



CS10 The Beauty and Joy of Computing

Lecture #8 : Concurrency

2011-09-28

UC Berkeley EECS
Lecturer SOE
Dan Garcia

"KOOOMEY'S LAW" – EFFICIENCY 2X EVERY 18 MO

Prof Jonathan Koomey looked at 6 decades of data and found that energy efficiency of computers doubles roughly every 18 months. This is even more relevant as battery-powered devices become more popular. Restated, it says that for a fixed computing load, the amount of battery you need drops by half every 18 months. This was true before transistors!



www.technologyreview.com/computing/38548/

Concurrency & Parallelism, 10 mi up...

Intra-computer

- Today's lecture
- Multiple computing "helpers" are cores within one machine
- Aka "multi-core"
 - Although GPU parallelism is also "intra-computer"

Inter-computer

- Week 12's lectures
- Multiple computing "helpers" are different machines
- Aka "distributed computing"
 - Grid & cluster computing



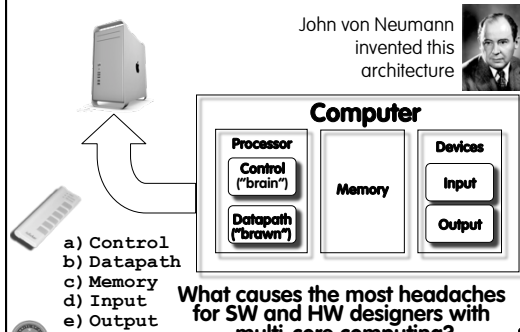
UC Berkeley CS10 "The Beauty and Joy of Computing" : Concurrency (9)



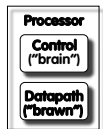
Anatomy: 5 components of any Computer



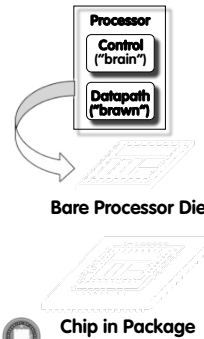
Anatomy: 5 components of any Computer



But what is INSIDE a Processor?



But what is INSIDE a Processor?



- Primarily Crystalline Silicon
- 1 mm – 25 mm on a side
- 2009 "feature size" (aka process) ~ 45 nm = 45×10^{-9} m (then 32, 22, and 16 [by yr 2013])
- 100 - 1000M transistors
- 3 - 10 conductive layers
- "CMOS" (complementary metal oxide semiconductor) - most common
- Package provides:
 - spreading of chip-level signal paths to board-level
 - heat dissipation.
- Ceramic or plastic with gold wires.

en.wikipedia.org/wiki/Moore's_law

Moore's Law

Predicts: 2X Transistors / chip every 2 years

What is this "curve"?

- Constant
- Linear
- Quadratic
- Cubic
- Exponential

Gordon Moore
Intel Cofounder
B.S. Cal 1950!

UC Berkeley CS10 "The Beauty and Joy of Computing": Concurrency (7)

Moore's Law and related curves

UC Berkeley CS10 "The Beauty and Joy of Computing": Concurrency (8)

Moore's Law and related curves

UC Berkeley CS10 "The Beauty and Joy of Computing": Concurrency (9)

Power Density Prediction circa 2000

Source: S. Borkar (Intel)

UC Berkeley CS10 "The Beauty and Joy of Computing": Concurrency (10)

Going Multi-core Helps Energy Efficiency

- Power of typical integrated circuit $\sim C V^2 f$
 - C = Capacitance, how well it "stores" a charge
 - V = Voltage
 - f = frequency. I.e., how fast clock is (e.g., 3 GHz)

In the same process technology...

Cache		Cache	
Core	→	Core	Core
Voltage = 1		Voltage = -15%	
Freq = 1		Freq = -15%	
Area = 1		Area = 2	
Power = 1		Power = 1	
Perf = 1		Perf = -1.8	

Activity Monitor (on the lab Macs) shows how active your cores are

William Holt, HOT Chips 2005

UC Berkeley CS10 "The Beauty and Joy of Computing": Concurrency (11)

Energy & Power Considerations

Courtesy: Chris Batten

UC Berkeley CS10 "The Beauty and Joy of Computing": Concurrency (12)

view.eecs.berkeley.edu

Parallelism again? What's different this time?

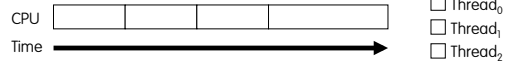
"This shift toward increasing parallelism is not a triumphant stride forward based on breakthroughs in novel software and architectures for parallelism; instead, this plunge into parallelism is actually a retreat from even greater challenges that thwart efficient silicon implementation of traditional uniprocessor architectures."

– Berkeley View, December 2006

- HW/SW Industry bet its future that breakthroughs will appear before it's too late

Background: Threads

- A **Thread** stands for "thread of execution", is a single stream of instructions
 - A program / process can split, or fork itself into separate threads, which can (in theory) execute simultaneously.
 - An easy way to describe/think about parallelism
- A single CPU can execute many threads by **Time Division Multiplexing**

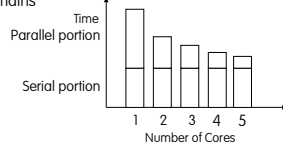


- Multithreading** is running multiple threads through the same hardware

en.wikipedia.org/wiki/Amdahl's_law

Speedup Issues : Amdahl's Law

- Applications can almost **never** be completely parallelized; some serial code remains



- s is serial fraction of program, P is # of cores (was processors)

Amdahl's law:

$$\text{Speedup}(P) = \text{Time}(1) / \text{Time}(P)$$

$$\leq 1 / (s + (1-s) / P), \text{ and as } P \rightarrow \infty$$

$$\leq 1 / s$$

- Even if the parallel portion of your application speeds up perfectly, your performance may be limited by the sequential portion

Speedup Issues : Overhead

- Even assuming no sequential portion, there's...**
 - Time to think how to divide the problem up
 - Time to hand out small "work units" to workers
 - All workers may not work equally fast
 - Some workers may fail
 - There may be contention for shared resources
 - Workers could overwriting each others' answers
 - You may have to wait until the last worker returns to proceed (the slowest / weakest link problem)
 - There's time to put the data back together in a way that looks as if it were done by one

Life in a multi-core world...

- This "sea change" to multi-core parallelism means that the computing community has to rethink:

- Languages
- Architectures
- Algorithms
- Data Structures
- All of the above

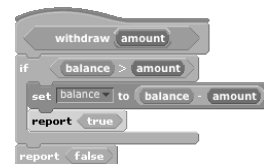


en.wikipedia.org/wiki/Concurrent_computing

But parallel programming is hard!

- What if two people were calling **withdraw** at the same time?

- E.g., balance=100 and two withdraw 75 each
- Can anyone see what the problem *could* be?
- This is a race condition



- In most languages, this is a problem.

- In Scratch, the system doesn't let two of these run at once.

Another concurrency problem ... deadlock!

- **Two people need to draw a graph but there is only one pencil and one ruler.**
 - One grabs the pencil
 - One grabs the ruler
 - Neither release what they hold, waiting for the other to release
- **Livelock also possible**
 - Movement, no progress
 - Dan and Luke demo



Summary

- "Sea change" of computing because of inability to cool CPUs means we're now in multi-core world
- This brave new world offers lots of potential for innovation by computing professionals, but challenges persist

