Introduction

CS 4740: Cloud Computing

Fall 2024

Lecture 1

Yue Cheng

University of Virginia

Introduction

- On the faculty of Data Science & Computer Science
  - Web: https://tddg.github.io
  - Email: mrz7dp@virginia.edu

- Current research: Designing **better data systems**
  - Cloud data systems (analytics & cloud storage that runs on serverless functions)

https://github.com/ds2-lab/Wukong

https://github.com/ds2-lab/infinicache
https://github.com/ds2-lab/infinistore

https://github.com/ds2-lab/LambdaFS

Course staff and getting help

- Instructor: Yue Cheng
    - Office hours: Thursday, 11am-12pm on Zoom

- GTAs:
    - Rui Yang
    - Email: qgh4hm@virginia.edu
    - Office hours: TBD

    - Yilong Yang
    - Email: kbh8sa@virginia.edu
    - Office hours: Friday 3pm-5pm

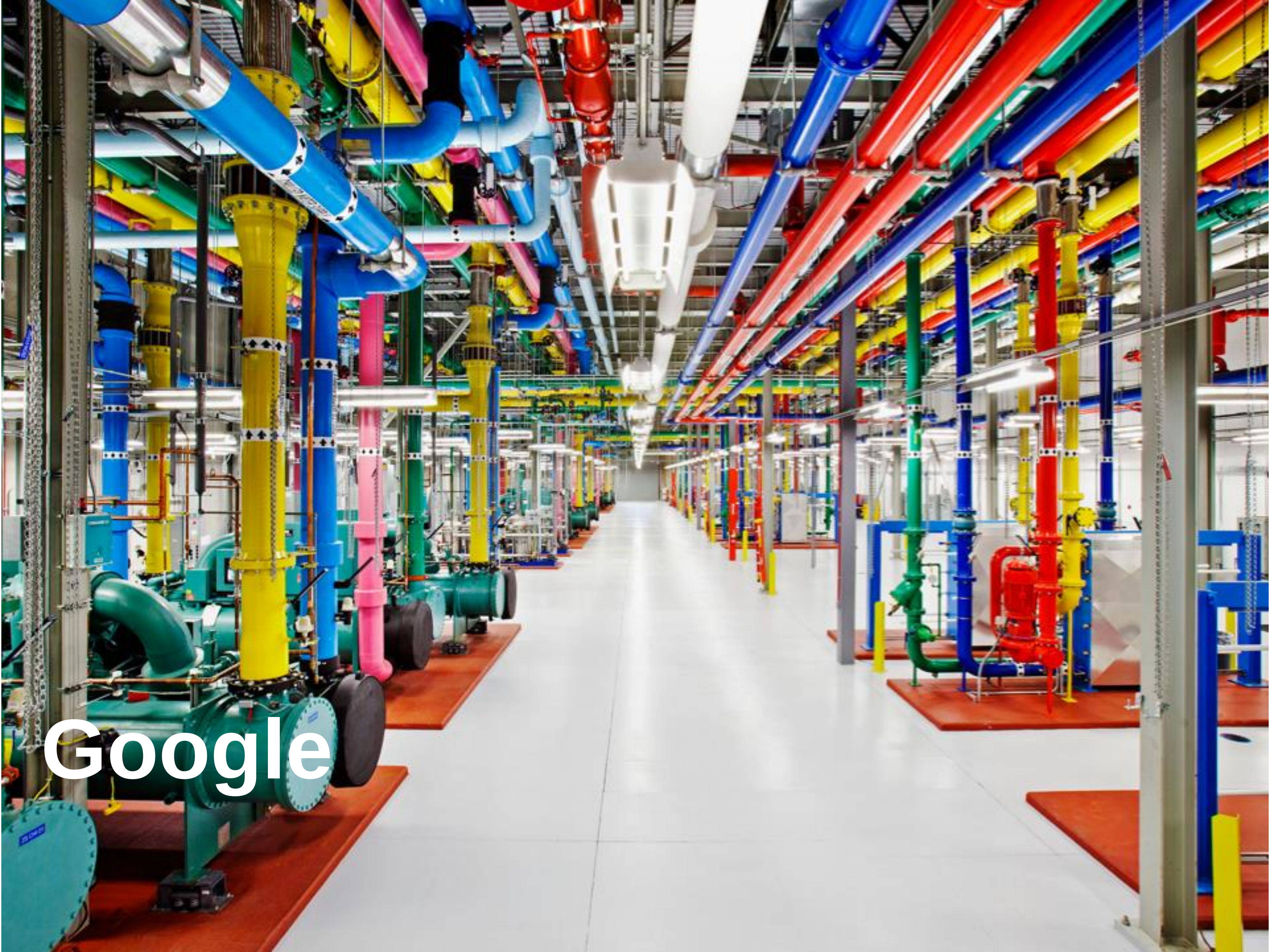
    - Han Ling
    - Email: mtd3tu@virginia.edu
    - Office hours: Monday 6pm-8pm

Course staff and getting help

- Discussion, questions: Ed
  - https://edstem.org/us/dashboard
  - Alternative place to ask questions about lab assignments and materials
  - No anonymous posts or questions
  - Can use private posts to instructor/GTA
  - We are monitoring Ed several times a day
  - We will respond to questions in a batch manner

Today’s agenda

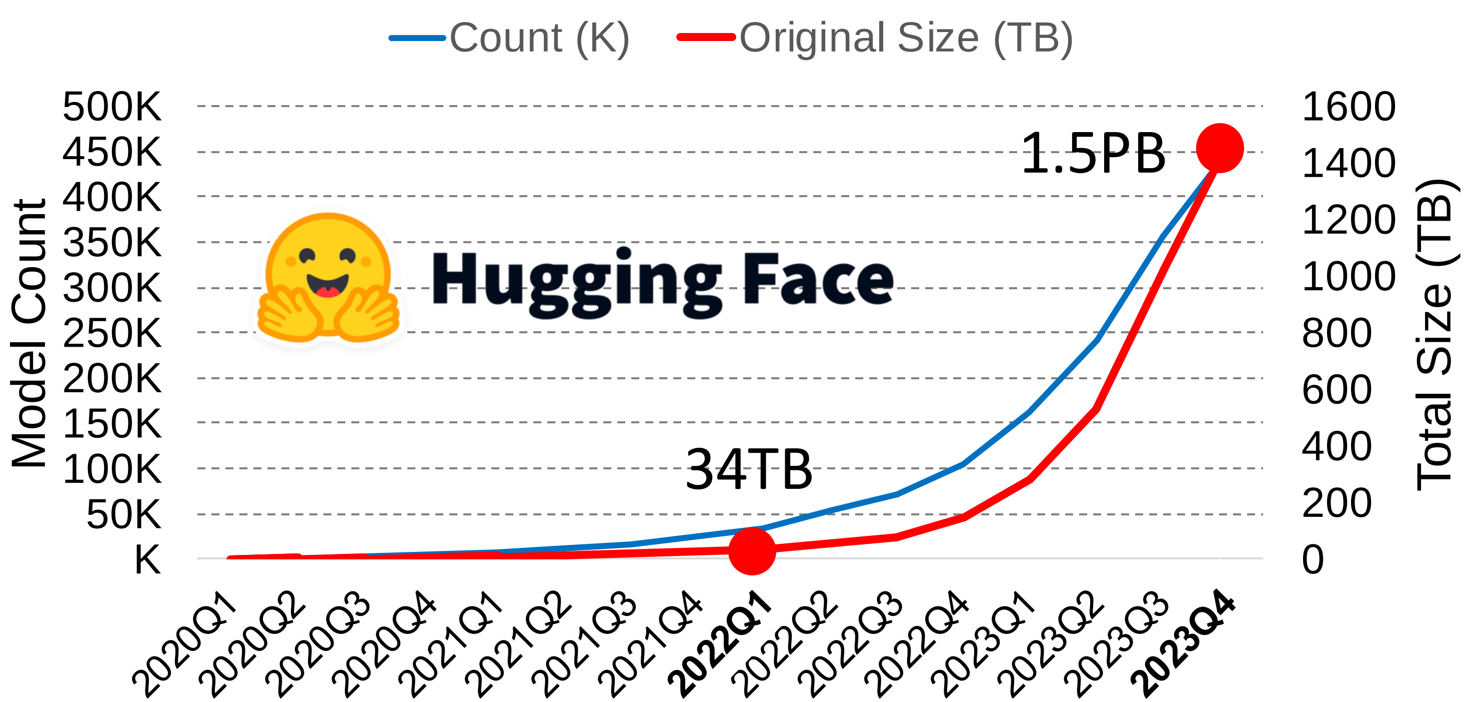
- Why are we studying Cloud Computing? What is this course about?
- What will you do in this course?


Google

A Google Datacenter in Hamina, Finland

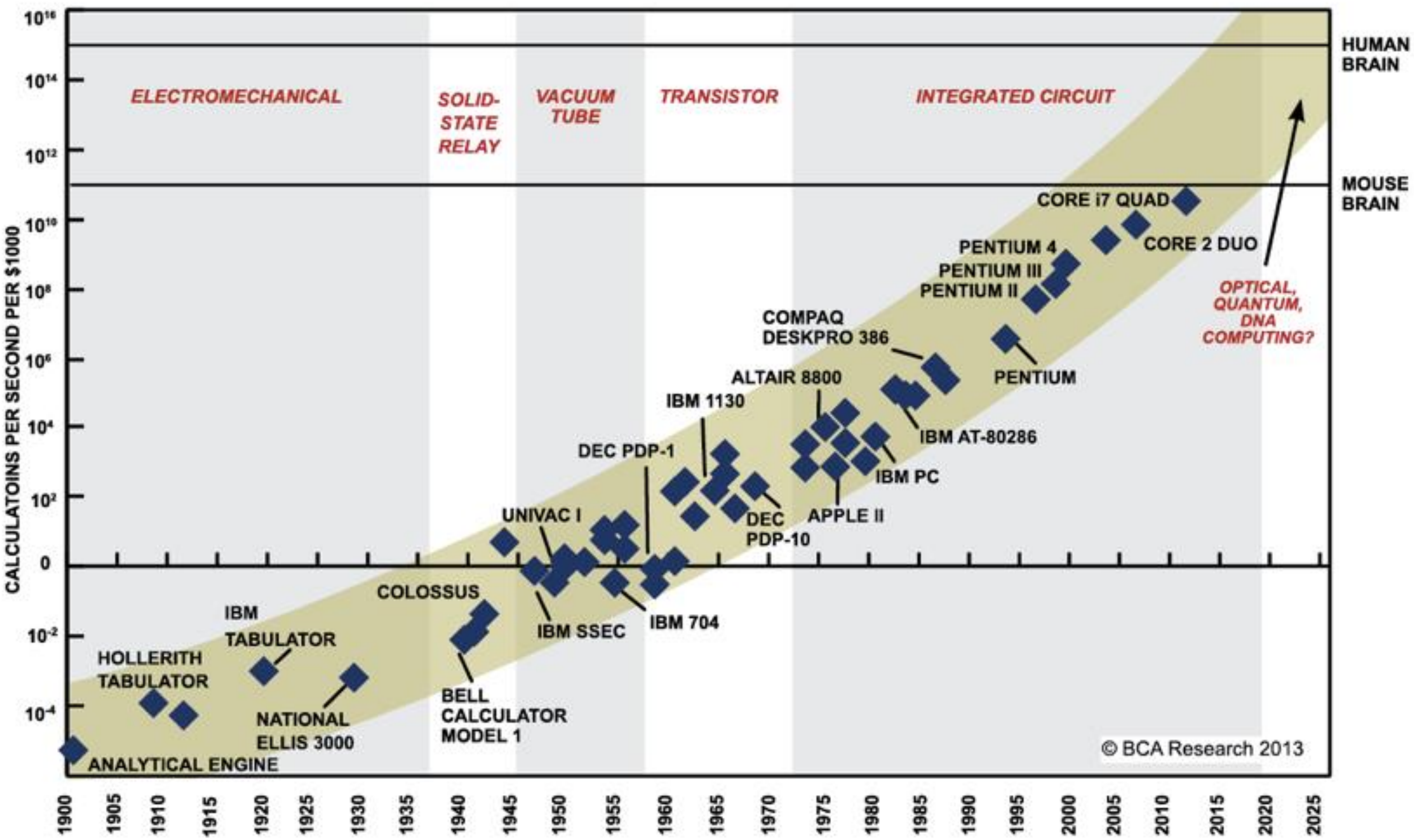
Data explosion

- Google web index: 10+ PB
- Hugging Face: ML model storage **exponentially** increasing



Exciting time in cloud & distributed systems

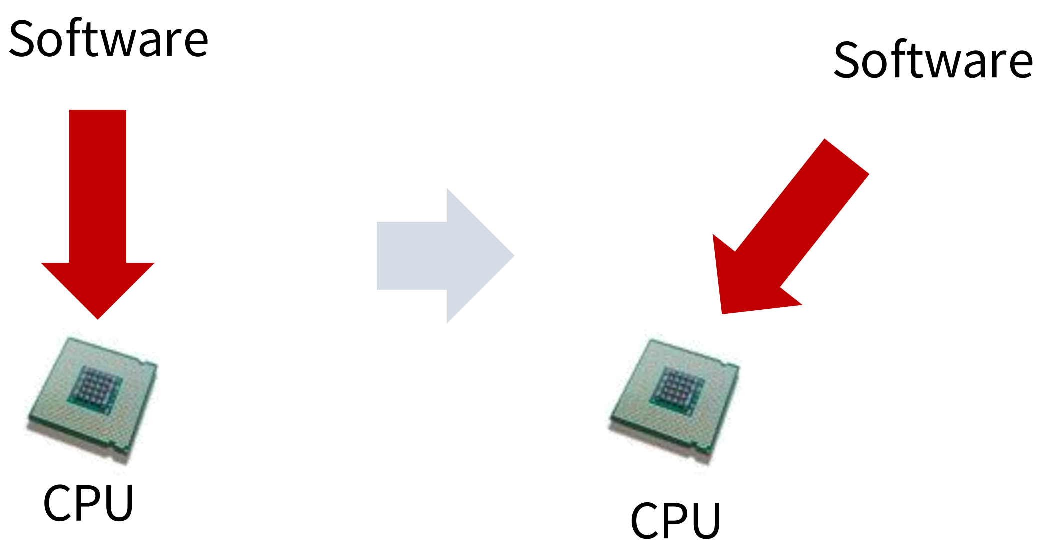
Moore's law ending → many challenges



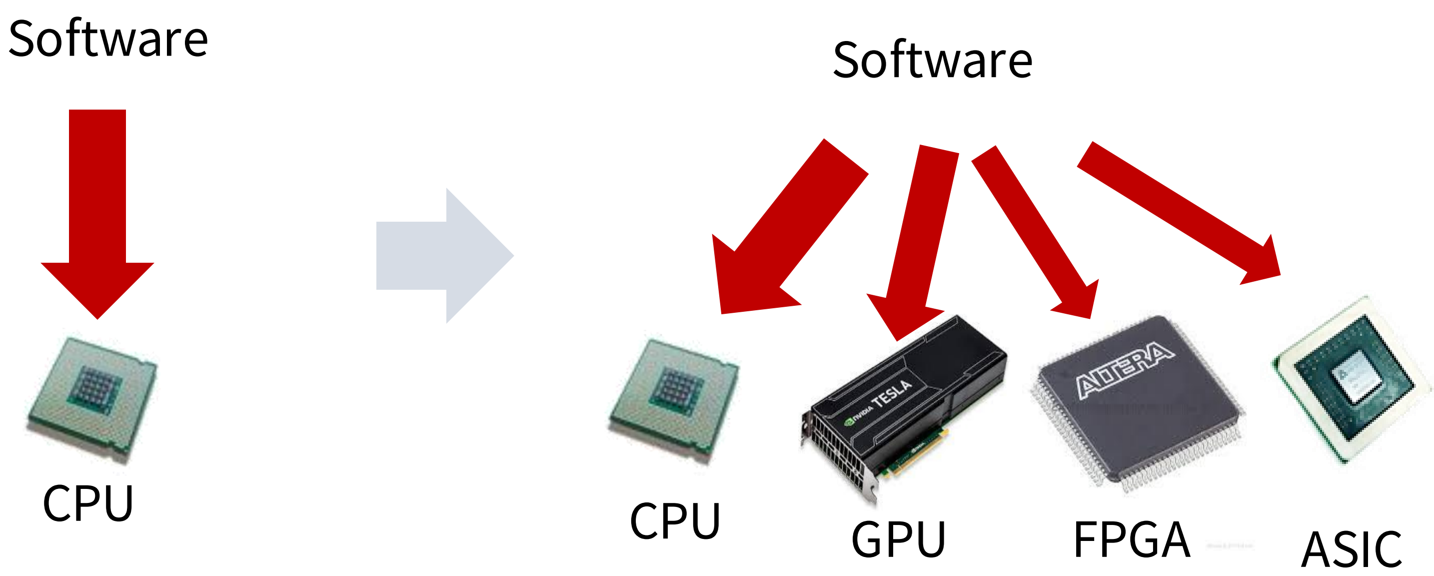
SOURCE: RAY KURZWEIL, "THE SINGULARITY IS NEAR: WHEN HUMANS TRANSCEND BIOLOGY", P.67, *THE VIKING PRESS*, 2006. DATAPOINTS BETWEEN 2000 AND 2012 REPRESENT BCA ESTIMATES.

Increased complexity – Computation

Increased complexity – Computation



Increased complexity – Computation



Increased complexity – Memory

2015

L1/L2 cache	~1 ns
L3 cache	~10 ns
Main memory	~100 ns / ~80 GB/s / ~100GB
NAND SSD	~100 usec / ~10 GB/s / ~1 TB
Fast HDD	~10 msec / ~100 MB/s / ~10 TB

Increased complexity – Memory

2015

- L1/L2 cache: ~1 ns
- L3 cache: ~10 ns
- Main memory: ~100 ns / ~80 GB/s / ~100GB
- NAND SSD: ~100 usec / ~10 GB/s / ~1 TB
- Fast HDD: ~10 msec / ~100 MB/s / ~10 TB

2024

- L1/L2 cache: ~1 ns
- L3 cache: ~10 ns
- HBM: ~10 ns / ~1TB/s / ~10GB
- Main memory: ~100 ns / ~80 GB/s / ~100GB
- NVM: ~1 usec / ~10GB/s / ~1TB
- NAND SSD: ~100 usec / ~10 GB/s / ~10 TB
- Fast HDD: ~10 msec / ~100 MB/s / ~100 TB

Increased complexity – More and more choices in clouds

Basic tier: A0, A1, A2, A3, A4
Optimized Compute : D1, D2, D3, D4, D11, D12, D13
D1v2, D2v2, D3v2, D11v2,…
Latest CPUs: G1, G2, G3, …
Network Optimized: A8, A9
Compute Intensive: A10, A11,…

Microsoft Azure

t2.nano, t2.micro, t2.small
m4.large, m4.xlarge, m4.2xlarge, m4.4xlarge, m3.medium,
c4.large, c4.xlarge, c4.2xlarge,
c3.large, c3.xlarge, c3.4xlarge,
r3.large, r3.xlarge, r3.4xlarge,
i2.2xlarge, i2.4xlarge, d2.xlarge
d2.2xlarge, d2.4xlarge,…

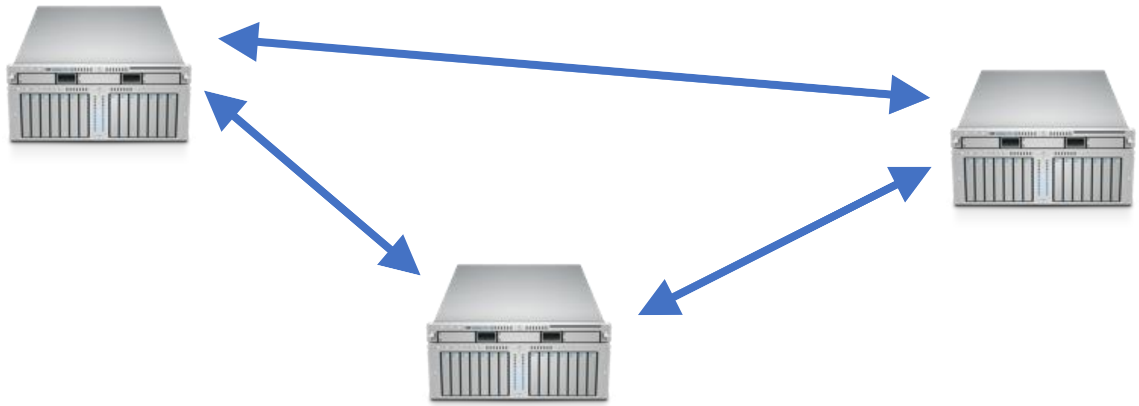
Amazon EC2

n1-standard-1, ns1-standard-2, ns1-standard-4, ns1-standard-8, ns1-standard-16, ns1highmem-2, ns1-highmem-4, ns1-highmem-8, n1-highcpu-2, n1-highcpu-4, n1-highcpu-8, n1-highcpu-16, n1-highcpu-32, f1-micro, g1-small…

Google Cloud

But how do we program cloud resources to tackle these challenges?

Cloud systems are distributed systems



- Multiple cooperating computers – distributed systems
  - Connected by a network
  - Doing something together
- Storage for big websites, MapReduce, etc.
- Lots of cloud infrastructures are distributed

Cloud systems: Why?

- Or, why not 1 computer to rule them all?

- To organize physically separate entities
- To tolerate faults via replication
- To scale up/out throughput via parallel CPUs/mem/disk/net

Goals of “cloud systems”

- Service with higher-level abstractions/interface
  - E.g., file system, database, key-value store, programming model, …
- High complexity
  - Scalable (scale-out)
  - Reliable (fault-tolerant)
  - Well-defined semantics (consistent)
- Do “heavy lifting” or “messy plumbing” so app developer doesn’t need to

Course syllabus

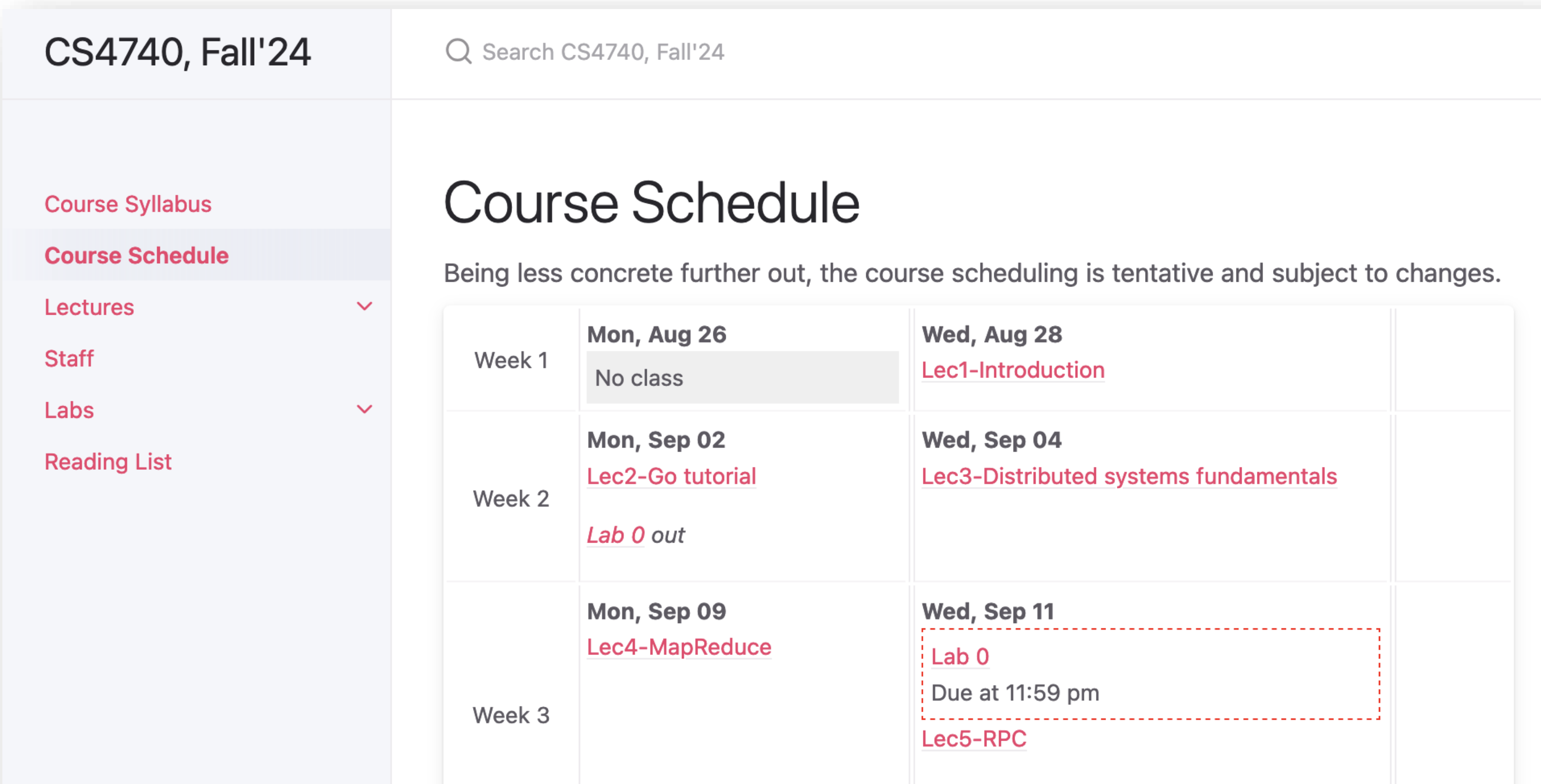
Big picture course goals

- Learn about some of the most influential works in distributed and cloud computing systems
- Get a sense of how massive scale systems "fit" together
- Learn how to manage writing highly concurrent and non-deterministic code
  - In my opinion, much harder than "just" parallel programming
- Learn how to approach, discuss, and communicate about difficult & technical subject matter

Schedule (tentative)

- Readings, lab assignments, due dates
- Less concrete further out; don't get too far ahead

  https://tddg.github.io/cs4740-fall24/



Lectures

- (Review) + lecture + (quiz, lab tutorial)
- Slides available on course website (night before)
- First five weeks: Fundamentals of concurrent & distributed systems
- Week 6-7: Fault tolerance and consensus
- After midterm: Week 9-11: Virtualization, cloud computing, serverless computing
- Week 12-14: Consistency and cloud storage

Textbooks?

- Papers and blog articles serve as reference for many topics that aren’t directly covered by a text

- Slides/lecture notes

- Three optional textbooks (first two are free)
  - “**Operating Systems: Three Easy Pieces (OSTEP)**” by Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau
  - “**Distributed Systems (3rd edition)**” by van Steen and Tenenbaum will supply optional alternate explanations
  - “**The Go Programming Language**” by Alan A. A. Donovan and Brian W. Kernighan (can be accessed via UVA library)

Programming labs

- Five lab assignments (Go) – all individually
  - **Lab 0:** Intro to Go (warmup)
  - **Lab 1:** MapReduce
  - **Lab 2:** Key-value server
  - **Lab 3:** Raft
  - **Lab 4:** Serverless and containerization

- Require comfort with (Go) concurrency that takes awhile to acquire

- Your labs will be autograded (Autolab); you can resubmit and view your score in real-time

Programming labs grading

- Labs are graded on functionality but not performance
  - Bad designs, however, may significantly affect the performance and thus force autograder to timeout


CAUTION
Heavy Programming
Next 13 Weeks...

By the end of the semester…

You will have built some sophisticated, functional, distributed systems using Go

Grading

- Labs (70% total)
  - Late turnings are graded with 10% deducted each day; no credit after 3 days
- Quizzes (5%)
- Midterm exam (10%)
- Final exam (15%)

TODOs

- Sign-ups
  - Sign up for Ed
  - Sign up for Autolab – TAs will set up the Autolab service before weekend

- Next class:
  - Go systems programming tutorial