15-418/15-618: Parallel Computer Architecture and Programming

Brian Railing	Nathan Beckmann
Gates-Hillman Center 6005	Gates-Hillman Center 9021
`www.cs.cmu.edu/~bpr`	`www.cs.cmu.edu/~beckmann`
`bpr@cs.cmu.edu`	`beckmann@cs.cmu.edu`

Carnegie Mellon University
Spring 2021

Organization

Class web page:	`www.cs.cmu.edu/~418`
Piazza:	`www.piazza.com/cmu/spring2021/15418618/home`
Autolab:	`https://autolab.andrew.cmu.edu/courses/15418-s21`
Lecture:	Monday, Wednesday, and Friday 4:00 – 5:20 REO
Office Hours:	Please see the class web page for instructor and TA office hours.

Course Summary

From smart phones, to multi-core CPUs and GPUs, to the world's largest supercomputers, parallel processing is ubiquitous in modern computing. The goal of this course is to provide a deep understanding of the fundamental principles and engineering trade-offs involved in designing modern parallel computing systems as well as to teach parallel programming techniques necessary to effectively utilize these machines. Because writing good parallel programs requires an understanding of key machine performance characteristics, this course will cover hardware design and how that affects software design.

Assumed Background

Introduction to Computer Systems (15-213 or equivalent) is a strict prerequisite for this course. We will build directly upon the material presented in 213, including memory hierarchies, memory management, basic networking, etc.

Students are expected to be strong C/C++ programmers as there will be exposure to a variety of "C-like" parallel programming languages in this course. We expect students to be able to read documentation of new languages and new programming environments on their own and to have the system skills to track down and fix problems on their own.

This course builds on much of the material presented in Parallel and Sequential Data Structures and Algorithms (15-210). Although this course is not a strict prerequisite, students who haven't taken this course may need to study some of its material on their own.

While Introduction to Computer Architecture (18-447) would be helpful for understanding the material in this course, it is not a prerequisite.

Getting Help

We will use the class website as the central repository for all information about the class.

For technical questions (lectures, exams, assignments), post a question on Piazza. By default, any question you post will be private to you and the instructors. We will put posts on Piazza answering some common questions. Be sure to check these before contacting an instructor.

We will use both Autolab and Gradescope to collect assignments and to manage grading.

If you want to talk to a staff member in person, the posted office hours are the best opportunity, as the they represent times when we guarantee that we will be in the location identified. If a meeting is needed outside of the office hours, please use email to arrange a time.

Your Involvement in the Class

Activities

Your participation in the class will include the following:

- *Attending lectures*. Although attendance will not be taken, you will be responsible for the material presented in class. We will arrange to have these lectures video taped. The slides for the lectures will be available from the class home page.
- *Lecture Quizzes*. For each lecture, after the first, there will be a short quiz available on Canvas for that lecture. The quiz will be available for 24 hours, and is designed to be a quick check-in on your learning. Completing the quiz (regardless of the score in Canvas) is considered full credit, and you are expected to complete 70% or more of the quizzes. There are no extensions for the quizzes.
- *Assignments*. Students will complete four assignments. Each will involve a combination of programming, benchmarking, analysis, and tuning. Assignment 1 will be performed individually. The remaining three assignments may be performed in groups of at most two. Although it is not necessary to have a partner in 15-418/618, it is highly recommended.
- *Final project*. Over the last six weeks of the course students will propose and complete a self-selected final project. The final project can be performed individually or in groups of two. Each team will present the project at a final poster session and produce a detailed write-up describing its work and results.
- *Exams*. There will be two synchronous exams.
- *Exercises*. There will be eight homework exercises covering the lecture material. These will be useful in preparing for the exams. Each exercise will be peer reviewed based primarily on effort. This

provides further connection to your peers and opportunity to evaluate answers, which will further support your learning through a differentiated mechanism. As a portion of the grade is based on reviewing your peers, there is no late submission mechanism, instead the lowest two scores will be dropped. Otherwise, your score is based on equal parts: submitting something reasonable, the maximum of your received scores, and completing the three assigned reviews.

Handin Policy

Each student is allotted a total of five late-day points for the semester. Late-day points are for use on programming assignments only. (They cannot be used for the final project or for exercises) Late-day points work as follows:

- A one-person team can extend a programming assignment deadline by one day using one point.
- A two-person team can extend a programming assignment deadline by one day using two points, either one point from each student, or two from one of the students.
- If a team does not have remaining late day points, late hand-ins will incur a 10% penalty per day
- No assignments will be accepted more than three days after the deadline.

Points and penalties are computed automatically by Autolab. You cannot decide to which assignments your points are applied.

Grading

Grading for the course will be based on a weighted average of the following:

Activity	Instance				Total
Assignments	Asst. 1 8	Asst. 2 12	Asst. 3 12	Asst. 4 8	40
Exams	Exam 1 12	Exam 2 12			24
Final Project					25
Quizzes and Exercises					11
TOTAL					100

Textbook

There is no required textbook for 15-418/618. However, you may find the following textbooks helpful. There are also plenty of great parallel programming resources available for free on the web.

- David E. Culler and Jaswinder Pal Singh, with Anoop Gupta, *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann, 1998

- John L. Hennessy and David A. Patterson, *Computer Architecture: A Quantitative Approach*. Sixth Edition, Morgan Kaufmann, 2017
- Jason Sanders and Edward Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*. 1st Edition. Addison-Wesley. 2010
- David Kirk and Wen-mei Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*. Third Edition. Morgan Kaufmann. 2016
- Brian Railing *Selected Lecture Notes for 15-418 / 618*. Available via Canvas. 2021.

Collaboration Policy

Students in 15-418/618 are absolutely encouraged to talk to each other, to the TAs, to the instructors, or to anyone else about course assignments. Any assistance, though, must be limited to discussion of the problems and sketching general approaches to a solution. Each programming project team must write its own code and produce its own writeup. Consulting another student's or team's solution (past or present) is prohibited on assignments.

You are encouraged to make use of the extensive material online, including documentation from companies, online forums such as Stackoverflow, and course material from other universities. However, you may not make use of any material specific to our assignments, such as old assignment solutions on Github. If you are uncertain about whether you can use some resource, post a question on Piazza, with a link to the material you would like to use.

You may not supply code, assignment writeups, or exams you complete during 15-418/618 to other students in future instances of this course. You may not make these items available (e.g., on the web) for use in future instances of this course (just as you may not use work completed by students who have taken the course previously). We encourage you to use public source control hosts, such as Github, for your assignments; however please be sure to make your programming assignment repositories private and to keep them private even when you have completed the course.

Students who are caught plagiarizing or cheating will be given a university-level Academic Integrity Violation (AIV). The penalty for an AIV will range from getting a -100% score for the particular assignment to failing the class.

Students who provide resources to students in future instances of the course can be subject to a retroactive AIV violation in this course. This can lead to a change in grade and even a revocation of your diploma.

Acknowledgements

Special thanks to the Intel Corporation, the NVIDIA Corporation, and to DELL for equipment donations and/or financial support for course development. Todd Mowry created the original version of 15-418, and Kayvon Fatahalian contributed greatly to the lectures and the assignments.