CPS343 Parallel and High Performance Computing

Spring 2020

General Information

Meeting Time and Place

Monday, Wednesday, Friday 2:10–3:10 p.m., KOSC 125 (KOSC 244 as noted on schedule).

Professor

Dr. Jonathan Senning, 246 Ken Olsen Science Center
978-867-4376, *jonathan.senning@gordon.edu*

Office Hours

Monday & Wednesday: 3:20–4:20 p.m.,
Tuesday & Thursday: 1:00–3:00 p.m.,
and by appointment.

Required Text

- ***Multicore and GPU Programming: An Integrated Approach***, Gerassimos Barlas, Morgan Kaufman/Elsevier, 2015.

Recommended and Reference Texts

- ***Pro Git*** 2nd Edition, Scott Chacon and Ben Straub, Apress, 2014.
  https://git-scm.com/book
- ***Introduction to High-Performance Scientific Computing***, Victor Eijkhout, 2016.
  https://bitbucket.org/VictorEijkhout/hpc-book-and-course/src/default/EijkhoutIntroToHPC.pdf
- ***Parallel Computing for Science and Engineering***, Victor Eijkhout, 2017.
  https://bitbucket.org/VictorEijkhout/parallel-computing-book/src/default/EijkhoutParComp.pdf
- ***Designing and Building Parallel Programs***, Ian Foster, 1995.
  http://www.mcs.anl.gov/~itf/dbpp/text/book.html
- ***MPI: The Complete Reference***, Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, and Jack Dongarra, 1996.
  http://www.netlib.org/utk/papers/mpi-book/mpi-book.html

Prerequisites

Ability to program in C/C++ and Python. Knowledge of computer organization, parallel programming constructs such as Java threads, and experience with linear algebra/matrices will all be helpful.

Online Materials

Online materials associated with this class can be found on the departmental web server at *http://www.math-cs.gordon.edu/course/cps343* and on Blackboard.

Academic Accommodations

Our academic community is committed to providing access to a Gordon education for students with disabilities. A student with a disability who intends to request academic accommodations should follow this procedure:

1. Meet with a staff person from the Academic Success Center (ASC) and provide them with current documentation of the disability;
2. Obtain a Faculty Notification Form from the ASC, listing appropriate accommodations; and
3. Submit this form to professors and discuss those accommodations with them, ideally within the first two weeks of classes.

Some accommodations need more time to arrange so communicating early in the semester is important. For more information consult http://www.gordon.edu/academicaccessibility or email asc@gordon.edu.

Academic Dishonesty

Academic dishonesty is regarded as a major violation of both the academic and spiritual principles of this community and may result in a failing grade or suspension. Academic dishonesty includes plagiarism, (see Plagiarism in Student Handbook), cheating (whether in or out of the classroom), and abuse or misuse of library materials when such abuse or misuse can be related to course requirements.

Course Description

Introduction

Three decades ago high performance computing was restricted to research universities, government labs, and big business. Supercomputers were built using specialized designs and hardware and were expensive to acquire and maintain. Today, however, virtually all computers have multi-core processors that can carry out tasks simultaneously. Parallel clusters built with commodity hardware and running open-source software can be built for a fraction of the cost of yesterday's supercomputers. In the late 2000's GPU (graphics processing unit) hardware began to be used for general numeric computation. Specialized General Purpose Graphics Processing Unit (sometimes denoted GPGPU, but usually just GPU even when not used for graphics) and MIC (Many Integrated Core) cards, both called “accelerators,” are now available and can be placed in a workstation that allow one to have a “desktop supercomputer.”

Taking advantage of the various forms of parallel hardware available today often requires specialized problem formulation and programming. Modern supercomputers can have multiple types of parallel hardware, complex interconnect systems, and hundreds or thousands of nodes. Managing them is a complex task, and special software software has been developed to help. On the software side, some algorithms are easily parallelized while others require clever restructuring to attain even a moderate level of parallelism. Modern programming languages and compilers are able to take some advantage of certain types of parallelism automatically, but careful, thoughtful, and creative programming is still required in many instances.

This course is designed to build on the concepts of multiprocessing and multi-threaded programming introduced in first and second year courses. Our focus will be high performance computing (HPC), which is applied to problems that require large amounts of computation, memory, or both. *Importantly,* *however, the understanding and skills we develop are generally applicable in any situation where execution time and hardware resources are limiting factors.*

Course Content and Objectives

We will cover the following topics, and perhaps others as well.

- Introduction and History: What is parallel and high performance computing?
- Tools for parallel and high performance computing (OpenMP, MPI, CUDA, OpenACC, scripting, numerical libraries, large file support via HDF5, etc.)
- High performance issues (memory hierarchy and caching, bandwidth, data I/O)
- Parallel computation issues (partitioning, synchronization, load balancing)
- Survey of parallel solutions to problems from areas such as differential equations, linear algebra, sorting, and searching.

Students completing this course will be able to design and implement parallel programs on multi-core machines, parallel clusters, and GPU/accelerator architectures. More specifically, they will be able to:

- analyze a programming task and identify what portions admit a parallel implementation
- use OpenMP to develop applications for multi-core computers
- use the MPI standard to develop applications for clusters
- use CUDA, OpenACC and/or Thrust to develop applications for GPU hardware

Procedure and Workload Expectation

Class will be a blend of lecture and practical, hands-on activities. Reading will be assigned regularly which you are expected to complete prior to the start of class. There will be written and computational homework assignments, several small projects, and a final project with a topic chosen by the student.

For each semester hour of credit, students should expect to spend a minimum of 2–3 hours per week outside of class in engaged academic time. This time includes reading, writing, studying, completing assignments, lab work, or group projects, among other activities.

Mobile Device Policy

Laptops, tablets, and other mobile devices may be used only when appropriate for the current classroom activity. You may not use a mobile phone or other device for texting or otherwise communicating with others during class. This activity prevents you from fully concentrating on our topic and is distracting to those around you, including the professor.

Course Requirements

Attendance and Participation

Regular and consistent attendance is expected. You are expected to have completed the assigned reading before class. You should read for comprehension and expect to discuss the content during class.

Homework Assignments

Homework will be assigned frequently. You are permitted to discuss the problems with one another, but the written work you turn in should reflect your own understanding of the material. Some of the homework assignments will involve programming and some may involve writing short reports.

Projects

Four programming projects will be assigned. The first will focus on analyzing program performance and working with large data files. The next three will focus on three main parallel architectures: shared memory multi-core processors, multiple processors with distributed memory, and GPU accelerators.

[Planned] An individual final project will selected in consultation with the professor. It will allow you to explore in greater depth an application of parallel programming that is interesting to you. Each student will make a presentation about their project to the class during the final exam period.

[Revised] Rather than complete a final programming project, students will prepare a 5 minute video presentation. You will select a topic in consultation with the professor, do any necessary research & write any necessary demonstration code, and then make a video to share with the class. Topics can be chosen from nearly any area of HPC but should be fairly narrow and focused. For example, you might talk about how HPC has been used to help deal with the COVID-19 pandemic, focus on a significant feature of MPI or Cuda that we didn't explore in our course work, or just about anything else.

Quizzes

Approximately seven quizzes will be during the semester **(Revised to four quizzes)**. These primarily will cover the assigned reading, but may also be based on class presentations, homework assignments, or projects.

Examinations

There will be two exams each covering roughly one half of the course material.

Grading Procedure

Your final average will be computed using the following table:

Component	**Planned Percentage**	**Revised Percentage**
Class preparedness and participation	10%	10%
Written assignments	15%	20%
Quizzes	10%	5%
Programming projects	30%	40%
Final project and presentation	15%	5%
Midterm exam	10%	10%
Final exam	10%	10%

The following table shows the correspondence between the final average and the letter grades that will be assigned.

100 – 96	A+	87 – 84	B+	75 – 72	C+	63 – 60	D+
95 – 92	A	83 – 80	B	71 – 68	C	59 – 56	D
91 – 88	A–	79 – 76	B–	67 – 64	C–	55 – 52	D

Tentative Schedule

Day	**Date**	**Topic**	
Wednesday	January 15	*Introduction*	
Friday	January 17	*A canonical problem: matrix-matrix multiplication*	
Wednesday	January 22	*Introduction to HPC*	**(Meet in KOSC 244)**
Friday	January 24	*A brief history and overview of HPC*	
Monday	January 27	*Performance metrics, prediction, and measurement*	
Wednesday	January 29	*Debugging and profiling programs*	**(Meet in KOSC 244)**
Friday	January 31	*Dense matrix algebra and libraries*	
Monday	February 3	*Dealing with data: data files and HDF5*	
Wednesday	February 5	*Reading and writing HDF5 files*	**(Meet in KOSC 244)**
Friday	February 7	*A model HPC problem: Finite difference solution of the unsteady heat equation in multiple dimensions*	
Monday	February 10	*Parallel algorithm analysis and design*	
Wednesday	February 12	*Memory hierarchy & data organization*	**(Meet in KOSC 244)**
Friday	February 14	*Parallel algorithm analysis and design; Parallel architectures*	
Monday	February 17	*Shared memory programming: threads, semaphores & monitors*	
Wednesday	February 19	*Using threads and OpenMP*	**(Meet in KOSC 244)**
Friday	February 21	*Shared memory programming made easy: OpenMP*	
Monday	February 24	*Distributed memory programming: Introduction to MPI*	
Wednesday	February 26	*Cluster computing with MPI*	**(Meet in KOSC 244)**
Friday	February 28	*MPI collective communication*	
Monday	March 2	*MPI derived datatypes*	
Wednesday	**March 4**	**Exam 1**	
Friday–Friday	March 6-13	***Quad Finals and Spring Break***	
Monday	March 16	*Reset Day*	

Day	**Date**	**Topic**	
Wednesday	March 18	*Project 2 Work Day*	
Friday	March 20	*MPI derived datatypes example: Cartesian grids*	
Monday	March 23	*Parallel I/O in MPI and HDF5*	
Wednesday	March 25	*Working with Cartesian grids in MPI*	**(Zoom, 2:10pm)**
Friday	March 27	*MPI example: Parallel sorting*	
Monday	March 30	(Professor ill – begin working on project 3)	
Wednesday	April 1	*Parallel sorting with MPI on Canaan cluster*	**(Zoom, 2:10pm)**
Friday	April 3	*Introduction to GPU programming and CUDA*	
Monday	April 6	*CUDA memory types*	
Wednesday	April 8	*Introduction to CUDA*	**(Zoom, 2:10pm)**
Friday	April 10	***Good Friday***	
Monday	April 13	***Easter Monday***	
Wednesday	April 15	*Global and shared memory in CUDA*	**(Zoom, 2:10pm)**
Friday	April 17	*CUDA optimization*	
Monday	April 20	*CUDA optimization example: parallel reduction*	
Wednesday	April 22	*CUDA profiling and debugging*	
Friday	April 24	*Introduction to the Thrust template library*	
Monday	April 27	*Thrust algorithms*	
Wednesday	April 29	*Using Thrust*	**(Zoom, 2:10pm)**
Friday	May 1	*OpenACC: Accelerator programming made easy*	
Monday	May 4	*OpenACC* (continued)	
Wednesday	May 6	*Using OpenACC*	**(Zoom, 2:10pm)**
Wednesday	May 13	**2:30–3:30 p.m. Exam 2** **3:30–4:30 p.m. Final Project Presentations**	