

CS 475: Parallel Programming
&
GRAD 510: High Performance Computing

Sanjay Rajopadhye
Colorado State University
Fall 2011

Name card info (outside)

- Fold your card LENGTHWISE so that it can be placed on your desk.
- Neatly write your name (how you want to be addressed) in BIG, BOLD LETTERS, using the markers that are circulating
- Prop it up on your desk so that it can be seen from the front of the classroom.

Name card info (inside)

- **Name:** Sanjay Rajopadhye
- **Pronunciation (optional):** In Indian names, “a” is almost always pronounced as a short “u” sound as in **gun**, **fun**, etc., or a long “aa” sound as in **calm**, **bard**, etc.
 - Sanjay is pronounced as **Sun-ju**y
 - Rajopadhye **Raaj**-Oh-**paath**-yay (in the “paath” make the t sound like a d). Don’t worry if you don’t get it right, it’s almost always mispronounced, even in India).
 - These pronunciation rules are used in many parts of Asia, e.g., pronounce “Bagdad?”
- **Major dept:** CS/ECE
- **Status:** Associate Professor (e.g., 3rd year Ph.D.)
- **Interesting fact:** **Everyone in my family is born in a different country**

Bio sketch

- **Education**
 - undergrad: B. Tech, IIT Kharagpur, India, 1980
 - grad: Ph.D. University of Utah, 1986
- **Professional:**
 - Assistant Prof, CS University of Oregon (86-91)
 - Assistant Prof, ECE Oregon State University (91-92)
 - Researcher/Prof Associé IRISA, Rennes (1993-2001)
 - Associate Prof, Colorado State University (2001-...)
- **Research contributions/interests:**
 - High-performance computing, embedded systems, VLSI, algorithms, compilers, programming languages, ...
 - Polyhedral model: a mathematical framework for describing, transforming and “compiling” massively parallel computations

Teaching Assistant

- **Name:** Thiyagarajan (Rajan) Chockalingam
- **What you want to be called:** Rajan
- **Major:** ECE
- **Status:** Grad (MS, 2nd year)
- **Interesting fact:**

Teaching Assistant

- **Name:** Nissa Osheim
- **Major:** Computer Science
- **Status:** Grad (PhD)
- **Interesting fact:** Osheim means “house in the field that overlooks the fjord.” It was changed from an unpronounceable word that meant, “house in the field so shoddy that not even goats would live there.”

Why CS475/510 (and CS575, CS560)

Moore's Law is dead

Long live Moore's Law

Moore's Law as formulated

- Empirical observation (1965) by Gordon Moore:
 - The number of transistors that can be inexpensively placed on an integrated circuit is increasing exponentially, doubling approximately every two years.
 - “If Al Gore invented the internet, I invented the exponential.” --- Gordon Moore
 - en.wikipedia.org/wiki/Moore's_law
- This has held true till now and is expected to hold until at least 2015 (news.cnet.com/New-life-for-Moores-Law/2009-1006_3-5672485.html).

Moore's Law (interpretations)

- **Chip designers:** “we had better ensure that the exponential growth is maintained (or else the competition will).”
 - Main reason that Moore's law is being sustained.
- Other features of semiconductors are also growing exponentially (but at different rates)
 - Frequency
 - Memory: density and speed (bandwidth and latency)
 - Hard drive (and other I/O devices): capacity and speed
 - Power
- **General public:** Performance is increasing, so we can expect/build more and more powerful applications.

Corollary of exponential growth

- When two quantities grow exponentially, but at different rates, their ratio also grows exponentially. Consider,

$$y_1 = a^x, \text{ and } y_2 = b^x \text{ for } a \geq b \geq 1$$

$$y = \frac{y_1}{y_2} = \left(\frac{a}{b}\right)^x = \alpha^x \text{ for } \alpha \geq 1$$

“Gaps or walls” of Moore’s Law

- Memory gap/wall:
 - Memory bandwidth grows much more slowly than processor speeds (since mid 80s, this was addressed by ever increasing on-chip caches).
Devoting such a large fraction of silicon resources to “just” storage is not very effective.
- ILP (instruction level parallelism) wall:
 - One way to exploit increased clock frequency was to increase the instruction-level parallelism on chip (deeply pipelined, out-of-order, VLIW, etc.) leading to complicated control logic.
Overhead for this as the degree of parallelism is scaled is too high
- Power wall
 - Power dissipation ability is also increasing exponentially, but at such a slow rate that it has effectively peaked. **This is a brick wall**

Long Live Moore’s Law

The goal is to deliver performance to the “masses.” The only solution to the power wall is NOT to increase frequency as aggressively, but to have multiple processor cores

- “The processor is the new transistor.”
 - Number of cores will continue to grow at an exponential rate

Challenges of multicores/manycores

- Parallel programming is critical:
 - move from the realm of a small number of expert programmers who have challenging applications that must be resolved to general purpose computing
 - A sequential program is a slow program
- Parallel programming is hard. Grand challenge:
 - Enable average programmers to write codes for massively parallel computers **without making the programming task more difficult.**
 - This requires long term research on automatic parallelization, languages, OS, fault tolerance, etc.

In the meantime there's this class

Class Objectives

- Study the foundations of parallel programming
- Write parallel programs
 - In C
 - Using MPI and OpenMP style programming paradigms
- Execute and measure performance
- Analyze performance
- Write reports

HPC & Embedded Systems at CSU

- Cluster of three relevant courses in Parallel/HPC Systems:
 - CS 575: parallel algorithms, general purpose HPC, computing
 - CS 560: Polyhedral model, Equational programming, program transformations, gateway for research in HPC/compilers at CSU
- ECE Embedded Systems Track: CS475, ECE/CS 561, ECE 661

Class Format

- Hands on
- Weekly Lab (Wed 10:00-11:40) covers many details.
- Write effective and scalable parallel programs
 - You will submit your programs a few days before the homework is due (half the story)
 - Measure, analyze and report performance (second half)

Break

Analyzing/Plotting Data

- Back to basics: study your data
- No fancy statistics (not even linear regression to fit functions)
- Gnuplot

Plotting Data: cardinal rule

- Visually, a straight line conveys the most information.
- If your data is not a linear function, massage it so that **what you plot** looks like a linear function. Then deduce the original function.

Massaging Data

- If (you think) the function is polynomial

$$y = f(x) = a_0 + a_1x + \cdots + a_nx^n$$

$$\approx a_nx^n \quad (\text{asymptotically})$$

$$\log y = \log a_n + n \log x$$

$$y' = a' + nx'$$

Massaging Data

- If the function is exponential

$$\begin{aligned}y &= f(x) = ba^x \\ \log y &= \log b + x(\log a) \\ &= b' + a'x\end{aligned}$$

Massaging Data

- If the function is hyperbolic

$$\begin{aligned}y &= f(x) = \frac{a}{x} \\ y' &= \frac{a}{y} = x \quad \text{scaled reciprocal of } y \\ &= \frac{f(1)}{f(x)} \quad \text{this is the speedup function}\end{aligned}$$

Massaging Data

- If the function is hyperbolic with a non-zero asymptote

$$y = f(x) = \frac{a}{x} + c$$

$$y' = \frac{f'(x)}{f(x)} = \frac{(a+c)x}{a+cx} \text{ (not really a straight line)}$$

but linear for small c and x