Lecture 1:
Why Parallelism?
One common definition

A parallel computer is a collection of processing elements that cooperate to solve problems fast.

We care about performance *

We’re going to use multiple processors to get it

* Note: different motivation from “concurrent programming” using pthreads in 15-213
DEMO 1

(15-418 Spring 2012’s first parallel program)
Speedup

One major motivation of using parallel processing: achieve a speedup

For a fixed problem size:

\[
\text{Speedup}(P \text{ processors}) = \frac{\text{Time (1 processor)}}{\text{Time (P processors)}}
\]
Class observations from demos 1

- Communication limited the maximum speedup achieved
- Minimizing the cost of communication improved speedup
  - Moved students (“processors”) closer together (or let them shout)
DEMO 2

(scaling up to four processors)
Class observations from demo 2

- Imbalance in work assignment limited speedup
  - Some processors ran out work to do (went idle), while others were still working

- Improving the distribution of work improved speedup
DEMO 3

(massively parallel execution)
Class observations from demo 3

- The problem I just gave you has a significant amount of communication compared to computation

- Communication costs can dominate a parallel computation, severely limiting speedup
Course theme 1: Designing and writing parallel programs ... that scale!

- Parallel thinking
  1. Decomposing work into parallel pieces
  2. Assigning work to processors
  3. Orchestrating communication/synchronization

- Abstractions for performing the above tasks
  - Writing code in popular parallel programming languages
Course theme 2:
Parallel computer hardware implementation: how parallel computers work

- Mechanisms used to implement abstractions efficiently
  - Performance characteristics of implementations
  - Design trade-offs: performance vs. convenience vs. cost

- Why do I need to know about HW?
  - Because the characteristics of the machine really matter
    (recall speed of communication issues in class demos)
  - Because you care about performance (you are writing parallel programs)
Course theme 3: Thinking about efficiency

- FAST != EFFICIENT

- Just because your program runs faster on a parallel computer, it doesn’t mean it is using the hardware efficiently
  - Is 2x speedup on 10 processors is a good result?

- Programmer’s perspective: make use of provided machine capabilities

- HW designer’s perspective: choosing the right capabilities to put in system (performance/cost, cost = silicon area?, power?, etc.)
Logistics
Logistics

- Kayvon’s office hours
  - Tues/Thurs 1:30-2:30 PM (right after class)
  - GHC 7005

- TAs
  - Michael Papamichael
  - Mike Mu

- Textbook
  - Culler and Singh, *Parallel Computer Architecture: A Hardware/Software Approach*
  - Yes, it’s old. But many parts are still very good.
Logistics: assignments

- Four programming assignments
  - First assignment individual, the rest are in pairs
  - Each in a different parallel programming environment

Assignment 1: ISPC programming on Intel quad-core CPU

Assignment 2: OpenCL programming on NVIDIA GPUs

Assignment 3: OpenMP programming on Supercomputing cluster

Assignment 4: MPI programming on Supercomputing cluster
Logistics: final project

- 6-week final project
- Done in pairs

- Announcing: the first annual 418 parallelism competition!
  - Non-CMU judges from (Intel, NVIDIA, etc.)
  - Expect non-trivial prizes... (e.g., high end GPUs, tablets)
Logistics: grades

40% assignments
30% exams
25% project
5% class participation
Why parallelism?
Why parallelism?

- The answer 10 years ago
  - To get performance that was faster than what clock frequency scaling would provide
  - Because if you just waited until next year, your code would run faster on the next generation CPU

- Parallelizing your code not always always worth the time
  - Do nothing: performance doubling ~ every 18 months
End of frequency scaling

Intel CPU Trends
(sources: Intel, Wikipedia, K. Olukotun)
Power wall

\[ P = CV^2F \]

- **P**: power
- **C**: capacitance
- **V**: voltage
- **F**: frequency

- Higher frequencies typically require higher voltages
Power vs. core voltage

Pentium M

Credit: Shimin Chin
Programmable invisible parallelism

- **Bit level parallelism**
  - 16 bit → 32 bit → 64 bit

- **Instruction level parallelism (ILP)**
  - Two instructions that are independent can be executed simultaneously
  - “Superscalar” execution
ILP example

\[ a = (x^2 + y^2 + z^2) \]
ILP scaling

![Graph showing ILP scaling with instruction issue capability on the x-axis and speedup on the y-axis. The graph indicates a steep increase in speedup as instruction issue capability increases from 0 to 4, followed by a more gradual increase as the capability increases to 8, and then a plateau as the capability reaches 16.](image_url)
Single core performance scaling

The rate of single thread performance scaling has decreased (essentially to 0)

1. Frequency scaling limited by power
2. ILP scaling tapped out

No more free lunch for software developers!
Why parallelism?

- The answer 10 years ago
  - To get performance that was faster than what clock frequency scaling would provide
  - Because if you just waited until next year, your code would run faster on the next generation CPU

- The answer today:
  - Because it is the only way to achieve significantly higher application performance for the foreseeable future
Intel Sandy Bridge (2011)

- Quad core CPU + GPU
NVIDIA Fermi GPU (2009)

- 16 processing cores
Mobile processing

- Power limits heavily influencing designs

Apple A5: (in iPhone 4s and iPad 2)
Dual Core CPU + GPU + image processor and more

NVIDIA Tegra:
Quad core CPU + GPU + image processor...
Supercomputing

- Today: clusters of CPUs + GPUs
- Pittsburgh Supercomputing Center: Backlight
- 512 eight core Intel Xeon processors
  - 4096 total cores
Summary (what we learned)

- Single thread performance scaling has ended
  - To run faster, you will need to use multiple processing elements
  - Which means you need to know how to write parallel code

- Writing parallel programs can be challenging
  - Problem partitioning, communication, synchronization
  - Knowledge of machine characteristics is important