Syllabus

High Performance Scientific Computing

University of Colorado Boulder

Course No. CSCI 4576/5576

Fall 2024 (August 27, 2024 through December 12, 2024)

Lectures on Tuesdays and Thursdays from 12:30pm-1:45pm in ECCE 283
Labs on Fridays from 1:25pm-3:05pm in ECEE 283

Instructor: Scott R. Runnels, Ph.D. ECOT 411
scott.runnels@colorado.edu 505-695-9241

Contents

1 Course Description

This course introduces high-performance computing ("HPC") systems, software, and methods used to solve large-scale problems in science and engineering. It will focus on the intersection of two separate aspects of scientific HPC: Codes that implement methods for performing complex simulations and HPC systems on which those codes run. Four classes of methods will be covered: Spatial discretization using meshes, particle methods, ray-tracing, and linear solvers. Each of these types of methods have different requirements and lead to different perspectives regarding data structures. They will be intersected with three elements of HPC: Shared-memory parallelism (or "threading" via OpenMP), distributed-memory parallelism (via MPI), and accelerator-based computing (e.g., on GPUs). The focus will be on how the methods are adapted to these lines of HPC to increase performance. Choice of data structures and strategies of coding will be key design decisions. Students will write and run software on a supercomputer, and will learn how to interact with the supercomputer environment, including managing batch jobs, queues, and allocations.

2 Recommended Prerequisites

APPM 4650 or CSCI 3656 or MATH 4650 or MCEN 3030.

3 Class Structure

3.1 Grading

This class will consist of lectures and laboratories. There are no exams in this class; all work for grading is through the laboratories' post-lab reports, some of which include a few straightforward by-hand homework type problems. There will be a new lab assignment most weeks, and correspondingly, post-lab reports will be due most weeks. Nearly all laboratory assignments will require some type of programming or execution of software on either a student's local computer or a CU Boulder supercomputer. Because programming is involved, it is possible for any student to encounter a bug that prevents them from getting the desired results. This problem is mitigated somewhat by the fact that most laboratory coding work will be done in teams of two (and sometimes three, depending on the number of students), where the teammates are randomly selected each week. In this pair-programming model, elusive bugs are more easily found. The instructor can also provide some guidance. Ultimately, however, students should take comfort in the fact that the instructor understands the nature of the "elusive bug." Thus, grading of the code is not based solely on the students producing a working version. It is also based on the process the students follow, in addition to the results they obtain. So when a student cannot find the bug, they are graded on how diligently they pursued the bug and how well they report their debugging strategy.

Generally, post-lab reports will be due either the Wednesday or Thursday of the week following the Friday lab. It is important for students to submit their best effort by the due date for two reasons. First, experience has shown the uncompleted labs will snowball and knowledge of those weeks' codes will grow stale, making it increasingly difficult for the student to excel on labs that are past due. Second, some lab assignments build on software that is supposed to be developed by students in the previous lab. The students will be given the required technology in that subsequent lab, to ensure everyone has an equal footing that week regardless of how they did in the previous lab. Clearly, this means students could simply copy the subsequent week's technology for the first lab assignment if they were allowed to turn it in late. So again, it is important that students submit their best effort on the date it is due.

3.2 Structure of the Lab Assignments

As stated above, most labs are done in pairs of randomly selected teammates, and most labs are accomplished in one week. However, there are a few weeks that have multi-week labs, and there are a few weeks where students work or report individually instead of on a team. When working on a team, only one student will be able to operate the computer at a given time. But it is expected that the team will work together on the coding

part of each laboratory to accomplish its goals. Students will often be required to share with each other all files they generate as a team during the Friday laboratory. Because lab teams will not remain constant throughout the semester, but will instead be shuffled, it is expected that each individual will have the opportunity to be the one writing code, while the other(s) serves as checks.

The students are not expected to complete all of the work for the lab during the Friday lab period. Finishing the coding and writing reports will occur outside of lab time. But the lab is important for developing understanding together as a team and asking the instructor for help and clarifications.

The laboratories will occur on Fridays and will be conducted via Zoom for remote participants and in person for those taking the lab in person.

3.3 Office Hours

Office hours will be scheduled using the results of a class scheduling survey.

4 Recommended Resources

- *Parallel and High Performance Computing* by Robert Robey and Yuliana Zamora, available from Manning publishers.
- CU Boulder Research Computing documentation

5 Estimated Schedule and Course Content

Note that the schedule and content are subject to change.

Week #1 Introduction and Basics (8/27 & 8/29)

In this first week, we acquire some basic skills to introduce the world of high-performance scientific computing.

How to use a supercomputer, compile nodes vs. compute nodes, queues, file systems, batch jobs, slurm

In-class development of a basic MPI code, demonstrating basic commands

Running a parallel code on a supercomputer

Math Primer: Review of the key mathematical ideas used in this course, derivatives, approximations to derivatives, gradients, systems of simultaneous equations

Environment Primer: Linux, C, accessing the supercomputer

Code from repository: `lookup`

Laboratory (5%): (no. 00) Students will log into a supercomputer, build a code on it, and run a parallel job on it.

Week #2 Spatial Discretization in 3D (9/3 & 9/5)

We will introduce diffusion and write a 3D diffusion code.

How diffusion works in nature

The diffusion equation

The finite difference method for solving the diffusion equation

A simple linear system solver

Writing a 3D finite difference code

Visualizing 3D results

Code from repository: `fd_3D`

Laboratory (5%): (no. 01) Students will complete a 3D finite difference code and perform an exercise with it.

Week #3 Application – MPI on Spatial Discretization (9/10 & 9/12)

This week, we apply MPI commands and the ability to run parallel problems to a numerical method that solves a partial differential equation.

In-class development of a finite difference code

In-class development of an iterative linear system solver

Parallelizing the finite difference code with MPI

Code from repository: `fd_mpi`

Laboratory (5%): (no. 02) Parallelize a finite difference code, including the linear solver, with MPI.

Week #4 Application – Particles in a Spatial Mesh (9/17 & 9/19)

This week, we apply our knowledge to a different kind of scientific computing application, one involving particles moving through a spatial mesh.

In-class development of a rudimentary particle transport code, where the particles have a prescribed velocity

Analyzing data structure and computational requirements of the particle transport code

Brainstorming on ways to parallelize it using MPI

Code from repository: `fp_mpi`

Laboratory (5%): (no. 03): Develop key part required to parallelize the prescribed velocity particle transport code using MPI.

Week #5 Combining two MPI Performance Requirements (9/24 & 9/26)

This week, we investigate what happens when we combine two different computing applications into one code and try to optimize performance.

In-class development of combined finite difference and particle-in-cell code, i.e., a ”PIC” code

Governing equations for electrostatic PIC code

Brainstorming on ways to parallelize it using MPI

Code from repository: `esPIC`

Laboratory (5%): (no. 04) Parallelize the PIC code using MPI.

Week #6 Details of Processor and Memory Architecture (10/1 & 10/3)

Distributed memory computing with MPI may be thought of as separate PCs operating on their own part of the problem, communicating when needed. But task parallelism, vectorization, and GPU programming require more intimate knowledge of CPUs and memory. This week, to prepare us for those aspects of HPC, we explore those details.

Structure of a multi-core CPU

Memory access through cache levels, lines

Cache misses, distribution of memory per core

Commands for exploring a computer’s architecture

Code from repository: none

Laboratory (5%): (no. 05) Students will use commands and concepts learned in class to learn about and report on the computational and memory architecture of Alpine.

Week #7 Measuring Performance (10/8 & 10/10)

This week, we learn how to measure and analyze the performance of our MPI parallel electrostatic PIC code (`esPIC`), including the linear solver.

Performance metrics

Analyzing a code for performance bottlenecks

Demonstrating how to remove bottlenecks

valgrind/callgrind, kcachegrind, perf

Code from repository: `esPIC`

Laboratory (10%): (no. 06): Write a report on the performance of the `esPIC` code using all of the metrics and tools learned in class. Identify and attempt to fix a bottleneck.

Week #8 Introduction to Threading and OpenMP (10/15 & 10/17)

This week, we learn about the difference between MPI programming, which is for distributed memory, and task parallel programming, which relies on shared memory.

Tools for ”threading”

Introduction to OpenMP

In-class development of a code using OpenMP directives

Code from repository: `omp_02`

Laboratory (5%): (no. 07) Students will complete a code that solves Laplace's equation using high-level threading, will measure its speed-up, and will explore methods for making it faster.

Week #9 Linear Solvers (10/22 & 10/24)

This week, we explore linear solvers in more depth, adding and parallelizing a new solver to the code base.

Cell-based matrix formation

Parallelization of Jacobi iteration without ghost nodes

The Conjugate Gradient algorithm

Non-linear systems

Successive Approximation and Newton-Raphson iteration

Code from repository: `solvers`

Laboratory (10%): (no. 08) Students will learn the conjugate gradient algorithm in coded form and will parallelize it using MPI.

Week #10 Combining Distributed and Shared Memory Parallelization (10/29 & 10/31)

This week, we combine shared- and distributed-memory parallelism in a nonlinear spatial discretization solver.

Newton-Raphson Iteration

Relaxation

Analyzing code for thread parallelism performance; Intel Advisor

Code from repository: `solvers_nonlinear`

Laboratory (10%): (no. 09) Students will analyze a code designed to solve a nonlinear spatial PDE under MPI to determine how to combine MPI and threading for performance enhancement.

Week #11 Parallel I/O (11/5 & 11/7)

This week, we learn about the design principles and methods for parallel input and output.

Parallel input

n-to-1, n-to-n, and n-to-m parallel output methods

MPIIO

Guest Lecture: Alpine Parallel File Systems

Code from repository: `transientDiffusion_1mat`

Laboratory (10%): (no. 10) Students will write an n-to-1 parallel output writer for a uniform spatial grid, i.e., the finite difference code from earlier, using MPIIO.

Week #12 Resource Management (11/12 & 11/14)

This week, we learn about how to manage large computations in an HPC time-sharing environment.

Restart dumps

Automatic restart

Guest Lecture: Alpine Queues and Partitions

Code from repository: `transientDiffusion_1mat` and `mySlurm`

Laboratory (10%): (no. 11) Students will add a restart capability to a 2D transient diffusion solver and complete an automatic restart script to go with it.

Week #13 Application – Ray Tracing (11/19 & 11/21)

This week, we learn about ray tracing, its application in scientific computing, and study how to optimize ray tracing performance.

Ray tracing theory and application

In-class development of a 3D ray-tracing code

Code from repository: `ovenWalls`

Laboratory (10%): (no. 12) This is an individual lab (not a team lab, but collaboration is encouraged). While it will be submitted for a separate grade, it is connected with the GPU Programming lab. Students are encouraged to work ahead over the break if they want. In the lab this week, students either complete the ray-tracing code developed in class or develop their own, to meet specified requirements.

Week #14 Break - Thanksgiving Week (11/26 & 11/28)

No class this week. Enjoy your break!

Laboratory (0%): None

Week #15 GPU Programming, Part 1 (12/3 & 12/5)

This week, we will introduce the GPU and develop an understanding of the various design constraints associated with it.

Guest Lecture: Opportunites in HPSC

Tools for GPU programming

Code from repository: `ovenWalls_acc`

Laboratory (2.5%): (no. 13) This is a two-week lab assignment. See next week for the actual lab assignment summary statement. This final lab will be an individual lab; students will not work on teams for this one, but will be encouraged to collaborate freely.

Week #16 GPU Programming, Part 2 (12/10 & 12/12)

This week, we will apply tools we learned last week to speed up one of the applications we have developed as part of this class.

Guest Lecture: Opportunities in HPSC

Tools for GPU programming

In-class application of GPU directives

Code from repository: `ovenWalls_acc`

Laboratory (2.5%): (no. 13) Students will work individually on this lab. Students will be using pragma directives to speed up a code using a GPU. They will use Alpine if they do not have an appropriate GPU on their local computer. Collaboration is encouraged.

6 Policies

Classroom Behavior

Students and faculty are responsible for maintaining an appropriate learning environment in all instructional settings, whether in person, remote, or online. Failure to adhere to such behavioral standards may be subject to discipline. Professional courtesy and sensitivity are especially important with respect to individuals and topics dealing with race, color, national origin, sex, pregnancy, age, disability, creed, religion, sexual orientation, gender identity, gender expression, veteran status, marital status, political affiliation, or political philosophy.

For more information, see the classroom behavior policy, the Student Code of Conduct, and the Office of Institutional Equity and Compliance.

Accommodation for Disabilities, Temporary Medical Conditions, and Medical Isolation

If you qualify for accommodations because of a disability, please submit your accommodation letter from Disability Services to your faculty member in a timely manner so that your needs can be addressed. Disability Services determines accommodations based on documented disabilities in the academic environment. Information on requesting accommodations is located on the Disability Services website. Contact Disability Services at 303-492-8671 or `DSinfo@colorado.edu` for further assistance. If you have a temporary medical condition, see Temporary Medical Conditions on the Disability Services website.

If you have a temporary illness, injury or required medical isolation for which you require adjustment, there is no need to notify the instructor except if you want an extension on a due date. In that case, notify the instructor as soon as possible so that an exception can be granted, accommodated, and documented.

Preferred Student Names and Pronouns

CU Boulder recognizes that students' legal information doesn't always align with how they identify. Students may update their preferred names and pronouns via the student portal; those preferred names and pronouns are listed on instructors' class rosters. In the absence of such updates, the name that appears on the class roster is the student's legal name.

Honor Code

All students enrolled in a University of Colorado Boulder course are responsible for knowing and adhering to the Honor Code. Violations of the Honor Code may include but are not limited to: plagiarism (including use of paper writing services or technology [such as essay bots]), cheating, fabrication, lying, bribery, threat, unauthorized access to academic materials, clicker fraud, submitting the same or similar work in more than one course without permission from all course instructors involved, and aiding academic dishonesty. Understanding the course's syllabus is a vital part in adhering to the Honor Code.

All incidents of academic misconduct will be reported to Student Conduct & Conflict Resolution: `StudentConduct@colorado.edu`. Students found responsible for violating the Honor Code will be assigned resolution outcomes from the Student Conduct & Conflict Resolution as well as be subject to academic sanctions from the faculty member. Visit Honor Code for more information on the academic integrity policy.

Sexual Misconduct, Discrimination, Harassment and/or Related Retaliation

CU Boulder is committed to fostering an inclusive and welcoming learning, working, and living environment. University policy prohibits protected-class discrimination and harassment, sexual misconduct (harassment, exploitation, and assault), intimate partner abuse (dating or domestic violence), stalking, and related retaliation by or against members of our community on- and off-campus. The Office of Institutional Equity and Compliance (OIEC) addresses these concerns, and individuals who have been subjected to misconduct can contact OIEC at 303-492-2127 or email `CUreport@colorado.edu`. Information about university policies, reporting options, and support resources including confidential services can be found on the OIEC website.

Please know that faculty and graduate instructors must inform OIEC when they are made aware of incidents related to these policies regardless of when or where something occurred. This is to ensure that individuals impacted receive outreach from OIEC about resolution options and support resources. To learn more about reporting and support for a variety of concerns, visit the Don't Ignore It page.

Religious Accommodations

Campus policy requires faculty to provide reasonable accommodations for students who, because of religious obligations, have conflicts with scheduled exams, assignments or required attendance. Please communicate the need for a religious accommodation in a timely manner. In this class, there are no exams or required attendance, and assignments may be submitted late by prior arrangement.

See the campus policy regarding religious observances for full details.

Mental Health and Wellness

The University of Colorado Boulder is committed to the well-being of all students. If you are struggling with personal stressors, mental health or substance use concerns that are impacting academic or daily life, please contact Counseling and Psychiatric Services (CAPS) located in C4C or call (303) 492-2277, 24/7.

Free and unlimited telehealth is also available through Academic Live Care. The Academic Live Care site also provides information about additional wellness services on campus that are available to students.

Compiled on October 29, 2024