Distributed Systems
CPSC 416
Winter 2022

Jan 11 Lecture (first class!)
Online

Oh yeah, pandemic

- Not a great time to be taking courses
- My 2nd time teaching a large course over zoom
- Lots of resources, but this course may not be the right one for you (timezone/workload/content/etc)
    - Please consider carefully before committing
    - First assignment is a litmus test

Oh yeah, ***still*** pandemic

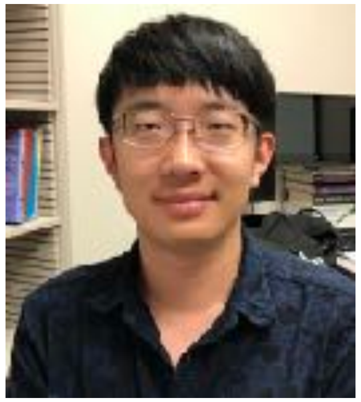
- January’22: zoom for all the things
- After January: unknown
  - Likely to be zoom for at least some part of Feb
  - But that’s my guess
- *My goal: support your learning regardless of format and person situation*

Course staff

- **Ivan** Beschastnikh, associate research professor
- At UBC since 2013
  - Previous taught 416 four times (in person), and over zoom in 2021
  - Research distributed systems, networks, security, program analysis

Course staff

- **Ivan** Beschastnikh, instructor
- TAs (all grad)
    - Mishaal
    - Mayank
    - Yanze



416 course evolution

- 2016: 77 students (open-ended project)
- 2017: 117 students (assignment hell)
- 2018W: 160 students (assignments + projects)
- 2018F: 44 students (mix of above)
- 2021W: 160 students (assignment… hell)
- 2022W: 120 students (assignments + projects) ← You are here

Waitlist

- Current waitlist has 61 people!
    - Keep joining and working on assignments, some people will drop, *but not everyone will get in*
- To others: consider dropping if you have other courses that look more interesting

Basic resources

- Everything on the website, updated continuously: https://www.cs.ubc.ca/~bestchai/teaching/cs416_2021w2/
- Use Piazza for **all** course-related communication
- Office hours (start next week, over zoom):
  - M,W,F: with TAs
  - Th: with Ivan

Quick zoom poll

- How well do you remember 317 (networking)?
- How well do you know Go lang?
- Do you want to do [assignments] in teams?

Course overview via the website

- Learning goals
- Go programming language (start learning!)
- Schedule (a work in progress)
  - Assignment 1 likely due Jan 21
- Exam (‘just’ a final)
- Advice for doing well
  - learn Go (a must to pass the course)
  - don’t hack, engineer
  - choose team, wisely
  - reach out on Pizza for help.
- Collaboration guidelines

Learning goals

- Understand key principles in designing and implementing distributed systems

- Reason about problems that involve distributed components

- Become familiar with important techniques for solving problems that arise in distributed contexts

- Build distributed system prototypes using the Go programming language

Learning goals

- Understand key principles in designing and implementing distributed systems
- Reason about problems that involve distributed components
- Become familiar with important techniques for solving problems that arise in distributed contexts
- Build distributed system prototypes using the Go programming language (the key to all the above)

Some student *workload* comments from previous offerings

- *The workload for this course is easily double that of any other course I had this term.*
- *Ivan has very high expectations of his students.*
- *I love and hate the fact that this class was a "sink or swim" approach to learning*

Assignment 1: UDP Networking with Go

- Implement a client that interactively plays the game of nim with a server
- Goal is to help you:
  - Learn Go
  - Learn Go
  - Learn Go
  - Remember some networking

Assignments note

- Typical 416 TA rant:

TEST YOUR CODE ON THE UGRAD MACHINES!!!!!!!!!!!!!!!!!!!

YOU WILL GET ZERO IF IT DOESN'T RUN OR COMPILE. WE HAVE NO SYMPATHY FOR THESE TYPES OF ERRORS.

… you’ve been warned

Examples of distributed systems

- What are some examples of distributed systems?
  - BitCoin, Blockchains
  - HDFS
  - Winery with temp controls that are coordinated (IoT: sensors/actuators, cyber-physical)
  - SETI: search for aliens at home: distribute compute
    - Floding@home: same but for proteins
  - Kafka: message system… better than a network? Distributed queues of msgs (+ policies over those messages); pub-sub
  - TOR: distributed system for privacy — hide your location (IP) from others
  - DNS: naming service — used for WWW; hierarchical and has weak consistency
  - Load balancers: take bunch of requests, decide who to send them to
  - AWS: cloud — collection of distributed systems
  - Raft: Consensus protocol (algorithm; etcd that realizes this alg)
  - CDNs: Global distributed systems for distributing content (dealing with flash crowds)
  - Zoom: cloud-cloud system
  - Git: weak consistent, async, support for disconnection operation
  - DHTs: distributed hash tables (Kademlia ~ KAD in Emule..)
  - BitTorrent, Cassandra (KV store)

Systems versus applications

- What are some examples of distributed systems?
- Why not a distributed **application**? (DApps on blockchains)
  - More scalability/concurrency — dealing with multiple connections/ clients who request service; application services … a single human?
  - Implicated abstraction are more at the API/protocol/semantics level.
  - Fault tolerance — application has downtime isn't the end of the world; fallout for a distributed system failure is much greater
  - Scales more naturally than an application

Systems versus applications

- What are some examples of distributed systems?
- Why not a distributed **application**? (DApps on blockchains)
  - Abstracted away from users
  - App is for clients, internals are systems
  - System provides a “service” to other programs / API
  - App usually interfaces with a person

Why distributed?

- What makes a system ***distributed***?
  - Distributed in space — removing reliance on centralized physical components = fault tolerance to failure of those components
  - Availability — higher for a distributed system (due to fault tolerance); geographic distribution
  - Communication/networking implicated in *every* distributed system: semantics/guarantees of the network are really important for every system (that you'll build in this course)

What .. distributed?

- What makes a system ***distributed***?
  - Communication (networking)
  - Concurrency/async (threads/processes/machines/Pis)
  - Multiple machines/decentralization
  - Replication (coordination) for fault tolerance/fail over
  - Division of tasks (compute)
  - Scalability/high perf ~ nice to have for a dist. sys

Distributed system examples

- YouTube
  - Videos are **replicated** (multiple machines host the same video)
  - **Scalable** wrt. client requests for videos (internally **elastic** — can throw more machines at the service to have it scale out further)

Distributed system examples

- DropBox (or google drive)
    - **Replicated** content across personal devices
        - Supports **disconnected operation** (can work while disconnected, and synchronize when re-connected)
        - Maintaining data **consistent** across devices
    - Supports sharing; **access control** policies (security!)

Distributed system examples

- NASDAQ
  - **Transactions** (e.g., ACID semantics from databases). Many DBMS concepts apply to distributed systems!
  - Strong **consistency** and **security** guarantees (otherwise people would not trust it with money)

Some D.S. challenges

- Synchronizing multiple machines (protocol complexity)
- Performance (how do you define/measure it?)
- Maintaining consistency: strong models (linearizable) to weak models (eventual) of consistency
- Failures: machine failures (range: failure stop to byzantine); network failures (just a few: disconnections/loss/corruption/delay/partitioning)
- Security (how to prevent malicious control of a single host in a system escalating into control of the entire system?)

For Thursday

- Install Go on your personal machine
- Work through *Tour of Go!* and other tutorials.
- **Practice Go!**