CS 181AI
Lecture 8

Performance + Parallelism

Arthi Padmanabhan

Feb 13, 2023

Notes about Assignment 2

- Please make sure you're subscribed to #assignmenthelp
- Sometimes ML is hard to explain
- A few percentage points can make a big difference!

Processes

- Instance of a running program
- Program contains:
  - Instructions
  - Data
  - Memory allocations
  - Symbols it uses
- Process contains:
  - Program + execution-specific information

Systems Concepts

- How do we run processes faster?
- How do we handle it smoothly when a process is interrupted?
- How do we schedule processes efficiently?
- We’ll be asking similar questions about machine learning jobs

Performance!

- What does it mean for a program to be fast?
- Program execution as some number of items, things to do
- Two concepts:
  - Items per unit of time -> bandwidth, or throughput (more is better)
  - Time per item, or latency -> less is better
- Improving either of these will make your program faster

Metrics

- Bandwidth, latency -> they are often related
- If you reduce the time to run a program from 5s to 4s, you increase the number per minute from 12 to 15
- If the conditions are right…
- Hopefully we can improve both metrics, sometimes we might need to pick one

Bandwidth

- Measures how much work can get done in one unit of time
- Parallelism improved bandwidth
- “Never underestimate the bandwidth of a station wagon full of tapes hurtling down the highway.”

Latency

- Measures how long it takes to do a task
- Also called response time
- Important for tasks that are real-time, like self-driving cars

Improving Latency

- Good way is to improve the time for a single process
  - There is a limit
  - Any improvements will also help with parallelized version
- However, faster sequential algorithms might not parallelize as well
- http://computers-are-fast.github.io
- Moral: don’t just guess which parts of your code are slow
  - Profile your code! We’ll get to methods of doing this later in the course

Exercise

- You need to make 100 paper airplanes. What’s the fastest way to do this?
  - What factors does your answer depend on?

Do Less Work

- Omit unnecessary work
  - Avoid calculating unnecessary intermediates
  - Calculate results only to the accuracy necessary for final output

Logging

- Producing text output to a log file or to a console screen is surprisingly expensive for a computer

Caching

- Store the results of expensive side-effect-free operations (e.g., I/O, computations) and reuse them if you know they're still valid

Be Prepared

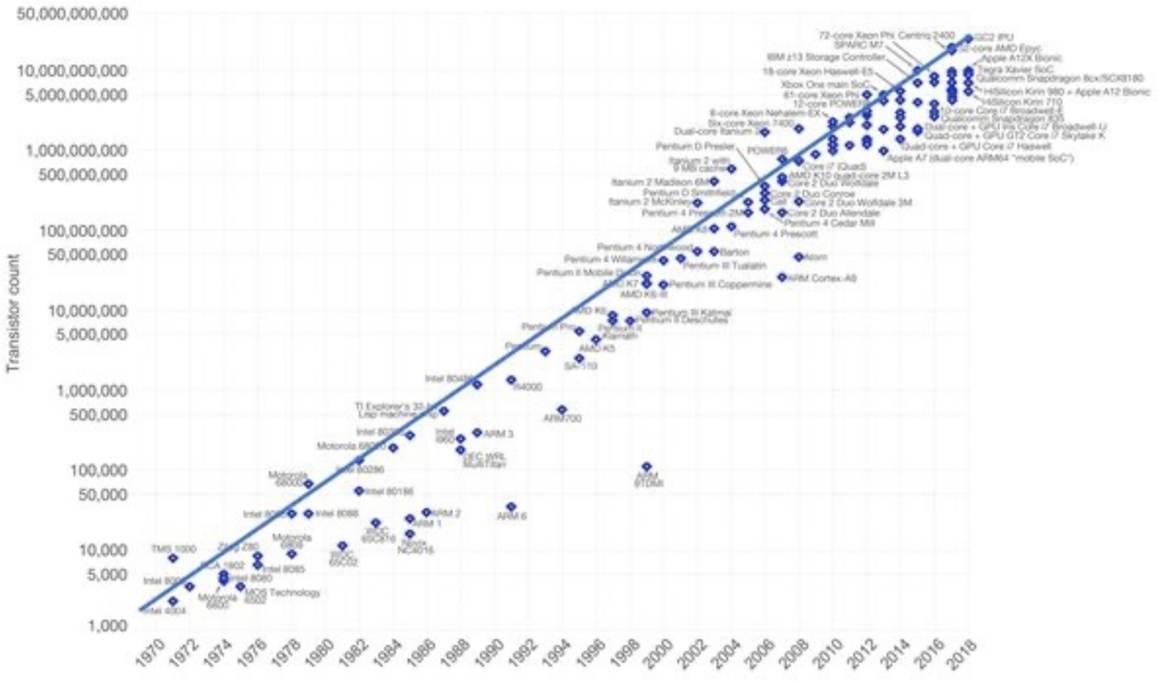
- If you know the user is going to ask for something, you can have it ready beforehand
  - ex/ you know customer is going to ask for an Excel report with some numbers and the numbers take a while to compute -> run the computations beforehand; then putting the necessary numbers in Excel is fast

Libraries

- Be careful in your choice of libraries
- Often library code can run faster than what you can write
- Other times, it’s better to optimize for your specific case

Wait for Better Machines?

- Back in the day, it was fine to write code with bad performance -> it was assumed that next year's CPUs would make it run acceptably
- Moore's law: observation that the number of transistors on a microchip will double about every two years



End of Moore's Law?

TECH

Intel says Moore's Law is still alive and well. Nvidia says it's ended.

The chips are down for Moore's law

The semiconductor industry will soon abandon its pursuit of Moore's law. Now things could get a lot more interesting.

Moore's Law Is Dead. Now What?

Intel co-founder Gordon Moore forever altered how we think about computing but, 55 years later, it's safe to say Moore's Law is finally dead. So what's next?

The End of Moore's Law and the Future of Computers: My Long-Read Q&A with Neil Thompson

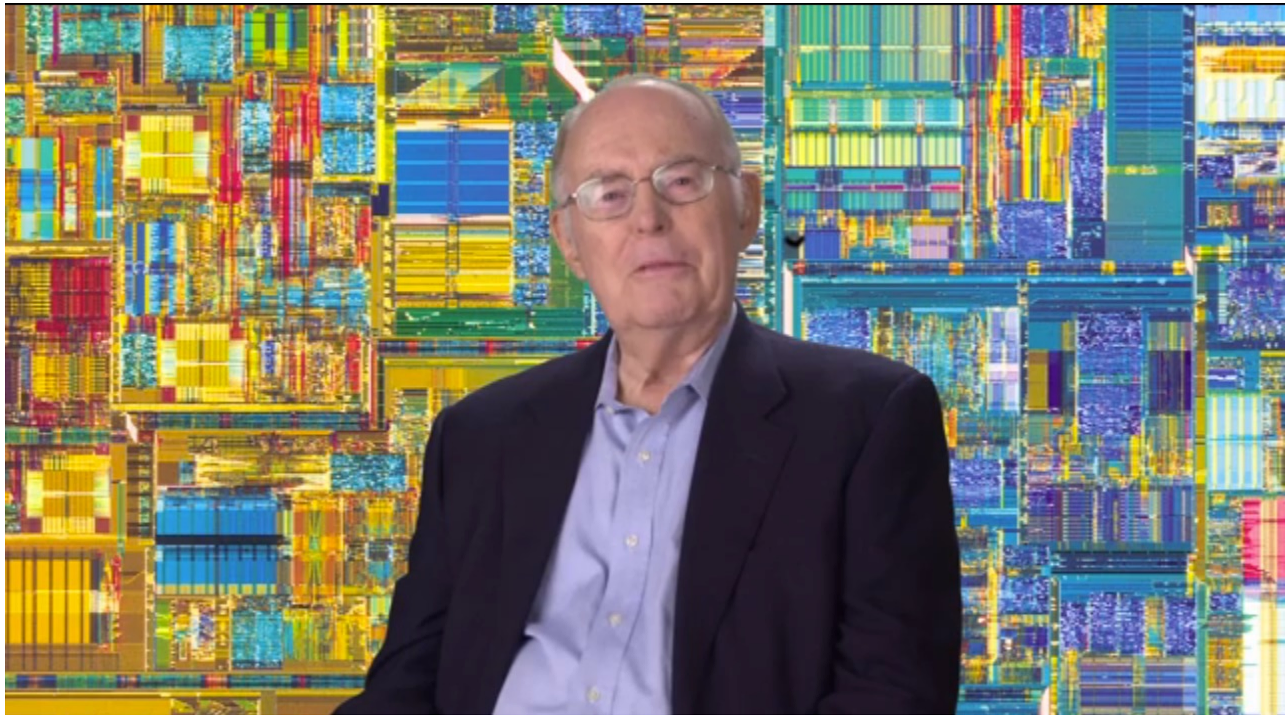
COMPUTING

We're not prepared for the end of Moore's Law

It has fueled prosperity of the last 50 years. But the end is now in sight.

Moore's Law

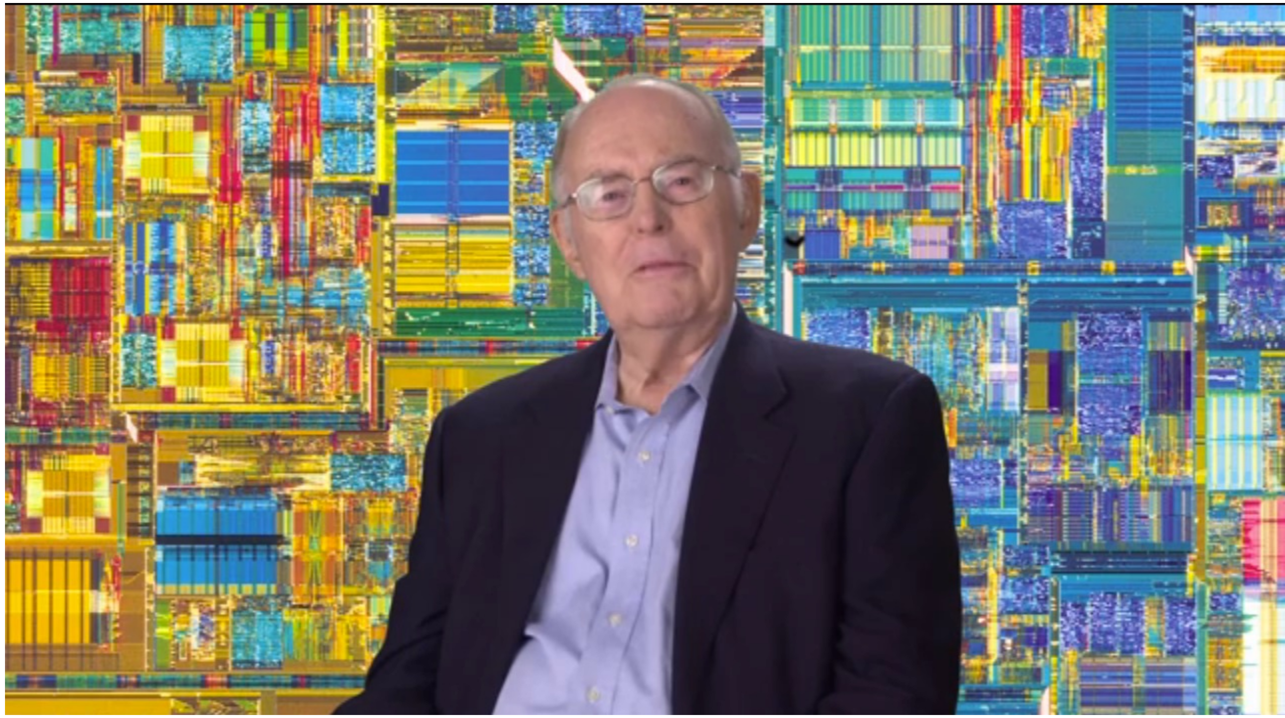
- "I guess I see Moore's law dying in the next decade or so"



Moore's Law

- "I guess I see Moore's law dying in the next decade or so"

  \- Gordon Moore



Throw More Resources at the Problem

- Use more CPUs!
- We will study how to effectively use more resources

Concurrency vs. Parallelism

- Concurrency:
  - Switching between two or more things (you can get interrupted)
    - Goal: make progress on multiple things at once
- Parallelism:
  - Running two or more things at the same time (they are independent)
    - Goal: finish multiple things as fast as possible

Concurrency vs. Parallelism

- You’re sitting at a table for dinner. You can:
  - Eat
  - Drink
  - Talk
  - Gesture
- Caveat: you’re so hungry that if you start eating, you won’t stop until you finish
  - Which tasks can and cannot be done concurrently?
  - Which tasks can and cannot be done in parallel?

More Parallelism, More Problems

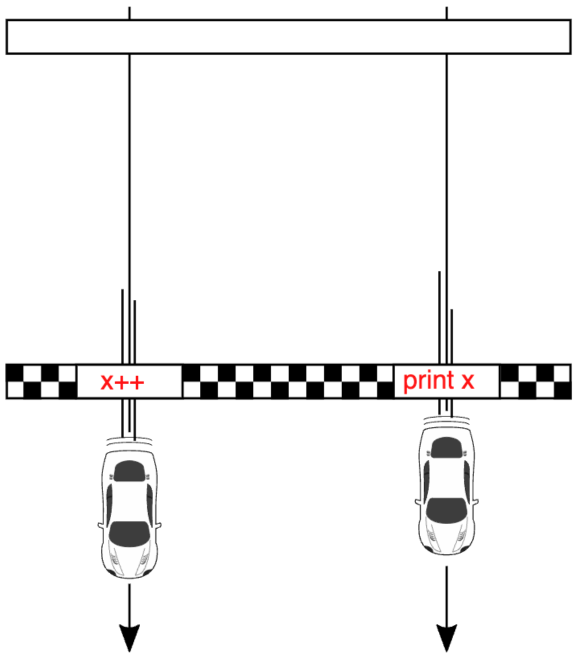
- It is often harder to write parallel code
- Some domains are “embarrassingly parallel”, others are “inherently sequential”
- There is often coordination overhead -> is it even worth it?
- Often, programs have a sequential part and a parallelizable part
  - If sequential dominates, running on multiple CPUs isn’t going to help
  - Known as Amdahl’s law

Complications

- There is no longer a total ordering between events
- Some events *A* are guaranteed to happen before others *B*
- Many events *X* and *Y* can occur in the order *XY* or *YX*

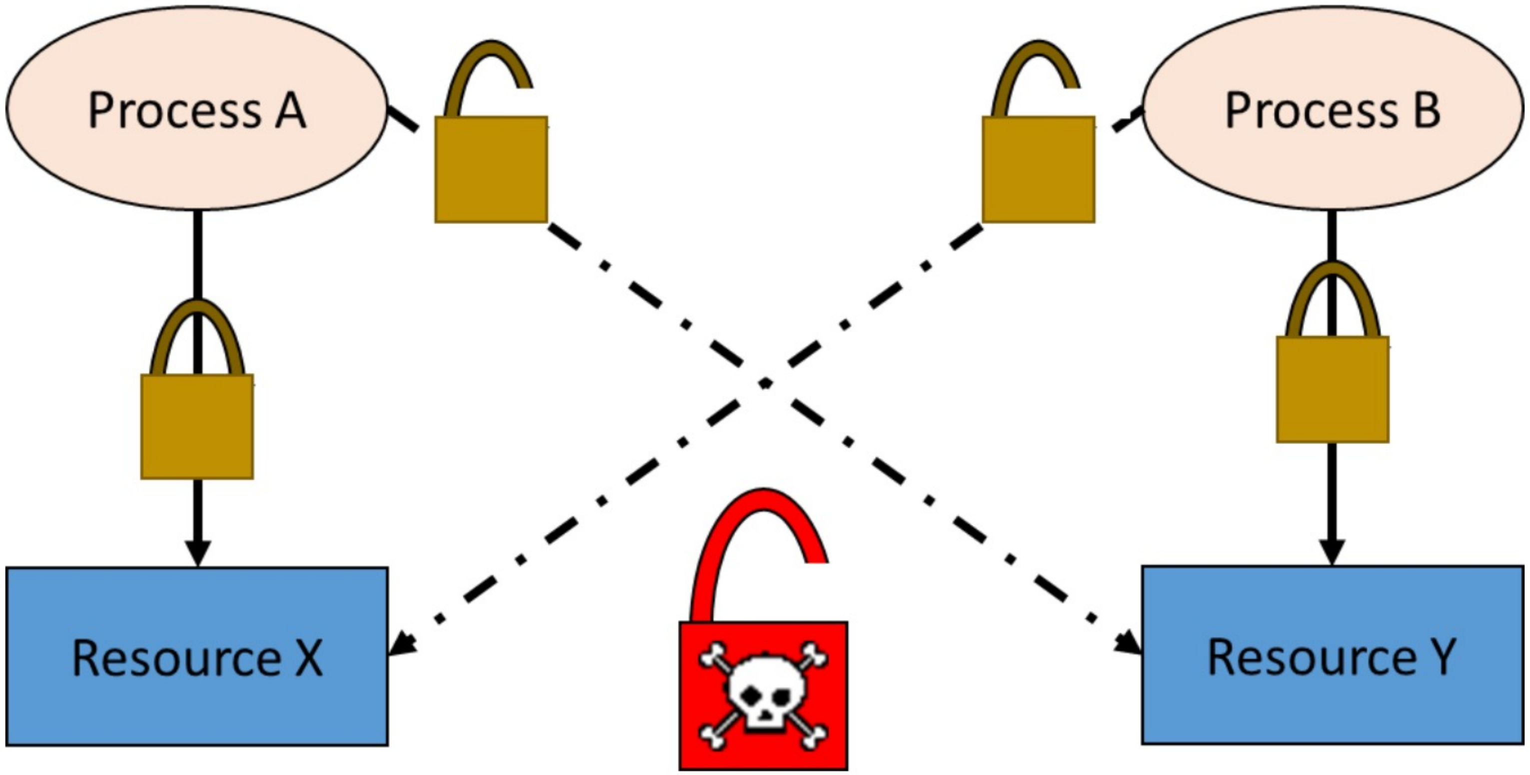
Races

- A **race** occurs when two or more processes try to access the same data and at least one of those accesses is a write
- Avoid races using locks, synchronization



Deadlocks

- A deadlock occurs when none of the processes can make progress because there is a cycle in the resource requests



General Principle

- The correctness of a parallel program should not depend on accidents of timing

Next Time: GPUs and Parallelization

- All about GPUs!

Acknowledgments

- Programming for Performance, University of Waterloo
- Operating Systems, UCLA