Course Overview

COS 418/518: Distributed Systems
Lecture 2

Wyatt Lloyd, Mike Freedman

Learning Objectives

- Reasoning about concurrency
- Reasoning about failure
- Reasoning about performance

- Building systems that correctly handle concurrency and failure

- Knowing specific system designs and design components

Lectures

- M/W 10-1050a
  - Slides posted just before class

- Core topics, system design components, system designs…

- You should be **actively** thinking during the lectures

Precepts

- Th or Fr
  - Slides will be posted after all precepts
- Helps with assignments and/or reinforces lecture material
- **Actively** work through problems together

Course Staff Intro 1

- Wyatt Lloyd
- Mike Freeman
- Samuel Ginzburg (P1)
- Ashwini Raina (P2)
- Oleg Golev (P2a)
- Yinwei Dai (P3)
- Jianan Lu (P4)
- Leon Schuermann (518)

Grading

- Two exams (50%)
    - Midterm 25%
    - Final 25%

- Assignments (50%)
    - Five assignments 10% each

Exams

- In person exams
  - 2-3 hours: should not have time pressure
  - Closed book
- Midterm
  - Registrar will schedule during week of Mar 4–8
  - Hopefully Mar 5ish
  - Either 2 or 2.5 or 3 hours
- Final
  - Registrar will schedule somewhere in Exam Period
  - 3 hours

Exams

- Test learning objectives mostly using designs covered in lectures
- And tests knowledge of specific design patterns and designs
- Recipe for success:
  - Attend lecture and actively think through problems
  - Ask questions during lecture and afterwards in my office hours
  - Attend precept and actively work through problems
  - Complete programming assignments
  - Study lecture materials for specific design patterns and designs
  - Run the system designs in your mind / on paper and see what happens

Midterm Review

- Date, time, location TBD
- Go over midterm together
- One and only opportunity to argue for points

Programming Assignments

- Reinforce / demonstrate all learning objectives!

- 1: “MapReduce” in Go
- 2: Distributed Snapshots
- 3: Raft Leader Election
- 4: Raft Log Consensus
- 5: Key-Value Storage Service

Programming Assignments

- All are individual assignments
- We will give you all the tests
  - Non-determinism for later tests though
    - Each failed test run => lose 50% of points for a test
- Daily grading script emails your grade
  - We use exactly this grade

Programming Assignments- Late Policy

- 3 “free” late days
  - We assign them at the end of the semester to maximize your score
  - Used in 1 day granularity
  - Cannot use for last assignment (Deans date)

- Late policy
  - G = Grade you would have earned if turned in on time
  - 1 day late => 90%*G
  - …
  - > 5 days late => 50%*G

Policies: Write Your Own Code

Programming is an individual creative process. At first, discussions with friends is fine. When writing code, however, the program must be your own work.

Do not copy another person's programs, comments, or any part of submitted assignment. This includes character-by-character transliteration but also derivative works. Cannot use another's code, etc. even while "citing" them.

Writing code for use by another or using another's code is academic fraud in context of coursework.

Do not publish your code e.g., on github, during/after course!

Policies: Write Your Own Code

Programming is an individual creative process. At first, discussions with friends is fine. When writing code, however, the program must be your own work.

Do not copy another person's programs, comments, or any part of submitted assignment. This includes character-by-character transliteration but also derivative works. Cannot use another's code, etc. even while "citing" them.

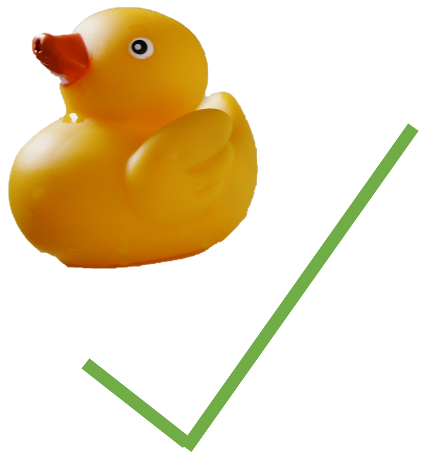
Writing code for use by another or using another's code is academic fraud in context of coursework.

Do not publish your code e.g., on github, during/after course!

Don't Plagiarize!

Policies: Generative AI is not Allowed

- Simple blanket ban on generative AI
  - Cannot use it to help write code
  - Cannot use it to help debug code
  - Cannot use it to discuss your design
  - Cannot use it at all!



Warnings

This is a 400-level course,
with expectations to match.

Warning #1:
Assignments are a LOT of work

- Assignment 1 is purposely easy to teach Go. Don't be fooled.
- Starting 3-4 days before deadline for later assignment => **Disaster.**
- Distributed systems are hard
    - Need to understand problem and protocol, carefully design
    - Can take 5x more time to debug than "initially program"
- Assignment #4 builds on your Assignment #3 solution, i.e., you can't do #4 until your own #3 is working! (That's the real world!)

Programming Assignment Statistics

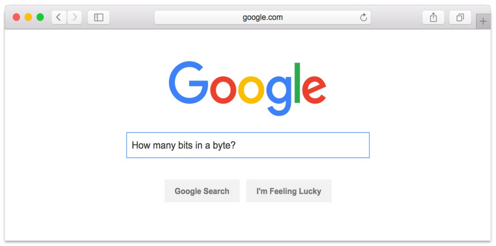
- Self Reported Hours Spent on Assignments: Median [Min-Max]

• 1-1:	2	[1-6]
• 1-2:	3	[1-8]
• 1-3:	4	[1-10]
• 2:	6	[2-15]
• 3:	12	[5-29]
• 4:	30	[10-100+]
• 5:	14	[3-25]

Warning #2:
Software engineering, not just programming

- COS 126, 217, 226 told you how to design & structure your programs. This class doesn't.
- Real software engineering projects don't either.
- You need to learn to do it.
- If your system isn't designed well, can be *significantly* harder to get right.
- Your friend: test-driven development
    - We'll supply tests, bonus points for adding new ones

Warning #3:
Don’t expect 24x7 answers



- Try to figure out yourself
- Ed discussion not designed for debugging
  - Utilize right venue: Go to office hours
  - Send detailed Q’s / bug reports, not “no idea what’s wrong”
- Staff are not on pager duty 24 x 7
  - Don’t expect response before next business day
  - Questions Friday night @ 11pm should not expect fast responses. Be happy with something before Monday.
- Implications
  - Students should answer each other
  - Start your assignments early!

Programming Assignments

- Recipe for disaster
  - Start day assignment is due
  - Write code first, think later
  - Test doesn’t pass => randomly flip some bits
  - Assume you know what program is doing

Programming Assignments

- Recipe for success
  - Start early (weeks early)
  - Think through a complete design
  - Progressively build out your design (using tests to help)
  - Checkpoint progress in git (and to github) frequently
  - Debug, debug, debug
    - Verify program state is what you expect (print it out!)
    - Write your own smaller test cases
    - Reconsider your complete design
  - Attend office hours

Programming Assignment Office Hours

- 20 hours of office hours per week!
- Schedule posted on Ed discussion
- Expectations
  - No: “You have a bug on line 17.”
  - Yes: Helping you think like they would think about a problem

Course Overview Conclusion

- Attend lecture, attend precept, think actively!
- Start programming assignments early, use the right strategy!
- Super cool distributed systems stuff starts Monday!