Chapter 6

Parallel Processors from Client to Cloud

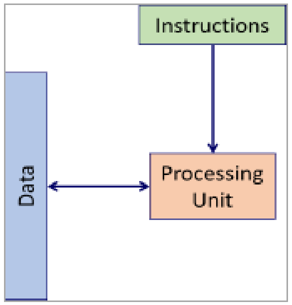
Introduction

- Goal: connecting multiple computers to get higher performance
  - Multiprocessors
  - Scalability, availability, power efficiency

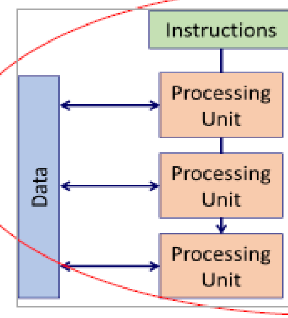
- Task-level (process-level) parallelism
  - High throughput for independent jobs
- Parallel processing program
  - Single program run on multiple processors
- Multicore microprocessors
  - Chips with multiple processors (cores), almost SMP
  - Moore's Law
- Reading : pp. 502~503

Instruction and Data Streams

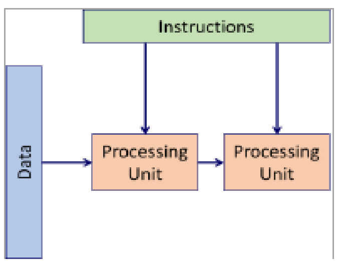
- An alternate classification ** Flynn's Taxonomy 1966*



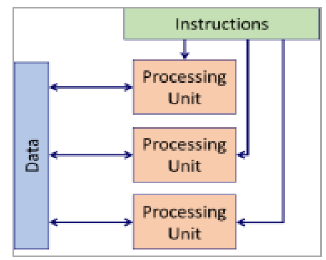
Single-Instruction
Single-Data
(Single-Core Processors)



Single-Instruction
Multi-Data
(GPUs, Intel SIMD Extensions)



Multi-Instruction
Single-Data
(Systolic Arrays,...)



Multi-Instruction
Multi-Data
(Parallel Processors)

- SIMD
  - All processors execute the same instruction at the same time
    - Each with different data address (elementwise), etc.
    - e.g., MMX and SSE instructions in x86
  - Simplifies synchronization
  - Works best for highly data-parallel applications

Hardware Multithreading

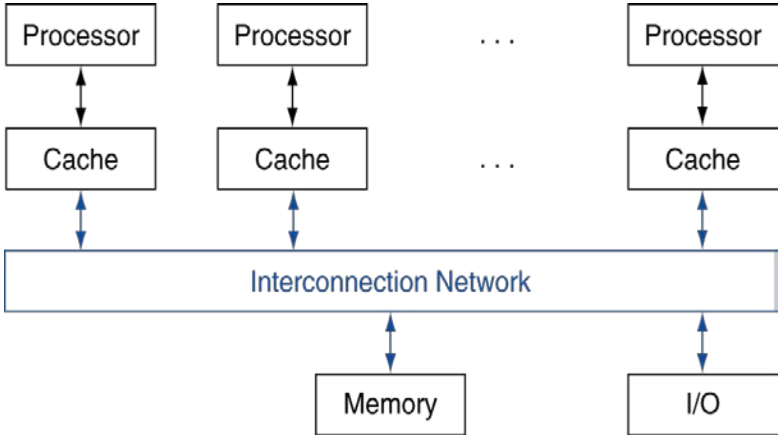
- Performing multiple threads of execution in parallel
  - Replicate registers, PC, etc.
  - Fast switching between threads

- Fine-grain multithreading
  - Switch threads after each cycle
  - Interleave instruction execution
  - If one thread stalls, others are executed

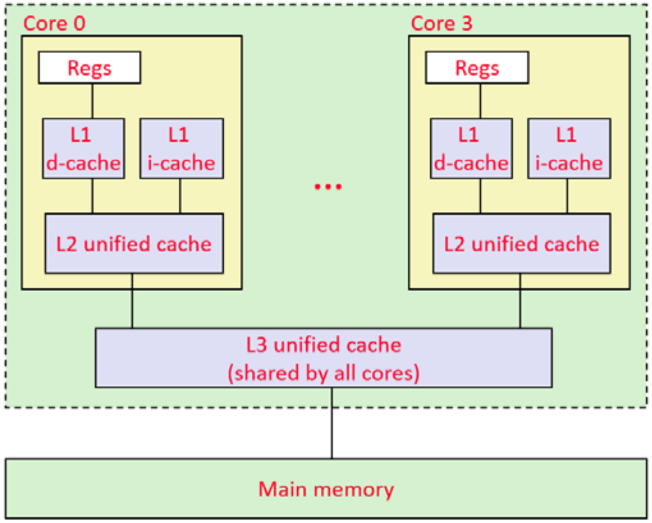
- Coarse-grain multithreading
  - Only switch on long stall (e.g., L2-cache miss)
  - Simplifies hardware, but doesn’t hide short stalls (eg, data hazards)

Shared Memory Multiprocessor (SMP)

- Hardware provides single physical address space for all processors



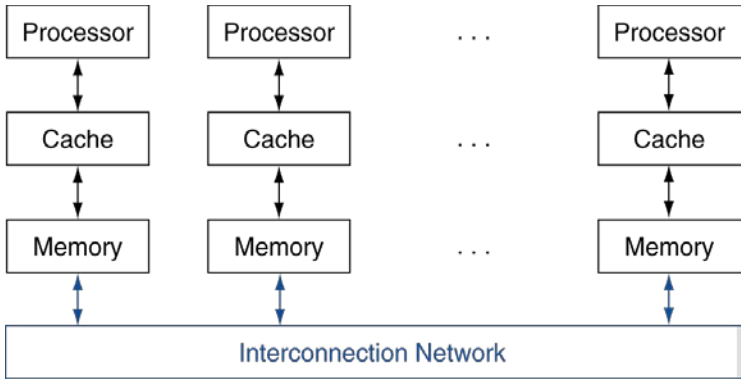
- Memory access time (pp. 520)
  - UMA (uniform) vs. NUMA (nonuniform)
- Synchronize shared variables using locks



Multicore?

Message Passing Multiprocessor

- Multiprocessor with multiple private address spaces



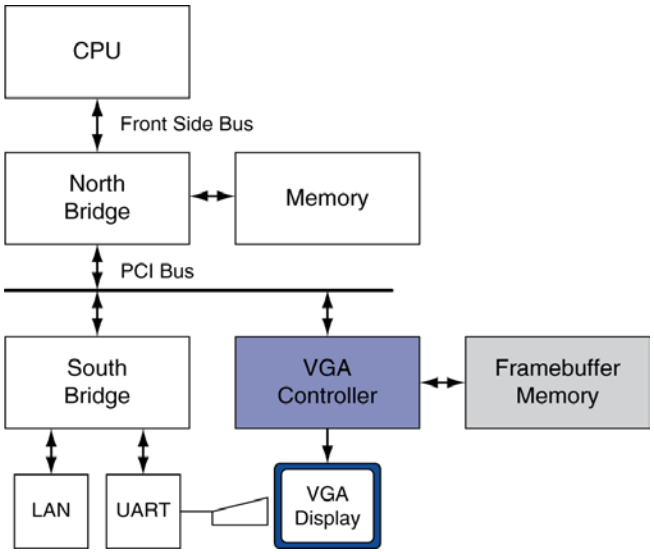
- Message passing
  - Send message routine
  - Receive message routine

Loosely Coupled Clusters

- Network of independent computers
  - Each has private memory and OS
  - Connected using I/O system
    - Ethernet/switch, Internet
- Suitable for applications with independent tasks
  - Web servers, databases, simulations, …
- High availability, scalable, affordable
- Problems
  - Administration cost (prefer virtual machines)
  - Low interconnect bandwidth
    - c.f. processor/memory bandwidth on an SMP

Graphics in the System (1)

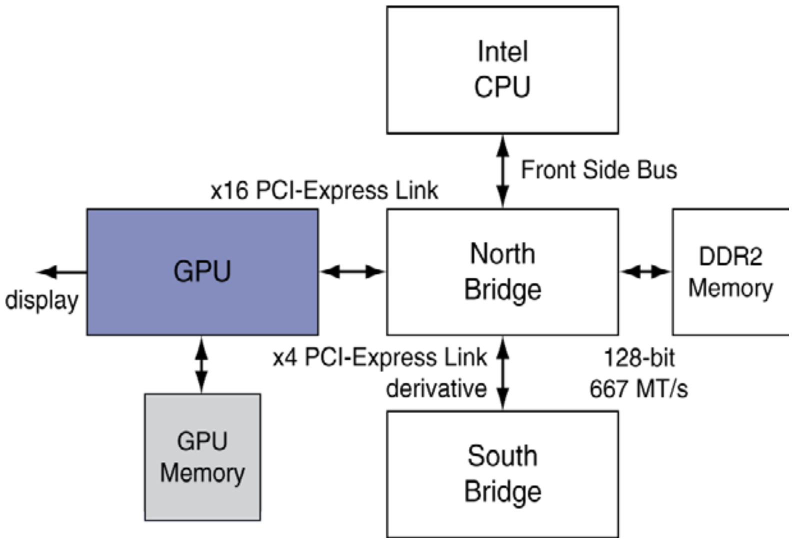
- Early video cards
  - Frame buffer memory with address generation for video output

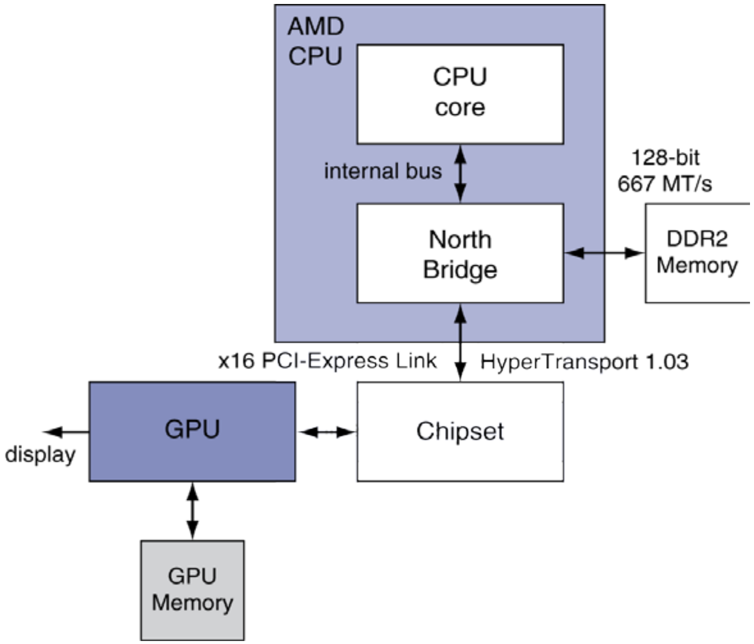


- 3D graphics processing
  - Originally high-end computers (e.g., SGI)
  - Moore's Law $\Rightarrow$ lower cost, higher density
  - 3D graphics cards for PCs and game consoles

Graphics in the System (2)

- Graphics Processing Units (GPU)
  - Processors oriented to 3D graphics tasks
  - Vertex/pixel processing, shading, texture mapping





- General-Purpose GPU
  - Thousands of simple cores with high floating-point processing capability
  - Very fast off-chip memory originally used for graphics processing

GPU Architectures

- Processing is highly data-parallel
  - GPUs are highly multithreaded
  - Use thread switching to hide memory latency
    - Less reliance on multi-level caches
  - Graphics memory is wide and high-bandwidth

- Trend toward general purpose GPUs
  - Heterogeneous CPU/GPU systems
  - CPU for sequential code, GPU for parallel code

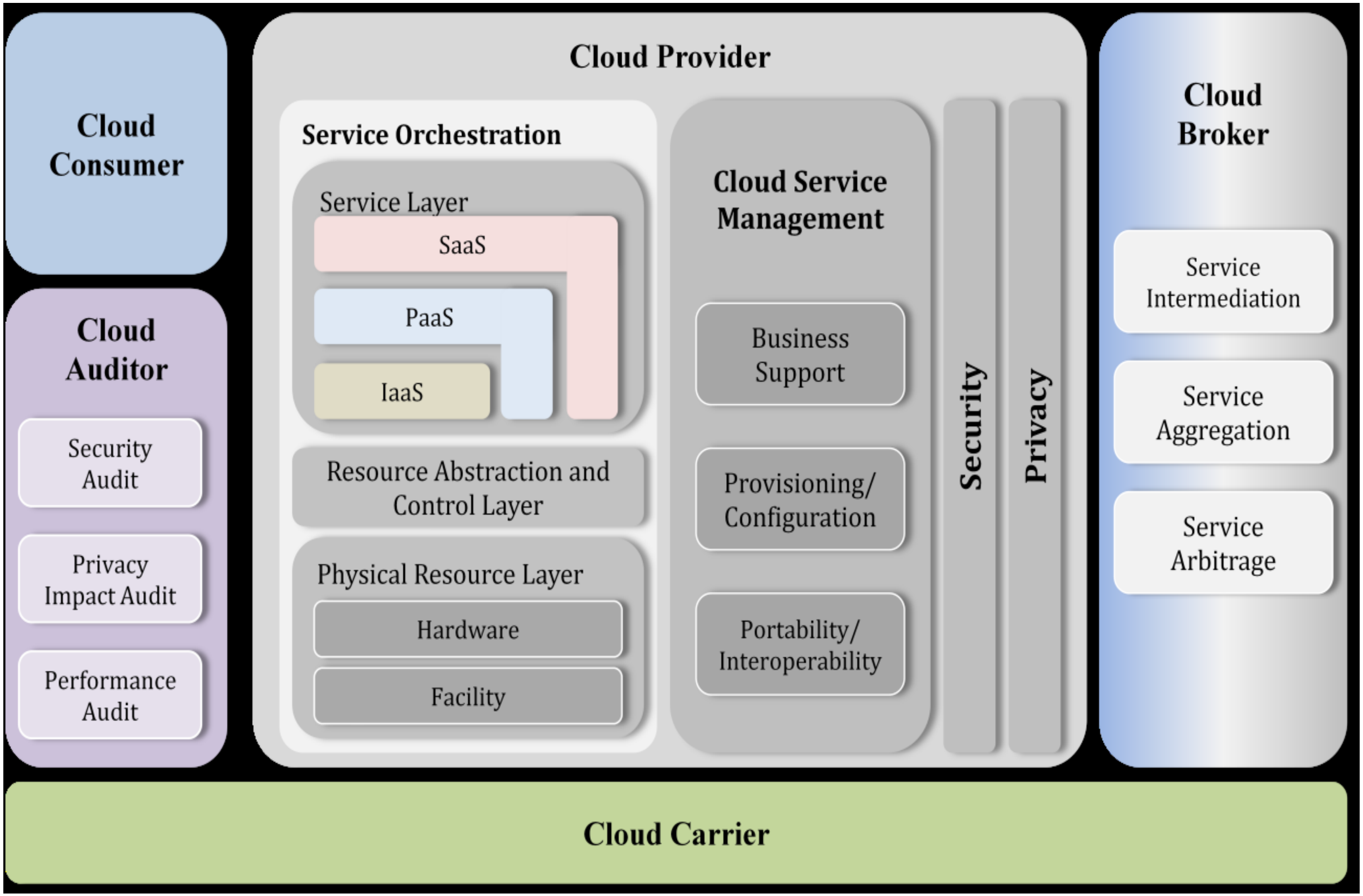
- Programming languages/APIs
  - DirectX, OpenGL
  - C for Graphics (Cg), High Level Shader Language (HLSL)
  - Compute Unified Device Architecture (CUDA)

Classifying GPUs

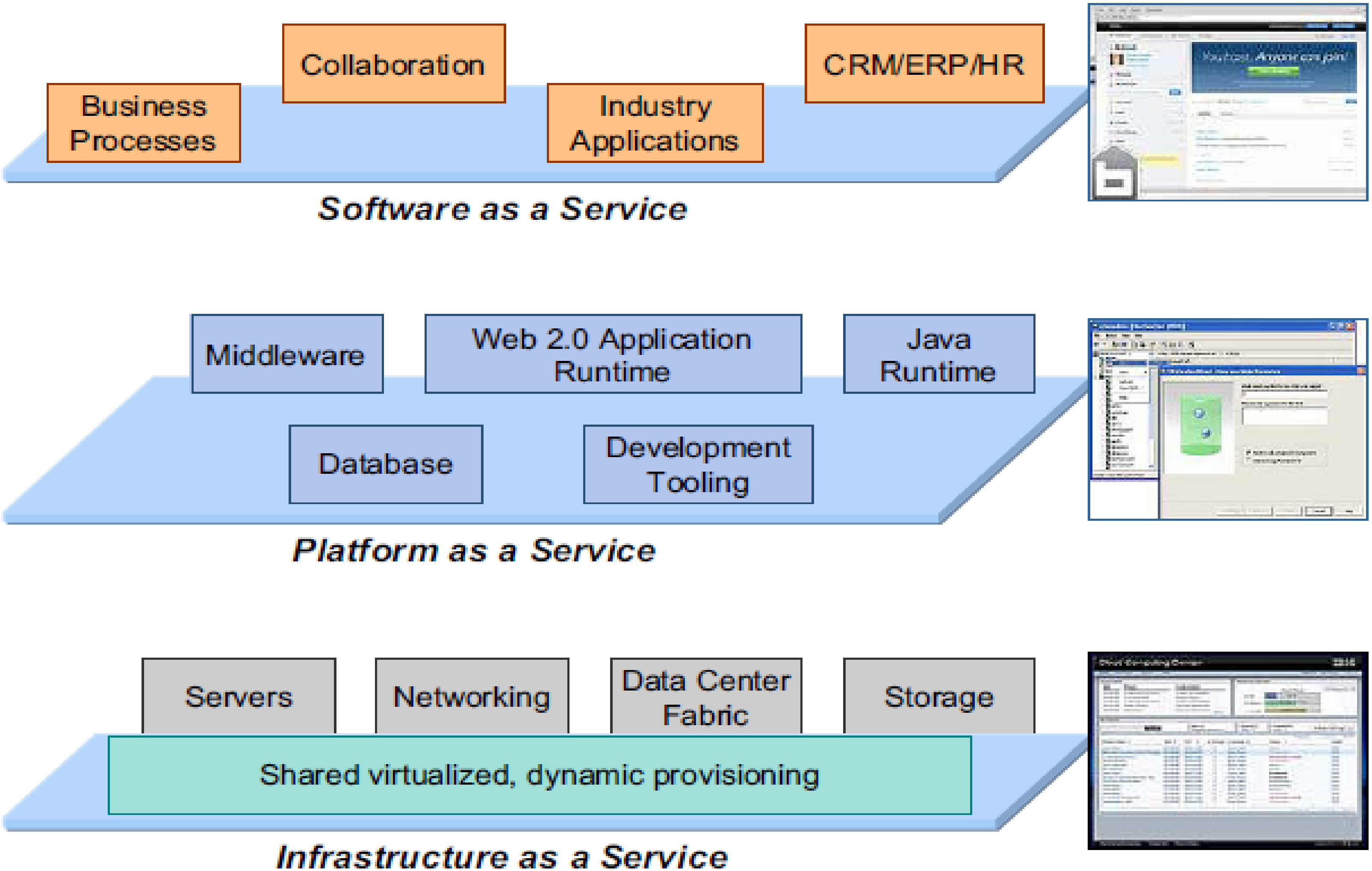
- How GPUs vary from CPUs : pp. 524 ~ 525

Feature	Multicore with SIMD	GPU
SIMD processors	4 to 8	8 to 16
SIMD lanes/processor	2 to 4	8 to 16
Multithreading hardware support for SIMD threads	2 to 4	16 to 32
Typical ratio of single precision to double-precision performance	2:1	2:1
Largest cache size	8 MB	0.75 MB
Size of memory address	64-bit	64-bit
Size of main memory	8 GB to 256 GB	4 GB to 6 GB
Memory protection at level of page	Yes	Yes
Demand paging	Yes	No
Integrated scalar processor/SIMD processor	Yes	No
Cache coherent	Yes	No

The NIST Cloud Computing Ref. Archi. five major actors



Cloud Service Layers



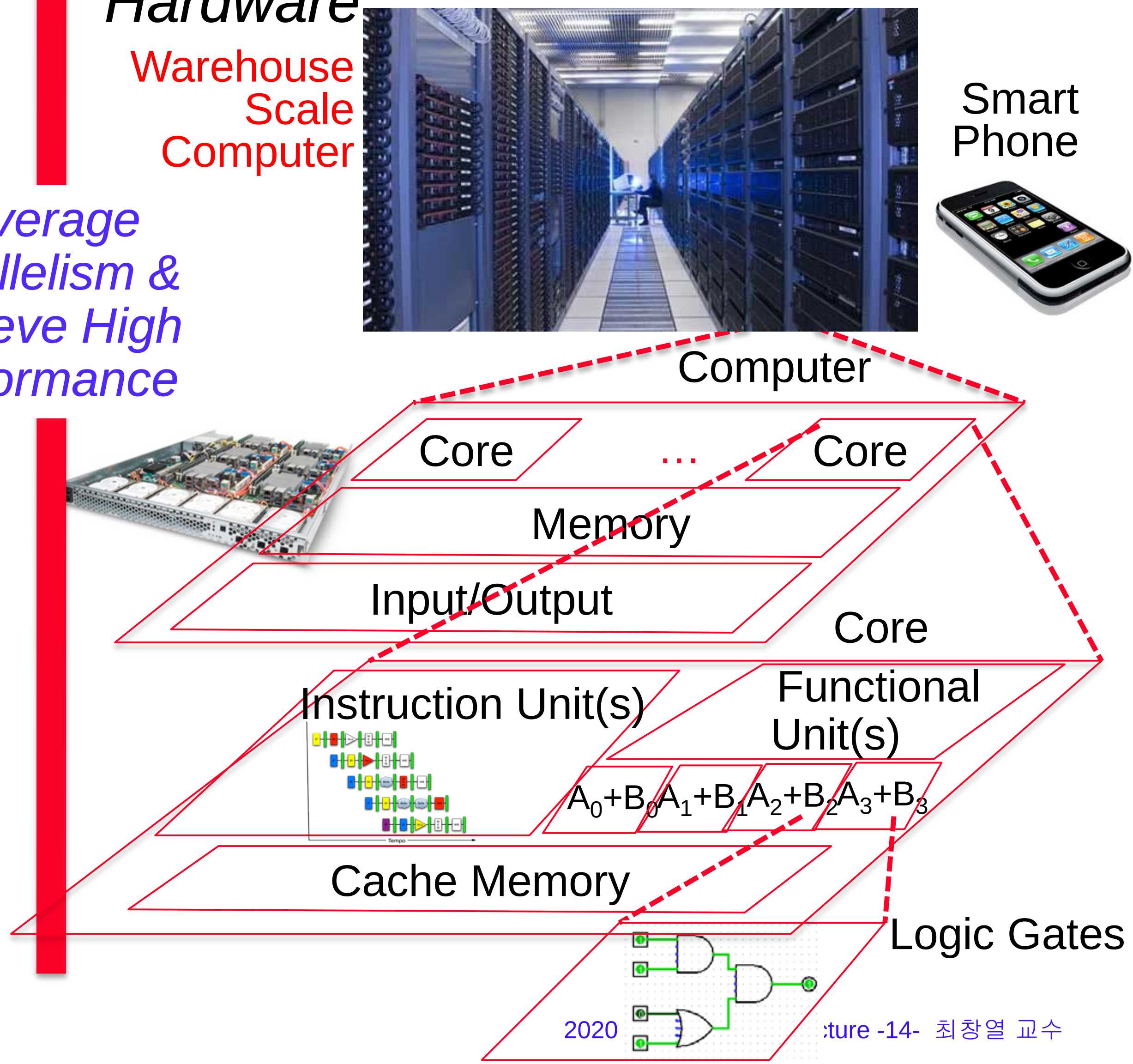
New "Great Ideas"

Software

- Parallel Requests
  - Assigned to computer
  - e.g., Search "Katz"
- Parallel Threads
  - Assigned to core
  - e.g., Lookup, Ads
- Parallel Instructions
  - >1 instruction @ one time
  - e.g., 5 pipelined instructions
- Parallel Data
  - >1 data item @ one time
  - e.g., Add of 4 pairs of words
- Hardware Descriptions
  - All gates functioning in parallel at same time
- Programming Languages

Leverage Parallelism & Achieve High Performance

Hardware



Concluding Remarks

- Goal: higher performance by using multiple processors

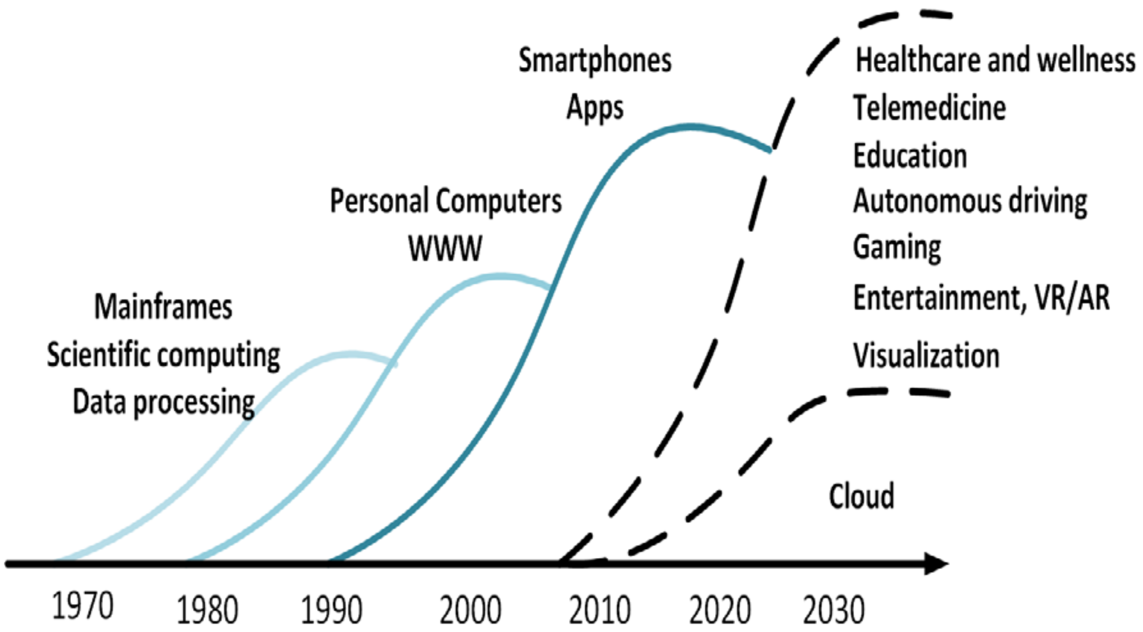
- Difficulties
  - Developing parallel software
  - Devising appropriate architectures

- SaaS importance is growing and clusters are a good match

- Performance per dollar and performance per Joule drive both mobile and WSC

Computer Architecture ?

Why is Architecture Exciting Today?

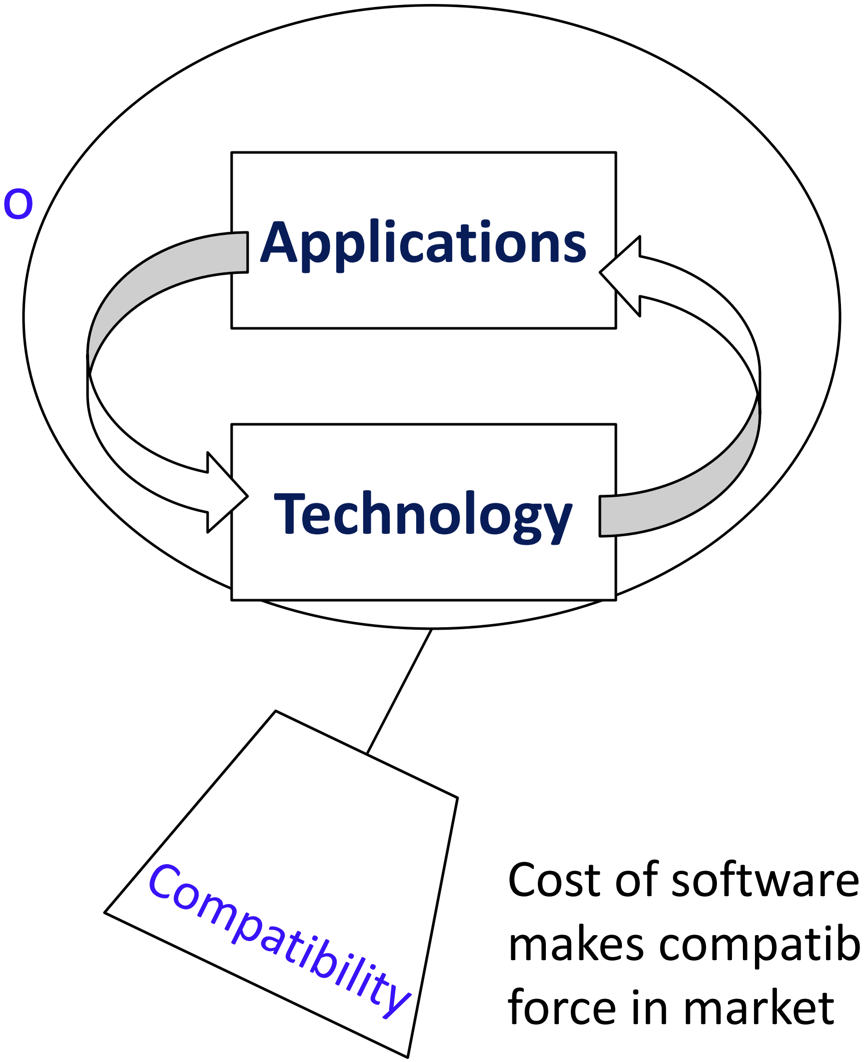


Number of deployed devices continues growing, but no single killer app.

- Diversification of needs, architectures

Architecture Continually Changing

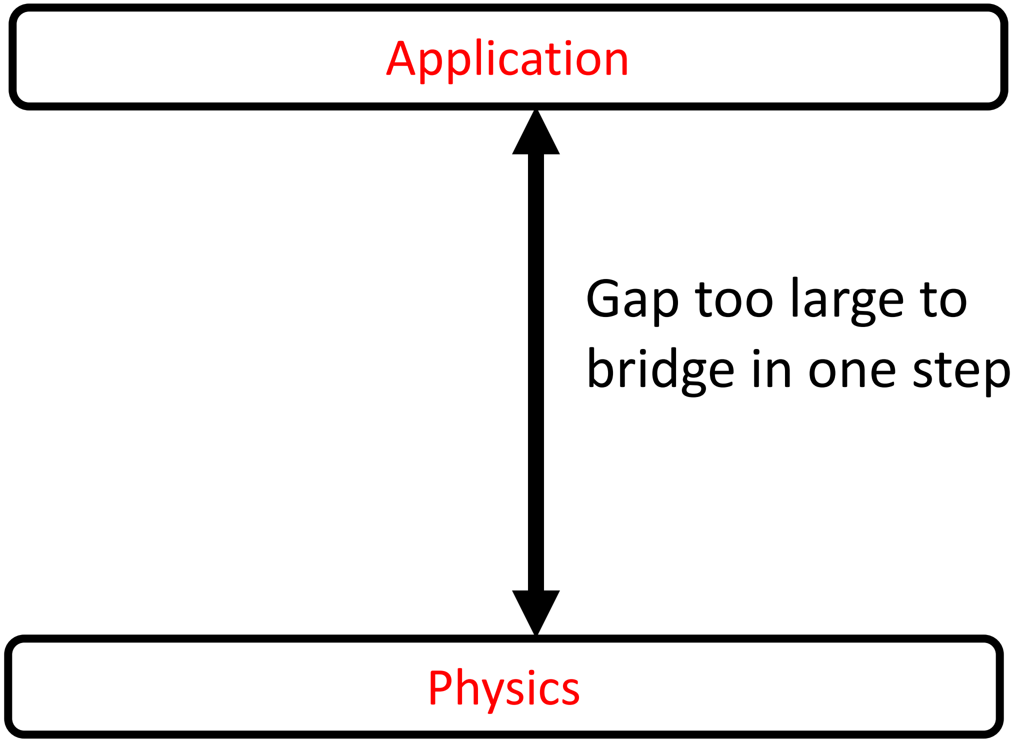
Applications suggest how to improve technology, provide revenue to fund development



Improved technologies make new applications possible

Cost of software development makes compatibility a major force in market

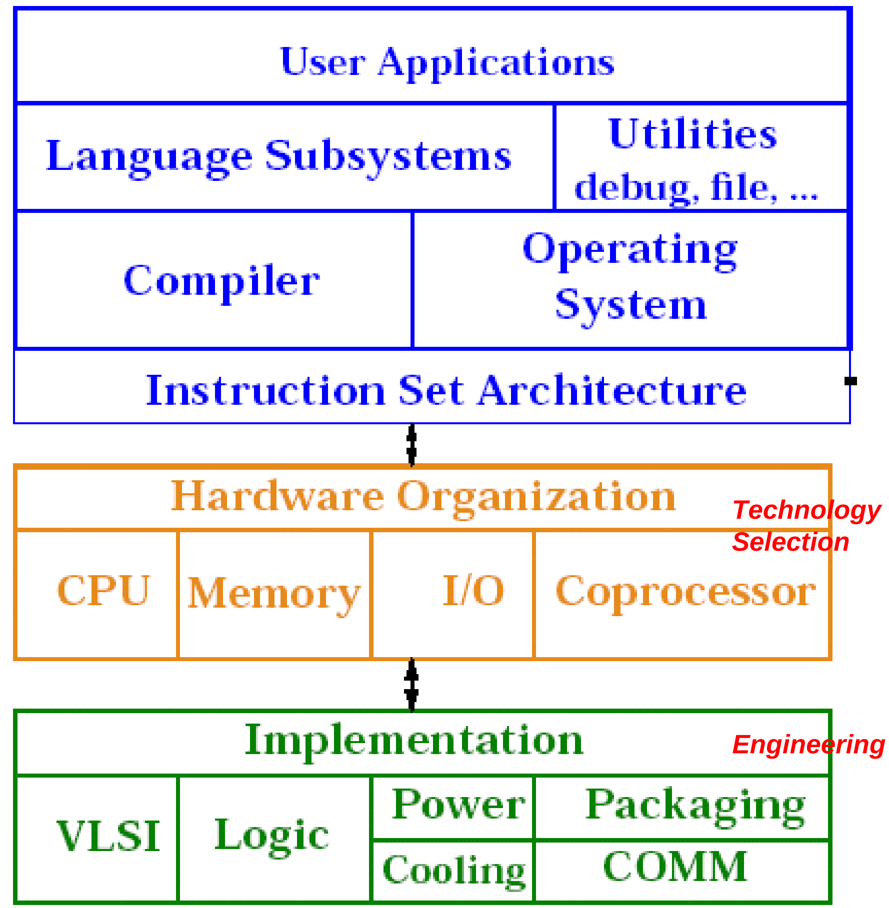
What is Computer Architecture? (1)



In its broadest definition, computer architecture is the *design of the abstraction layers* that allow us to implement information processing applications efficiently using available manufacturing technologies.

Abstraction Layers in Modern Systems

- Application
- Algorithm
- Programming Language
- Operating System/Virtual Machines
- Instruction Set Architecture (ISA)
- Microarchitecture
- Gates/Register-Transfer Level (RTL)
- Circuits
- Devices
- Physics



What is Computer Architecture? (2)

“*Computer architecture, like any other architecture, is the **art** of determining the needs of the user of a structure and then designing to meet those needs as effectively as possible within economic and technological constraints.*”

- It is an important and exciting subject
  - A combination between Science and **Art**
    - How to **utilize** technology appropriately
    - Performance can be enhanced by creativity
  - Many assessment goals
    - Performance (goal: increase)
    - Cost (goal: decrease)
    - Scalability (goal: improve)
    - Power/Heat (goal: decrease)
    - Reliability (goal: improve)
- Fact : Good programmers tend to write efficient software!
  - To do that, you need to understand the hardware, the architecture, and know how your program is executed…
    - knowing architecture will help you write more efficient programs
  - Today, we are entering multicore era, acclerators, ARM, cloud, etc…

고맙습니다!