CSC 447: Parallel Programming for Multi-Core and Cluster Systems

Introduction and Administrivia

Haidar M. Harmanani

Spring 2020

Course Introduction

- Lectures
  - TTh, 11:00-12:15 from January 21, 2019 until May 4, 2019
  - Prerequisites
    - Competency in a high-level programming language
    - CSC 310, Data Structures and Algorithms
    - CSC320, Computer Organization

Grading and Class Policies

- Grading
  - Midterm (1-2): 25%
  - Final: 40%
  - Programming Assignments [3-4]: 10%
  - Three milestones projects (OpenMP, OpenACC, and CUDA): 25%
- Exams Details
  - Exams are closed book, closed notes
- All assignments must be your own original work.
  - Cheating/copying/partnering will not be tolerated

Teaching Methodology

- We will use multiple choice questions to initiate discussions and reinforce learning
- We will also use a a flipped-classroom methodology for one part of the course

Programming Assignments

- All assignments and handouts will be communicated via `piazza`
  - Make sure you enable your account
- Use `piazza` for questions and inquiries
  - No questions will be answered via email
- All assignments must be submitted via `github`
  - `git` is a distributed version control system
  - `git` or its variations have become a universal standard for developing and sharing code
  - Make sure you get a private repo
    - Apply for a free account: https://education.github.com/discount_requests/new

Policy on Assignments and Independent Work

- With the exception of laboratories and assignments (projects and HW) that explicitly permit you to work in groups, all homework and projects are to be YOUR work and your work ALONE.
- It is NOT acceptable to copy solutions from other students.
- It is NOT acceptable to copy (or start your) solutions from the Web.
- PARTNER TEAMS MAY NOT WORK WITH OTHER PARTNER TEAMS
- You are encouraged to help teach other to debug. Beyond that, we don't want you sharing approaches or ideas or code or whiteboarding with other students, since sometimes the point of the assignment is the "algorithm" and if you share that, they won't learn what we want them to learn). We expect that what you hand in is yours.
- It is NOT acceptable to leave your code anywhere where an unscrupulous student could find and steal it (e.g., public GITHUBs, walking away while leaving yourself logged on, leaving printouts lying around,etc)
- The first offense is a zero on the assignment and an F in the course the second time
- Both Giver and Receiver are equally culpable and suffer equal penalties

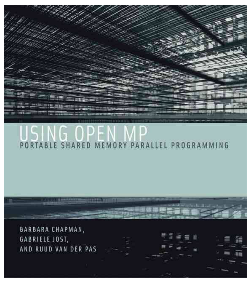
Contact Information

- Haidar M. Harmanani
  - Office: Block A, 810
  - Hours: TTh 9:00-10:30 or by appointment.
  - Email: haidar@lau.edu.lb

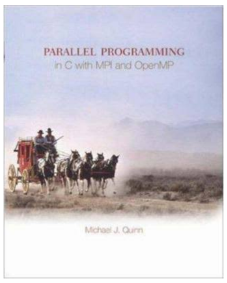
Topics

- Introduction to parallel programming
- Parallel programming performance
- Programming using Pthreads, OpenMP, OpenACC, and CUDA
- Introduction to Neural Networks and Deep Learning using Python

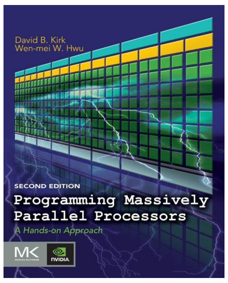
Course Materials


USING OPEN MP
PORTABLE SHARED MEMORY PARALLEL PROGRAMMING
BARBARA CHAPMAN, GABRIELE JOST, AND RUUD VAN DER PAS

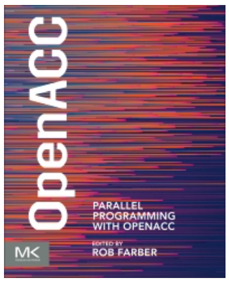
Optional



Optional



Required



Optional

Computers/Programming

- You will be using:
  - The lab or your own machines for Pthreads and OpenMP
  - The Computer Science Lab for OpenACC and CUDA Programming, or your own machine if you have a CUDA-capable GPU
  - The *Cloud* for GPU Tutorial Labs

Administrative Questions?

Definitions

- What is parallel?
  - Webster: "An *arrangement* or *state* that permits several operations or tasks to be performed simultaneously rather than consecutively"
- Parallel computing
  - Using parallel computer to solve single problems faster
- Parallel computer
  - Multiple-processor system supporting parallel programming
- Parallel programming
  - Programming in a language that supports concurrency explicitly

Parallel Processing – What is it?

- A parallel computer is a computer system that uses multiple processing elements simultaneously in a cooperative manner to solve a computational problem
- Parallel processing includes techniques and technologies that make it possible to compute in parallel
  - Hardware, networks, operating systems, parallel libraries, languages, compilers, algorithms, tools, …
- Parallel computing is an evolution of serial computing
  - Parallelism is natural
  - Computing problems differ in level / type of parallelism
- Parallelism is all about performance! Really?

Concurrency

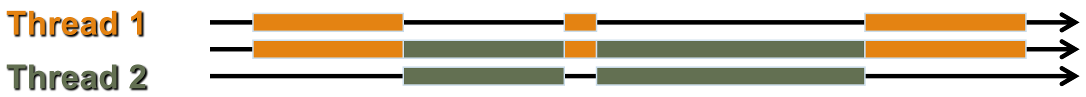
- Multiple executing tasks are concurrent with respect to each if
  - They can execute asynchronously
  - Implies that there are no dependencies between the tasks
- Example
  - Two threads can run *concurrently* on the same core by interleaving instructions
- Dependencies
  - If a task requires results produced by other tasks in order to execute correctly, the task's execution is *dependent*
  - If two tasks are dependent, they are not concurrent
  - Some form of synchronization must be used to enforce (satisfy) dependencies
- Concurrency is fundamental to computer science
  - Operating systems, databases, networking, …

Concurrency and Parallelism

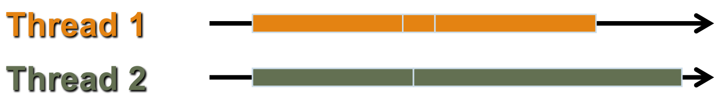
- Concurrent is not the same as parallel! Why?
- Parallel execution
  - Concurrent tasks *actually* execute at the same time
  - Multiple (processing) resources have to be available
- Parallelism = concurrency + "parallel" hardware
  - Both are required
  - Find concurrent execution opportunities
  - Develop application to execute in parallel
  - Run application on parallel hardware
- Is a parallel application a concurrent application?
- Is a parallel application run with one processor parallel? Why or why not?

Concurrency vs. Parallelism

- Concurrency: two or more threads are in progress at the same time:



- Parallelism: two or more threads are executing at the same time



Tunnel Vision by Experts!

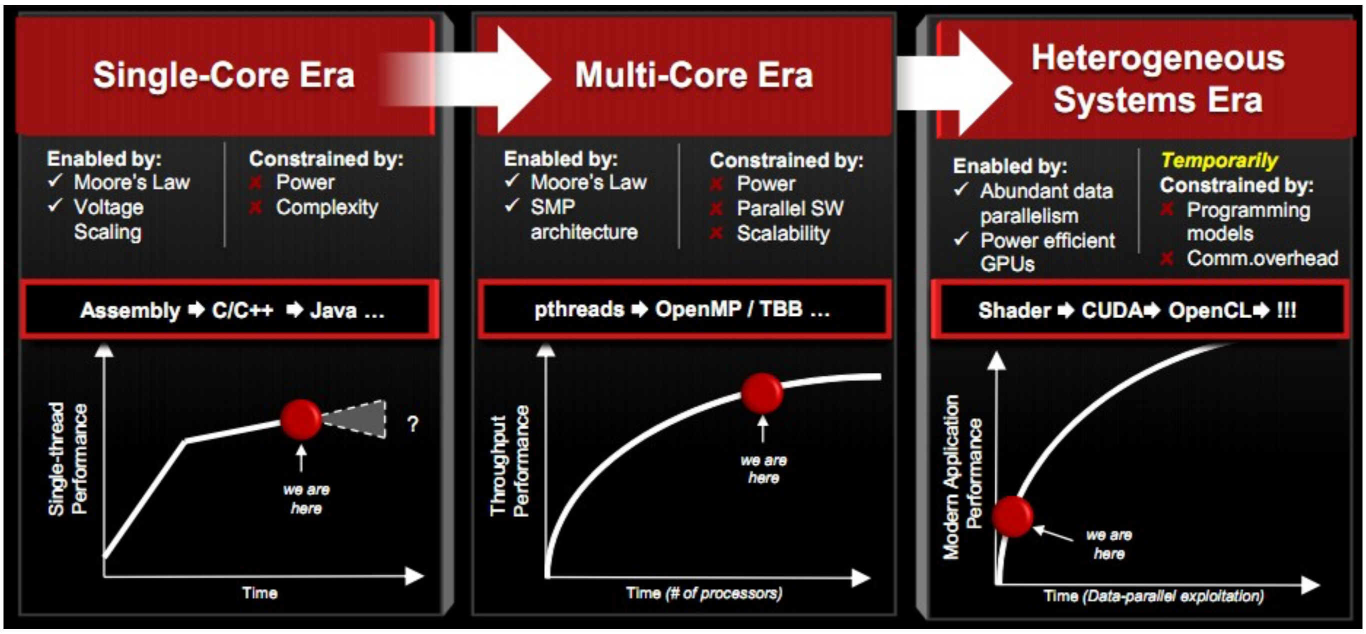
- "On several recent occasions, I have been asked whether parallel computing will soon be relegated to the trash heap reserved for promising technologies that never quite make it."
  - Ken Kennedy, CRPC Directory, 1994
- "640K [of memory] ought to be enough for anybody."
  - Bill Gates, chairman of Microsoft,1981.
- "There is no reason for any individual to have a computer in their home"
  - Ken Olson, president and founder of Digital Equipment Corporation, 1977.
- "I think there is a world market for maybe five computers."
  - Thomas Watson, chairman of IBM, 1943.

Why Parallelism (2020)?

- All major processor vendors are producing multicore chips
  - All machines are actually parallel machines
  - All programmers will be parallel programmers???
- New software model
  - Want a new feature? Hide the "cost" by speeding up the code first
  - All programmers will be performance programmers???
- Some may eventually be hidden in libraries, compilers, and high- level languages
- Rise of artificial intelligence and machine learning!

Supercomputing/Parallel Architecture Timeline

- Phase 1 (1950s): sequential instruction execution
- Phase 2 (1960s): sequential instruction issue
  - Pipeline execution, reservations stations
  - Instruction Level Parallelism (ILP)
- Phase 3 (1970s): vector processors
  - Pipelined arithmetic units
  - Registers, multi-bank (parallel) memory systems
- Phase 4 (1980s): SIMD and SMPs
- Phase 5 (1990s): MPPs and clusters
  - Communicating sequential processors
- Phase 6 (>2000): many cores, accelerators, scale, …


Single-Core Era
Enabled by:
✓ Moore's Law
✓ Voltage Scaling
Constrained by:
✗ Power
✗ Complexity
Assembly ➔ C/C++ ➔ Java …
Single-thread Performance
?
we are here
Time
Multi-Core Era
Enabled by:
✓ Moore's Law
✓ SMP architecture
Constrained by:
✗ Power
✗ Parallel SW
✗ Scalability
pthreads ➔ OpenMP / TBB …
Throughput Performance
we are here
Time (# of processors)
Heterogeneous Systems Era
Enabled by:
✓ Abundant data parallelism
✓ Power efficient GPUs
Temporarily Constrained by:
✗ Programming models
✗ Comm.overhead
Shader ➔ CUDA➔ OpenCL➔ !!!
Modern Application Performance
we are here
Time (Data-parallel exploitation)