CS4230 Parallel Programming

Lecture 1: Introduction

Mary Hall
August 21, 2012

Course Details

- Time and Location: TuTh, 9:10-10:30 AM, WEB L112
- Course Website
  - http://www.eng.utah.edu/~cs4230/
- Instructor: Mary Hall, mhall@cs.utah.edu, http://www.cs.utah.edu/~mhall/
  - Office Hours: Mon 11:00-11:30 AM; Th 10:45-11:15 AM
- TA: TBD
  - Office Hours: TBD
- SYMPA mailing list
  - cs4230@list.eng.utah.edu
  - https://sympa.eng.utah.edu/sympa/info/cs4230
- Textbook
  - "An Introduction to Parallel Programming," Peter Pacheco, Morgan-Kaufmann Publishers, 2011.
  - Also, readings and notes provided for other topics as needed

Administrative

- Prerequisites:
  - C programming
  - Knowledge of computer architecture
  - CS4400 (concurrent ok for seniors)
- Please do not bring laptops to class!
- Do not copy solutions to assignments from the internet (e.g., wikipedia)
- Read Chapter 1 of textbook by next lecture
- First homework handed out next time

Basis for Grades

- 35% Programming projects (P1,P2,P3,P4)
- 20% Written homeworks
- 5% Participation (in-class assignments)
- 25% Quiz and Final
- 15% Final project

Today's Lecture

- Overview of course
- Important problems require powerful computers …
  - … and powerful computers must be parallel.
  - Increasing importance of educating ***parallel programmers*** (you!)
  - Some parallel programmers need to be performance experts – my approach
- What sorts of architectures in this class
  - Multimedia extensions, multi-cores, SMPs, GPUs, networked clusters
- Developing ***high-performance*** parallel applications
  - An optimization perspective

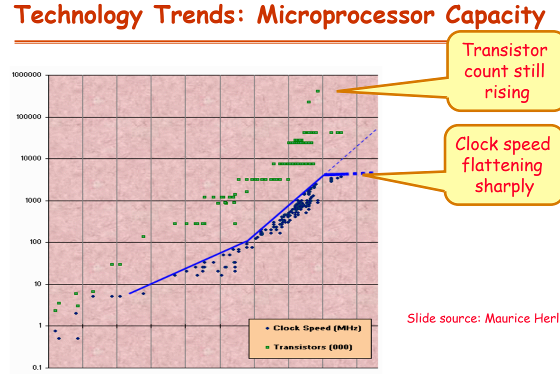
Course Objectives

- Learn how to program parallel processors and systems
  - Learn how to think in parallel and write correct parallel programs
  - Achieve performance and scalability through understanding of architecture and software mapping
- Significant hands-on programming experience
  - Develop real applications on real hardware
  - Develop parallel algorithms
- Discuss the current parallel computing context
  - Contemporary programming models and architectures, and where is the field going

Why is this Course Important?

- Multi-core and many-core era is here to stay
  - Why? Technology Trends
- Many programmers will be developing parallel software
  - But still not everyone is trained in parallel programming
  - Learn how to put all these vast machine resources to the best use!
- Useful for
  - Joining the work force
  - Graduate school
- Our focus
  - Teach core concepts
  - Use common programming models
  - Discuss broader spectrum of parallel computing

Technology Trends: Microprocessor Capacity



Slide source: Maurice Herlihy

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.

The Multi-Core or Many-Core Paradigm Shift

What to do with all these transistors?

- Key ideas:
  - Movement away from increasingly complex processor design and faster clocks
  - Replicated functionality **(i.e., parallel)** is simpler to design
  - Resources more efficiently utilized
  - Huge power management advantages

All Computers are Parallel Computers.

Scientific Simulation: The Third Pillar of Science

- Traditional scientific and engineering paradigm:
  1)Do theory or paper design.
  2)Perform experiments or build system.
- Limitations:
  - **Too difficult -- build large wind tunnels.**
  - **Too expensive -- build a throw-away passenger jet.**
  - **Too slow -- wait for climate or galactic evolution.**
  - **Too dangerous -- weapons, drug design, climate experimentation.**
- Computational science paradigm:
  3)Use high performance computer systems to simulate the phenomenon
  - **Base on known physical laws and efficient numerical methods.**

Slide source: Jim Demmel

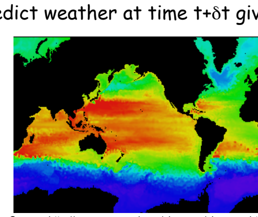
The quest for increasingly more powerful machines

- Scientific simulation will continue to push on system requirements:
  - To increase the precision of the result
  - To get to an answer sooner (e.g., climate modeling, disaster modeling)
- The U.S. will continue to acquire systems of increasing scale
  - For the above reasons
  - And to maintain competitiveness
- A similar phenomenon in commodity machines
  - More, faster, cheaper

Slide source: Jim Demmel

Example: Global Climate Modeling Problem

- Problem is to compute:
  f(latitude, longitude, elevation, time) → temperature, pressure, humidity, wind velocity
- Approach:
  - *Discretize* the domain, e.g., a measurement point every 10 km
  - Devise an algorithm to predict weather at time t+δt given t
- Uses:
  - Predict major events, e.g., El Nino
  - Use in setting air emissions standards



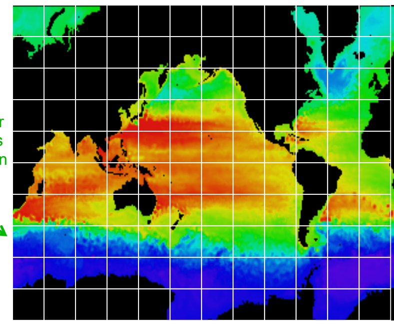
Source: http://www.epm.ornl.gov/chammp/chammp.html

Slide source: Jim Demmel

Some Characteristics of Scientific Simulation

- Discretize physical or conceptual space into a grid
  - Simpler if regular, may be more representative if adaptive
- Perform local computations on grid
  - Given yesterday's temperature and weather pattern, what is today's expected temperature?
- Communicate partial results between grids
  - Contribute local weather result to understand global weather pattern.
- Repeat for a set of time steps
- Possibly perform other calculations with results
  - Given weather model, what area should evacuate for a hurricane?

Example of Discretizing a Domain



Processors in adjacent blocks in the grid communicate their result.

Parallel Programming Complexity: An Analogy

- Enough Parallelism (Amdahl's Law)
- Parallelism Granularity
  - Independent work between coordination points
- Locality
  - Perform work on nearby data
- Load Balance
  - Processors have similar amount of work
- Coordination and Synchronization
  - Who is in charge? How often to check in?

Course Goal

- Most people in the research community agree that there are at least two kinds of parallel programmers that will be important to the future of computing
  - Programmers that understand how to write software, but are naïve about parallelization and mapping to architecture (Joe programmers)
  - Programmers that are knowledgeable about parallelization, and mapping to architecture, so can achieve high performance (Stephanie programmers)
  - Intel/Microsoft say there are three kinds (Mort, Elvis and Einstein)
  - This course is about teaching you how to become Stephanie/Einstein programmers

Course Goal

- Why OpenMP, Pthreads, MPI and CUDA?
  - These are the languages that Einstein/Stephanie programmers use.
  - They can achieve high performance.
  - They are widely available and widely used.
  - It is no coincidence that both textbooks I've used for this course teach all of these except CUDA.