# ECE 563: Programming Parallel Machines

## Spring 2019

*last updated January 29 (instructor's Tuesday office hours changed)*

| | |
|---|---|
| Instructor | Prof. S. Midkiff |
| Email | smidkiff@purdue.edu (put **ECE 563** in the subject line) |
| Office location | EE 310 |
| Office Hours | Tuesdays 2:10 to 3:40, Wednesdays 11 – 12:30 |
| Secretary | Mary Ann Satterfield, EE 326B |
| Course Web Page | http://www.ecn.purdue.edu/~smidkiff/ece563 |

**Prerequisites**: Proficiency in C or Fortran is desirable. Optionally, the shared memory part of the project can be done in Java, but the distributed memory part will need to be done in C/C++ or Fortran unless you want to venture into the world of Java *isolates*, which may require significant extra work on your part.

**Text:** There is no required text. The lectures will follow the book *Parallel Programming in C with MPI and OpenMP*, along with supplemental material I will provide. This book may be at the local bookstores.

**Additional References:** Additional information will be provided from time to time. You will be expected to know this material. Some of this material may be at a tutorial level, and some may be research papers.

**Piazza** will be used to answer online questions and distribute information. I prefer questions be submitted via piazza instead of email so that everyone in the class, including online students, has the benefit of the answers.

**Email:** If you send me email, put "563" in the subject line. This increases the chances that the email doesn't get put into my spam folder.

**Description:** This course will enable you to write programs targeting parallel machines, using any of the four major parallel programming paradigms: MPI (message passing), OpenMP (for shared memory machines), Pthreads thread programming (for shared memory machines.) and, GPU programming (using Cuda). We will also discuss system architecture and memory and programming language coherency models, as these are necessary to develop correct parallel programs and to debug parallel programs when they are not correct. We will also spend time on sequential performance optimizations.

This is not a course in parallel algorithms, although you will need implement one or more parallel algorithms for the course project.

**Objectives:** At the end of the course you will be able to

- Write a parallel program using MPI
- Write a parallel program using OpenMP
- Write a parallel program using explicit threads
- Write a GPU program using Cuda (assuming I can get GPU simulators installed for you to use.)
- Compute the performance, efficiency and performance of a parallel program
- Decide on the suitability of a parallel algorithm for a particular parallel programming model.

**Course Grading:**

I do not grade on a curve, thus it is possible for everyone to make an A, or everyone to make an F. The decision is largely yours. I do +/- grading. As you may well know, Purdue gives only 4 points for an A+, and less than four points for an A-, with a similar pattern for Bs and Cs. Therefore, if I spread my As across A+, A and A- the average GPA for the course would drop relative to not doing +/- grading. Therefore, I tend to use "-" grades for students that are very close to the next grade up, i.e. an A-, if given, would go to a student that had a B that was very close to an A.

| | | |
|---|---|---|
| Tests | 30% | (15% each two midterm exams)<br>The first exam will be a take home exam and involve programming and will done done after we finish MPI. It is tentatively scheduled for Feb. 22nd but may be delayed. The second exam will be held on April 22nd and will be in-class. |
| Class participation and exercises/homework | 30% | |

| | | |
|---|---|---|
| Project (Tentative) | 40% | The project will consist of a larger MPI/shared memory program for a more substantial algorithm.<br><br>**Students can, with my approval, do another project. I would like it to have an MPI and shared memory component.**<br><br>There will be a 5 - 10 page report that discusses both of these projects together and the good and bad points of your implementation.<br><br>Students may work in two-person teams, but each team member should participate in all parts of the project. |

**Regrading Policy:** I am willing to regrade tests if given the test and a reason for the regrade request. I am allowed to correct errors in your favor, and errors not in your favor.

**Academic Honesty:** You are expected to do all programming and homework assignments on your own or within the groups designated by the instructor. You may not copy files or solutions from other students/groups or have anyone do all, or some part, of any assignment for you.

You may discuss concepts and ideas with other students. You may answer general questions about programming language rules, help someone interpret an error message, or locate a bug. In general, if the instructor would provide a particular form of assistance, you may provide it too. When in doubt, check with the instructor.

Punishment for academic misconduct can be severe, including receiving an F in the course or being expelled from the University. By departmental rules all cases of cheating must be reported to the Dean's office, and I will do this. My personal feeling is that graduate students have passed beyond the need for mercy, and I will prosecute cheaters to the maximum extent possible.

**Tentative Course Schedule:** 45 lectures (effectively 15 weeks) plus a final exam. ***Note that exam dates are tentative, and may change.***

**Topics by number of lectures (approximate)**

- Introduction to parallelism and the course: 2 lectures
- Shared and distributed memory architectures, multicores, dependence, and the relationship to parallelism: 4 lectures

- Hardware and software coherence in shared memory systems. The focus will be on software memory models as 565 does a better job of teaching hardware coherence than I can hope to do: 3 lectures
- Sequential Optimizations - 3 lectures
- OpenMP and shared memory programming: 4 lectures
- Pthreads, Java and shared memory programming: 4 lectures
- MPI and distributed memory programming: 2 lectures
- GPU architecture and programming: 4 lectures
- Tuning applications and speedup theory, Amdahl's law, strong and weak scaling, etc. theory: 12 lectures
- Algorithms and techniques for fast reductions, recurrences, parallel prefix, divide and conquer, super linear speedup, etc.: 2 lectures
- Parallelizing compilers and their limitations: 1 lectures
- New programming models: some of Cilk, Stream, UPC, Galois, X-10: 3 or fewer lectures
- Tests (may be take home, may be in-class): 2 lectures

**Staying current with the material:** The goal of an instructor is to show you the direction. Learning the material is something you have to do internally. Reading and trying to understand the material, sometimes more than once, is necessary to gain a real understanding of the material.

**Exercises:** Homework will be assigned periodically. You will be graded on whether or not you turn in something that looks reasonable, not correctness. Solutions will be posted for homeworks, and you should talk to me during office hours if you are not sure about the correctness of your solution.

**Exams:** There will be two exams. All exams are comprehensive. For midterm exams, the emphasis is on the recently learned material. The in-class exam will be "open textbook and notes".

**Class Projects:** Each student or team will be assigned one or more algorithms to implement using MPI, OpenMP and threads. Students are expected to do the programming on their own. Students that have a problem that they are working on as part of their research that they would like to implement all, or part of, as the programming project should see the instructor. Also, I'm happy to have students suggest problems of interest to implement that are not among the projects I suggest -- in fact I encourage this. Students/teams will turn in a 5-10 page report at the end of the semester on their project.