# Introduction

# Background

- Purpose of the class: Concurrency!
- What is concurrency?

# Concurrency

- A *concurrent* program consists of several independent tasks
  - These can be unordered with respect to each other
  - Or they can be partially ordered with respect to each other
- That's not the same as parallelism
  - Parallelism = Performing several tasks simultaneously
- "Concurrency is about *dealing* with lots of things at once. Parallelism is about *doing* lots of things at once."
  - https://blog.golang.org/concurrency-is-not-parallelism

# Concepts

- Three main concepts that give beginning programmers trouble...

# Iteration

- Life isn't one ____ thing after another. It's the same ____ thing over and over again.
  (Edna St. Vincent Millay)

# Recursion

- To understand recursion, you must first understand recursion.
  (Unknown)

# Concurrency

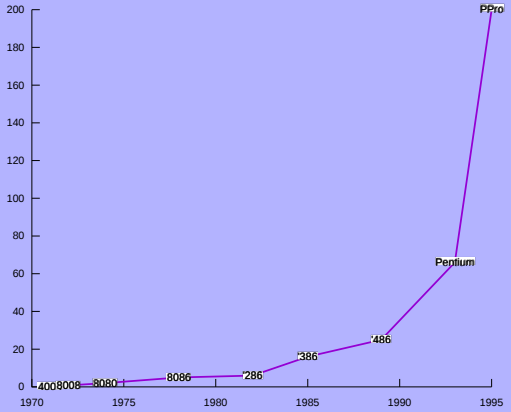
- Major Premise: Sixty men can do a piece of work sixty times as quickly as one man.
  Minor Premise: One man can dig a post-hole in sixty seconds
  Conclusion: Sixty men can dig a post hole in one second.
  (Ambrose Bierce)

# Motivation

- Why concurrency?
- Two main reasons:
  - Speedup
  - Organization

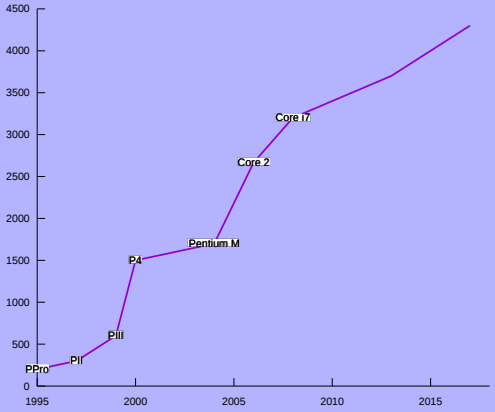
# Speed!

- Consider CPU speeds from early 70's through 1995: Exponential increase



# Speed!

- But then a funny thing happened...The rate of increase slowed down!



# Analysis

- Speeds are (slowly) hitting a brick wall
  - They creep upwards, but relative increases are getting smaller
- Limits of physics apply
  - Speed of light: 100 picoseconds (10GHz): 1.2 inches
  - Electrical charge: Move electrons on or off mosfet
    - There's a floor to how few electrons we can require (tunnelling)
  - SEU (cosmic rays) & transistor size

# The Past

- Is your program too slow?
  - Old joke: “You could spend a year rewriting it with a better algorithm or you could take a year off surfing.”
  - Either way, program will be fast enough next year!
- Not anymore...

# The Future

- Multiple cores are the trend
- As of late 2018:
  - AMD has announced a 48 core CPU (with 64 to follow)
  - Intel sells a 28 core unit (A bargain at $9,990)
  - Intel has demonstrated a *72 core* unit

# Make It Go Faster!

- Can’t just wait a year!
  - Multiple cores require programmer to do the work
  - Break task into pieces; separate piece per core

# Complication

- The way the processor industry is going, is to add more and more cores, but nobody knows how to program those things.
(Steve Jobs, as reported by Peter Clarke [https://www.eetimes.com/author.asp?section_id=36&doc_id=1266023])

# Challenge

- We need to use different programming techniques
- Concurrency demands a different tool set than nonconcurrent programming
- No industry consensus (yet) on the “best” way to approach the problem

# Measurements

- How do we know if we're making things better?
- Given two versions of same program:
  - Sequential one: Takes time

$$T_s$$

  - Parallel one: Takes time

$$T_p$$

- Concept: Speedup:

$$S = \frac{T_s}{T_p}$$

- Ideally, if we throw n cores at problem, time taken is $\frac{1}{n}$ original time
  - Thus, speedup = n

# Limitation

- Any program can be divided into two parts:
  - Parallelizable part
  - *Inherently sequential* part
    - *Must* be done sequentially
    - Ex: Initialization, task decomposition, I/O, etc.
- Suppose program 10% inherently sequential and 90% parallelizable
- What can we expect?

# Theoretically

- Let $T_s = 1$
- Suppose we have n CPU's
- Speedup:

$$S = \frac{T_s}{T_p} = \frac{1}{0.1 \cdot 1 + \frac{0.9 \cdot 1}{n}}$$

- Maximum possible speedup:

$$\lim_{n \to \infty} \frac{1}{0.1 \cdot 1 + \frac{0.9 \cdot 1}{n}} = \frac{1}{0.1} = 10$$

# Hmmm...

- Let that sink in a moment
- We can *never* get more than 10x speedup!

# Moral

- We must work to reduce sequential part of code to absolute minimum if we want to get good performance
- Note that often parallel programming allows us to solve larger problems
  - So even if we don't see speedup, the fact that we can solve a larger problem $\rightarrow$ still a benefit.

# Distributed Systems

- What if we need more speedup than we can get on a desktop system?
- Ex: If we have a 16 core machine and every last bit of our code is parallelizable, we can get at most 16× speedup
- But what if we need to go faster?
- Answer: Distributed systems

# Distributed System

- What is a distributed system?
  - Several independent computers working together
  - Independent CPU's, separate RAM, usually independent storage (HD,SSD)
- Cloud computing
  - Many companies provide "rentable computing power"
    - Amazon AWS, Microsoft Azure, Digital Ocean, ...

# Concurrency

- Sometimes, we don't really care (so much) about speedup
  - We can use concurrent programming to *organize* our code
  - Here the goal is not so much speeding things up as helping us manage several activities simultaneously

# Concurrency

- Example: In a game program: What do we have going on at once?
  - Rendering
  - Audio
  - AI/Pathfinding
  - Network communication
  - I/O: Loading game world/textures/etc. in the background
- We need to manage these, even if we only have one core

# Topics

- What this class will cover:
  - How to create parallel/concurrent programs
  - Organization: How to organize these programs
  - Message passing: Communicating between tasks
  - Work distribution: How to divide up tasks

# Assignment

- Install either:
  - Visual Studio + C# toolchain
  - Visual Studio Code + C# module
- Run the obligatory Hello, World program:

```
using System;
class Hello{
    public static void Main(string[] args){
        Console.WriteLine("Hello, world");
        Console.Readline();
    }
}
```

- Turn in a screenshot of the program running on your computer

# Sources


- Intel Corporation:
  http://www.intel.com/pressroom/kits/quickrefyr.htm ,
  http://www.intel.com/pressroom/kits/quickreffam.htm ,
  http://www.intel.com/pressroom/kits/quickrefyr.htm ,
  http://www.intel.com/content/www/us/en/history/history-intel-chips-timeline-poster.html?wapkw=processor+timeline
  http://ark.intel.com/products/93790/Intel-Xeon-Processor-E7-8890-v4-60M-Cache-2_20-GHz
- Intel Corporation. The Evolution of a Revolution.
  http://download.intel.com/pressroom/kits/IntelProcessorHistory.pdf
- Intel Corporation. Intel Ark. http://ark.intel.com/
- AMD, inc.
  http://products.amd.com/en-us/OpteronCPUDetail.aspx?id=814
- Wikipedia. Transistor Count.
- http://www.extremetech.com/extreme/171678-intel-unveils-72-core-x86-knights-landing-cpu-for-exascale-supercomputing
- http://www.amd.com/en-us/products/server/opteron/6000/6300#
- https://www.reddit.com/r/gamedev/comments/44fux4/multi_threading_in_game_development/
- https://www.extremetech.com/computing/245462-amds-new-32-core-naples-server-cpu-will-attack-intels-datacenter-monopoly

Created using LaTeX.

Main font: Gentium Book Basic, by Victor Gaultney. See http://software.sil.org/gentium/
Monospace font: Source Code Pro, by Paul D. Hunt. See https://fonts.google.com/specimen/Source+Code+Pro and http://sourceforge.net/adobe
Icons by Ulisse Perusin, Steven Garrity, Lapo Calamandrei, Ryan Collier, Rodney Dawes, Andreas Nilsson, Tuomas Kuosmanen, Garrett LeSage, and Jakub Steiner. See http://tango-project.org