



《计算机系统结构》课程直播

2020. 3.3

请将ZOOM名称改为“姓名”；

听不到声音请及时调试声音设备；签到将在课间休息进行

通知

- 课间还可以签到、但有课堂练习参与统计
- Canvas: 正版电子书链接
- 爱课程spoc注册，注册方式：Canvas文件、QQ群文件
- 爱课程spoc 第一次练习截止期
- 本次无书面作业，请做好课堂笔记.

第一章 概述

▼ 第一章 单元测验：计算机系统概述 截止时间：2020年03月10日 16:00

[前往测验](#)

计算机系统概述：单元测验。本次测验由10道选择题组成。

截止时间 2020年03月10日 16:00
请务必在截止时间之前提交，截止时间后的提交不再计分

有效分数 0.00/10.00
你的每一次测验系统都将为你计分，并提取最高得分作为你的有效分数

有效提交次数 0/3

讲课内容

1

计算机系统结构的发展趋势

Trends in Computer Architecture

2

性能评价指标

Performance Metrics

3

性能模型

Performance Models



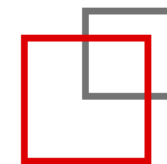
PART 01

计算机系统结构的发展趋势

Trends in computer architecture



上海交通大学
SHANGHAI JIAO TONG UNIVERSITY



What is architecture



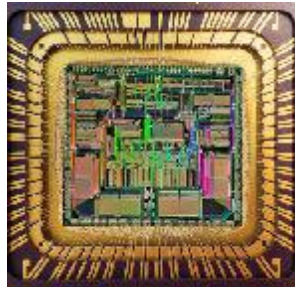
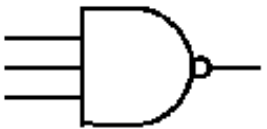
- Architecture
 - Science and art of **interconnecting** building materials to construct various buildings, **subject to constraints**
 - – Materials: brick, concrete, glass, etc.
 - – Buildings: house, office, auditorium, etc.
 - – Constraints: cost, safety, time, etc.



What is computer architecture



- Computer Architecture
 - Science and art of interconnecting hardware components to create computers, subject to constraints
 - – Hardware components: circuits, gates, chips, etc.
 - – Computers: desktop, server, mobile phone, etc.
 - – Constraints: performance, energy, cost, etc.



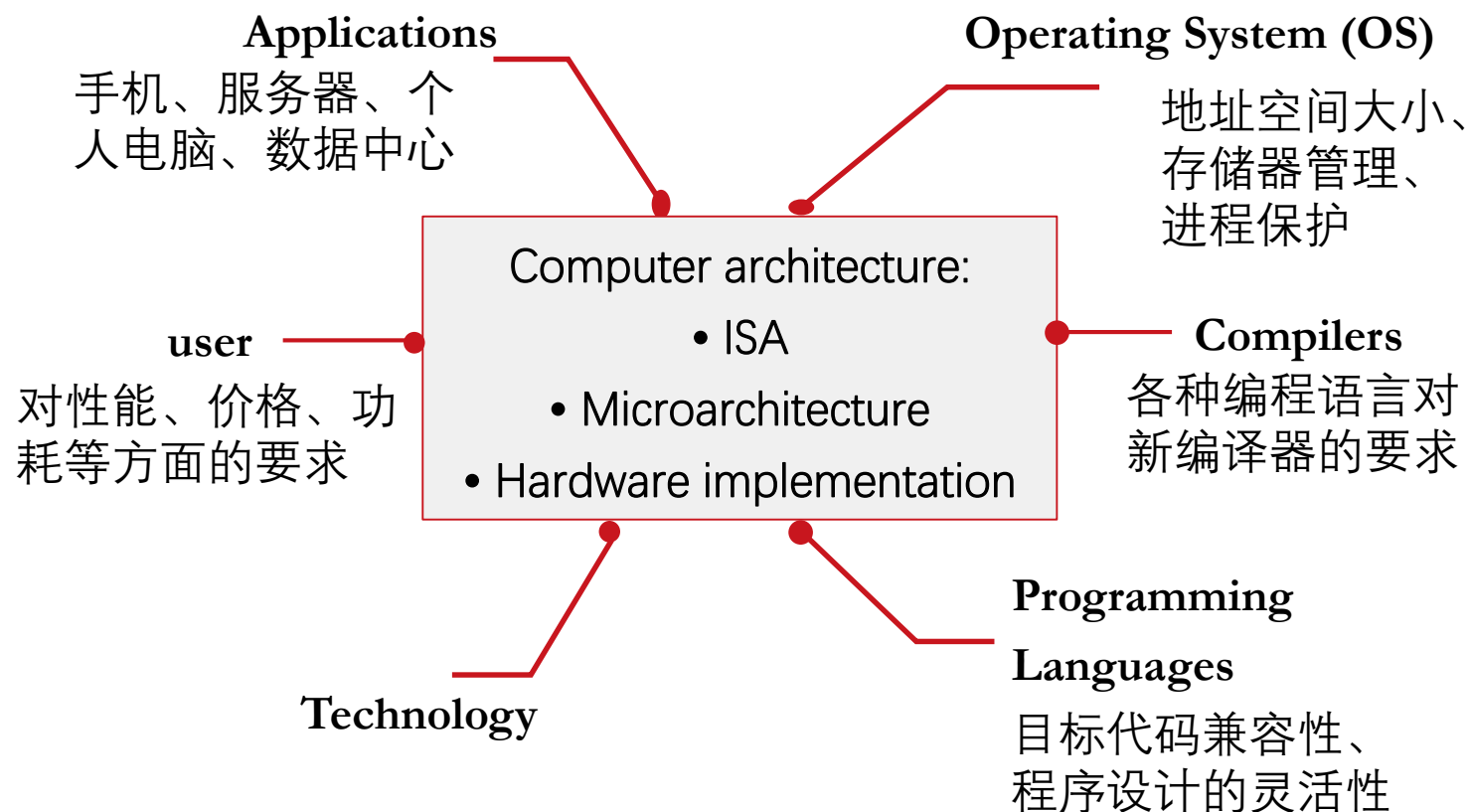
Three Aspects of Computer Architecture

- **Instruction Set Architecture (ISA)**
 - – Programmer/compiler view
 - – Instructions visible to the (system) programmer
 - – Opcode, architectural registers, address translation, etc.
- **Microarchitecture**
 - – Processor designer view
 - – Logical organization that implements the ISA
 - – Pipelining, functional units, caches, registers, etc.
- **Circuits**
 - – Circuit/chip designer view
 - – Detailed logic design and packaging technology
 - – Gates, cells, CMOS process, etc.

Course
focus

What is Genuine CA?

CA: Designing the Organization and Hardware to Meet Goals and Functional Requirements - Hennesy and Patterson, CAAQA 5th ed.

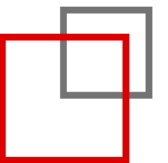




Technology and Trends



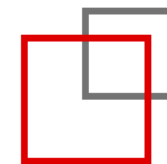
上海交通大学
SHANGHAI JIAO TONG UNIVERSITY



提问：摩尔定律？



上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

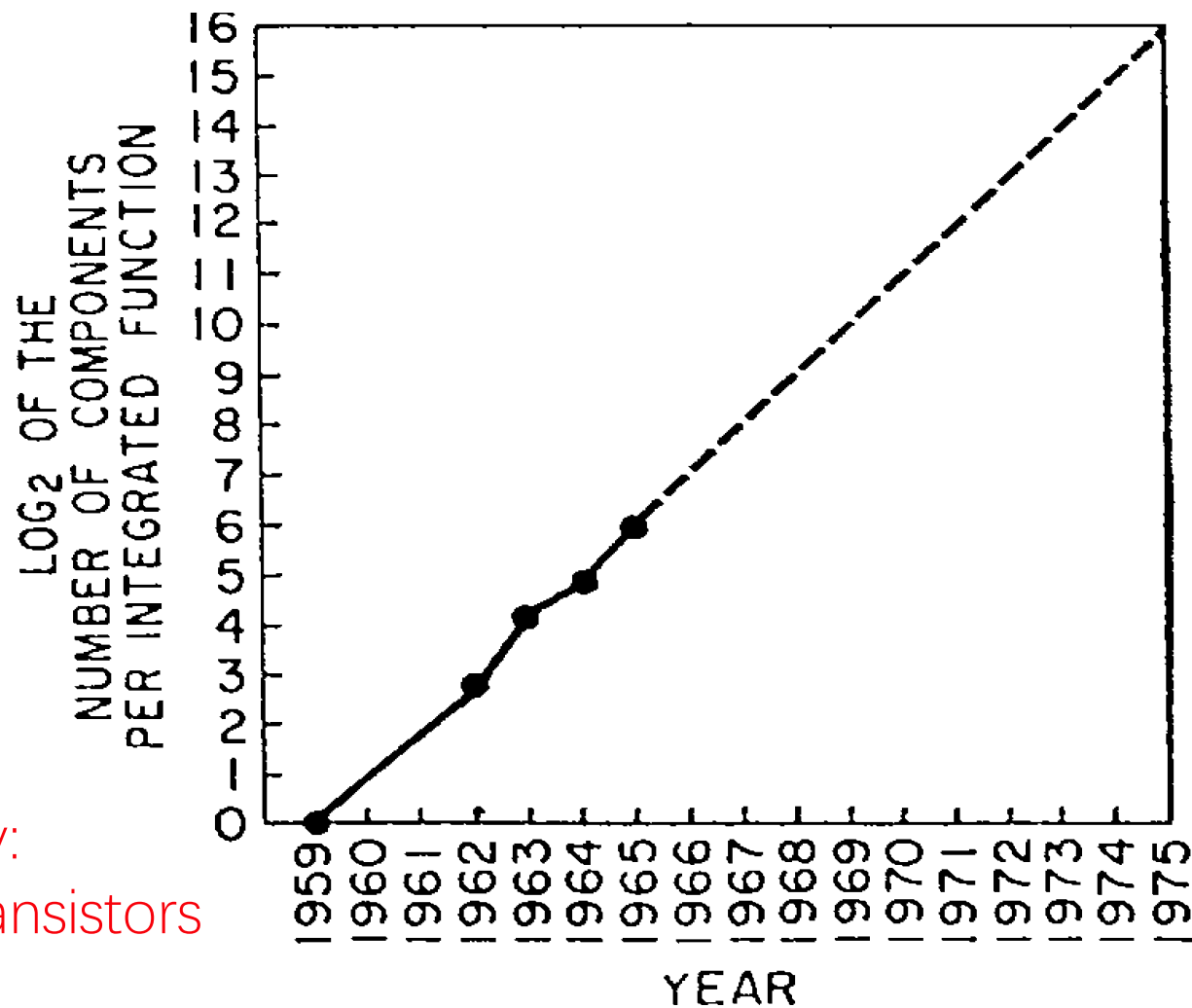


“Technology”

- Basic element
 - Solid-state **transistor** (i.e., electrical switch)
 - Building block of **integrated circuits (ICs)**
- What’s so great about ICs? Everything
 - High performance, high reliability, low cost, low power
- Several kinds of integrated circuit families
 - **SRAM/logic**: optimized for speed (used for processors)
 - **DRAM**: optimized for density, cost, power (used for memory)
 - **Flash**: optimized for density, cost (used for storage)
- Non-transistor storage and inter-connection technologies
 - Magnetic disks, optical storage, ethernet, fiber optics, wireless



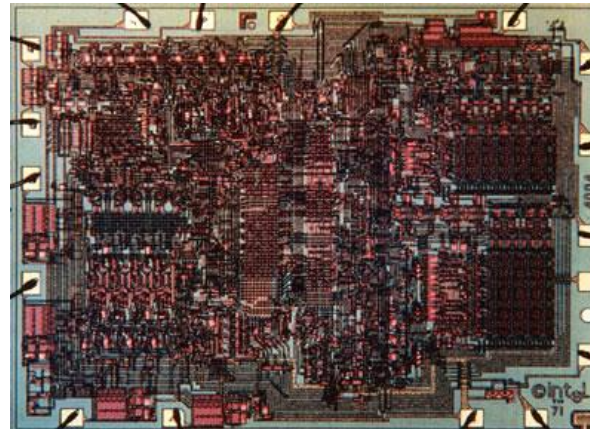
Moore's Law - 1965



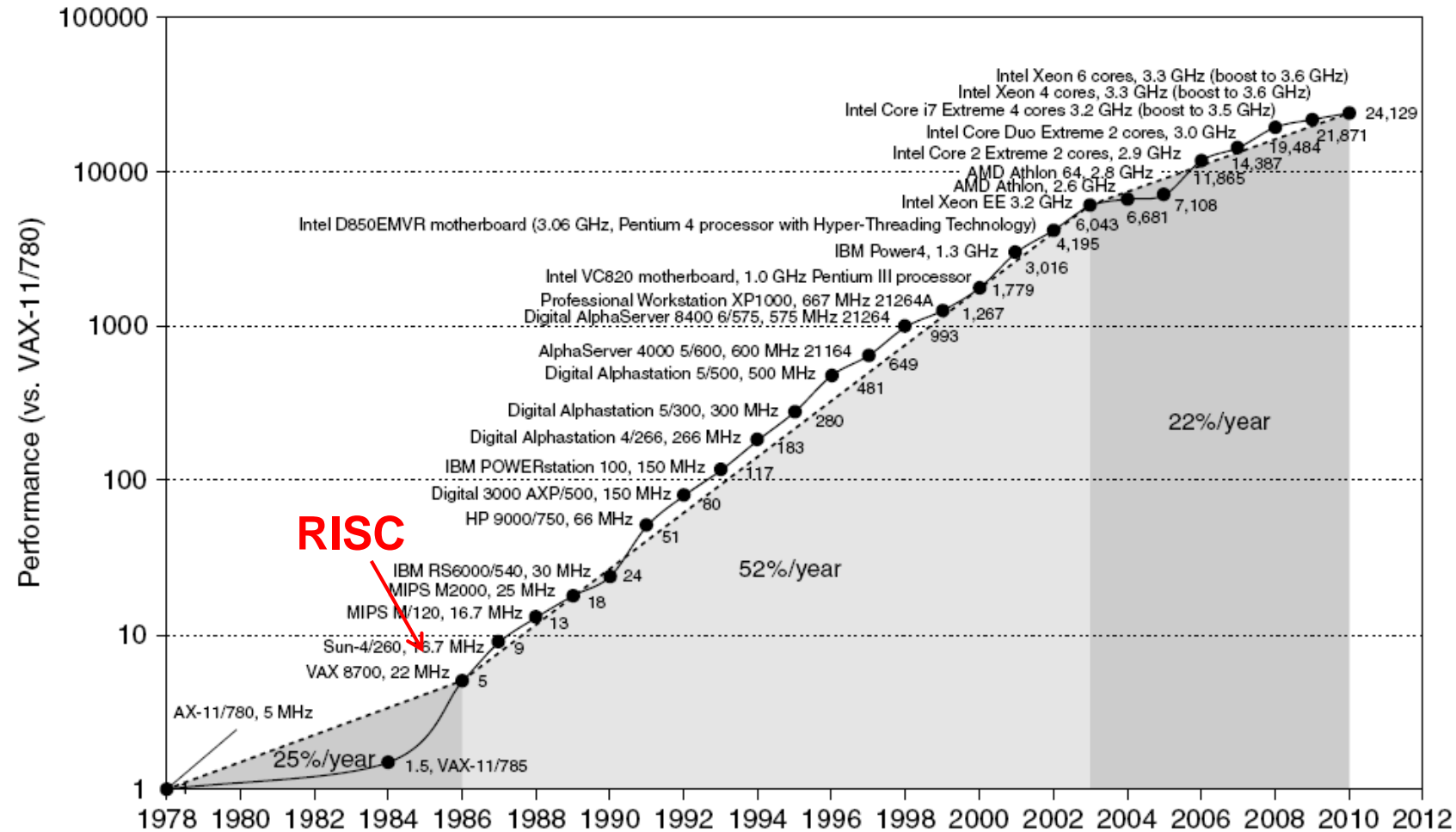
Today:
 2^{33} transistors

Revolution I: The Microprocessor

- Microprocessor revolution
 - One significant technology threshold was crossed in 1970s
 - Enough transistors (~25K) to put a 16-bit processor on one chip
 - Huge performance advantages: fewer slow chip-crossings
 - Even bigger cost advantages: one “stamped-out” component
- First Microprocessor: Intel 4004 (1971)
 - Application: calculators
 - Technology: 10,000 nm
 - 2300 transistors、13 mm²
 - 108 KHz、12 Volts
 - 4-bit data

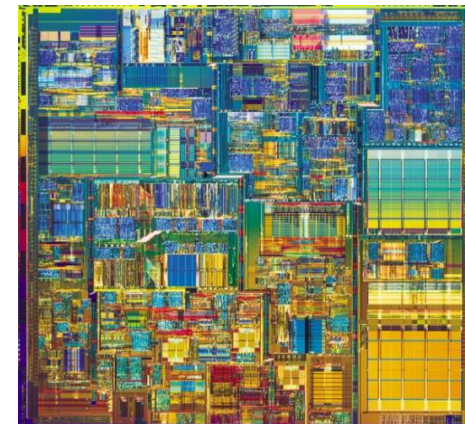


Processor Performance

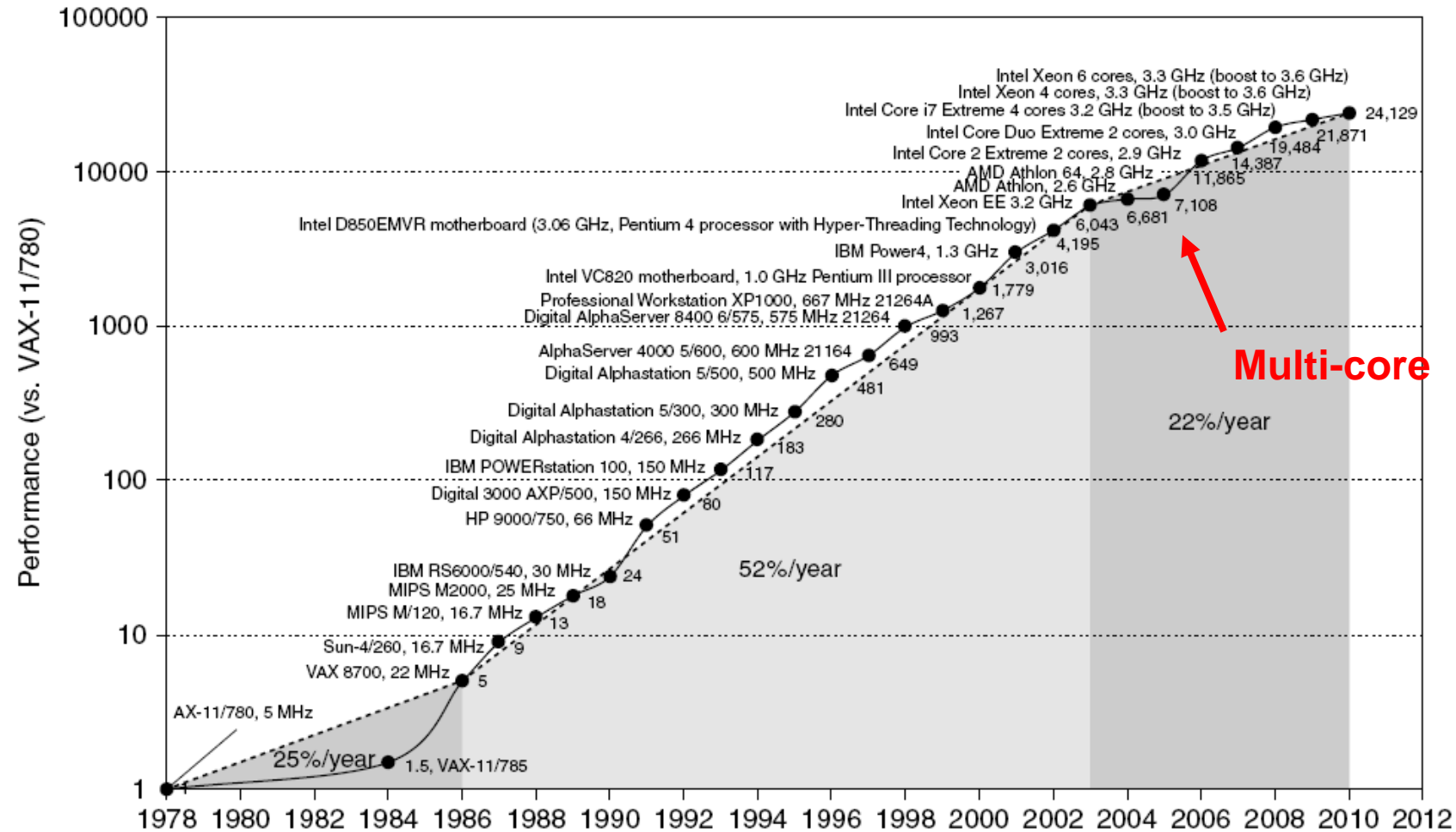


Revolution II: Implicit Parallelism

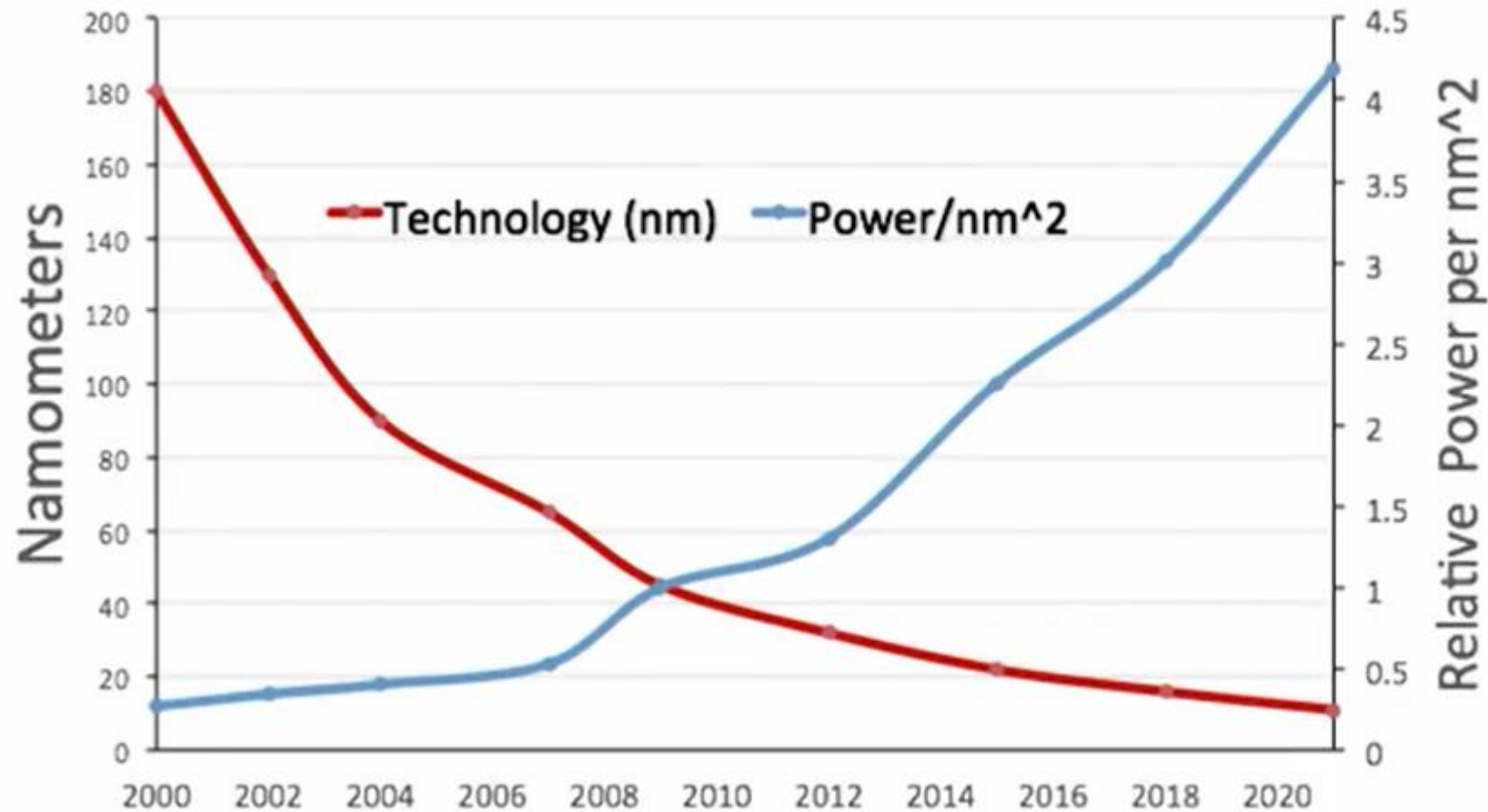
- Then to **extract implicit instruction-level parallelism**
 - Hardware provides parallel resources, figures out how to use them
 - Initially using pipelining ...cache... and integrated floating-point
 - Then deeper pipelines and branch s speculation
 - Then multiple instructions per cycle (superscalar)
 - Then dynamic scheduling (out-of-order execution)
 - We will talk about these things
- Intel Pentium4 (2003)
 - Application: desktop/server
 - Technology: 90nm (1/100x)、55M transistors (20,000x)
 - 101 mm² (10x)、3.4 GHz (10,000x)
 - 1.2 Volts (1/10x)、32/64-bit data (16x)
 - 22-stage pipelined datapath、3 instructions per cycle (superscalar)
 - Two levels of on-chip cache



Processor Performance



Technology & Power: Dennard Scaling

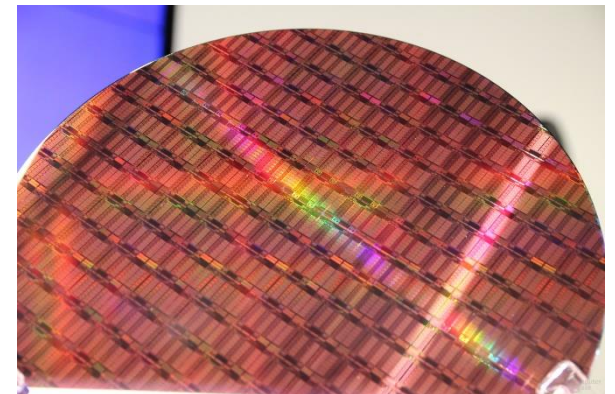


Power consumption based on models in "[Dark Silicon and the End of Multicore Scaling](#)," Hadi Esmaeilzadeh, ISCA, 2011

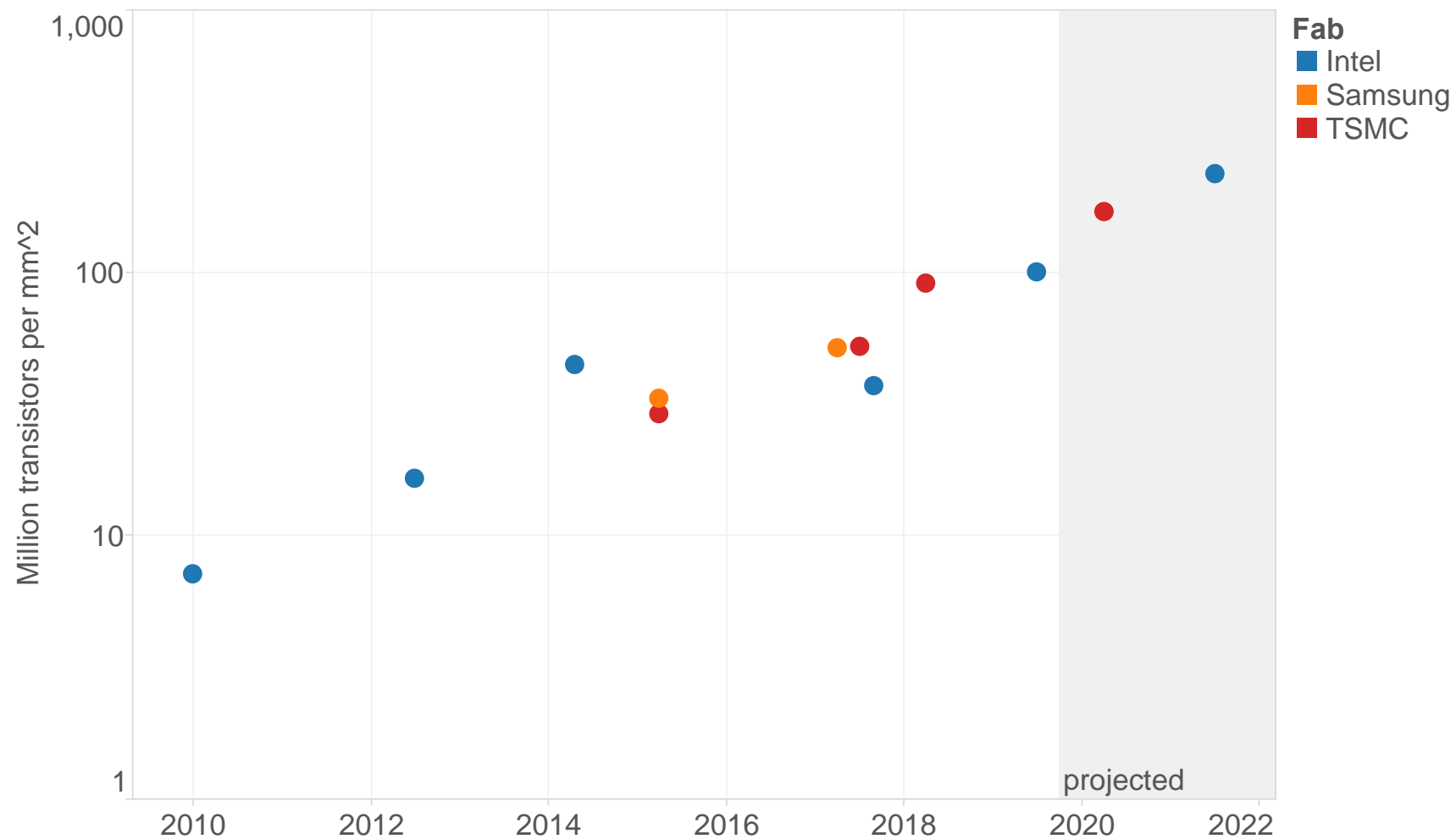
Energy scaling for fixed task is better, since more and faster transistors

Revolution III: Explicit Parallelism

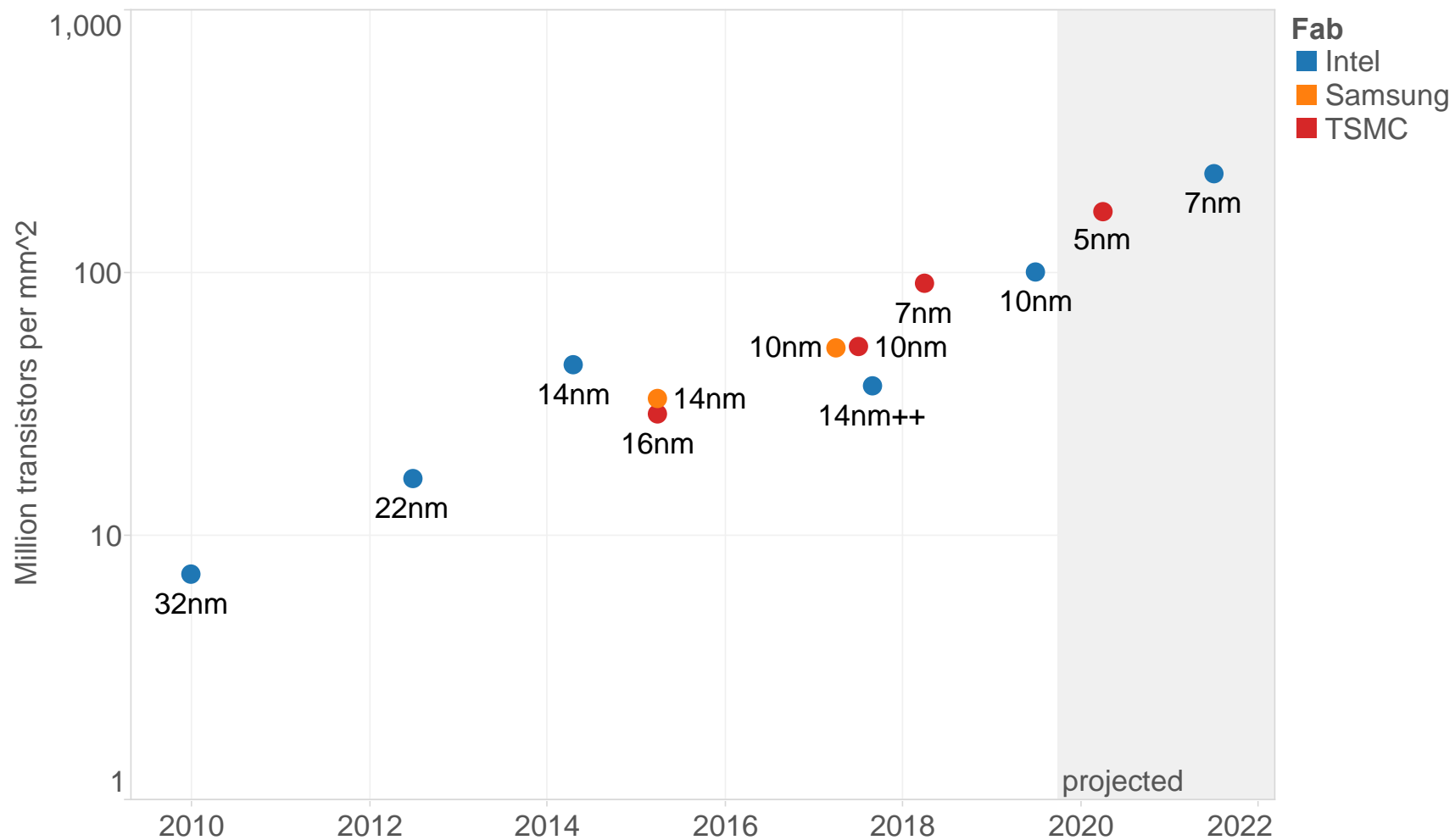
- Then to support **explicit data & thread level parallelism**
 - Hardware provides parallel resources, software specifies usage
 - Why? diminishing returns on instruction-level-parallelism
 - First using (subword) vector instructions..., Intel's SSE
 - One instruction does four parallel multiplies
 - ... and general support for multi-threaded programs
 - Then using support for multiple concurrent threads on chip
 - First with single-core multi-threading, now with multi-core
- Intel Xeon E5-2699 V4 (2016)
 - Application: server
 - Technology: 14nm (16% of P4)、7.2B transistors (130x)、456 mm² (4.5x)
 - 2.4 to 3.6 Ghz (~1x)、1.8 Volts (~1x)
 - 256-bit data (2x)、14-stage pipelined datapath (0.5x)
 - 4 instructions per cycle (1x)、Three levels of on-chip cache
 - data-parallel vector (SIMD) instructions, hyperthreading
 - 22-core multicore (22x)



Moore's Law today

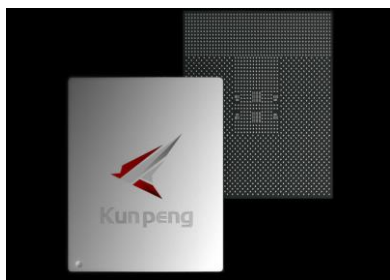


Moore's Law today



State-of-the-art - 2019

华为鲲鹏920处理器



- 32/48/64 cores
- 20B transistors,
- 7nm, 400 mm²
- ~200W @ 2.6GHz



iPhone11-A13



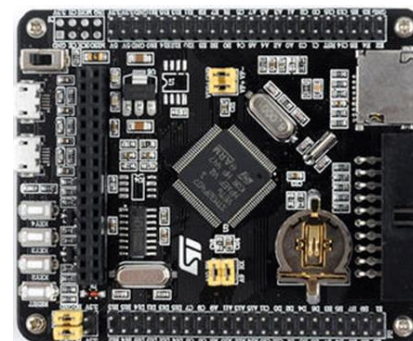
- 6 CPU cores + 4 GPU cores + 8 "Neural Engine"
- 8.5B transistors, 7nm,
- 98.48 mm²
- 2W @ 2.66GHz



ARM Cortex-M4

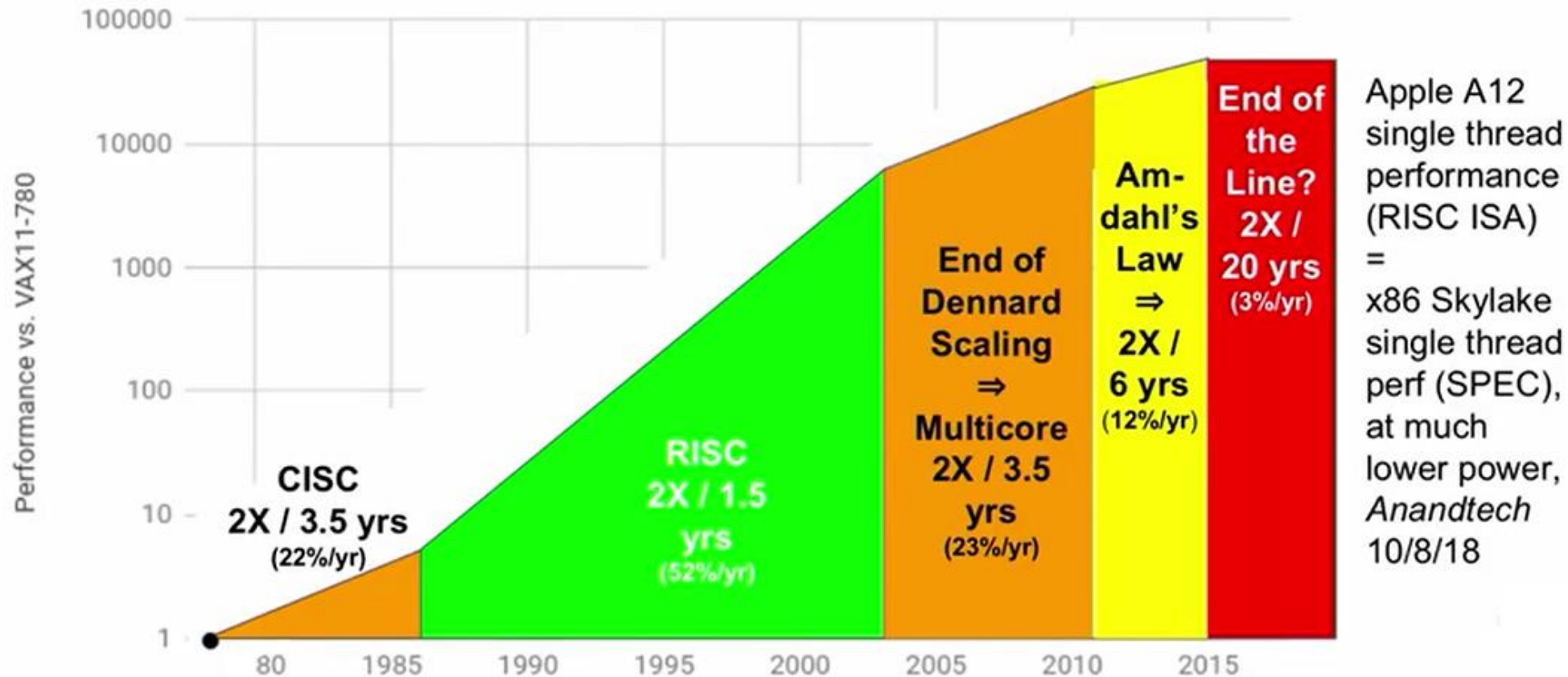


- 3-stage pipeline
- Floating Point Unit
- < 1 mm²
- 4mW @ 100MHz



End of Growth of Single Program Speed?

40 years of Processor Performance

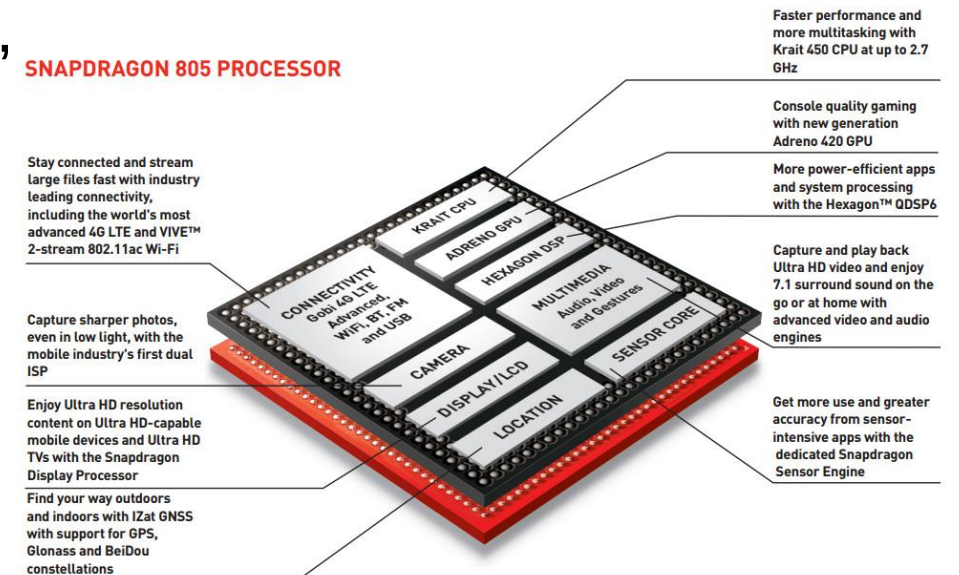


Based on SPECintCPU. Source: John Hennessy and David Patterson, Computer Architecture: A Quantitative Approach, 6/e. 2018

Revolution IV: Heterogeneous Processing

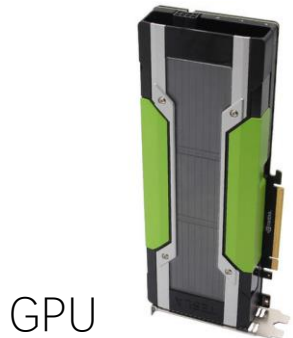
- Lots of stuff on the chip beyond just CPUs
 - Graphics Processing Units (GPUs)
 - throughput-oriented specialized multicore processors
 - good for gaming, machine learning, computer vision, ...
 - Special-purpose logic
 - media codecs, radios, encryption, compression,
 - machine learning
- Excellent energy efficiency and performance

SNAPDRAGON 805 PROCESSOR



Current trends in Computer Architecture

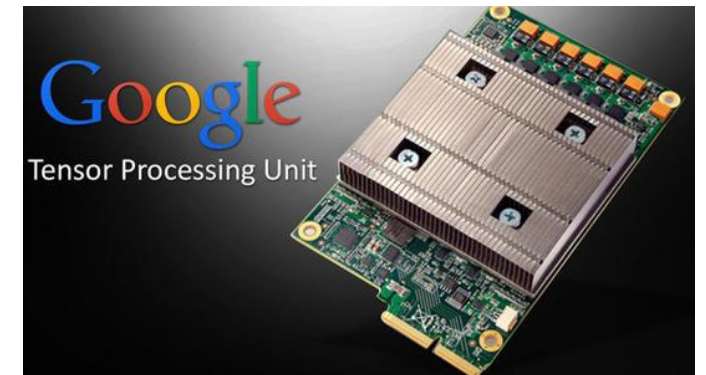
- Very complex processor designs
- Parallelism at the chip level
- Power-conscious designs
- Specialization: domain-specific processors
- Open-source hardware: RISC-V
- Security



GPU



Vision
proc.





Why study CA?

- Technology is always changing
 - CA should adapt
 - E.g., 10B vs 10K transistor budget
- Requirements are always changing
 - Speed, power, cost, reliability etc.
 - E.g., emergence of smartphones
- Understand computer performance
 - Essential for development of high-performance and/or energy-efficient software

互动

- 选出你最感兴趣的/觉得最神秘的内容？ 可以多选

1

GPU



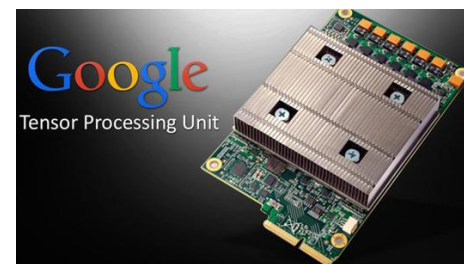
Vision
proc.

2



TPU

3



4

Warehouse computing,
cloud computing



5



6



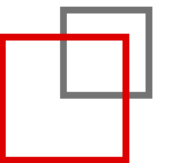


PART 02

Performance Metrics



上海交通大学
SHANGHAI JIAO TONG UNIVERSITY





Performance Metrics

- Metrics:
- **Execution time, response time:**
 - overall time for a given computation (e.g. full program execution, one transaction)
- **Latency:** time to complete a given task、 a fixed task
 - (e.g. memory latency, I/O latency, instruction latency)
- **Bandwidth, throughput:** rate of completion of tasks
 - number of tasks per unit time
 - (e.g. memory bandwidth, transactions per second, MIPS, FLOPS)



Performance: Latency vs. Throughput



- Latency (response time, execution time):
- Throughput (bandwidth):
 - exploit parallelism for throughput, not latency
 - e.g. increase # of cores can increase throughput but not latency
- Choose definition of performance that matches your goals
- Example: move people 10 miles
 - Car: capacity = 5, speed = 60 miles/hour
 - Bus: capacity = 60, speed = 20 miles/hour
 - Latency: car = 10 min, bus = 30 min
 - **Throughput:** car = 15 PPH (count return trip), bus = 60 PPH

课本练习题

- 假设某个使用桌面客户端和远程服务器的应用受到网络性能的限制，那么对于下列方法，哪个是同时改进了吞吐率和响应时间的？
 - A. 在客户端和服务端之间增加一条额外的网络信道（现在有两条网络信道了）
 - B. 改进网络软件，从而减少网络通信延迟，但并不增加吞吐率
 - C. 增加计算机内存
 - D. 更换运算速度更快的处理器

课堂提问：

- Fastest way to send 10TB[💬](bytes) of data?
- (bandwidth: 1+ gbits /second)

Amazon Does This!

Available Internet Connection	Theoretical Min. Number of Days to Transfer 100TB at 80% Network Utilization	When to Consider AWS Snowball?
T3 (44.736Mbps)	269 days	2TB or more
100Mbps	120 days	5TB or more
1000Mbps	12 days	60TB or more

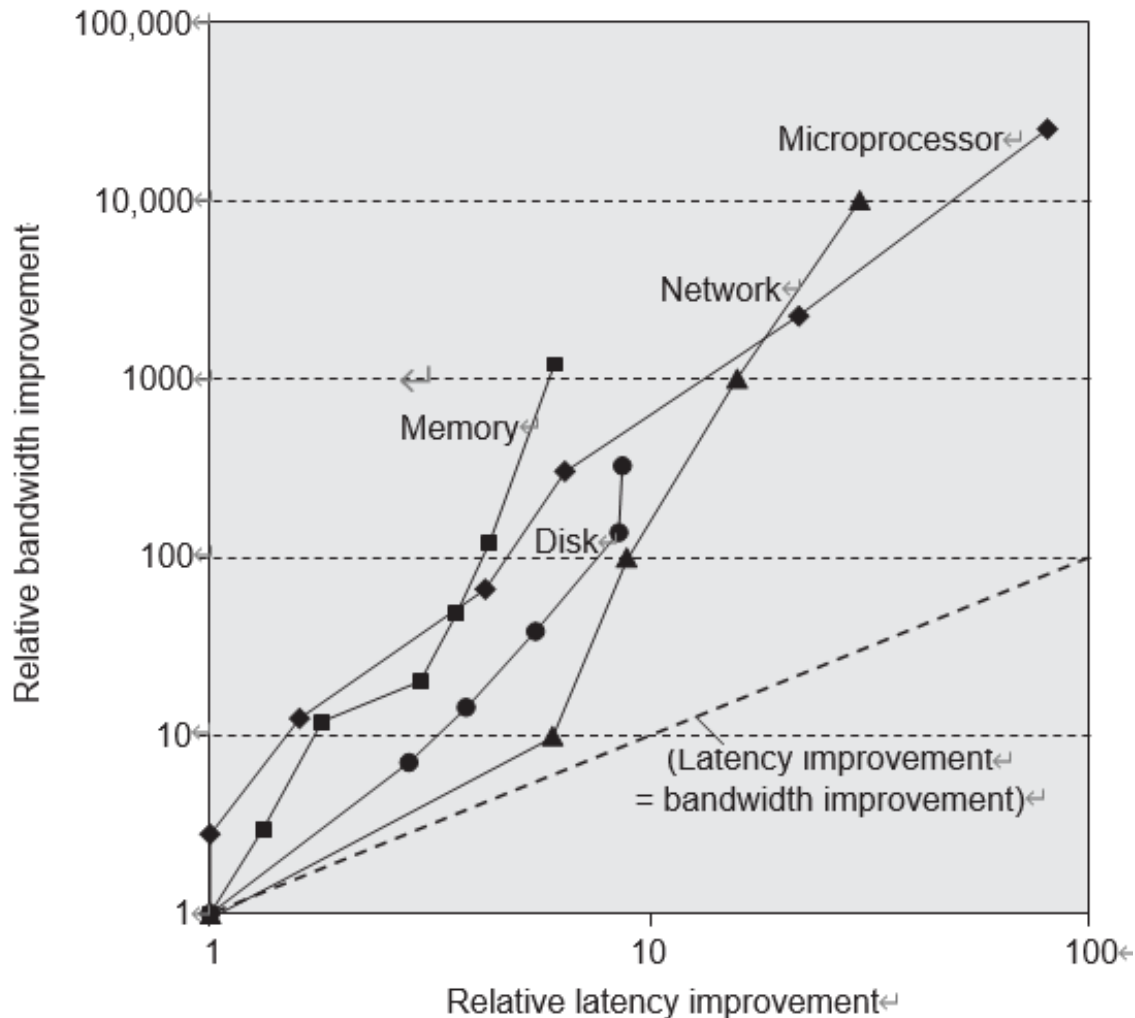


“Never underestimate the bandwidth of a station wagon full of tapes hurtling down the highway.”

-- Andrew Tanenbaum, *Computer Networks*, 4th ed., p. 91



Performance Trends: Bandwidth over Latency



Log-log plot of bandwidth and latency milestones

Bandwidth / throughput


- 10,000-25,000X improvement for processors
- 300-1200X improvement for memory and disks

Latency / response time

- 30-80X improvement for processors
- 6-8X improvement for memory and disks

Why?

Improving Performance: Principles of CA

- **Take advantage of parallelism (利用并行性)**
 - **System level:** multiple processors, multiple disks ,multiple chanel
 - **Processor level:** operate on multiple instructions at once (e.g., pipelining, superscalar issue)
 - **Circuit level:** operate on multiple bits at once (e.g., carry-lookahead ALU)
- Focus on the common case
 - Amdahl's law, CPU Performance equation
 - E.g. RISC design principle
- Principle of locality
 - – Spatial and Temporal Locality
 - – 90% of program executing in 10% of code
 - – E.g. Caches

Comparing Performance - Speedup

- Speedup of A over B
 - $X = \text{Latency}(B) / \text{Latency}(A)$ (divide by the faster)
 - $X = \text{Throughput}(A) / \text{Throughput}(B)$ (divide by the slower)
- A is X% faster than B if
 - $X = ((\text{Latency}(B) / \text{Latency}(A)) - 1) * 100$
 - $X = ((\text{Throughput}(A) / \text{Throughput}(B)) - 1) * 100$
 - $\text{Latency}(A) = \text{Latency}(B) / (1 + (X/100))$
 - $\text{Throughput}(A) = \text{Throughput}(B) * (1 + (X/100))$
- Car/bus example
 - Latency? Car is 3 times (and 200%) faster than bus
 - Throughput? Bus is 4 times (and 300%) faster than car

Speedup and % Increase and Decrease

- Program A runs for 200 cycles
- Program B runs for 350 cycles
- Percent increase and decrease are **not the same**.
 - % increase: $((350 - 200)/200) * 100 = 75\%$
 - % decrease: $((350 - 200)/350) * 100 = 42.3\%$
- Speedup:
 - $350/200 = 1.75$ – Program A is 1.75x faster than program B
 - As a percentage: $(1.75 - 1) * 100 = 75\%$
- If program C is 1x faster than A, how many cycles does C run for? – 200 (the same as A)
 - What if C is 1.5x faster? 133 cycles (50% faster than A)

The CPU Performance Equation

$$\text{CPU time} = \text{IC} \times \text{CPI} \times \text{Clock time}$$

where: CPU time = execution time

IC = number of instructions executed (instruction count)

CPI = number of average clock cycles per instruction

Clock time = duration of processor clock

$$\text{CPU time} = \left(\sum_{i=1}^n \text{IC}_i \times \text{CPI}_i \right) \times \text{Clock time}$$

where: IC_i = IC for instruction (instruction group) i

CPI_i = CPI for instruction (instruction group) i



Improving CPU Performance

- IC:
 - Compiler optimizations (constant folding, constant propagation)
 - ISA (More complex instructions)
- CPI:
 - Microarchitecture (Pipelining, Out-of-order execution, branch prediction)
 - Compiler (Instruction scheduling)
 - ISA (Simpler instructions)
- Clock period:
 - Technology (Smaller transistors)
 - ISA (Simple instructions that can be easily decoded)
 - Microarchitecture (Simple architecture)



误区1: Frequency as a performance metric



- 1 Hertz = 1 cycle per second
1 Ghz is 1 cycle per nanosecond, 1 Ghz = 1000 Mhz
- ... but general public (mostly) also ignores CPI
- and instead equate clock frequency with performance!
- Which processor would you buy?
 - Processor A: CPI = 2, clock = 5 GHz
 - Processor B: CPI = 1, clock = 3 GHz
 - Probably A, but B is faster (assuming same ISA/compiler)
- Classic example
 - Core i7 faster clock-per-clock than Core 2
 - Same ISA and compiler!
- partial performance metrics are dangerous!

误区2：只看MIPS，忽略IC

- (Micro) architects often ignore dynamic instruction count
 - Typically work in one ISA/one compiler → treat it as fixed
- CPU performance equation becomes
 - Throughput: $\text{insn} / \text{second} = (\text{insn} / \text{cycle}) * (\text{cycles} / \text{second})$
 - **MIPS** (millions of instructions per second)
 - **Cycles / second**: clock frequency (in MHz)
 - Example: $\text{CPI} = 2, \text{clock} = 500 \text{ MHz} \rightarrow 0.5 * 500 \text{ MHz} = 250 \text{ MIPS}$
- Pitfall: may vary inversely with actual performance
 - Compiler removes insns, program gets faster, MIPS goes down
 - Work per instruction varies (e.g., multiply vs. add, FP vs. integer)

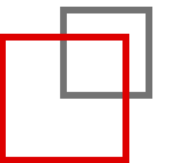


PART 03

Performance Laws



上海交通大学
SHANGHAI JIAO TONG UNIVERSITY



Amdahl's Law

$$\frac{1}{(1 - P) + \frac{P}{S}}$$

How much will an optimization improve performance?

P = proportion of running time affected by optimization

S = speedup

What if I speedup 25% of a program's execution by 2x?

1.14x speedup

What if I speedup 25% of a program's execution by ∞ ?

1.33x speedup

Build a balanced system

- Don't over-optimize 1% to the detriment of other 99%
- System performance often determined by slowest component

Amdahl's Law for Parallelization



$$\frac{1}{(1 - P) + \frac{P}{N}}$$

How much will parallelization improve performance?

P = proportion of parallel code
 N = threads

What is the max speedup for a program that's 10% serial?

What about 1% serial?



Increasing proportion of parallel code



- Amdahl's Law **requires *extremely* parallel code** to take advantage of large multiprocessors
- two approaches:
 - **strong scaling**: shrink the serial component
 - + same problem runs faster
 - becomes harder and harder to do
 - **weak scaling**: increase the problem size
 - + natural in many problem domains: internet systems, scientific computing, video games
 - doesn't work in other domains

How long am I going to be in this line?



Little's Law

$$L = \lambda W$$

L = items in the system : 同时被服务的用户数量

λ = average arrival rate: 用户到达系统的速度

W = average wait time: 每个用户在系统中驻留的时间

- Assumption:
 - system is in steady state : 稳定的系统
 - i.e., average arrival rate = average departure rate
- No assumptions about:
 - arrival/departure/wait time distribution or service order
- Works on **any** queuing system
- Works on **systems of systems**

Little's Law实例

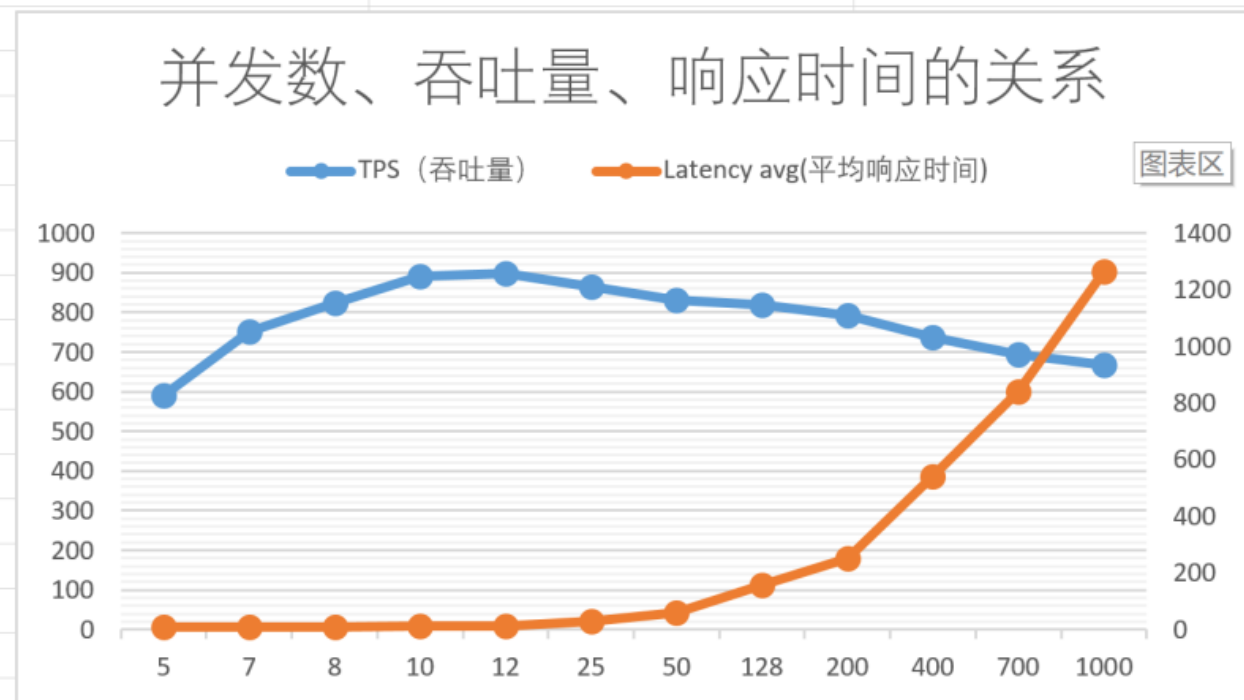
- 1.准备一个腾讯云mysql服务（5.6版本，配置为2核4G）
- 2.一台云服务器（腾讯云内网测试）
- 3.用sysbench(0.5版本)对mysql服务进行测试

并发数	TPS（吞吐量）	QPS	Latency avg(平均响应时间)	99th percentile	TPS（吞吐量）*Latency avg(平均响应时间)	错误率
5	591.1	8275.38	8.45	16.12	4.994795	0
7	750.69	10509.61	9.32	17.01	6.9964308	0
8	822.94	11521.09	9.72	22.69	7.9989768	0
10	891.59	12482.22	11.21	33.72	9.9947239	0
12	898.32	12576.43	13.35	44.17	11.992572	0
25	865.82	12121.53	28.86	78.6	24.9875652	0
50	831.16	11636.25	60.13	110.66	49.9776508	0
128	820.06	11480.87	156.06	297.92	127.9785636	0
200	793.64	11110.96	251.88	580.02	199.9020432	0
400	738.47	10338.64	541.28	1032.01	399.7190416	0
700	692.97	9701.58	839.32	1327.91	581.6235804	0
1000	667.59	9356.55	1264.6	2082.91	844.234314	0

<https://cloud.tencent.com/developer/news/462956>

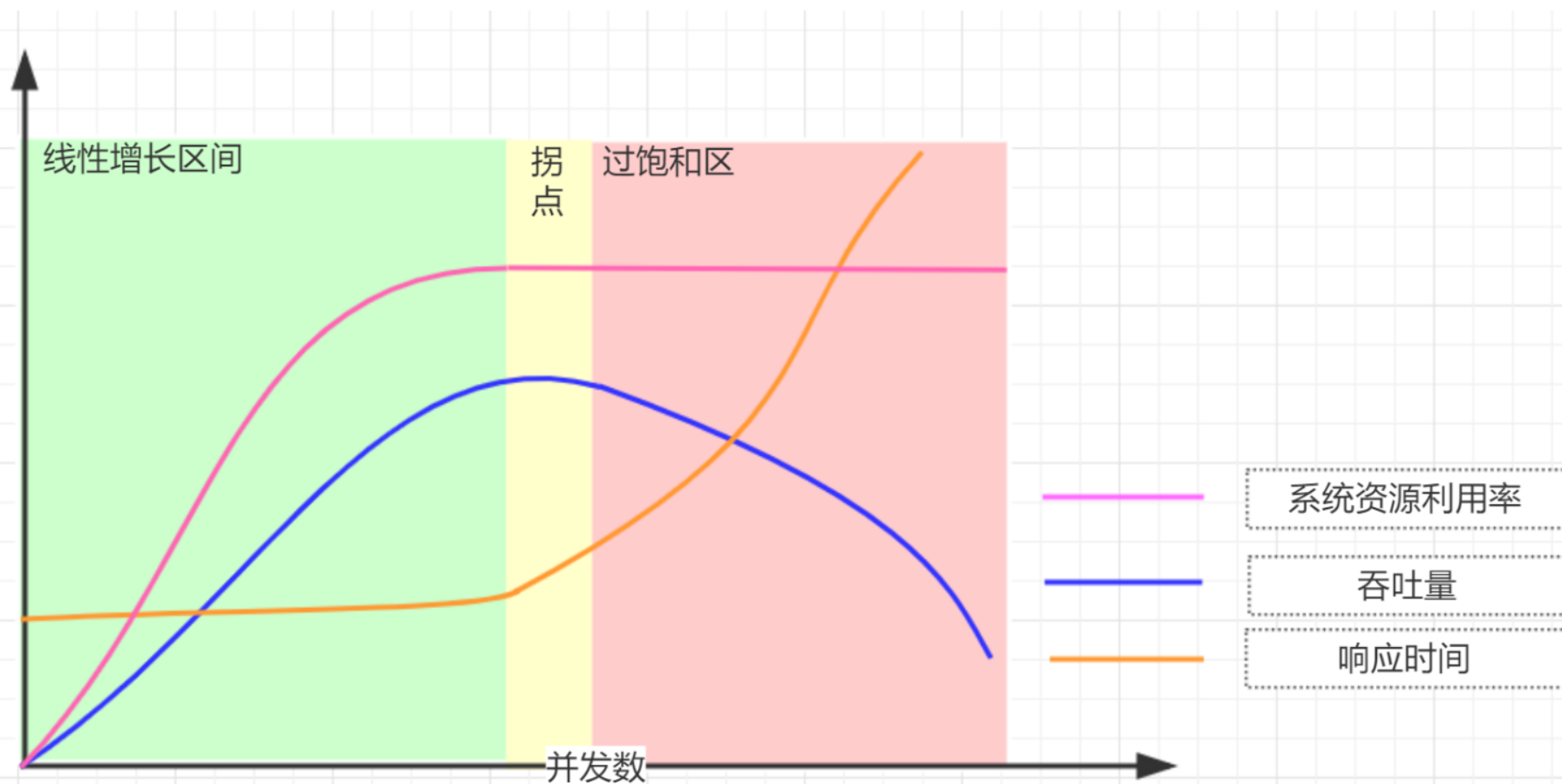
Little's Law实例

腾讯云 并发数	TPS (吞吐量)	Latency avg(平均响应时间)
5	591.1	8.45
7	750.69	9.32
8	822.94	9.72
10	891.59	11.21
12	898.32	13.35
25	865.82	28.86
50	831.16	60.13
128	820.06	156.06
200	793.64	251.88
400	738.47	541.28
700	692.97	839.32
1000	667.59	1264.6



<https://cloud.tencent.com/developer/news/462956>

Little's Law实例





Little's Law for Computing Systems

- Only need to measure two of L , λ and W
- Describes how to meet performance requirements
 - e.g., to get high throughput (λ), we need either:
 - low latency per request (small W 低延迟)
 - service requests in parallel (large L 并发数)
- Addresses many computer performance questions
 - sizing queue of L1, L2, L3 misses
 - sizing queue of outstanding network requests for 1 machine
 - or the whole datacenter
 - calculating average latency for a design

The top of the slide features a decorative header with a red background. On the left, there is a small image of a traditional Chinese building with a tiled roof. The rest of the header is a solid red bar.

提问时间

课堂练习

问卷星链接

<https://ks.wjx.top/jq/61477362.aspx>





下一节

- 周四8: 00
- 整数的表示和运算
- 习题为主
- 请做好准备

再见

