

COMP322

| | | | | | |
|----------------------|------------------------------|----------------------------|--------------------------|----------------------------------|---------------------------------|
| Home | Office Hours | HJlib Info | edX site | Autograder Guide | Other Resources |
|----------------------|------------------------------|----------------------------|--------------------------|----------------------------------|---------------------------------|

COMP 322: Fundamentals of Parallel Programming (Spring 2023)

| | | | |
|--------------------------|---|-----------------------|---|
| Instructor: | Mackale Joyner, DH 2063 | TAs: | Mohamed Abeam, Chase Hartsell, Taha Hasan, Harrison Huang, Jerry Jiang, Jasmine Lee, Michelle Lee, Hung Nguyen, Quang Nguyen, Ryan Ramos, Oscar Reynozo, Delaney Schultz, Tina Wen, Raiyan Zannat, Kailin Zhang |
| Piazza site: | https://piazza.com/rice/spring2023/comp322 (Piazza is the preferred medium for all course communications) | Cross-listing: | ELEC 323 |
| Lecture location: | Herzstein Amphitheater | Lecture times: | MWF 1:00pm - 1:50pm |
| Lab locations: | Mon (Herzstein Amp), Tue (Keck 100) | Lab times: | Mon 3:00pm - 3:50pm (Raiyan, Oscar, Mohamed, Ryan, Michelle, Taha, Jasmine) Tue 4:00pm - 4:50pm (Tina, Delaney, Chase, Hung, Jerry, Kailin) |

Course Syllabus

A summary PDF file containing the course syllabus for the course can be found [here](#). Much of the syllabus information is also included below in this course web site, along with some additional details that are not included in the syllabus.

Course Objectives

The primary goal of COMP 322 is to introduce you to the fundamentals of parallel programming and parallel algorithms, by following a pedagogic approach that exposes you to the intellectual challenges in parallel software without enmeshing you in the jargon and lower-level details of today's parallel systems. A strong grasp of the course fundamentals will enable you to quickly pick up any specific parallel programming system that you may encounter in the future, and also prepare you for studying advanced topics related to parallelism and concurrency in courses such as COMP 422.

The desired learning outcomes fall into three major areas:

- 1) *Parallelism*: functional programming, Java streams, creation and coordination of parallelism (async, finish), abstract performance metrics (work, critical paths), Amdahl's Law, weak vs. strong scaling, data races and determinism, data race avoidance (immutability, futures, accumulators, dataflow), deadlock avoidance, abstract vs. real performance (granularity, scalability), collective & point-to-point synchronization (phasers, barriers), parallel algorithms, systolic algorithms.
- 2) *Concurrency*: critical sections, atomicity, isolation, high level data races, nondeterminism, linearizability, liveness/progress guarantees, actors, request-response parallelism, Java Concurrency, locks, condition variables, semaphores, memory consistency models.
- 3) *Locality & Distribution*: memory hierarchies, locality, data movement, message-passing, MapReduce

To achieve these learning outcomes, each class period will include time for both instructor lectures and in-class exercises based on assigned reading and videos. The lab exercises will be used to help students gain hands-on programming experience with the concepts introduced in the lectures.

To ensure that students gain a strong knowledge of parallel programming foundations, the classes and homework will place equal emphasis on both theory and practice. The programming component of the course will use the [Habanero-Java Library \(HJ-lib\)](#) pedagogic extension to the Java language developed in the [Habanero Extreme Scale Software Research project](#) at Rice University. The course will also introduce you to real-world parallel programming models including Java Concurrency, MapReduce. An important goal is that, at the end of COMP 322, you should feel comfortable programming in any parallel language for which you are familiar with the underlying sequential language (Java or C). Any parallel programming primitives that you encounter in the future should be easily recognizable based on the fundamentals studied in COMP 322.

Prerequisite

The prerequisite course requirements are [COMP 182](#) and [COMP 215](#). COMP 322 should be accessible to anyone familiar with the foundations of sequential algorithms and data structures, and with basic Java programming. [COMP 321](#) is also recommended as a co-requisite.

Textbooks and Other Resources

There are no required textbooks for the class. Instead, lecture handouts are provided for each module as follows. You are expected to read the relevant sections in each lecture handout before coming to the lecture. We will also provide a number of references in the slides and handouts. The links to the latest versions of the lecture handouts are included below:

- Module 1 [handout](#) (*Parallelism*)
- Module 2 [handout](#) (*Concurrency*)

There are also a few optional textbooks that we will draw from during the course. You are encouraged to get copies of any or all of these books. They will serve as useful references both during and after this course:

- [Fork-Join Parallelism with a Data-Structures Focus \(FJP\)](#) by Dan Grossman (Chapter 7 in [Topics in Parallel and Distributed Computing](#))
- [Java Concurrency in Practice](#) by Brian Goetz with Tim Peierls, Joshua Bloch, Joseph Bowbeer, David Holmes and Doug Lea
- [Principles of Parallel Programming](#) by Calvin Lin and Lawrence Snyder
- [The Art of Multiprocessor Programming](#) by Maurice Herlihy and Nir Shavit

Lecture Schedule

| Week | Day | Date (2023) | Lecture | Assigned Reading | Assigned Videos (see Canvas site for video links) | In-class Worksheets | Slides | Work Assigned | Work Due | Worksheet Solutions |
|------|-----|-------------|--|-----------------------------|---|-----------------------------|------------------------------|-------------------|-------------------|-------------------------------|
| 1 | Mon | Jan 09 | Lecture 1: Introduction | | | worksheet1 | lec1-slides | | | WS1-solution |
| | Wed | Jan 11 | Lecture 2: Functional Programming | | | worksheet2 | lec02-slides | | | WS2-solution |
| | Fri | Jan 13 | Lecture 3: Higher order functions | | | worksheet3 | lec3-slides | | | WS3-solution |
| 2 | Mon | Jan 16 | No class: MLK | | | | | | | |
| | Wed | Jan 18 | Lecture 4: Lazy Computation | | | worksheet4 | lec4-slides | | | WS4-solution |
| | Fri | Jan 20 | Lecture 5: Java Streams | | | worksheet5 | lec5-slides | Homework 1 | | WS5-solution |
| 3 | Mon | Jan 23 | Lecture 6: Map Reduce with Java Streams | Module 1: Section 2.4 | Topic 2.4 Lecture, Topic 2.4 Demonstration | worksheet6 | lec6-slides | | | WS6-solution |
| | Wed | Jan 25 | Lecture 7: Futures | Module 1: Section 2.1 | Topic 2.1 Lecture , Topic 2.1 Demonstration | worksheet7 | lec7-slides | | | WS7-solution |
| | Fri | Jan 27 | Lecture 8: Async, Finish, Computation Graphs | Module 1: Sections 1.1, 1.2 | Topic 1.1 Lecture, Topic 1.1 Demonstration, Topic 1.2 Lecture, Topic 1.2 Demonstration | worksheet8 | lec8-slides | | | WS8-solution |
| 4 | Mon | Jan 30 | Lecture 9: Ideal Parallelism, Data-Driven Tasks | Module 1: Section 1.3, 4.5 | Topic 1.3 Lecture, Topic 1.3 Demonstration, Topic 4.5 Lecture, Topic 4.5 Demonstration | worksheet9 | lec9-slides | | | WS9-solution |
| | Wed | Feb 01 | Lecture 10: Event-based programming model | | | worksheet10 | lec10-slides | | Homework 1 | WS10-solution |
| | Fri | Feb 03 | Lecture 11: Scheduling/executing computation graphs Abstract performance metrics | Module 1: Section 1.4 | Topic 1.4 Lecture , Topic 1.4 Demonstration | worksheet11 | lec11-slides | Homework 2 | | WS11-solution |
| 5 | Mon | Feb 06 | Lecture 12: GUI programming, Parallel Speedup, Amdahl's Law | Module 1: Section 1.5 | Topic 1.5 Lecture , Topic 1.5 Demonstration | worksheet12 | lec12-slides | | | WS12-solution |
| | Wed | Feb 08 | Lecture 13: Accumulation and reduction. Finish accumulators | Module 1: Section 2.3 | Topic 2.3 Lecture Topic 2.3 Demonstration | worksheet13 | lec13-slides | | | WS13-solution |
| | Fri | Feb 10 | No class: Spring Recess | | | | | | | |
| 6 | Mon | Feb 13 | Lecture 14: Recursive Task Parallelism | | | worksheet14 | lec14-slides | | | WS14-solution |
| | Wed | Feb 15 | Lecture 15: Data Races, Functional & Structural Determinism | Module 1: Sections 2.5, 2.6 | Topic 2.5 Lecture , Topic 2.5 Demonstration, Topic 2.6 Lecture, Topic 2.6 Demonstration | worksheet15 | lec15-slides | Homework 2 | | WS15-solution |

| | | | | | | | | | | |
|-----------|-----|--------|---|----------------------------------|--|-------------|--------------|-------------------|-------------------|---------------|
| | Fri | Feb 17 | Lecture 16: Limitations of Functional parallelism. Abstract vs. real performance. Cutoff Strategy | | | worksheet16 | lec16-slides | Homework 3 | | WS16-solution |
| 7 | Mon | Feb 20 | Lecture 17: Midterm Review | | | | lec17-slides | | | |
| | Wed | Feb 22 | Lecture 18: Midterm Review | | | | lec18-slides | | | WS18-solution |
| | Fri | Feb 24 | Lecture 19: Fork/Join programming model. OS Threads. Scheduler Pattern | | Topic 2.7 Lecture, Topic 2.7 Demonstration, Topic 2.8 Lecture, Topic 2.8 Demonstration, | worksheet19 | lec19-slides | | | WS19-solution |
| 8 | Mon | Feb 27 | Lecture 20: Confinement & Monitor Pattern. Critical sections Global lock | Module 2: Sections 5.1, 5.2, 5.6 | Topic 5.1 Lecture, Topic 5.1 Demonstration, Topic 5.2 Lecture, Topic 5.2 Demonstration, Topic 5.6 Lecture, Topic 5.6 Demonstration | worksheet20 | lec20-slides | | | WS20-solution |
| | Wed | Mar 01 | Lecture 21: Atomic variables, Synchronized statements | Module 2: Sections 5.4, 7.2 | Topic 5.4 Lecture, Topic 5.4 Demonstration, Topic 7.2 Lecture | worksheet21 | lec21-slides | | Homework 3 | WS21-solution |
| | Fri | Mar 03 | Lecture 22: Parallel Spanning Tree, other graph algorithms | | | worksheet22 | lec22-slides | Homework 4 | | WS22-solution |
| 9 | Mon | Mar 06 | Lecture 23: Java Threads and Locks | Module 2: Sections 7.1, 7.3 | Topic 7.1 Lecture, Topic 7.3 Lecture | worksheet23 | lec23-slides | | | WS23-solution |
| | Wed | Mar 08 | Lecture 24: Java Locks - Soundness and progress guarantees | Module 2: 7.5 | Topic 7.5 Lecture | worksheet24 | lec24-slides | | | WS24-solution |
| | Fri | Mar 10 | Lecture 25: Dining Philosophers Problem | Module 2: 7.6 | Topic 7.6 Lecture | worksheet25 | lec25-slides | | | WS25-solution |
| | Mon | Mar 13 | No class: Spring Break | | | | | | | |
| | Wed | Mar 15 | No class: Spring Break | | | | | | | |
| | Fri | Mar 17 | No class: Spring Break | | | | | | | |
| 10 | Mon | Mar 20 | Lecture 26: N-Body problem, applications and implementations | | | worksheet26 | lec26-slides | | | WS26-solution |
| | Wed | Mar 22 | Lecture 27: Read-Write Locks, Linearizability of Concurrent Objects | Module 2: 7.3, 7.4 | Topic 7.3 Lecture, Topic 7.4 Lecture | worksheet27 | lec27-slides | | | WS27-solution |
| | Fri | Mar 24 | Lecture 28: Message-Passing programming model with Actors | Module 2: 6.1, 6.2 | Topic 6.1 Lecture, Topic 6.1 Demonstration, Topic 6.2 Lecture, Topic 6.2 Demonstration | worksheet28 | lec28-slides | | | WS28-solution |
| 11 | Mon | Mar 27 | Lecture 29: Active Object Pattern. Combining Actors with task parallelism | Module 2: 6.3, 6.4 | Topic 6.3 Lecture, Topic 6.3 Demonstration, Topic 6.4 Lecture, Topic 6.4 Demonstration | worksheet29 | lec29-slides | | | WS29-solution |
| | Wed | Mar 29 | Lecture 30: Task Affinity and locality. Memory hierarchy | | | worksheet30 | lec30-slides | | Homework 4 | WS30-solution |
| | Fri | Mar 31 | Lecture 31: Data-Parallel Programming model. Loop-Level Parallelism, Loop Chunking | Module 1: Sections 3.1, 3.2, 3.3 | Topic 3.1 Lecture, Topic 3.1 Demonstration, Topic 3.2 Lecture, Topic 3.2 Demonstration, Topic 3.3 Lecture, Topic 3.3 Demonstration | worksheet31 | lec31-slides | Homework 5 | | WS31-solution |
| 12 | Mon | Apr 03 | Lecture 32: Barrier Synchronization with Phasers | Module 1: Section 3.4 | Topic 3.4 Lecture, Topic 3.4 Demonstration | worksheet32 | lec32-slides | | | WS32-solution |
| | Wed | Apr 05 | Lecture 33: Stencil computation. Point-to-point Synchronization with Phasers | Module 1: Section 4.2, 4.3 | Topic 4.2 Lecture, Topic 4.2 Demonstration, Topic 4.3 Lecture, Topic 4.3 Demonstration | worksheet33 | lec33-slides | | | WS33-solution |
| | Fri | Apr 07 | Lecture 34: Fuzzy Barriers with Phasers | Module 1: Section 4.1 | Topic 4.1 Lecture, Topic 4.1 Demonstration | worksheet34 | lec34-slides | | | WS34-solution |

| | | | | | | | | | | |
|----|-----|--------|---|--|--|-------------|--------------|--|-------------------|---------------|
| 13 | Mon | Apr 10 | Lecture 35: Eureka-style Speculative Task Parallelism | | | worksheet35 | lec35-slides | | | WS35-solution |
| | Wed | Apr 12 | Lecture 36: Scan Pattern, Parallel Prefix Sum | | | worksheet36 | lec36-slides | | | WS36-solution |
| | Fri | Apr 14 | Lecture 37: Parallel Prefix Sum applications | | | worksheet37 | lec37-slides | | | |
| 14 | Mon | Apr 17 | Lecture 38: Overview of other models and frameworks | | | | lec38-slides | | | |
| | Wed | Apr 19 | Lecture 39: Course Review (Lectures 19-38) | | | | lec39-slides | | Homework 5 | |
| | Fri | Apr 21 | Lecture 40: Course Review (Lectures 19-38) | | | | lec40-slides | | | |

Lab Schedule

| Lab # | Date (2023) | Topic | Handouts | Examples |
|-------|-------------|---------------------------------|--|----------|
| 1 | Jan 09 | Infrastructure setup | lab0-handout lab1-handout | |
| - | Jan 16 | No lab this week (MLK) | | |
| 2 | Jan 23 | Functional Programming | lab2-handout | |
| 3 | Jan 30 | Futures | lab3-handout | |
| 4 | Feb 06 | Data-Driven Tasks | lab4-handout | |
| - | Feb 13 | TBD | | |
| - | Feb 20 | No lab this week (Midterm Exam) | | |
| 5 | Feb 27 | Async / Finish | lab5-handout | |
| 6 | Mar 06 | Recursive Task Cutoff Strategy | lab6-handout | |
| - | Mar 13 | No lab this week (Spring Break) | | |
| 7 | Mar 20 | Java Threads | lab7-handout | |
| 8 | Mar 27 | Concurrent Lists | lab8-handout | |
| 9 | Apr 03 | Actors | lab9-handout | |
| 10 | Apr 10 | Loop Parallelism | lab10-handout | |
| - | Apr 17 | No lab this week | | |

Grading, Honor Code Policy, Processes and Procedures

Grading will be based on your performance on four homework assignments (weighted 40% in all), two exams (weighted 40% in all), lab exercises (weighted 10% in all), online quizzes (weighted 5% in all), and in-class worksheets (weighted 5% in all).

The purpose of the homework is to give you practice in solving problems that deepen your understanding of concepts introduced in class. Homework is due on the dates and times specified in the course schedule. No late submissions (other than those using slip days mentioned below) will be accepted.

The slip day policy for COMP 322 is similar to that of COMP 321. All students will be given 3 slip days to use throughout the semester. When you use a slip day, you will receive up to 24 additional hours to complete the assignment. You may use these slip days in any way you see fit (3 days on one assignment, 1 day each on 3 assignments, etc.). Slip days will be tracked using the README.md file. Other than slip days, no extensions will be given unless there are exceptional circumstances (such as severe sickness, not because you have too much other work). Such extensions must be requested and approved by the instructor (via e-mail, phone, or in person) before the due date for the assignment. Last minute requests are likely to be denied.

Labs must be submitted by the following Wednesday at 4:30pm. Labs must be checked off by a TA.

Worksheets should be completed by the deadline listed in Canvas so that solutions to the worksheets can be discussed in the next class.

You will be expected to follow the Honor Code in all homework and exams. The following policies will apply to different work products in the course:

- In-class worksheets: You are free to discuss all aspects of in-class worksheets with your other classmates, the teaching assistants and the professor during the class. You can work in a group and write down the solution that you obtained as a group. If you work on the worksheet outside of class (e.g., due to an absence), then it must be entirely your individual effort, without discussion with any other students. If you use any material from external sources, you must provide proper attribution.
- Weekly lab assignments: You are free to discuss all aspects of lab assignments with your other classmates, the teaching assistants and the professor during the lab. However, all code and reports that you submit are expected to be the result of your individual effort. If you work on the lab outside of class (e.g., due to an absence), then it must be entirely your individual effort, without discussion with any other students. If you use any material from external sources, you must provide proper attribution (as [shown here](#)).
- Homework: All submitted homework is expected to be the result of your individual effort. You are free to discuss course material and approaches to problems with your other classmates, the teaching assistants and the professor, but you should never misrepresent someone else's work as your own. If you use any material from external sources, you must provide proper attribution.
- Quizzes: Each online quiz will be an open-notes individual test. The student may consult their course materials and notes when taking the quizzes, but may not consult any other external sources.
- Exams: Each exam will be a open-book, open-notes, and open-computer individual test, which must be completed within a specified time limit. No external materials may be consulted when taking the exams.

For grade disputes, please send an email to the course instructors within 7 days of receiving your grade. The email subject should include COMP 322 and the assignment. Please provide enough information in the email so that the instructor does not need to perform a checkout of your code.

Accommodations for Students with Special Needs

Students with disabilities are encouraged to contact me during the first two weeks of class regarding any special needs. Students with disabilities should also contact Disabled Student Services in the [Ley Student Center](#) and the [Rice Disability Support Services](#).