EECS 491
Introduction to Distributed Systems

Fall 2019

Harsha V. Madhyastha
Course Staff

Harsha Madhyastha
Instructor

Ayush Goel
GSI

Muhammed Uluyol
GSI
Agenda for Today

- Why study distributed systems?
- Course syllabus and logistics
What is a Distributed System?

Multiple interconnected computers that cooperate to provide some service.
Why study distributed systems?

- Ubiquitous
Applications as Services

Google Docs

Dropbox

Pokémon GO

Albert Einstein

Instagram
Amazon Web Services

Available via license CC BY 4.0 from https://www.researchgate.net/figure/Map-showing-the-location-of-Amazon-cloud-datacenters_fig1_301309025
Data Centers

- Spread services and data storage/processing across 100s of thousands of machines
Why Make a System “Distributed”?
Why Make a System “Distributed”?

- Build reliable systems with unreliable components
- Aggregate systems for higher capacity
  - CPU cycles, memory, disks, network bandwidth
  - Cost grows non-linearly
- Conquer geographic separation
  - Facebook and Google customers span the planet
- Customize computers for specific tasks
  - Example: cache server, speech-to-text conversion server
Why study distributed systems?

- Ubiquitous
- Hard to develop and reason about
Jeff Dean “Facts”

Jeff Dean writes directly in binary. He then writes the source code as a documentation for other developers.

Compilers don’t warn Jeff Dean. Jeff Dean warns compilers.

Jeff Dean builds his code before committing it, but only to check for compiler and linker bugs.
Challenge 1: Partial failures

“A distributed system is one where you can’t get your work done because some machine you’ve never heard of is broken.” – Leslie Lamport
Facebook’s Prineville Data Center

- Contents (approx.):
  - 200K+ servers
  - 500K+ disks
  - 10K network switches
  - 300K+ network cables

- At any instant, likelihood that all components correctly functioning?
Partial failures make it hard to sort 100 TB!

2010, 0.582 TB/min

**TritonSort**
100 TB in 10,318 seconds
47 nodes x
(2 Quadcore processors, 24 GB memory, 16x500GB disks)
Cisco Nexus 5020 switch

Alex Rasmussen, Radhika Niranjan Mysore,
Harsha V. Madhyastha, Michael Conley,
George Porter, Amin Vahdat,
University of California, San Diego

Alexander Pucher
Vienna University of Technology
Challenge 2: Ambiguous failures

- ATM does not receive response within 5 secs
  - Potential causes?
  - Problems if ATM resends request after 5 secs?
Challenge 2: Ambiguous failures

- If a server doesn’t reply, how to tell if
  - The server has failed
  - The network is down
  - Neither; they are both just slow

- Makes failure detection hard
Challenge 3: Concurrency

Why not partition users across machines?

Shared State
Challenge 3: Concurrency

● How to ensure consistency of distributed state in the face of concurrent operations?

● Use mutex, cv, semaphore, etc.?

● Need to synchronize based on unreliable messages
Other Challenges

- Performance at scale
  - Example: Amazon redesigns software services for every order of magnitude change in scale

- Testing
  - Infeasible to test/reproduce every possible scenario
  - Cannot test at production scale
Objectives of this class

- Unify several machines from app perspective
- Enable concurrent use, hide failures
- Do “heavy lifting” so developers don’t need to
Contrast with EECS 482

Give every application illusion of running on its own machine

App$_1$  App$_2$  App$_3$
Distributed systems classes

- **EECS 591**: Study several instances of distributed systems and look for common principles

- **This class**: Study basic principles and examine case studies to understand how they are applied
Class topics

- **Before mid-term:**
  - How to maintain consistency of replicated state in the face of:
    - Failures
    - Concurrency

- **After mid-term:**
  - Scaling up deployments
  - Case studies
Class outcomes

- Be able to reason about uncertainty

- Be able to identify properties offered by a system
  - When will the system work well?
  - When will the system break down?
Class Material

- Class webpage
  - https://lamport.eecs.umich.edu
  - Also linked from Canvas

- Syllabus, lecture slides, homeworks, and projects will be posted on class webpage

- Subscribe yourself to Piazza
  - Announcements and class discussion
Lectures

- Print lecture slides before coming to class
- Lecture recordings will be posted online

2% extra credit for class participation
- 2 points for reasonable answer whenever you are called upon
- 0 points if never “present”
- 1 point for anything in between
Discussion Sections

- Will be used in three ways:
  - Explanation of project specs
  - Recap of lecture material
  - Discuss questions to help confirm understanding of lecture material

- Okay to attend either section
Projects

- 4 projects
  - MapReduce master
  - Primary backup replication
  - Paxos based replication
  - Scaling via sharding

- First one individually, others in groups of 2
  - Declare your group (by 9/16) via course web page
  - Post to Piazza if you don’t know anyone
  - We’ll assign private github repositories
Project recommendations

- Choose group partner carefully
- We’ll evaluate every member’s contributions
  - Peer feedback
  - git log and github statistics
- Monitor and participate in discussion on Piazza
- Attend discussion sections on Fridays
- Think carefully before you code!
Differences versus 482

- Program in Go

- Most of the test cases included in handout
  - Run locally as set of processes exchanging RPCs
  - You need to understand test cases to debug

- Exams are (probably) harder
Policies

● Submission
  ◆ 1 submission per day + 3 bonus submissions
  ◆ Due at midnight on deadline
  ◆ 3 late days across all projects

● Collaboration
  ◆ Okay to clarify problem or discuss Go syntax
  ◆ Not okay to discuss solutions
Exams

- Midterm: November 7\textsuperscript{th} (6:30 - 10:30 pm)

- Final: December 13\textsuperscript{th} (10:30am - 12:30 pm)

- No makeup exams
  - Unless dire circumstances
  - Make sure you schedule interviews appropriately
Grading breakdown

- Projects:
  - Project 1: 5%
  - Projects 2: 15%
  - Projects 3 and 4: 20% each

- Mid-term exam: 25%
- Final exam: 15%
Recipe for success

- Start early on projects
- Make sure to attend lectures and discussions
- Ask questions when something is unclear
- Take advantage of available help
  - Go to office hours, post/monitor questions on Piazza
Setting expectations

- First offering of EECS 491
  - So, course content still in flux

- Expect a hiccup or two!
TODOs

- Learn Go
  - Tutorial on Go at discussion section on Friday

- Subscribe to discussion forum on Piazza

- Find partner for project group