CSE 6220 – Introduction to High Performance Computing
OMSCS

About

This course is a graduate-level introduction to parallel computing. Its goal is to give you the foundations to develop, analyze, and implement parallel and locality-efficient algorithms and data structures.

What You'll Learn

When we talk about scalability, "scale" has two senses: efficiency as the problem size grows and efficiency as the system size—as measured by the numbers of cores or compute nodes—grows.

To really scale your algorithm in both senses, you can start by reducing asymptotic complexity (remember your days as a CS 101 n00b?). But then you also need to reduce communication and data movement. This course is about the basic algorithmic techniques you'll need, especially for the latter.

The course videos focus on theory. But that only gets you so far. So, you will need to supplement this algorithmic theory with hands-on labs on parallel shared-memory and distributed-memory machines; otherwise, how will you know if something that works in theory also works well in practice?

The specific techniques you will encounter cover the main algorithm design and analysis ideas for three major classes of machines:

1. sequential or parallel machines with deep memory hierarchies (e.g., caches), via external memory models;
2. multicore and manycore shared memory machines, via the work-span (or "multithreaded DAG") model;
3. and distributed memory machines like clusters and supercomputers, via network models.

You will see these techniques applied to fundamental problems, like sorting, searching on trees and graphs, and linear algebra, among others. You will implement several of them on real parallel and distributed systems, using practical programming models such as Cilk Plus, OpenMP, MPI, or possibly others.

Why?

The "standard" undergraduate CS curriculum begins, and usually ends, with the sequential (or serial) random access machine ("RAM") model. This course helps to fill the gap between

algorithm design for serial RAM machines and real machines, which will always have multiple cores, multiple nodes, vector units, and deep memory hierarchies.

Prerequisites

You should be comfortable designing and analyzing basic sequential algorithms and data structures using "big-Oh-my" notation. That's the stuff you learned in a first or second course in algorithms and data structures, a la Georgia Tech's CS 3510 or Udacity's Intro to Algorithms.

You should also be comfortable programming in C or C++ for the programming assignments. Experience using command-line interfaces in *nix environments (e.g., Unix, Linux) is also helpful, though there are software carpentry materials that may help with that, a la https://software-carpentry.org, for instance.

Please also review the course readiness survey for CSE 6220. It includes some topics in calculus, linear algebra, probability, and algorithms, which are required in many undergraduate CS programs. It's not so much that you need to remember how to solve exactly those sorts of problems without looking things up. Rather, you should feel confident that you can make time to brush-up as needed. Being able to do so is strongly correlated with the level of mathematical maturity and independence necessary for this class.

Logistics

Your first and best source of questions and answers is the class discussion forum. We make all critical announcements there, including when assignments go out, reminders about when they are due, clarification of various policies, errata, and hints.

The teaching staff strongly prefers that, if you need to contact them, you post a private note to the instructors on Ed Discussions rather than sending email. (You'll get a much faster response by doing so.)

Office Hours

We will hold weekly office hours and record them for students that are unable to attend. You are responsible for information conveyed in them. Please see the class forums for details.

Assignments and Exams

Beyond the videos, there will be supplemental readings and a set of programming assignments. See the schedule for a list of assignments, their due dates, and their associated lessons.

Submissions. Programming assignments must be submitted on Canvas. We only consider the last submission for each assignment so please make sure you've validated and performance tested your solution before submission.

Please make note of this. We will not accept submissions in any other format (Ed posts, emails, etc).

Due date convention. For all assignments and exams, the posted due date should be interpreted as "23:59 anywhere on earth" (11:59 pm AOE) unless otherwise noted. For example, if an assignment is due on January 31, as long as there is any place on the planet Earth where it is 11:59 pm or earlier, your submission is on time. (You are responsible for taking signal transmission delays into account, especially if you are connecting from outer space.)

For the ultimate in precision timing, here's a handy AOE clock:
http://www.timeanddate.com/time/zones/aoe

Grading. For each programming assignment you submit, we will check your implementation against some test cases for correctness and we will also measure how fast your implementation is. The minimum grade for a correct (but possibly slow) implementation is roughly equivalent to a 'C'; to get a higher grade, your implementation should also be fast, with a 'B' meaning reasonable performance and 'A' for high performance.

But how do you know whether your implementation is slow or fast? We will post guidelines with each assignment.

Late policy. Programming assignments may be submitted up 72 hours after the official due date. Every 24 hour period results in a progressive 25% grade reduction. Any assignments submitted after 72 hours will receive zero credit. We have to enforce a hard limit so that we can grade the assignments and return results within a reasonable timeframe.

The Late policy may not be used for exams. No late exams will be graded or accepted.

Collaboration policy. All Georgia Tech students are expected to uphold the Georgia Tech Academic Honor Code. Honest and ethical behavior is expected at all times. All incidents of suspected dishonesty will be reported to and handled by the Dean of Students office. Penalties for violating the collaboration policy can be severe; alleged violations are adjudicated by the Dean of Students office and not by the instructor.

Collaboration on assignments is encouraged at the "whiteboard" level. That is, you may share ideas and have technical conversation, which we especially encourage on the class forums, but you must write and submit your own code.

Exams must be done individually but are open-book and "open-internet" (for looking up information only). That is, while taking the exam you are not allowed to use the internet or to actively get help by, for instance, posting questions or messaging with others. We will use Honorlock to administer the exams. (The students shown on the Honorlock website all look so happy, so it must be good, right?)

This shouldn't need to be said, but you may not use something like artificial intelligence to help write your exam solutions or complete your project assignments.

We require you to complete a quiz each term acknowledging you have read and understand these policies. Violating them will result in receiving a zero on the entire assignment or exam. You will also be ineligible for any score adjustments/curves for the entire term.

Please note the emphasis on the acknowledgement being required. You will not receive credit for any assignment or exam submitted before you complete it.

Regrade policy. We'll be happy to review your submissions and recalculate your grade, provided your request is specific and you submit your request (as a private Ed Discussions post) within 14 days of grades being released. The caveat with this though is your score has a possibility of going down if your time/speedup gets worse. If the assignment has a reference serial execution time, it will be recalculated at the time of regrading to ensure consistency.

The above holds similarly when asking for regrades on exam questions. If you ask us to regrade a question and we notice something that should have been deducted, but wasn't, then we reserve the right to adjust your score accordingly.

Regrades do not apply to performance scores for projects where you are scaled against other students.

Accommodation policy. If you have any accommodations you need to inform us during the first week of classes, and provide us with the detailed accommodation approval letter from the Office of Disability Services (or Dean's Office). We need to confirm during the first week of classes (by the close of registration) that we can accommodate your requests. If you don't get approval from us by the close of registration then we cannot accommodate your requests.

Course Grades

The grade cutoffs for the course will be as follows:

- A: [90%, 100%]
- B: [80%, 90%)
- C: [70%, 80%)
- D: [60%, 70%)
- F: [0%, 60%)

So, to guarantee an A, get 90% or better overall (not 89.9). To guarantee at least a B grade, get 80% or better overall, etc.

These cutoffs might be adjusted, but only in the downward direction (to make letter grades higher).

Changes and Adjustments

The instructors reserve the right to make changes as needed term to term. Any modifications will be shared during the first week of class.