ECE 465 - Cloud Computing

Spring 2025 Independent Study

General Course Information

Instructor: Prof. Rob Marano
Email: rob@cooper.edu
Semester of the indepedent study: Spring 2025
Dates: 21 Jan 2025 – 16 May 2025

"**Cloud computing** — the phenomenon by which services are provided by huge collections of remote servers." — Andrew Tanenbaum & Maarten van Steen

Weekly course notes

Course Catalog Description

From the catalog: "Critical, foundational technology components that enable cloud computing, and the engineering advancements that have led to today's ecosystem. Students design, build and test representational software units that implement different distributed computing components. Multi-threaded programming in Java. Functional programming (MapReduce). Hadoop: a programmer's perspective; building and configuring clusters; Flume as an input engine to collect data; Mahout as a machine learning system to perform categorization, classification and recommendation; Zookeeper for systems coordination." *Note: We will update course catalog description to focus on topic vs software implementation. We may or may not use MapReduce, Hadoop, Zookeeper, and ML.*

From the instructor: You will dive deep, with hands-on approach, to study and implement critical, foundational technology components that enable distributed (cloud) computing, and the engineering advancements that have led to today's thriving cloud computing ecosystem. Students will understand, design, build and test representational software units that implement different distributed components, e.g., concurrent logic execution that uses software-defined communications, compute, storage, and security. Distributed computing topics include multi-threaded programming; architecture designs; communication designs; coordination; naming; consistency & replication; fault tolerance; and security.

You will be introduced to many of the system design tenets used in the operation of large-scale distributed systems operated by modern commercial cloud computing providers (hyperscalers), e.g., Google Cloud Platform (GCP), Amazon Web Services (AWS), and Microsoft Azure. These tenets include reliability; scalability; availability; fault tolerance; data replication; caching; consistency; partition tolerance; load balancing; asynchronous communications; instrumentation; and monitoring.

This course prepares the student with foundational knowledge and experience to

prepare for modern cloud computing software design and development professions and academic research.

3 credits. 3 hours per week (45 total hours).

Course Prerequisites

Minimum ECE 251 and ECE 264, or approval of EE Department Chair.

Course Structure/Method

Lectures and Live Coding Labs: This class meets TBD ET on the following days: TBD, for a total of 15 sessions. I encourage each of you to schedule office hours with me. Once appointment is confirmed, we will meet either in person in the Engineering Adjunct's Office on the 2nd floor of the NAB at 41 Cooper Square or remotely using Microsoft Teams video call.

Anticipated Schedule

Dates	Topic
Class 1	Intro; centralized vs distributed systems; development environment setup
Class 2	Multi-processing & network programming — Part 1
Class 3	Multi-processing & network programming — Part 2
Class 4	Multi-processing & network programming — Part 3
Class 5	Containerization: Docker and Kubernetes
Class 6	DevOps and CI/CD
Class 7	Integrate application to infrastructure
Class 8	Distributed Architectures
Class 9	Communication and Coordination
Class 10	Consistency & Replication
Class 11	Fault Tolerance
Class 12	Security
Class 13	Deploying on k8s on cloud-based virtual bare metal nodes
Class 14	Deploying on k8s on cloud-based k8s
Class 15	Final individual projects due

Course Learning Outcomes

Given that it's possible to build distributed systems, it does not always mean that it's a good idea.

A distributed system's design goals; it should

1. make resources easily accessible — Resource Sharing
2. mask the fact these resources are distributed across a network — Distribution Transparency
3. be open, offering components that can be easily used by or integrated into other systems — Openness
4. scale up and down based upon use — Scalability

Upon successful completion of this course, you will be able to understand the design goals that make building a distributed (cloud) system worth all the effort:

1. Learning how to scale program logic from a uniprocessor to a multiprocessor to networked nodes
2. Understanding the core concepts of distributed systems as well as tradeoffs, such as how multiple machines work together to solve problems in an efficient, reliable, and scalable manner
3. Applying knowledge of distributed systems techniques and methodologies
4. Gaining experience in the design and development of distributed systems and applications
5. Understanding how independent network and machine failure can make reliable distributed systems difficult to achieve
6. Understanding the core concepts in distributed computing, such as logical clocks, consistent cuts, consensus, replication, and fault tolerance

Communication Policy

The best way to contact me is first by a short summary via chat on Microsoft Teams followed immediately by a detailed email to `rob.marano@cooper.edu`. I will do my best to respond within 24 hours. Communication and participation in class is not only encouraged, but required. I seek to understand your individual understanding of the material each class. Advocate for yourself, early and often.

Course Expectations

Class Preparation

Each session will consist of two components: discussion and in-class lab work on your computers. Come prepared with your laptop and the Linux environment. See the "Software" section below.

Each class discussion consists of a mix of lectures, programming examples, and question-driven group analysis of one or more large programming problems. Lab

will consist of either group or individual work on exercises or projects. Questions arising during lab may be used to fuel additional discussion as time permits.

Attendance

Success as a student begins with attendance. Class time serves not only for learning new concepts and skills but also for practicing what you have learned with active feedback. Some assignments and demos may be completed in class, but practice and study are required outside of class. Students are expected to attend classes regularly, arrive on time, and participate. I take attendance during every session, and it forms part of your grade. Students are encouraged to e-mail me when they are absent. Students are responsible for all academic work missed as a result of absences. It is at my discretion to work with students outside of class time in order to make-up any missed work.

Materials

Reference Books

To understand the journey of distributed systems (that lead to today's current implementation called "The Cloud" - hence "cloud computing"):

- Tanenbaum, Andrew, and Maarten van Steen. Distributed Systems, 4th ed. Upper Saddle River, NJ: Prentice Hall, 2023. ISBN: 9789081540636. Get your official electronic copy here

We will be using other resources available on the Internet for our course. These will be shared throughout the semester based upon covered topics and assignments.

With that said, the following books provide helpful and historical background for our course and may help with programming. None are required.

- Stevens, W. Richard, Bill Fenner, and Andrew M. Rudoff. UNIX Network Programming, Vol. 1: The Sockets Networking API. 3rd ed. Reading, MA: Addison-Wesley Professional, 2003. ISBN: 9780131411555.
- Tanenbaum, Andrew. Modern Operating Systems. 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2001. ISBN: 9780130313584.
- McKusick, Marshall Kirk, Keith Bostic, Michael J. Karels, and John S. Quarterman. The Design and Implementation of the 4.4 BSD Operating System. Reading, MA: Addison-Wesley Professional, 1996. ISBN: 9780201549799.
- Stevens, W. Richard, and Stephen Rago. Advanced Programming in the UNIX Environment. 2nd ed. Reading, MA: Addison-Wesley Professional, 2005. ISBN: 9780201433074.

Software

You run your distributed systems development environment in Linux, which can be operated natively or within a virtual machine using VirtualBox, Vagrant, or Docker, for example. At a minimum, you'll need a terminal emulator running the Bash shell, Java 18 or higher with Maven, and git installed.

All software used during this course will be open source-based. We will also be using Amazon Web Services through the AWS Academy site of which Cooper Union has been given access. For many of our programming assignments, the Java programming language will be used, especially since this language is one of the most used ones for implementing modern cloud services given its portability across CPU instruction set architectures.

Assessment Strategy and Grading Policy

All assignments and the individual final project per student must be completed by the end of this course. Programming assignments and the projects will be handed-in individually via GitHub.

Assignment	Title	Points	Given On	Due Date	Link to Solution
1	Program 1	10	Class 1	Class 2	TBD
2	Program 2	10	Class 2	Class 3	TBD
3	Program 3	10	Class 3	Class 4	TBD
4	Program 4	10	Class 4	Class 5	TBD
5	Program 5	10	Class 5	Class 6	TBD
6	Program 6	10	Class 6	Class 7	TBD
7	Program 7	10	Class 7	Class 8	TBD
8	Program 8	10	Class 8	Class 9	TBD
9	Program 9	10	Class 9	Class 10	TBD
10	Program 10	10	Class 10	Class 11	TBD
11	Program 11	10	Class 11	Class 12	TBD
12	Program 12	10	Class 12	Class 13	TBD
13	Final Project	180	Class 6	Last Class	TBD

Final Projects MVP

One final project:

- Solo project; one student per project.
- Possibility to reuse strictly one subsystem implementing one design tenet will be shared across all projects; for example, naming, coordination, security. To be discussed during semester.
- Source code to be maintained in a GitHub repository; you will be invited by a GitHub invitation from the instructor's email address `robmarano@gmail.com`.

- Design and document in GitHub Wiki the software architecture and source code of your MVP; it's critical to document design in diagrams (draw.io) and in words on your repo project's wiki.
- Breakdown the MVP design in manageable sets of tasks, and track high-level via GitHub Project.
- Document the design of each subsystem for your MVP in its appropriate GitHub Wiki section.
- Demonstrate the MVP as part of your final presentation in an unlisted YouTube video, which will be due with your project on **Last Class 11:59:59pm ET**.

Your Coding Portfolio

Before you leave for break, ensure that you clean up your personal GitHub respository so that you can showcase the work you have developed. This will be helpful in any employment interviews you may have in the future. Like an artist, you know have a portfolio of software you have designed and implemented. No matter what you decide in your career, work and life is better through coding!

Research, tinker, and automate so that you have more time for the fun stuff in life!

Enjoy the course! /prof.marano