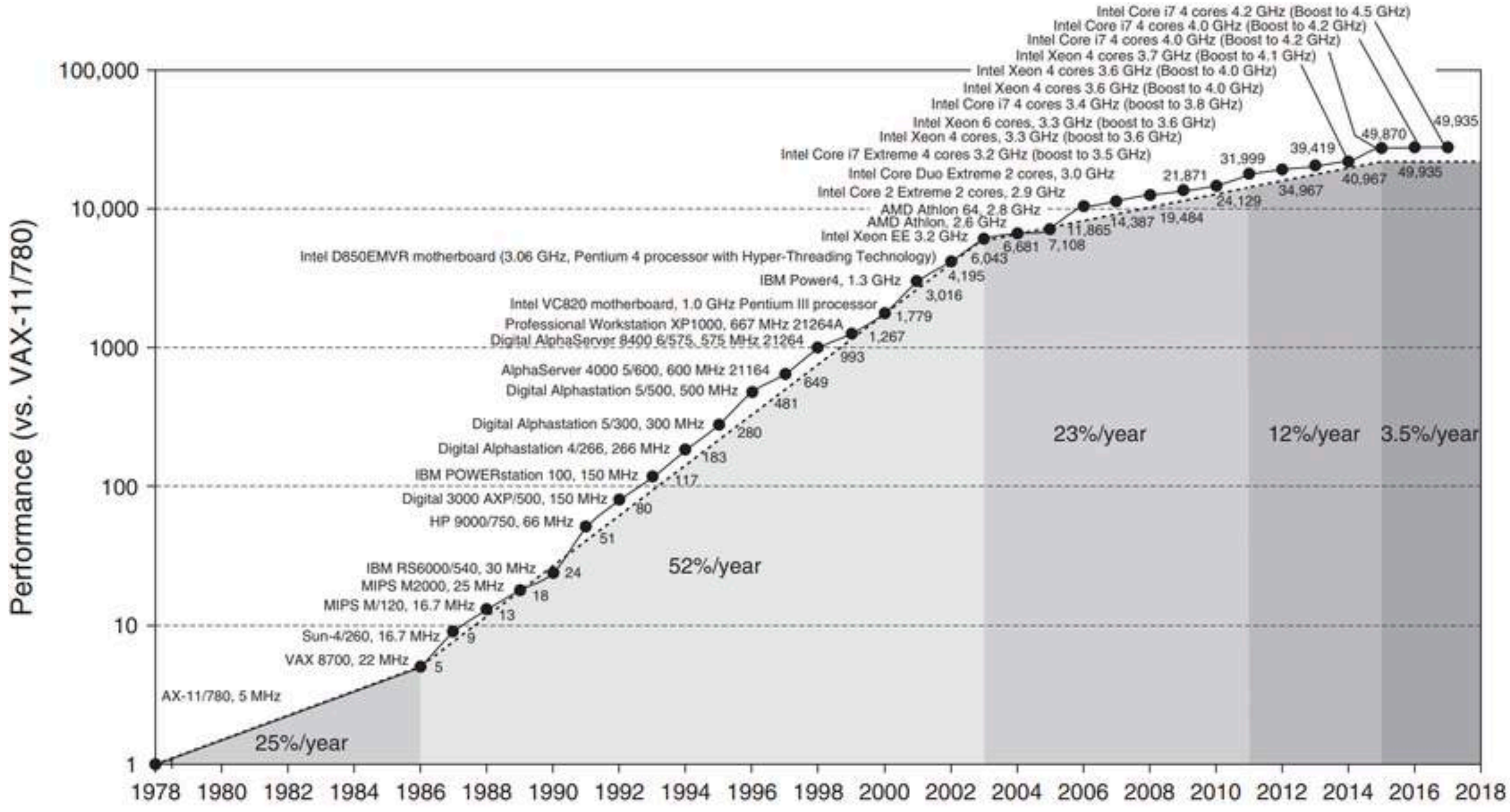
张献伟

xianweiz.github.io

DCS5367, 9/7/2021

Technology Improvement[技术提升]

- Computer technology has greatly improved
  - A $500 cellphone today outperforms the fastest supercomputer in 1993 ($50 million)
  - Improvement from both advances in the tech used to build computers and from innovations in computer design

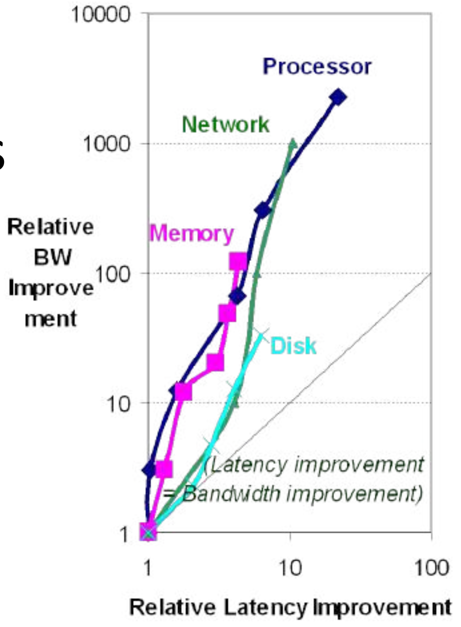


Trends in Technology (§1.4)[技术趋势]

- Integrated circuit (IC) logic[集成电路]
  - Transistor density: +35%/year (feature size decreases)
  - Die size: +10-20%/year
  - Integration overall: +40-55%/year (Moore's Law)
- DRAM capacity: +25-40%/year (growth is slowing)[内存]
  - Memory usage quadruples every three years
- Flash capacity: +50-60%/year[闪存]
  - 8-10X cheaper/bit than DRAM
- Magnetic disk: +40%/year[磁盘]
  - 8-10X cheaper/bit then Flash and 200-300X cheaper/bit than DRAM
- Network

Performance Trends[性能趋势]

- Bandwidth or throughput[带宽/吞吐]
  - Total work done in a given time
  - 32,000 - 40,000X improvement for processors
  - 400 - 2400X improvement for memory and disks
- Latency or response time[时延/响应时间]
  - Time between start and completion of an event
  - 50 - 90X improvement for processors
  - 8 - 9X improvement for memory and disks
    - ▫ Memory wall[内存墙]



- Latency lags bandwidth (in the last 30 years)

Transistors and Wire[晶体管/线路]

- IC processes are characterized by feature size[特征尺寸]
- **Feature size**: minimum size of transistor or wire
  - 10um in 1971 to 22nm in 2012 to 7nm in 2017 (5nm is being developed)
- Moore’s Law: aka “technology scaling”[缩放]
  - Literally: density (transistors/area) doubles every 18 months
  - Public interpretation: performance doubles every 18 months
  - Continued miniaturization (esp. reduction in channel length)
    - + Improves switching speed, power/transistor, area(cost)/transistor
    - – Reduces transistor reliability

Power and Energy (§1.5)[功耗/能耗]

- **Energy** is a biggest challenge facing computer design
  - Bring power in with 100s of pins
  - Power is dissipated as heat and must be removed
- **Power/energy** are increasingly important
  - Battery life for mobile devices[电池续航]
    - Laptops, phones, cameras
  - Tolerable temperature for devices without active cooling[温度]
    - Power means temperature, active cooling means cost
    - No room for a fan in a cell phone, no market for a hot cell phone
  - Electric bill for compute/data centers[电费]
    - Pay for power twice: once in, once out (to cool)
  - Environmental concerns[环保]
    - “Computers” account for growing fraction of energy consumption

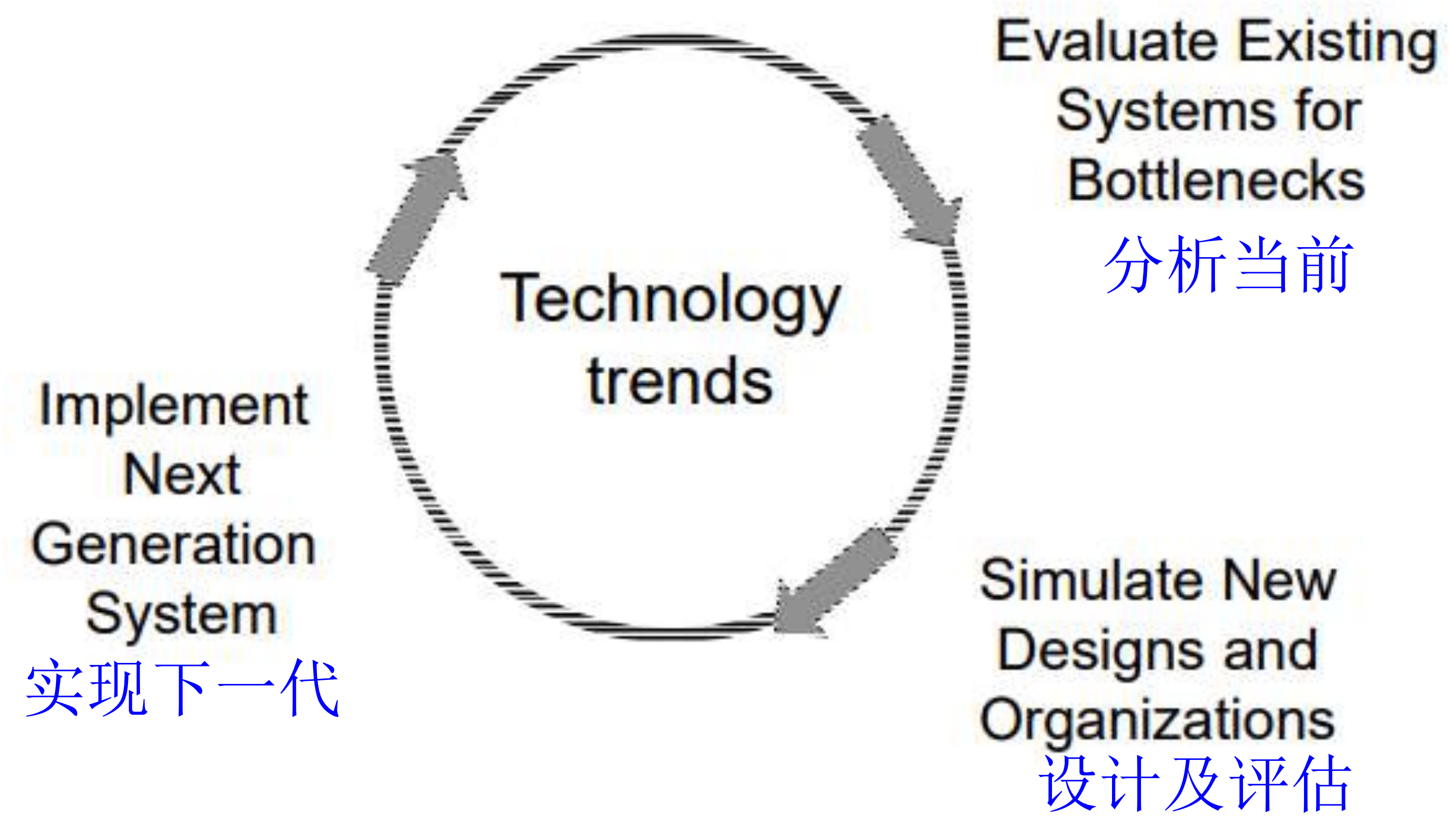
Power and Energy (cont.)

- **Energy**: measured in Joules or Watt-seconds[焦耳]
  - Total amount of energy stored/used
  - Battery life, electric bill, environmental impact
- **Power**: energy per unit time (measured in Watts)[瓦特]
  - Joules per second
  - Power impacts power supply and cooling requirements (cost)
  - Peak power vs average power
- Two sources[来源]
  - Dynamic power: active switching of transistors
  - Static power: leakage of transistors even while inactive
- Calculation
  - Energy is proportional to Voltage^2
  - Power is proportional to (Voltage^2 x Frequency)

Example：天河2号

- 配置
  - 16000个节点，每个2*12-核CPU + 2 Matrix-2000 + 64GB内存
    - 系统峰值运算速度为每秒10.07亿亿次，持续速度每秒6.14亿亿次
- 峰值功耗：17.6MW （加散热系统20+MW）
  - 17.6MW x 24h x 365 = 1.5亿度电/年
    - 40万度/天 → 30万元电费/天
- 水冷散热
  - 机柜内循环水冷的模式：8°C进水， 21°C出水

Methodology: Design/Evaluation[方法]


Technology trends
Evaluate Existing Systems for Bottlenecks
分析当前
Simulate New Designs and Organizations
设计及评估
Implement Next Generation System
实现下一代

Design Goals[设计目标]

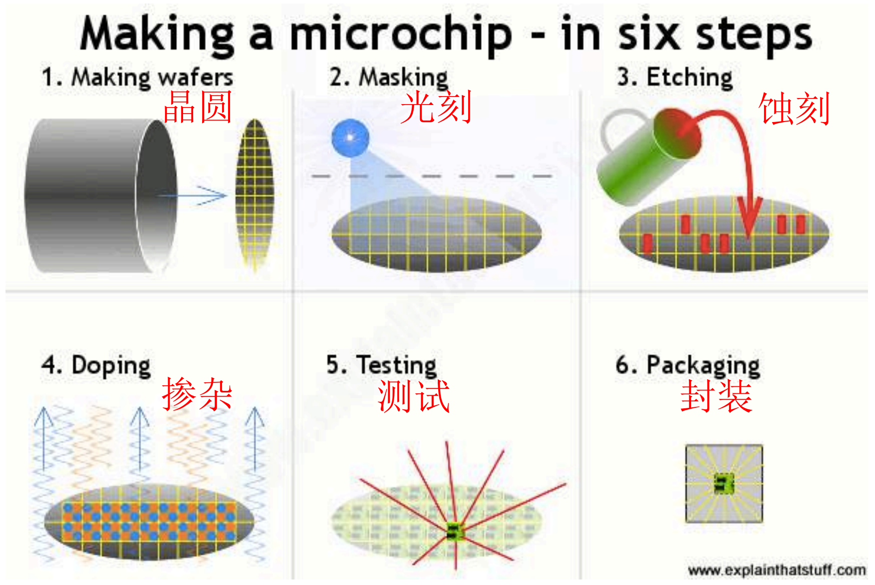
- Functional[功能性]
  - What functions should it support？
  - Needs to be correct
    - Unlike software, difficult to update once deployed
- High performance[高性能]
  - “Fast” is only meaningful in the context of a set of important tasks
  - Not just “Gigahertz”
  - Impossible goal: fastest possible design for all programs
- Reliable[可靠性]
  - Does it continue to perform correctly?
  - Hard fault vs. transient fault
    - Example：memory errors and sun spots
  - Space satellites vs. desktop vs. server reliability

Design Goals (cont.)

- Low cost[低成本]
  - Design cost (huge design teams, why?)[设计]
  - Cost of making first chip after design (mask cost)[流片]
  - Per unit manufacturing cost (wafer cost)[量产]
- Low power/energy[低能耗]
  - Energy in (battery life, cost of electricity)
  - Energy out (cooling and related costs)
  - Cyclic problem, very much a problem today
- **Challenge**: balancing the relative importance of these goals
  - And the balance is constantly changing
  - No goal is absolutely important at expense of all others
  - Our focus: performance, only touch on cost, power, reliability

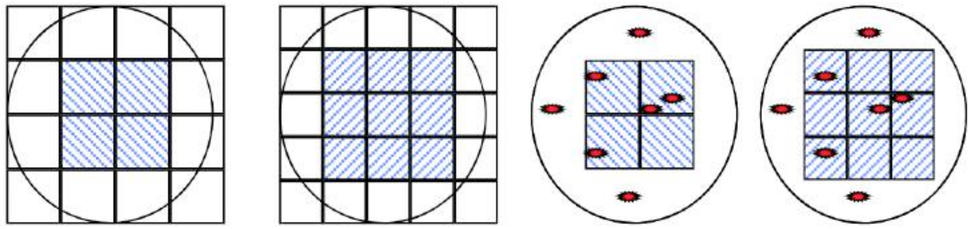
Manufacturing Process[制造流程]

- Silicon wafers[晶圆] undergo many processing steps so that different parts of the wafer behave as insulators, conductors, and transistors (switches)
- Multiple metal layers on the silicon enable connections between transistors
- The wafer is chopped into many dies[裸晶或裸片] – the size of the die determines yield and cost



Integrated Circuits Costs (§1.6)[成本]

- $Dies\ per\ wafer = \frac{\pi * (Wafer\ diameter/2)^2}{Die\ area} - \frac{\pi * Wafer\ diameter}{\sqrt{2 * Die\ area}}$



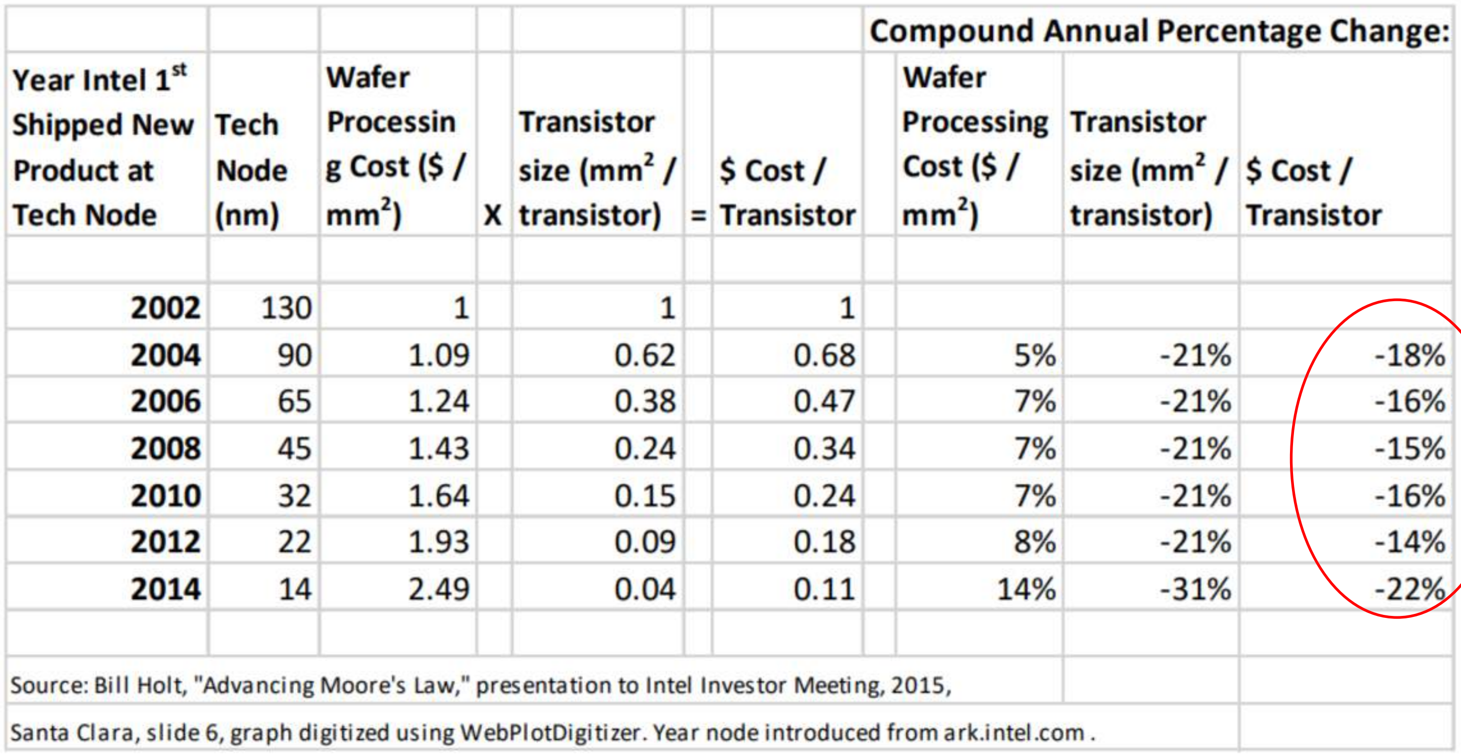
- $Die\ yield = \ Wafer\ yield * \frac{1}{(1 + Defects\ per\ unit\ area * Die\ area)^N}$

  Where *N*= process-complexity factor=7.5-9.5(28nm,2017)

- $Cost\ of\ die = \frac{Cost\ of\ wafer}{Dies\ per\ wafer * Die\ yield}$
- $Cost\ of\ IC = \frac{Cost\ of\ die + Cost\ of\ testing\ die + Cost\ of\ packaging\ and\ final\ test}{Final\ test\ yeild}$

Integrated Circuits Costs (cont.)

- Real-world examples



							Compound Annual Percentage Change:		
Year Intel 1st Shipped New Product at Tech Node	Tech Node (nm)	Wafer Processin g Cost ($ / mm^2)	X	Transistor size (mm^2 / transistor)	=	$ Cost / Transistor	Wafer Processing Cost ($ / mm^2)	Transistor size (mm^2 / transistor)	$ Cost / Transistor
2002	130	1		1		1			
2004	90	1.09		0.62		0.68	5%	-21%	-18%
2006	65	1.24		0.38		0.47	7%	-21%	-16%
2008	45	1.43		0.24		0.34	7%	-21%	-15%
2010	32	1.64		0.15		0.24	7%	-21%	-16%
2012	22	1.93		0.09		0.18	8%	-21%	-14%
2014	14	2.49		0.04		0.11	14%	-31%	-22%

Source: Bill Holt, "Advancing Moore's Law," presentation to Intel Investor Meeting, 2015,
Santa Clara, slide 6, graph digitized using WebPlotDigitizer. Year node introduced from ark.intel.com .

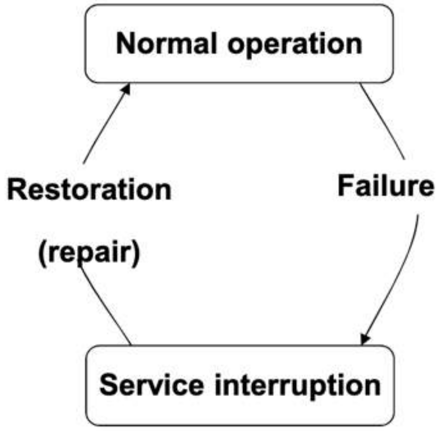
Wafer size conversions offset Intel's increased wafer-processing cost

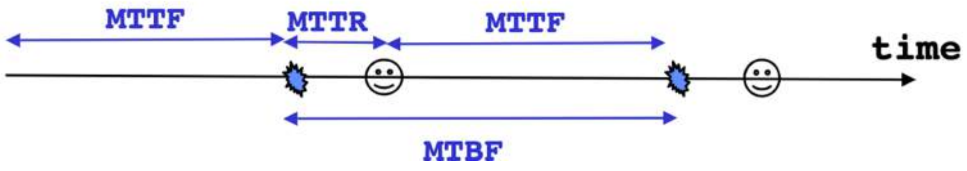
https://www.imf.org/~/media/Files/Conferences/2017-stats-forum/session-6-kenneth-flamm.ashx

SUN YAT-SEN UNIVERSITY

Dependability (§1.7)[可靠性]

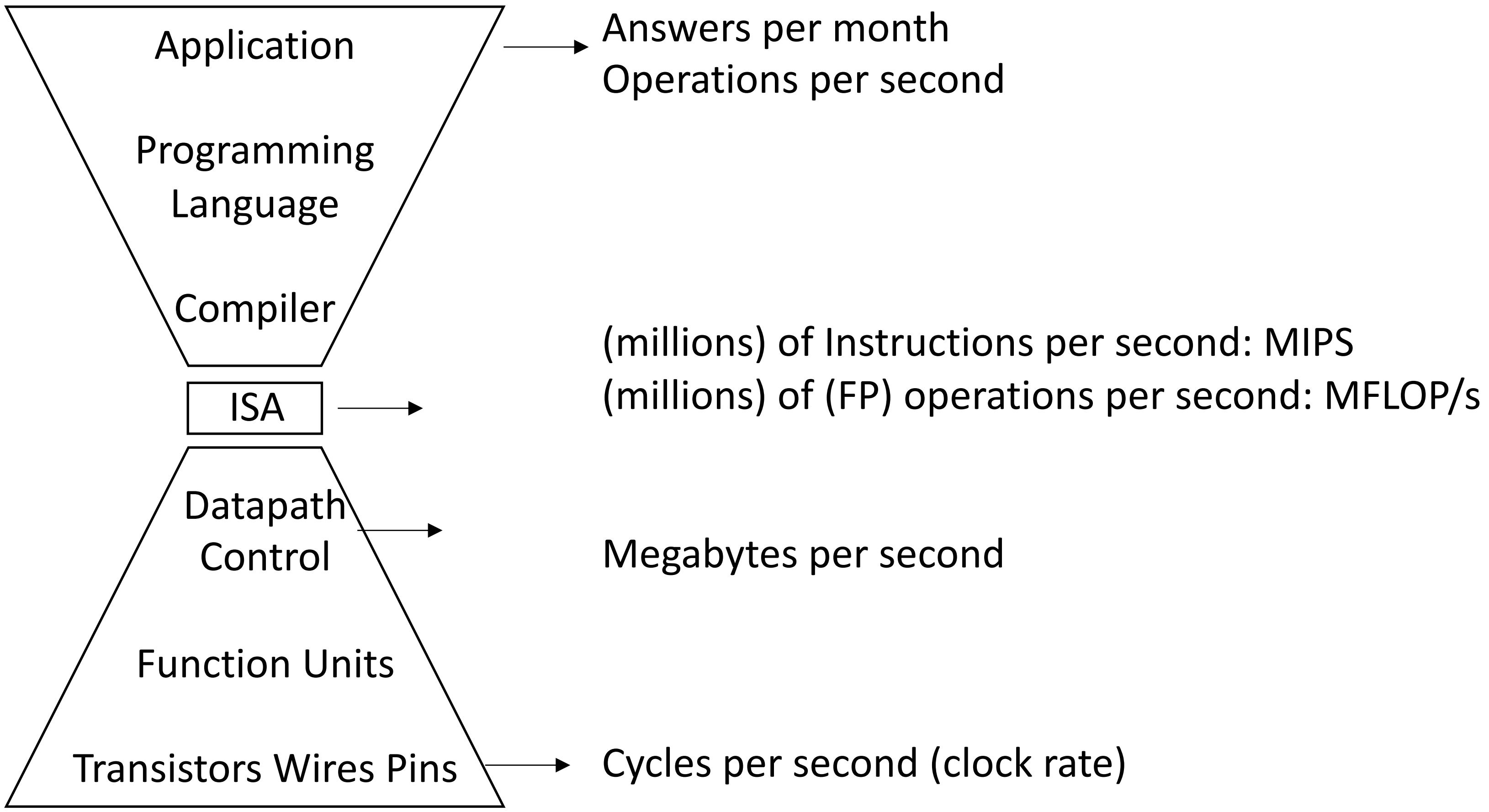
- Fault vs. error
  - Fault: failure of a component
  - Error: manifestation of a fault
  - Faults may or may not lead to system failure
- Metrics
  - Reliability measure: mean time to failure (MTTF)
  - Repair efficiency: mean time to repair (MTTR)
  - Mean time between failures
    - MTBF = MTTF + MTTR
  - Availability = MTTF / MTBF
- Improving availability
  - Increase MTTF: fault avoidance, fault tolerance, fault forecasting
  - Reduce MTTR: improved tools and processes for diagnosis/repair





Performance (§1.8)[性能]

- The performance metric may mean different things

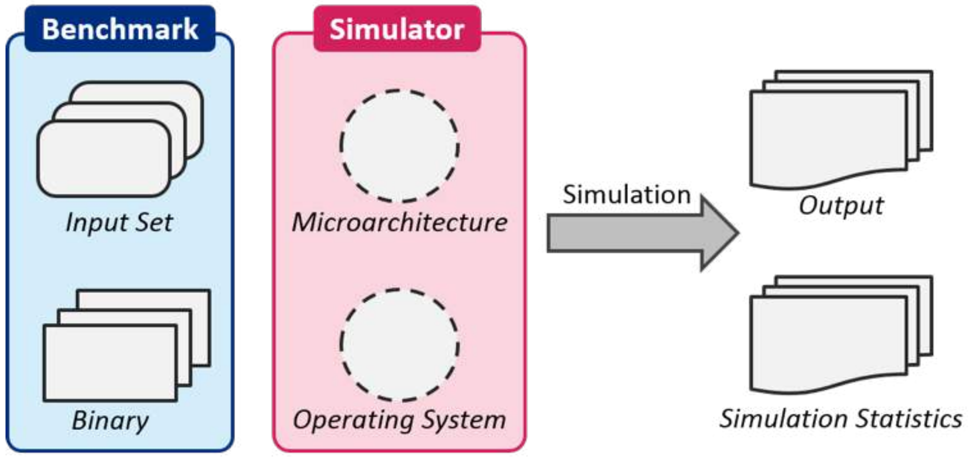


Measuring Performance[评估性能]

- Time to run the task (latency)
  - Execution time, response time, CPU time, …
- Tasks per day, hour, week, sec, ns, …
  - Throughput, bandwidth
- Performance measurement[测试]
  - Hardware prototypes : Cost, delay, area, power estimation
  - Simulation (many levels, ISA, RT, Gate, Circuit, …)
  - Benchmarks (Kernels, toy programs, synthetic), Traces, Mixes
  - Analytical modeling and Queuing Theory

Simulator[模拟器]

- What is an architecture (or architectural) simulator?
  - A tool that reproduces the behavior of a computing device
- Why use a simulator?
  - Leverage faster, more flexible software development cycle
  - Permits more design space exploration
  - Facilitates validation before hardware becomes available
  - Possible to increase/improve system instrumentation

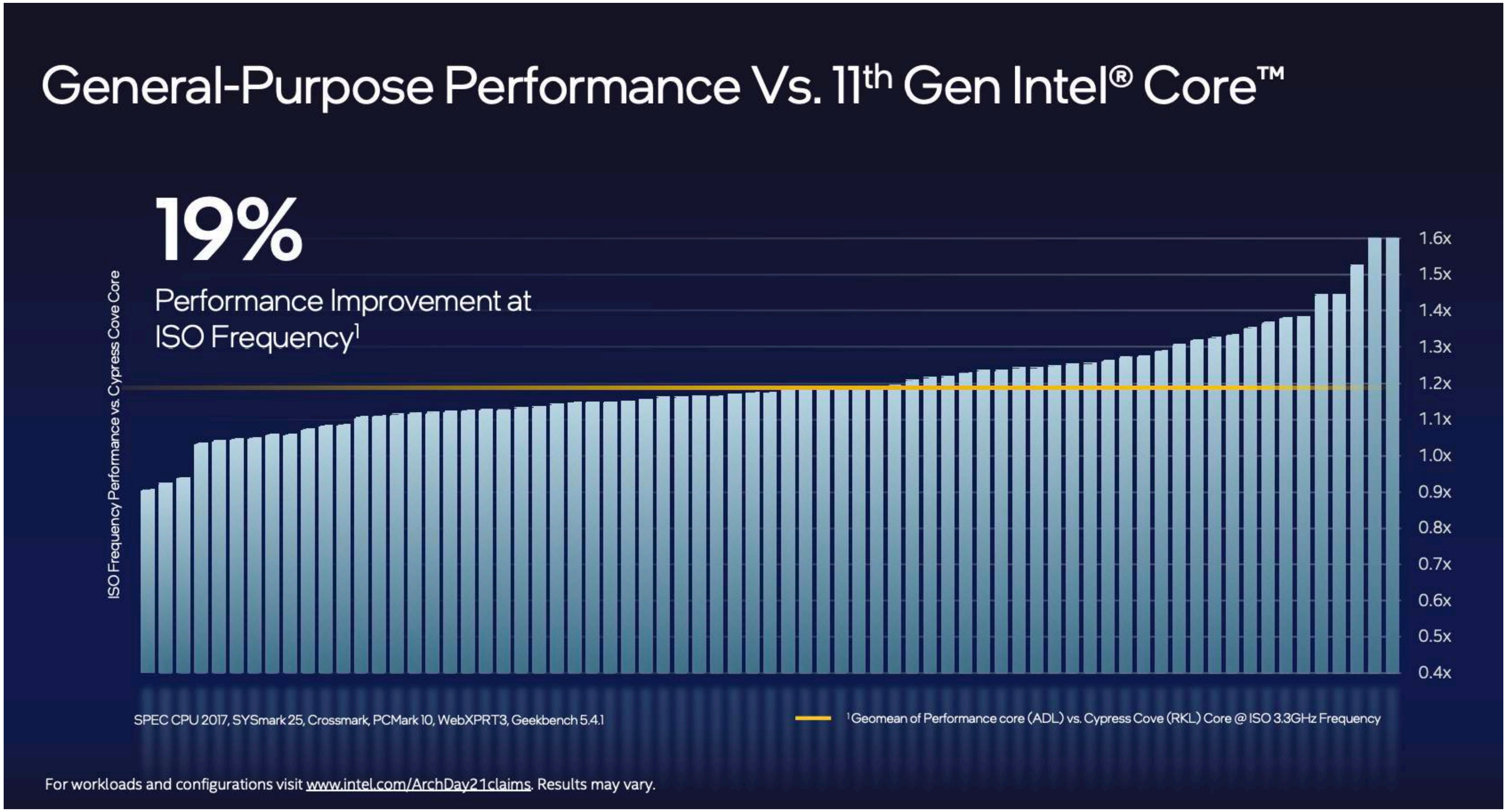


Benchmarks[基准测试集]

- SPEC: Standard Performance Evaluation Corporation
- PARSEC: Princeton Application Repository for Shared Memory Computers
- Rodinia: GPGPU applications
- HPL: a High-Performance Linpack benchmark implementation
- MLPerf: a suite of performance benchmarks that cover a range of leading AI workloads widely in use
- MediaBench: Multimedia and embedded applications
- Transaction processing: TPC-C, SPECjbb
- EEMBC: embedded microprocessor benchmark consortium

Benchmarks (cont.)

- Example: performance of Intel's newest CPU (2021)

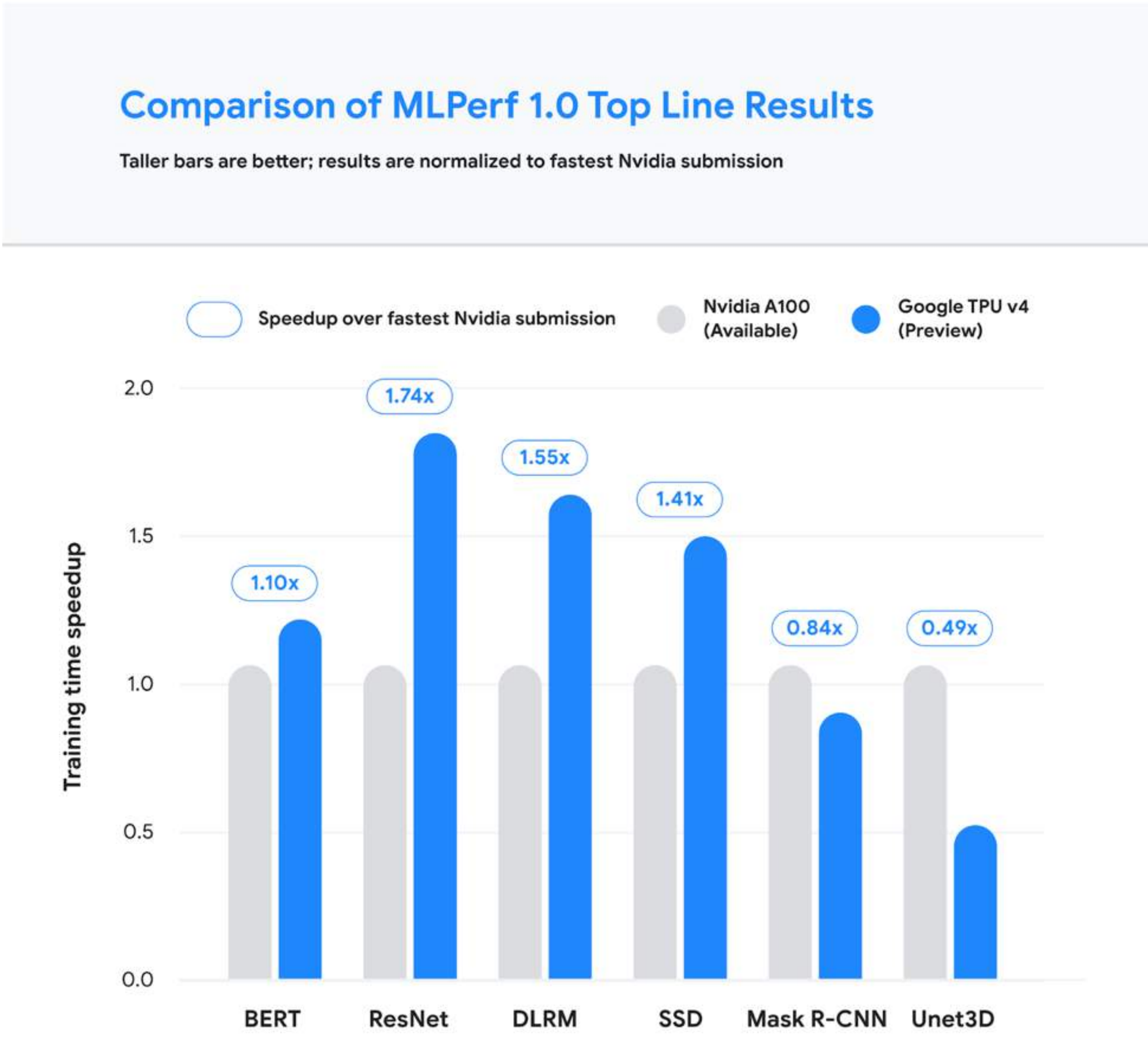


https://download.intel.com/newsroom/2021/client-computing/intel-architecture-day-2021-presentation.pdf

Benchmarks (cont.)

- MLPerf
  - A broad ML benchmark suite for measuring performance of ML software frameworks, ML hardware accelerators, and ML cloud platforms

MLPerf


Comparison of MLPerf 1.0 Top Line Results
Taller bars are better; results are normalized to fastest Nvidia submission
Speedup over fastest Nvidia submission
Nvidia A100 (Available)
Google TPU v4 (Preview)
Training time speedup
1.10x
1.74x
1.55x
1.41x
0.84x
0.49x
BERT
ResNet
DLRM
SSD
Mask R-CNN
Unet3D