Spring Syllabus for:
CSCI 6360: Parallel Computing
CSCI 4320: Parallel Programming

Prof. Christopher D. Carothers
Department of Computer Science
Rensselaer Polytechnic Institute
110 8th Street Troy, New York 12180
e-mail: chris.carothers@gmail.com or chrisc@cs.rpi.edu

Course Materials: Posted in Submitty "Course Materials" under CSCI4320 Course
Class Lecture: Via WebEx Events until Classroom Scheduled - URL(s) posted in Submitty Course Materials.

Office Hours: Via WebEx Personal Room:
https://rensselaer.webex.com/meet/carotc;
Mon./Thurs., 2:00 p.m. to 3:30 p.m. and by appointment.

January 10, 2022

1 Course Description and Textbook Information

This course is an introduction to parallel computing and programming. Topics include, but are not exclusively limited to:

- Introduction and Motivation: Amdahl's Law, and review of uni-processor memory and CPU organization.
- Parallel architectures: Message passing, shared-memory system, vector/SIMD and communications networks.
- Parallel Programming: CUDA and Message Passing Interface (MPI).
- Parallel Filesystems: `MPI_File` interface.
- Performance Analysis Tools: Tau.
- Other Parallel Programming Paradigms: MapReduce, Transactional Memory.
- Fault Tolerance
- Applications: from across the spectrum of computational science and engineering (CSE).

The overall goal for this course is for students to gain expert knowledge of the design, development and execution of massively parallel programs on the *AiMOS* supercomputer located at the *Center for Computational Innovations (CCI)*. This new supercomputer is a hybrid CPU/GPU architecture and so many of the programming assignments will involve GPU programming. This is a big shift in the course from previous offerings.

1.1 Prerequisite

The principle prerequisite for this class is CSCI 2500, *Computer Organization*. However, I understand that many of you come from fields of study that are outside of Computer Science. So below I have listed some prerequisites and course assumptions:

- **Some programming experience in FORTRAN, C, C++.** Java is great but not for HPC. You will have a choice to do your assignment in C, C++ or FORTRAN subject to the language support of the programming paradigm.
- **This course assumes you have never touched a parallel or distributed computer**. However, we do assume you have touched a computer and have some knowledge of Linux/Unix as all of our parallel computing systems only use the Linux OS.
- **You should possess a strong desire and love to write software.** While both theory and practice are presented in lecture, the class assignments and group project is focused on getting real programs to execute in parallel such that performance improvement is demonstrated.

1.2 Optional Textbooks

- *Introduction to Parallel Computing*, by Grama, Gupta, Karypis and Kumar. Make sure you have the 2nd edition. This book is available online at `Amazon.com`.
- *An Introduction to Parallel Programming*, by Peter S. Pancheo, 2011.

2 Performance Expectations

As a professor, student adviser and course instructor, I get asked, "Is Parallel Computing/Programming Hard?".

If you like writing software and you enjoy the challenging task of debugging programs that have a non-deterministic execution order when not properly made to execute in parallel, this is the course for you. Also, if you are a "performance junky" and like to tweak programs to make them run as fast as possible, this course is for you. If you are a weak programmer or you do not really enjoy writing software, then I would say this course is probably not your cup of tea.

If you find yourself getting "deadlocked" (pun intended) on an assignment and you are unable to make progress, please contact Prof. Carothers and/or the TA(s) early. Do not wait until the last minute. Also, do not try to stay up all night in a marathon debugging session. Try to work smarter by asking questions and getting help from the instructor and our class TA(s).

3 Graduate Teaching Assistants

We have the following Graduate TA assigned to our class.

1. **Ian Bogle. Email:** `boglei@rpi.edu`. Office hours: Via WebEx Personal Room, `https://rensselaer.webex.com/meet/boglei`; Time - Tuesday and Wednesday, 2-3 p.m.

4 Course Format and Schedule of Topics

This course is largely a lecture format where class readings will come from the conference, journal articles and technical reports. The anticipated order of topics is:

- Introduction and Motivation: Amdahl's Law, and review of uni-processor memory and CPU organization.
- Parallel architectures: Message passing, shared-memory system, vector/SIMD and communications networks.
- Parallel Programming: CUDA and Message Passing Interface (MPI).
- Parallel Filesystems: `MPI_File` interface.
- Performance Analysis Techniques & Tools:
- Other Parallel Programming Paradigms: MapReduce, Transactional Memory.
- Fault Tolerance
- Applications: Distributed AI/Machine Learning, Parallel Discrete-Event simulation, Neuromorphic Computing and other CSE applications.

5 Schedule of Homeworks, Project and NO CLASS days

All homeworks and class project will be submitted using `submitty.cs.rpi.edu` grading system. The approximate schedule of class assignments is as follows:

- Assignment 1 assigned on Thursday, January 13th, due on Monday, January 31st.
- Assignment 2 assigned on Monday, January 31st, due on Monday, February 14th.
- Assignment 3 assigned on Monday, February 14th, due on Monday, February 28th.
- Assignment 4 assigned on Monday, February, 28th due on Monday, March 14th.
- Assignment 5 assigned on Monday, March 14th, due on Monday, March, 28th.

- Project assigned on Monday March 28th, due on Wednesday, April 27th (Last Day of Classes).

The NO CLASS days are as follows:

- Monday, January 17th in observance of MLK Day.
- Monday, February 21st (President's Day) but takes place on Tuesday, February 22nd.
- Monday, March 7th and Thursday, March 10th for Spring Break.
- Monday, April 25th, Last Day of Class. No Lecture but hand-in last lecture summary and Q&A session for project.

6 Grading and Other Class Policies

- 50%: 5 programming assignments worth 10 pts each. Some maybe group assignments. Submitted using `submitty.cs.rpi.edu` grading system.
- 20%: 20 reading/lecture summaries (across approx. 25 lectures) each counts 1pt. Submitted using `submitty.cs.rpi.edu` grading system.
- 30%: Group project. Submitted using `submitty.cs.rpi.edu` grading system.

Attendance Policy: Attendance at lectures is not required, but it will be very hard to write the summary without having attended lecture.

Late Assignments Policy: Late assignments and summaries will not be graded. You will get a zero for that assignment, except under extenuating circumstances, such as illness, family death etc. If you are ill, please be prepared to provide a note from the health center or your own family physician.

Grade Modifiers Policy: Grade modifiers will be used in this class. Precise break points will be determine by overall class performance. Nominally, for example, you expect to earn a B- if your score is greater than 79.5 and less than 83.0, B if your score is greater than 83 and less than 86, B+ if your score is greater than 86 and less than 89.5. The similar modifier points occur for the A, C and D ranges subject to overall class performance except that there is no A+ nor is a D- allowed under the RPI Grade Modifier Policy.

Assignment Grading Criteria: Programming assignments are graded as follows: 15% for proper comments (e.g., each function should indicate what it does) and 85% for a correct working implementation. We typically divide the correctness points over key functions working. For example, *file reading - worth 10 points*, *file writing – worth 10 points*, and then *doing the calculation correctly – worth 65 points*. Note that programs that either don't compile or generate a "core dump" typically get no more than 20 points of the 85. **Thus, your max score for a "properly commented" program that fails in some fundamental way is only 35 points even if you spent 100 hours of time on it**.

Note: a more specific grading rubric will be made available on the `submitty` grading system used in the class.

7 Academic Integrity

While I strongly encourage you to form study groups and work together in learning this material, the programming assignments are to be done individually unless otherwise noted by the assignment/project specification. What this means is that you should do whatever is necessary to ensure your work remains your work. For example, in doing programming assignments you might want to prepend variable names with your initials. If during the grading process, it is determined that students shared or duplicated work, those students will automatically take a zero for the offense plus a 5 point total average deduction. For a second offense, the student or students involved will fail this course and a report will be sent to the Dean of Students office which could result in additional disciplinary action.

8 Learning Outcomes

By the end of this course, you will be able to:

1. *Apply the concept of the* **Amdahl's Law** *to the estimation of the fraction of a program that can be serialized and still yield a "good" speedup / program performance improvement.*
2. *Apply the concepts of a* **Parallel Computer Architecture** *by creating a parallel program that will maximize the performance of the parallel program when executed on that class of parallel computing systems.*
3. *Apply the concepts of* **Message Passing** *to the creation of a program that executes efficiently on this class of parallel computer architecture.*
4. *Apply the concepts of* **Threads** *to the creation of a program that executes efficiently on this class of parallel computer architecture.*
5. *Apply the concepts of* **CUDA** *to the creation of a program that executes efficiently on this class of parallel computer architecture.* Note that the concept of "threads" will be covered in the context of CUDA programming.
6. *Apply the concepts of* **Parallel File I/O** *to the creation of a parallel program that efficiently reads and writes data to disk.*
7. *Apply the concepts of* **Performance** *to the analysis of computer performance problems.*
8. *Apply the concepts of* **Performance Counters** *to the analysis of parallel program performance.*
9. *Apply the concepts from* **Various Research Papers** *to the development of a group research project and workshop/conference quality paper.*