CSCI 480/580: Parallel Computing

Course information
Spring 2016
Meeting time: T/Th 3:40-5:00PM
Location: Social Science 344
Final Exam: none

Course material/submissions/grades are in Moodle (http://umonline.umt.edu)

Instructor information
Instructor: Travis Wheeler
Office: Social Science 412
E-mail: travis.wheeler@umontana.edu
Phone: 406-243-6219
Office Hours:
Mon 11:00-12:30 (SS 412): General office hours
Or by appointment (see schedule at http://wheelerlab.org/calendar)
Sadly, I'm not available before class; I teach another class immediately prior to this one.

Course Overview
This course is an introduction to parallelism and parallel programming. Topics include the various forms of parallelism on modern computer hardware (e.g. SIMD vector instructions, GPUs, multiple cores, and networked clusters), with coverage of locality and latency, shared vs non-shared memory, and synchronization mechanisms (locking, atomicity, etc). We will address computer architecture at a high level, sufficient to understand the relative costs of operations like arithmetic and data transfer. We will discuss patterns that appear in essentially all programs that need to run fast, as well as tools for debugging correctness and performance of parallel programs. The emphasis will be on implementing actual parallel applications, rather than mere theory.

Course Prerequisites
CSCI 205 and 232, or instructor consent.

Regarding instructor consent: this course may prove useful for students from many departments and with different backgrounds. I will assume mature programming skills in a conventional (non-parallel) language, as well as enough mathematical skills to understand the problems and algorithmic solutions presented. Course projects will be implemented in C – I strongly recommend having familiarity with C prior to the beginning of class, for example using this online tutorial:
http://www.cprogramming.com/tutorial/c-tutorial.html.

Course Learning outcomes
- Able to recognize and analyze patterns/applications that benefit from parallelism
- Able to implement basic parallel computing in multithreaded environment (OpenMP, Pthreads)
- Able to implement basic parallel computing in GPU environment (CUDA)

- Able to implement basic SIMD computing approaches
- Able to implement basic parallel computing in a cluster environment (MPI)
- Able to analyze and measure performance of parallel computing systems, and to debug those systems
- Able to analyze the impact of latency and resource contention on throughput

Additionally, for graduate students:
- Experience presenting complex computer science (algorithm) methods and results to an audience

Required textbook
An Introduction to Parallel Programming, by Peter Pacheco

Other resources
An open source textbook by Norm Matloff:
http://heather.cs.ucdavis.edu/~matloff/158/PLN/ParProcBook.pdf
OpenMP
https://www.youtube.com/playlist?list=PLLX-Q6B8xqZ8n8bwjGdzBJ25X2utwnoEG
http://openmp.org/mp-documents/Intro_To_OpenMP_Mattson.pdf
http://openmp.org/mp-documents/Mattson_OMP_exercises.zip
CUDA
https://www.udacity.com/course/intro-to-parallel-programming--cs344
https://github.com/udacity/cs344

Grading

Undergraduate	
Homework:	35%
Exams:	35%
In-class activities	30%

Graduate	
Homework:	25%
Exams:	25%
In-class activities	25%
Work/Presentations	25%

Grade cutoffs:
Determined based on my opinion of the work of students at the boundary.

Tentative schedule (confirm in moodle)

Week	Content	Reading / Videos	Assignment (due)
Jan 25 – Jan 29	Intro OpenMP	Book: pp 1-18 OpenMP videos 1-6	
Feb 1 – Feb 5	OpenMP	Book: pp 209-232 OpenMP videos 7-17	
Feb 8 – Feb 12	OpenMP	Book: pp 232-259 OpenMP videos 18-27	
Feb 15 – Feb 19	GPU Parallelism basics	CUDA videos lesson 1 Book: pp 47-76	HW 1 (Tue)
Feb 22 – Feb 26	**Exam Tuesday** GPU **(no class)**	CUDA videos lesson 2	CUDA prog 1 (Th)
Feb 29 – Mar 4	GPU/CUDA	CUDA videos lesson 3+4	CUDA progs 2+3 (Th)
Mar 7 – Mar 11	GPU/CUDA	CUDA videos lesson 5+6	CUDA progs 4+5 (Th)
Mar 14 – Mar 18	GPU/CUDA	CUDA videos lesson 7 Stratton 2012	
Mar 21 – Mar 25	Pthreads	Book ch 4	CUDA progs 6 (Tu)
Mar 28 – Apr 1	**Exam Tuesday** MPI	Book ch 3	
Apr 4 – Apr 8	**Spring break**		
Apr 11 – Apr 15	MPI + SIMD / SSE	Book ch 6.1	HW 2 (Thu)
Apr 18 – Apr 22	Optimization Debugging	TBD	
Apr 25 – Apr 29	Overflow + Presentations		HW 3 (Thu)
May 2 – May 6	Presentations **Exam Thursday**		

(TBD = to be determined)

Late policy

Submissions for programming and homework assignments are due at the beginning of class. Late submissions will not be accepted. Every student will get one free extension on an assignment (programming or homework) for up to a week. You do not have to ask for this – just write that you are using your free extension when you turn it in. Don't waste this extension or feel obligated to use it; another extension will be given only in exceptional circumstances.

Exams

My exam strategy is as follows: all homework and exam problems will be drawn from the textbook, perhaps with minor modification. Thus the correct way to study for this course is to review these problems and figure out how to solve them. The more you work, the better your grade will be. Exams will be closed book.

Working in groups (homework)
The best way to learn the material is by solving problems. You are encouraged to work together - the best way to understand the subtleties of the homework problems is to argue about the answers. Each of you should look at all the problems independently, and not just divide the list in two parts each time. After discussing problems and coming up with solutions, you will each write up a separate submission. Though the ideas behind your solutions may be quite similar, the text should not be identical – demonstrate your command of the problem with a personalized solution. I retain the right to question you about the material turned in. If it is evident that you don't understand what you turned in, your grade will be lowered.

(Don't be a leech and let your partner do all the work. Unless you learn how to solve problems, you will get burned on the exams and thus for your final grade.)

Working in groups (programming assignments)
I encourage discussion with others regarding programming assignments, as well. As with homework, these should be high-level discussions. Code should be written independently. If I suspect copying or plagiarism, I will ask you to explain each piece of the code to me, possibly resulting in a reduced grade or removal from class.

Cheating
It should go without saying that academic dishonesty (including plagiarism and cheating) will not be tolerated. Consult the university's student conduct code for more details. I will follow the guidelines given there. I will seek out the maximum allowable penalty for any academic dishonesty that occurs in this course. If you have questions about which behaviors are acceptable, please ask me.

Disabilities
Students with disabilities are encouraged to meet with me to discuss *any* accommodations they require.

Electronic devices
Turn off your cellphone, or set it to vibrate during class. Take calls outside the classroom. Students texting during class will be asked to leave.

Personal contact
I like to establish personal contact in my classes. Please feel welcome to stop by my office (during office hours or otherwise) to ask questions or say hello.