High Performance Computing

Ashish Sharma
Department of Biomedical Informatics
Emory University, Atlanta, GA

Logistics

- Class: Mon, Wed: 4 – 5:15
- Drop date: 01/19/2016
- Questions / Discussions:

(Questions, project discussions, assignments…)
Team: hpccourse.slack.com

- Office Hours: Send an email to ashish.sharma@emory.edu
  - Office: #573, Psychology Building
- Grading:
  - Class Participation: 30%
  - Assignments: 30%
  - Final Project: 40%

- Lecture Slides, Readings etc., will be posted on github
  - Lookout for an announcement on slack

Class Participation

- 15min presentations on current state of HPC
  [Class reading list & your own research]
- Guest Lectures on HPC applications: Varies
- Occasional 5min quizzes
- Active Participation:
  - Attend regularly
  - Participate (Class & Slack)
  - Start early with project

Academic Integrity

- Review the Emory Honor Code
- Penalty for academic dishonesty is a 0 on the assignment **and** half of your class participation grade. **For all involved.**
- It is your responsibility to protect your work
- All assignments are individual unless prior permission is explicitly granted. Any evidence of cooperation or plagiarism will be considered academic dishonesty
  - **You are allowed** to discuss and seek help on systems, installation, getting-started issues. **When in doubt ASK ME**
    1. Only from fellow classmates (~~Stackoverflow~~, ~~Quora~~, …)
    2. Suggest using slack for this, so that others can benefit from this (**also counts towards class participation grade**)

Acknowledgements

Similar courses that were helpful in preparing these lectures

- G63.2011.002/G22.2945.001: High Performance Scientific Computing
  - Marsha Berger & Andreas Klöckner – NYU
- CS 759: High Performance Computing for Engineering Applications
  - Dan Negrut – Univ. of Wisconsin
- CS267: Applications of Parallel Computers
  - Jim Demmel – Berkeley
- CS525: Parallel Computing
  - Ananth Grama – Purdue
- Seminar Series on High Performance Computing at The National Institute for Computational Science, University of Tennessee, Knoxville

Textbooks

- Introduction to Parallel Computing by Grama, Gupta, Kumar, Karypis
- Introduction to High-Performance Scientific Computing by Eijkhout

Introductions

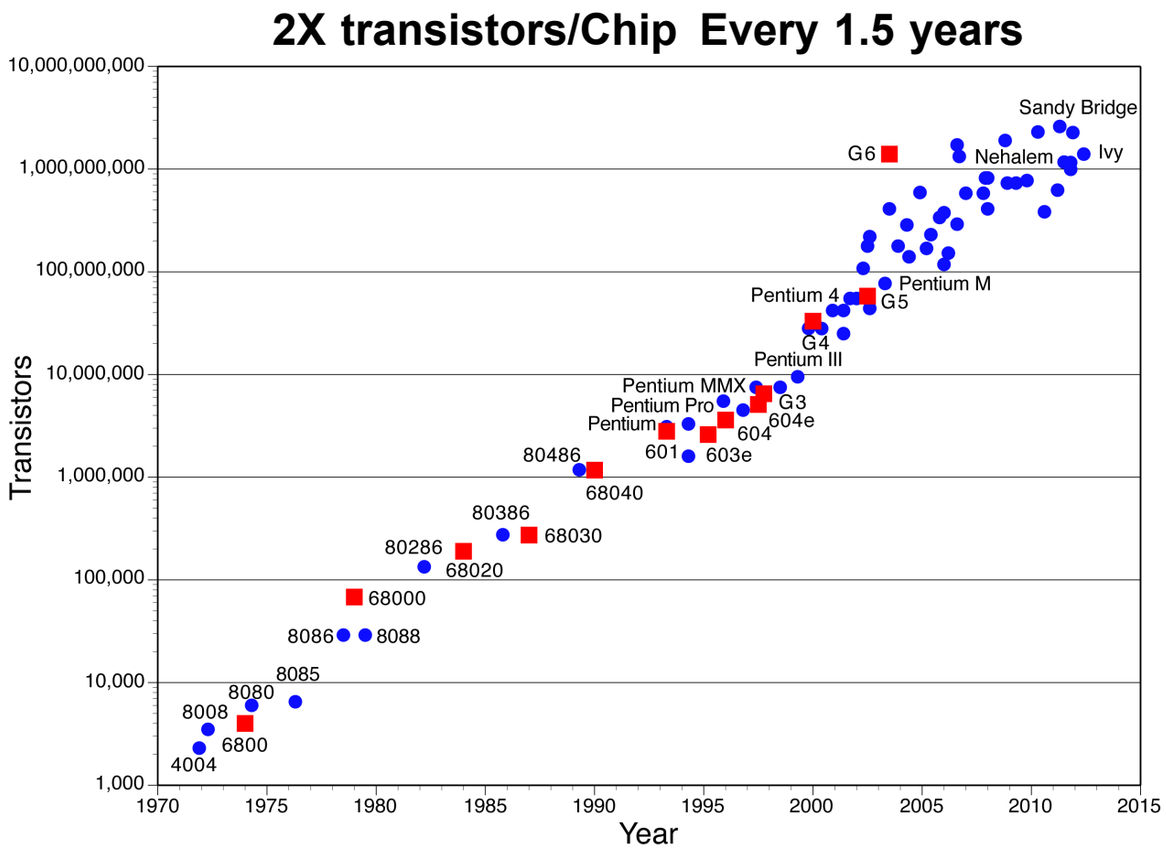
- Name:
- Program:
- Describe what you expect to gain from this course:

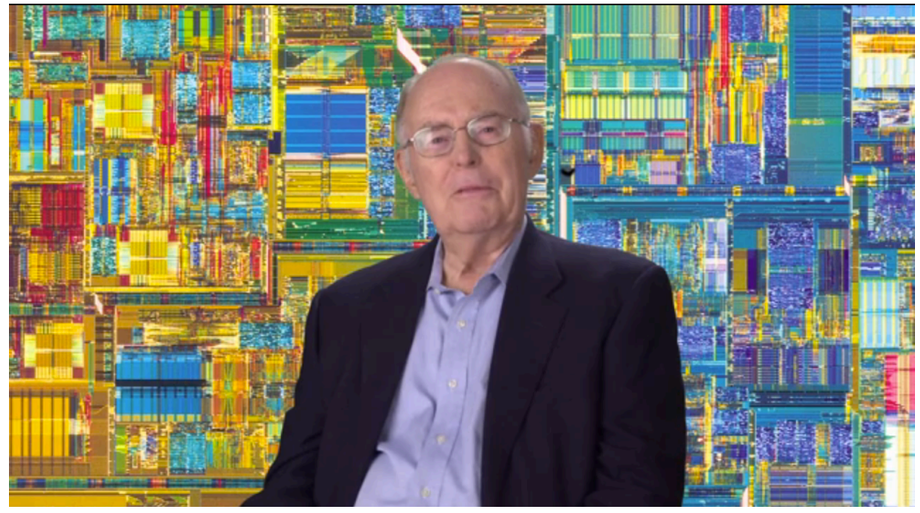
01 - Introduction

Basic Terminology

Prefix	Symbol	Size
Kilo	K	10^3
Mega	M	10^6
Giga	G	10^9
Tera	T	10^{12}
Peta	P	10^{15}
Exa	E	10^{18}
Zeta	Z	10^{21}

Moore's Law

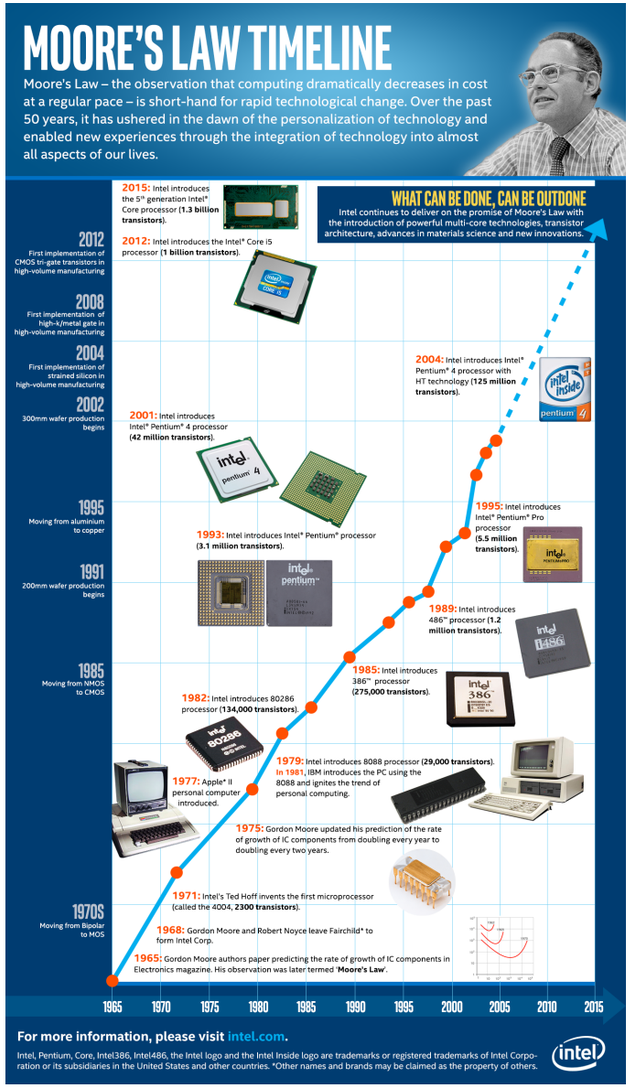


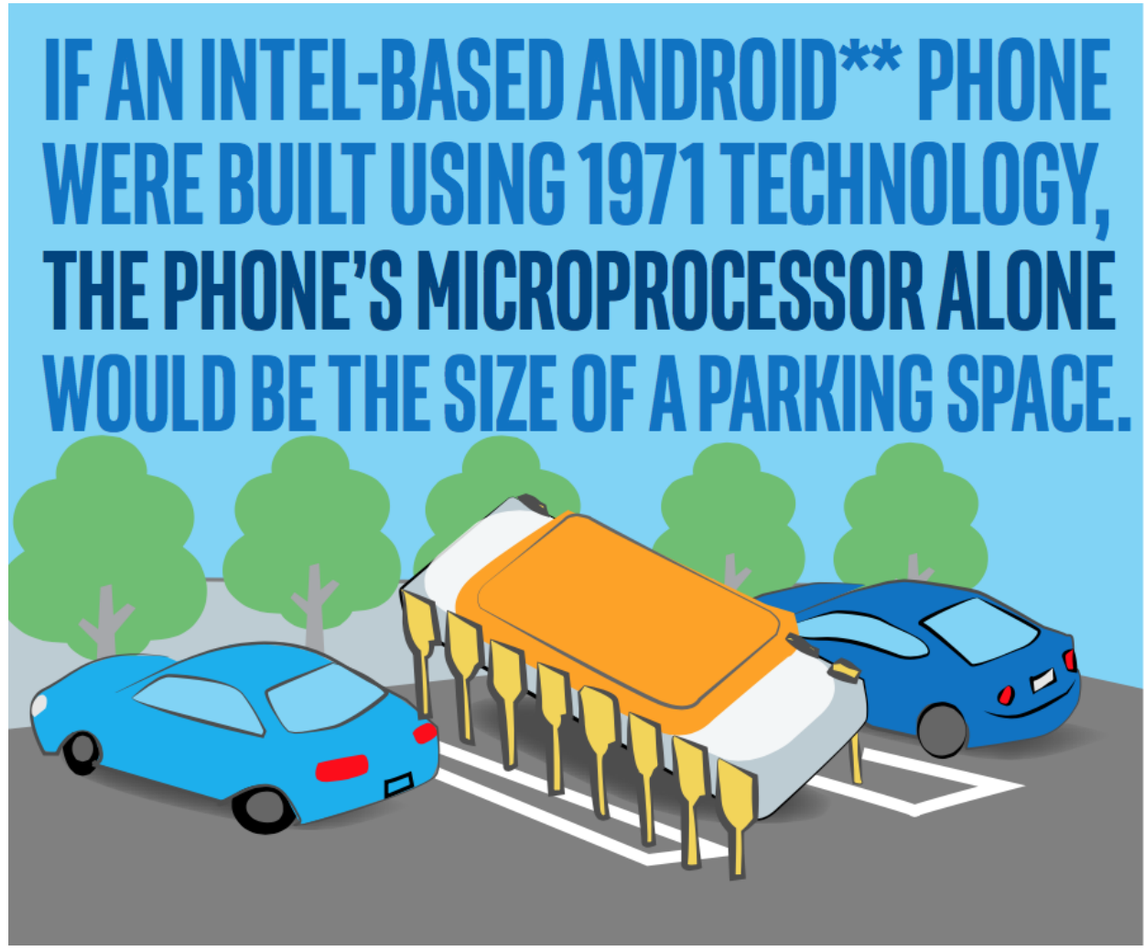


Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

Source: University of Wisconsin

In other words





http://www.techspot.com/news/60418-50-years-moore-law-fun-facts-timeline-infographic.html

How does it impact me?

Moore's Law: Same Chip → 2x transistors

1. What happens when transistor shrinks by a factor of x ?
2. Clock rate goes up by x because wires are shorter
   actually less than x, because of power consumption
3. Transistors per unit area goes up by **x^2**
4. Die size also tends to increase
   typically another factor of ~x
5. Raw computing power of the chip goes up by ~ x^4 !
   typically x^3 is devoted to either on-chip

So most programs x^3 times faster, without changing them

In that case, why bother?

If performance is doubling, then why bother with all this.
Just wait for computing to catch up??

- All major processor vendors are producing *multicore* chips
  - Every machine will soon be a parallel machine
  - To keep doubling performance, parallelism must double
- Which (commercial) applications can use this parallelism?
  - Do they have to be rewritten from scratch?
- Will all programmers have to be parallel programmers?
  - New software model needed
  - Try to hide complexity from most programmers – eventually
  - In the meantime, need to understand it

Slide from Berkeley

How do we measure this performance gain?

FLOPS — **Fl**oating-point **O**perations **P**er **S**econd — The unit of measuring RAW computer performance

Intel Xeon E5 2600 Sandy Bridge 2.6Ghz processor

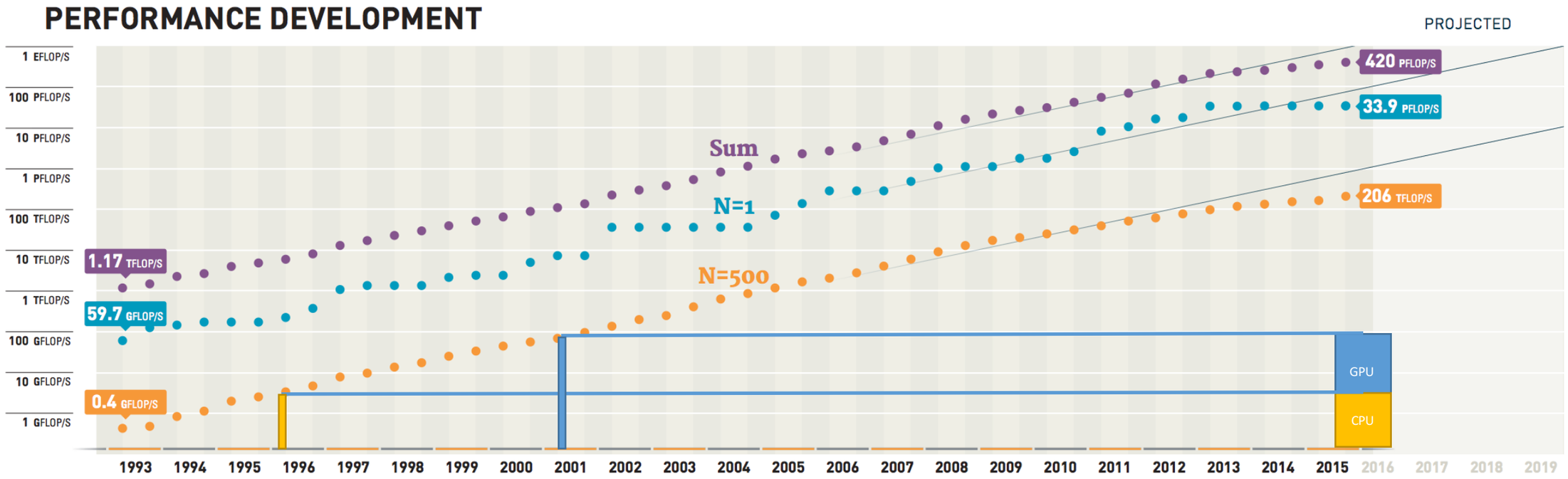
- 8 cores
- single precision FLOPS: 16/cycle
- double precision FLOPS: 8/cycle

What is the theoretical peak performance?

$2.6 * 10^9 * 16 * 4 * 2$ = 332.8Gflops (single precision)

$2.6 * 10^9 * 8 * 4 * 2$ = 166.4Gflops (double precision)

Top500.org



	NAME	SPECS	SITE	COUNTRY	CORES	RMAX PFLOP/S	POWER MW
1	Tianhe-2 (Milkyway-2)	Intel Ivy Bridge (12C 2.2 GHz) & Xeon Phi (57C 1.1 GHz), Custom interconnect	NUDT	China	3,120,000	33.9	17.8
2	Titan	Cray XK7, Opteron 6274 (16C 2.2 GHz) + Nvidia Kepler GPU, Custom interconnect	DOE/SC/ORNL	USA	560,640	17.6	8.2
3	Sequoia	IBM BlueGene/Q, Power BQC (16C 1.60 GHz), Custom interconnect	DOE/NNSA/LLNL	USA	1,572,864	17.2	7.9
4	K computer	Fujitsu SPARC64 VIIIfx (8C 2.0 GHz), Custom interconnect	RIKEN AICS	Japan	705,024	10.5	12.7
5	Mira	IBM BlueGene/Q, Power BQC (16C 1.60 GHz), Custom interconnect	DOE/SC/ANL	USA	786,432	8.59	3.95

Top500.org – Nov 2015 list

What is HPC?

High Performance Computing most generally refers to the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation in order to solve large problems in science, engineering, or business — InsideHPC

Tools to help do large scale, compute and/or data intensive problems

What is HPC…

- CPU Intensive → Distributed Memory (GPU, OpenMP) vs. Shared Memory (MPI)
- CPU Intensive vs. Data Intensive
  - Data interchange vs. exploit data locality
    MPI…. Hadoop…
- Architecture: CPU / GPU / Hybrid / coProcessors

Course Outline **(Tentative)**

Hardware, programming methodologies and algorithms for high performance and large-scale computing

Theme	Specific Topics
Introduction	Overview of The Course Incl. Prerequisites; HPC In The Age of Big Data; Architectural And Networking Advances; Parallel And Distributed Programming Models
HPC Hardware and Systems	Parallel Architectures; Networking; High Speed Interconnects; GPUs; Measuring Performance and Scalability; Storage Systems; Containers
Parallel Computing Frameworks	Designing algorithms for parallelism; Problem/Data Partitioning; Optimization Techniques; Load Decomposition;
Traditional Parallel Programming Models	Programming with OpenMP; Programming with MPI
Distributed/Cloud Computing	Cloud Computing environments (IaaS, PaaS); Hadoop; Apache Spark
Applications of HPC	Material Genome, Retail, Genomics and Bioinformatics, Medical Imaging, Clinical Informatics

Learning Objectives (*aka* what will be covered)

- Understand what is HPC — strengths and limitations
- Hardware and Programming Methodologies
- Develop an intuition for CPU-intensive vs. data-intensive computing
- Learn from the experiences of others
- Learn through hands-on problems and projects

What is not covered in this course

- GPUs, CUDA…
- Machine learning algortihms
- Deep dive into parallel processing algortihms
- Optimization techniques

Assignment #1

Due: 11:59pm, 01/17/2016

Assignment #1 (**Due 01/17/2016**)

1. Signup on class team: hpccourse.slack.com
   - Introduce yourself on the #general channel
2. Complete Class survey
3. Two programming assignments to help me establish a baseline
   - No help from instructor on this one
   - No help from fellow students on any aspect of this assignment

Assignment #1 (**Due 01/17/2016**)

Submission:

1. Create a **private** github repo: **name**-hpc2016
   - https://education.github.com
2. Add me as a collaborator to the repo (sharmaashish)
3. Create folders per assignment: HW1/HW2…
   - Case sensitive
4. Commit your assignment to the appropriate folder
5. I will checkout your latest commit made on or before the due date
   - Always include a brief README.md with instructions on how to test your code

Assignment #1 (**Due 01/17/2016**)

- Matrix Multiplication
  - Create a random *n x n* matrix **A**
  - Multiply **A** x **A**
  - Give user the ability to specify the size of the matrix as an input parameter (**-n 3**)
  - Give user the option to display the input and product (-**display**)
  - Display how long it takes to run this on a 2x2, 4x4, 8x8, 16x16…4096x4096 (-**benchmark**)

- Submission:
  1. Profile your submission to determine what parts take up the most amount of time. Are their parts of your code that are impacted by the size of n? Any optimization strategies?
  2. Submit C/C++/Python code w/ makefile (C/C++) → make and or/run.sh

sh run.sh –n 3 –display	sh run.sh –benchmark

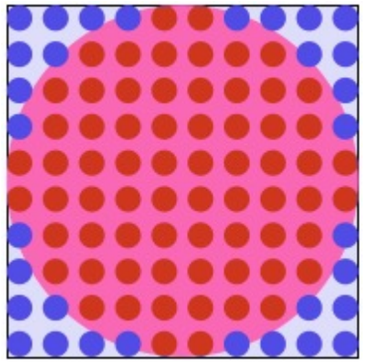
Assignment #1 (**Due 01/17/2016**)

- Calculate the value of **π**

  Throw *n* random points on a square

  Count those points that land in a circle

  PI = 4 * num of points in the circle / num of points



- Submission:
  1. Profile your submission to determine what parts take up the most amount of time. Are their parts of your code that are impacted by the size of n? What is the % error between your PI and π? Any optimization strategies?
  2. Submit C/C++/Python code w/ makefile (C/C++) → make and or/run.sh

```
sh run.sh –n 10000
```