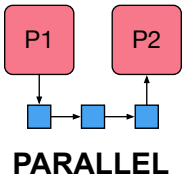


CISC 372: Parallel Computing

University of Delaware, Fall 2022

Syllabus

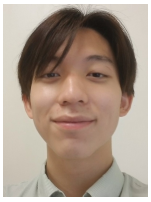
v. 1.0 (28-Aug-2022)

All information here is subject to change. Changes will be announced in class, Slack, and Canvas.

1. Fundamentals

Instructor: Stephen Siegel, `siegel@udel.edu`
Office hours: Tuesday 2:00–3:00 PM (Smith 432 or Zoom); Thursday 9:00–10:00 AM (Zoom only); or by appointment
Office Hours Zoom: `https://udel.zoom.us/j/97146059059`

Teaching Assistant: Zane Greenholt, `zgrnhlt@udel.edu`.
Office hours: Thursday 3:30–4:30 PM (Smith 203 or Zoom); Friday 12:00–1:00 PM (Smith 203 or Zoom); or by appointment
Office Hours Zoom: `https://udel.zoom.us/j/97420401921`



Teaching Assistant: Alex You, `youalex@udel.edu`
Office hours: Monday and Wednesday 9:00–10:00 AM (Smith 203 or Zoom); or by appointment
Office Hours Zoom: `https://udel.zoom.us/j/93550766708`

Class meeting times: 12:30–1:45 PM, Tue/Thu from Aug. 30 to Dec. 8 except Nov. 8, 22, and 24. Total number of classes: 27.

How class meets: In person, Alison Hall Room 133. If an exceptional situation arises, class may be held on Zoom.

What you need: Any reasonable desktop or laptop computer and a good, stable internet connection. You will be given accounts on the machines needed for the work in this class. You can connect to these machines using `ssh`.

Course Canvas page: `https://udel.instructure.com/`, site name `22F-CISC372-010`. Canvas will be used to post grades, videos, and solutions to homework, and for announcements and quizzes.

Slack workspace: `https://cisc372.slack.com`. Join this slack space using the following link: `https://join.slack.com/t/cisc372/signup`. We will use Slack for asynchronous discussion of material and to ask and answer questions. There are channels for the different course topics and homework assignments. For homework, feel free to ask questions, discuss problems generally, and give hints or pointers, but please do not give the answer to a problem or post code from your solution. Also, do not use Slack for discussion involving any confidential information, such as your grades or academic record—for these things, use email, phone, or Zoom conference.

Exams: Midterm 1: Tuesday, October 11. Midterm 2: Thursday, November 17. Both held in class. Final exam: Friday, December 16, 1:00–3:00 PM, Gore 318. *These dates are all tentative at this time. They may change. Changes will be announced on Canvas and Slack with plenty of warning.*

Version control: This class will use the Subversion version control system ("svn") for (1) distributing notes, code, and other material from the instructor to all students, and (2) submitting solutions to the homework. Each student will be assigned their own private repository for (2); do not share this repository password with anyone. Detailed instructions for these systems will be provided.

2. Learning Outcomes

Parallel computing arises whenever multiple computer processors or concurrent threads of control cooperate to solve a problem. This may involve the cores in a modern multicore processor, nodes in a networked cluster, or threads in a general-purpose graphical processor, among many other cases. Students completing this course will be able to...

- ...design, implement, and analyze parallel programs that can be executed on these various platforms.
- ...use several different parallel programming APIs to construct such programs, including the Message Passing Interface, OpenMP, POSIX threads, and CUDA.
- ...use a state-of-the-art supercomputer to submit and monitor jobs, including jobs at very large scale.
- ...accurately measure performance of parallel programs, and analyze performance and scalability.
- ...ensure that their parallel programs are functionally correct as well as performant.

The following is an approximate list of the topics that will be covered in this class. It is subject to change. The list is structured logically; the topics will not necessarily be covered in this order.

(1) Introduction
  (a) What is *concurrency*? What is meant by *parallel* or *distributed* computing?
  (b) Historical background: time-sharing operating systems, early parallel machines, ...
  (c) Architecture: Flynn's taxonomy, vector processors, multicores, GPUs, networks
  (d) Programming models: message-passing, shared variables
  (e) Practical issues: using parallel machines
(2) Message-passing concurrency
  (a) Basic message-passing concepts and the MPI model
  (b) Distributing data: block distribution of arrays
  (c) Point-to-point communication
  (d) Message ordering and synchronization
  (e) Collective communication
  (f) Nonblocking communication
  (g) Wildcards; determinism and nondeterminism in message-passing programs
  (h) Manager-worker pattern
(3) Performance
  (a) Performance models
  (b) FLOPs, "speedup", "peak" performance, "scalable", "efficiency"
  (c) Measuring performance, benchmarking
  (d) Analyzing performance data
  (e) Load balance
  (f) Amdahl's Law and Gustafson's Law
  (g) Weak scaling and strong scaling
  (h) Cache effects
(4) Shared-variable concurrency
  (a) Synchronization
  (b) Deadlock
  (c) Data races and race conditions
  (d) Barrier implemenations

(e) Reduction implementations
(f) The critical section problem
(g) Locks/mutexes
(h) Fairness
(i) Monitors, condition variables
(j) Fine-grained concurrency and concurrent data structures
(k) Languages/APIs: Pthreads, OpenMP, and CUDA

(5) Applications
(a) Adding numbers in an array
(b) Finding prime numbers (sieve)
(c) Diffusion (or "heat") equation
(d) Wave equation
(e) Gaussian elimination for solving a linear system $Ax = b$
(f) Matrix multiplication and other matrix operations
(g) Graph algorithms: searches, shortest path, etc.
(h) Sorting
(i) n-body problems
(j) Monte Carlo algorithms: computing π, finance, best-move poker
(k) Numerical integration

3. Texts and other resources

There is no required text for the class. However, if you want a good text, I recommend *An Introduction to Parallel Programming*, by Peter Pacheco. This is a very clear, readable introduction to the subject and discusses MPI, Pthreads, and OpenMP. Unfortunately, it does not discuss GPU programming.

`https://www.elsevier.com/books/an-introduction-to-parallel-programming/pacheco/978-0-12-374260-5`

Other resources:

- Pacheco's web site for his book, including source code for examples used in the book: `http://www.cs.usfca.edu/~peter/ipp/`
- Errata for Pacheco's book: `http://www.cs.usfca.edu/~peter/ipp/errata.pdf`
- *Parallel Programming for Multicore and Cluster Systems*, by Thomas Rauber and Gudula Rünger, is another excellent text. The online PDF version is available free to UD students, faculty, and staff through the UD subscription to SpringerLink: `https://link-springer-com.udel.idm.oclc.org/book/10.1007%2F978-3-642-37801-0`
- *Using MPI* is a excellent, up-to-date book on MPI: `http://www.mcs.anl.gov/research/projects/mpi/usingmpi/`
- Blaise Barney of Lawrence Livermore National Laboratory has written a number of high-quality tutorials which are free and available online:
  - an *Introduction to Parallel Computing*: `https://computing.llnl.gov/tutorials/parallel_comp/`
  - MPI tutorial: `https://computing.llnl.gov/tutorials/mpi/`
  - OpenMP tutorial: `https://computing.llnl.gov/tutorials/openMP/`
  - Pthreads tutorial: `https://computing.llnl.gov/tutorials/pthreads/`
- The MPI Standard, the ultimate authority on all things MPI: `https://www.mpi-forum.org/docs/`
- The OpenMP Specification: `https://www.openmp.org/specifications/`
- Pthreads is specified in the POSIX standard: `http://pubs.opengroup.org/onlinepubs/9699919799/`
- NVIDIA's

  - CUDA documentation: `http://docs.nvidia.com/cuda/`
  - CUDA C Programming Guide: `http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html`
- The Academic Enrichment center offers various kinds of support and can set you up with a tutor: `https://www.ae.udel.edu`
- Ask questions on Slack, and visit the Zoom office hours of the instructor and/or TA.

4. Machines

Two machines are available for use by this class. The first is `cisc372.cis.udel.edu` (also aliased as `grendel.cis.udel.edu`). This can be used for "light" work such as compilation, debugging, and small tests. The machine can be accessed by `ssh`:

```
ssh USER@cisc372.cis.udel.edu
```

where `USER` is your EECIS user name (usually, the same as your UD user name). You must obtain an EECIS account by filling out the form at `https://accounts.eecis.udel.edu` before you can use this machine. Note your EECIS password is probably different from your UD password. If you are off-campus, you must either use UD VPN or log on from `go.cis.udel.edu`.

The second machine is the *Bridges-2* supercomputer at the Pittsburgh Supercomputing Center, with 504 nodes, each with 128 CPU cores, and 24 GPU nodes, each with eight NVIDIA Tesla V100-32GB SXM2 GPUs. Documentation for Bridges-2 can be found here:

```
https://www.psc.edu/resources/bridges-2/
```

Instructions for accessing the machine will be provided later.

All tools necessary for the assignments have been installed, configured, and tested on these machines. Detailed instructions for accessing and using these machines will be provided in the homework assignments. Students may also use their own computers/laptops for development, debugging, and small runs; use Slack for pointers on installation and configuration.

5. Grading

Quizzes	5%
Participation	5%
Homework	40%
Midterm 1	15%
Midterm 2	15%
Final Exam	20%

Quizzes: Before each class, you must watch the video lecture and (optionally) read the instructor notes, and then take a short quiz on Canvas. The purpose of the quiz is to check that you really watched the material before class. It is important that everyone prepare for class in this way so that they can participate in the breakout problem sessions, described below. The quizzes and videos will be available at least one week before the class to give you plenty of time.

Participation: Participation points are awarded for attending class, submitting breakout problem solutions, asking/answering questions (including on Slack), and filling out the course evaluation. For online classes you must attend class by Zoom and remain logged on and responsive for the entire meeting to earn participation points.

In most class meetings you will work on exercises in small groups ("breakouts") based on the material presented in the last video. After an allotted time, the breakouts will end and everyone will rejoin to compare and discuss solutions with the instructor. The instructor will also be available during the breakout to help as needed. To earn participation points you just need to show you

tried your best on the problem(s). Your participation will be assessed by the commits you made to your personal repository during the class, as well as interactions with the professor in class.

If you are sick or think you might be: don't come to class. Just send an email to the professor and TAs (i.e., include all 3 email addresses in your message) as early as possible. You will be excused for participation on that day. If this happens an unusual number of times, we will speak with you.

Exams are more serious and you should not miss them unless illness really prevents you from showing up. If you have to miss an exam for serious illness you should email the instructor and TAs at least 24 hours before the exam, or, if that is not possible, by 8:00 AM the day of the exam at the absolute latest, with an explanation. Failure to follow this protocol may result in a 0 for the exam.

Homework: Homework assignments are to be completed either individually or with a partner. Partners will be assigned at the beginning of the semester. Do not attempt to look for solutions on the internet or anywhere else. Feel free to discuss general issues with your other classmates (e.g., on Slack), but do not give them your answers or share code from your solution. For help with homework, contact a TA or instructor, or attend their office hours.

Completed homework assignments are submitted using your personal Subversion repository. The due date and time will be clearly printed at the top of each assignment description. The solution that will be graded is whatever you have committed to your repository at that time. There is no need to submit solutions by Canvas, email, or in any other way.

Most of the homework problems involve writing programs. Here is some general guidance on how they are graded:

(1) The program must compile without errors. A program that cannot be compiled is not much use, and usually will get 0 points.
(2) The program should compile without warnings. Compiler warnings almost always indicate a problem with the program; you should resolve those issues before proceeding.
(3) The program must be correct, which we will measure by executing the program on a number of tests. We may give you a few of these tests, but not all of them—it is your job to come up with a good test suite and convince yourself your program is correct. We may also examine the code to see that it is correct in other ways, that it uses APIs correctly, etc.
(4) The program should be *performant*. Since the whole idea of parallel computing is to make programs run faster, if your parallel program performs the same (or worse) than a similar sequential program, it won't get much credit. If the performance does not improve with more processors or cores, it also will not get much credit in the performance category. More specific performance guidelines will be included in each assignment.

Late policy: Homework deadlines are strict. You can commit after the deadline, but a 10% penalty per day will be applied. You must notify the TAs if you wish to exercise this late option. After the solution has been published on Canvas and/or presented in class, solutions will no longer be accepted.

Grade disputes: If you have any question or disagreement about how something was graded, see/email the instructor or TA as soon as possible. Otherwise, two weeks after an assignment or exam has been graded and returned, the grade becomes "frozen" and will not be changed for any reason other than clerical error.

Letter grades are obtained from the numerical score as follows:

Minimum score	93	90	87	83	80	77	73	70	67	63	60	0
Letter Grade	A	A-	B+	B	B-	C+	C	C-	D+	D	D-	F

6. Academic honesty

Copying any other person's work (off the Internet, for example) without proper acknowledgment is **plagiarism**, a serious offense, and the one most common to computer science courses. Anyone that aids another student with work that is expected to be done without collaboration is as guilty as the person who seeks help. Both will be prosecuted. It is strongly recommended that you familiarize yourself with the University's Policy on Academic Honesty.

Any student who in any way facilitates another student's access to someone else's classwork is cheating, whether the classwork is written, electronic, verbal, or any other form.

Furthermore, there have been rare instances of people claiming that their work was stolen. In these cases it is very hard to determine if the person gave their work to someone else, or if it was taken without their permission. If there is any doubt, I will usually assume that the work was deliberately shared. It is thus your responsibility to safeguard your papers, your passwords, your computers, and any other means by which your work can be copied.

7. Class Policies

(1) Treat all participants in this course (students, TAs, and instructor) with respect. If a dispute arises, try to resolve it in a civil way, or ask a TA or the instructor for help.
(2) Support and help each other. Students in this course are not competing against each other, because the grading is based on a fixed standard, not by comparing students to each other. If everyone does "A" work, everyone will get an "A".
(3) Come to class, and come prepared to participate by watching the video and/or reading the notes before class.
(4) Come to class on time. If you are occassionally late, just try to enter the room and find a seat quietly and without disturbing the class.
(5) Follow all university Covid policies, including those concerning masks.
(6) All cellphones and other distracting devices should be turned completely off and stowed away completely out of view for the duration of the class. Laptops can be used in class only for class activities. Violations will result in negative participation points.
(7) When working in a team, all members should contribute roughly equally.
(8) When working with a partner on a homework set, it is **not** acceptable to divide the problems up, have each member solve their half, and then copy the partner's solutions for the other half. Instead, both members should work together on each problem. This can be done in person, by Zoom (screen-sharing is very effective in this regard), or by exchanging messages.
(9) No chewing gum in class.

You are encouraged to upload a picture of yourself to your Zoom and Slack profiles. This will help others recognize and get to know you.

8. My Accounts & Passwords Cheat Sheet

For your convenience: fill in this table and keep in a secure location.

System	Identifier	Username	Password or hint
UD	`*.udel.edu`		
EECIS	`*.cis.udel.edu`		
Public repository	`svn://vsl.cis.udel.edu/cisc372/372-2022F`		
Private repository	`svn://vsl.cis.udel.edu/cisc372/372-`		
ACCESS			
PSC	`bridges2`		
Slack	`https://cisc372.slack.com`		
Class Zoom	`https://udel.zoom.us/j/99612923161`		threads

9. Schedule

Class	Day	Date	Topic	Due
1	Tue	08/30	Introduction to Parallel Computing	
2	Thu	09/01	Systems: Unix, Subversion	
3	Tue	09/06	C 1: preprocessor, types, pointers	HW1 (Systems)
4	Thu	09/08	C 2: allocation, multi-dim. arrays	
5	Tue	09/13	MPI 1: message-passing model; hello, world; reduction, barrier	
6	Thu	09/15	Performance concepts, measurement and analysis	HW2 (C)
7	Tue	09/20	MPI 2: point-to-point communication	
8	Thu	09/22	MPI 3: array distributions and nearest neighbor communication	
9	Tue	09/27	MPI 4: collective operations	HW3 (MPI 1)
10	Thu	09/29	MPI 5: wildcards and nondeterminism, manager-worker	
11	Tue	10/04	Pthread 1: shared-memory model; hello, world; create, join	
12	Thu	10/06	Review	HW4 (MPI 2)
13	Tue	10/11	Midterm 1 (Unix, C, MPI) — *tentative*	
14	Thu	10/13	Pthread 2: data races, mutexes, critical sections	
15	Tue	10/18	Pthread 3: condition variables	
16	Thu	10/20	Pthread 4: implementing barriers and reductions	
17	Tue	10/25	OpenMP 1: pragmas, syntax; hello, world; parallel regions	
18	Thu	10/27	OpenMP 2: private vs. shared data, for loops	HW5 (Pthreads)
19	Tue	11/01	OpenMP 3: data races, reduction, worksharing	
20	Thu	11/03	OpenMP 4: synchronization, MPI/OpenMP hybrid programs	
21	Thu	11/10	CUDA 1: model, memory hierarchy; hello, world; blocks/threads	
22	Tue	11/15	Review	HW6 (OpenMP)
23	Thu	11/17	Midterm 2 (Threads) — *tentative*	
24	Tue	11/29	CUDA 2: 1d, 2d, 3d blocks, indexing	
25	Thu	12/01	CUDA 3: synchronization, dot product	
26	Tue	12/06	CUDA 4: matrix multiplication	
27	Thu	12/08	Future directions	HW7 (CUDA)
	Fri	12/09		HW8 (Hybrid)
	Fri	12/16	Final Exam (Gore 318 1:00–3:00 PM) — **Tentative**	

Read and sign the following, then scan this page and submit using Canvas as instructed.

I, _____ (print your full name), have read this syllabus and agree to abide by the policies described herein.

_____

Signature and date