CS 194: Lecture 1

Department of Electrical Engineering and Computer Sciences
University of California
Berkeley

Today's Lecture

- Opening Remarks
- Administrivia
- Overview
- Background Questionnaire

Administrivia

- Course web page:
  - http://inst.eecs.berkeley.edu/~cs194/sp05
  - Check periodically to get the latest information
- Office Hours: TBD
- Deadline means deadline
  - Unless otherwise specified, it means 5:00pm on the date
  - Special circumstances should be brought to our attention <u>well before</u> the deadline
- Closed book exams (but with single crib sheet)

Grading

- Homework: 20% (4 x 5%)
- Project: 40%
- Midterm: 20%
- Final: 20%

Promise

- Course won't be as boring as the first chapter!

Two Views of Distributed Systems

- **Optimist**: A distributed system is a collection of independent computers that appears to its users as a single coherent system
- **Pessimist**: "You know you have one when the crash of a computer you've never heard of stops you from getting any work done." (Lamport)

History

- First, there was the mainframe
- Then there were workstations (PCs)
- Then there was the LAN
- Then people wanted to make the collection of PCs look like a mainframe
- They built some neat systems (DFS, TDBs, ….)
- But the web blew them away!

Why?

- The vision of distributed systems:
  - Enticing dream
  - Very promising start, theoretically and practically
- But the impact was limited by:
  - Autonomy (fate sharing, policies, cost, etc.)
  - Scaling (some systems couldn't scale well)
- The Internet provided:
  - Extreme autonomy
  - Extreme scale
  - Poor consistency (nobody cared!)

Recurring Theme

- Academics like:
  - Clean abstractions
  - Strong semantics
  - Things that prove they are smart
- Users like:
  - Systems that work (most of the time)
- Scale is important to reach many users
- Perfect consistency isn't important to many users

Inherent Tension

- Goals:
  - **C**onsistent (broadly defined)
  - **A**vailable
  - **P**artitions don't stop system
- CAP Thm: Can only have two! (Brewer)
  - Systems embody each of the three choices
- More generally:
  - Consistency and availability hard to achieve as system gets larger

What Makes a DS Hard?

- Delays (asynchrony)
- Failures (stop failures)
- Byzantine failures
- Different incentives

Course Overview

- Algorithmic challenges:
  - Synchronization, exclusion, consistency, CAP, commitment, fault tolerance, security,…
- How the real world works:
  - Protocols, RPC, RMI, processes, naming, CORBA/DCOM, distributed file systems, Web, Jini, P2P, DHTs, Internet design, incentives, (sensornets?)

Project: DBay

- Distributed auction
  - Synchronize bidding
  - Secure transaction on sale
- Will be done in stages
  - Starting easy
  - Let us know if you are in trouble **early**!

Sample Problem I

- Consider n generals, each with a certain number of troops.
- Assume m of them are traitors, who will lie.
- Design an algorithm (assuming reliable communication) so that every loyal general knows the number of troops of every other loyal general
- Extra credit: design it so that the loyal generals agree on a total troop strength (perhaps incorrect)

Sample Problem II

- What's the fastest way to spread a rumor?
  - N people
  - Each has a phone
  - Place a random phone call every morning
  - They don't remember who they talked to the previous day, and they don't know N
  - But they can remember how many times they've repeated the rumor, etc., and must use that information to decide when to stop spreading it
  - Want to minimize the number of people who haven't heard, as a function of the number of times the rumor is retold.

Questionnaire