CSE 490h/CSE M552: Distributed Systems

Instructor: Tom Anderson (tom@cs.washington.edu)
TAs: Vincent Liu (vincent@cs.washington.edu)
Allison Obourne (aeobourn@cs.washington.edu)
Class meetings: MWF 9:30 – 10:20, EEB 003
Sections: Thursdays 8:30-9:20, 9:30 – 10:20, MOR 225
Instructor Office Hours: Mondays, 2:30 – 3:30
http://www.cs.washington.edu/490h/

Motivation: Distributed systems have become central to many aspects of how computers are used, from web applications to e-commerce to content distribution. This senior-level course will cover abstractions and implementation techniques for the construction of distributed systems, including client server computing, the web, cloud computing, peer-to-peer systems, and distributed storage systems. Topics will include remote procedure call, preventing and finding errors in distributed programs, maintaining consistency of distributed state, fault tolerance, high availability, and distributed security. As we believe the best way to learn the material is to build it, there will be a substantial programming project. Two versions of the course will be offered simultaneously: CSE 490h to undergraduates, and CSE M552 to students in the fifth year masters program.

Student Deliverables:

Course project to be done in teams of 2-3
An optional (490h) or required (M552) add-on to the project
Foundational readings in distributed systems
Blog entry answering one thought question per paper (M552 only)
Two problem sets
Final exam

Textbook, Papers and Blogs: There is no textbook for this course. Instead, we will assign research papers for specific lectures; the readings are to be done **before** class, as we will take the papers as a starting point, not the end point of the class discussion.

For each assigned paper except the first, we will post a discussion question, linked off the course web page. M552 students are required to post a short, **unique** comment or observation to the discussion by 8am on the day of the class. Valid posts can be to take a position on a design choice in the paper, to discuss the applicability of the paper to an important problem, to point out a problem in some other student's argument, etc. CSE 490h students are invited to add comments for extra credit but are not required to do so. Everyone should read the posted comments before class.

Project: The core of the course is the project: to design and build a fault tolerant peer-to-peer Facebook application by the end of the quarter. The project is to be done in groups of 2-3 people. The project has five pieces, each building on the previous ones, due roughly every two weeks. This is a very aggressive schedule and so we need every group to get an early start on the assignments. We will provide you some basic message passing code as well as a framework for simulating and emulating a distributed system to aid in debugging.

The first project assignment is to build a simple client-server storage system. We then add cache coherence, crash recovery and high availability to this basic system. The final assignment is to use these pieces to construct a fault tolerant Facebook application that can run without a central server, on a cluster of nodes in the undergraduate PC lab. M552 students will be required to extend the basic design in some interesting way, e.g., to add support for disconnected operation. CSE 490h students may do so for extra credit.

To help provide you time to complete the project, we set aside weeks 8 and 10 as “hack weeks” with no lectures or sections. We will also automatically grant each group three slip days for the project assignments, for you to use at your discretion. Regardless, the final assignment must be turned by March 15.

Details about the project assignments will be provided in a separate handout.

Problem Sets: There will be two problem sets handed out during the quarter. The problem sets are to be done individually and are intended to reinforce the basic material covered in the class, and to help prepare students for the final exam. Problem sets are due in section.

Collaboration/Cheating: In our experience, students often learn more from each other than they do from the instructor. Thus, we encourage you to collaborate with your classmates, in all aspects of the course. The grading in the class is emphatically not curved; we would like nothing better than for all of you to get a 4.0.

To draw a very clear line, you may use any idea from any other person or group in the class or out, provided you clearly state what you have borrowed and from whom. If you do not provide a citation -- that is, you turn other people’s work in as your own -- that’s cheating. Anything else is fair game. Of course, we’ll be grading you on the ideas you’ve added, but you should always borrow as much as you can as a starting point – there’s no point in reinventing the wheel.

Final: The final for this class will be held Wednesday, March 16 at 8:30am. Project demonstrations will be held at various times on Tuesday, March 15.

Grading (490h): Project: 50%; problem sets 15%; final: 35%

Grading (M552): Project: 50%; problem sets 15%; blogs: 5%; final: 30%

Tentative Syllabus: Readings to be done **before** class

1/3: Introduction

> Read: Google. Introduction to Distributed System Design. http://code.google.com/edu/parallel/dsd-tutorial.html

Part 1: Client/Server

1/5: Remote procedure call

> Read: Andrew Birrell and Bruce Nelson. Implementing Remote Procedure Calls. ACM TOCS 1984.
>
> Read: Google, Protocol Buffers Developer Guide. http://code.google.com/apis/protocolbuffers/docs/overview.html

1/7: Remote method invocation

> Read: Wollrath, Riggs and Waldo. A Distributed Object Model for the Java System. USENIX Computing Systems, 1996.

1/10: Distributed debugging (guest lecture: TBD)

> Read: Lamport, Time, Clocks and the Ordering of Events in a Distributed System, CACM 1978. (up to, not including, the section on physical clocks)

Part 2: Memory consistency and cache coherence

1/12: Cache coherence, part 1

> Read: Kai Li and Paul Hudak. Memory Coherence in Shared Virtual Memory Systems. ACM TOCS 1989.

1/14: Cache coherence, part 2

1/14: Project part 1 due: client-server

1/17: MLK

1/19: Eventual consistency

> Read: Jay Kistler and M. Satyaranayaman. Disconnected Operation in the Coda File System. ACM TOCS1992.

Part 3: Failure recovery and Fault Tolerance

1/21: Transactions

> Read: Robert Hagmann. Reimplementing the Cedar File System Using Logging and Group Commit. SOSP 1987.

1/24: Distributed Transactions, part 1

> Read: Bernstein, Hadzilacos, and Goodman. Distributed Recovery. Chapter 7 in Concurrency Control and Recovery in Database Systems. (up to, and not including, three phase commit)

1/26: Distributed Transactions, part 2

1/28: State Machine Replication (guest lecture: TBD)

> Read: Bernstein and Newcomer. Replication. Chapter 9 in Principles of Transaction Processing.

1/28: Project part 2 due: cache coherence

1/31: Paxos, part 1

> Read: Lamport, Paxos Made Simple, ACM SIGACT News, 2001.

2/2: Paxos, part 2

2/4: GFS

> Read: Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google File System. SOSP 2003.

Part 4: Cloud Programming

2/7: MapReduce

> Read: Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. OSDI 2004.

2/8: Problem set #1 due

2/9: Dryad

> Yu et al. DryadLINQ: A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language. OSDI 2008.

2/11: BigTable

Read: Chang et al. BigTable: Distributed Storage System for Structured Data. OSDI 2006.

2/11: Project part 3 due: distributed transactions

2/14: Dynamo

Read: DeCandia et al. Dynamo: Amazon's Highly Available Key-Value Store. SOSP 2007.

Part 5: Beyond the Cloud

2/16: PNUTS

Read: Cooper et al. PNUTS: Yahoo's Hosted Data Serving Platform, VLDB 2008

2/18: DNS

Read: Mockapetris. Development of the Domain Name System. SIGCOMM 1988.

Presidents Day/Hack Week: 2/21 – 2/25

2/25: Project part 4 due: paxos

2/28: BitTorrent

Read: Piatek et al. Do Incentives Build Robustness in BitTorrent? NSDI 2007.

3/2: Distributed Security

Lampson, Computer Security in the Real World, 2001.

3/4: Wrapup

Lampson. Hints for Computer System Design. ACM SIGOPS OSR 1983.

Hack Week: 3/7 – 3/11

3/8: Problem set #2 due

3/14: Project part 5 due: peer to peer Facebook

3/14: M552 project add-on due

3/15: Project demonstrations

3/16: Final Exam, 8:30am – 10:20