Introduction

DS 5110/CS 5501: Big Data Systems

Spring 2024

Lecture 1

Yue Cheng

Some material taken/derived from:

- Wisconsin CS 744 by Shivaram Venkataraman.

Welcome!

- Two sessions combined in one
  - DS + CS

- New content this semester
  - Will be discussed in today's lecture

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:
  - Arup Sarker
  - Email: djy8hg@virginia.edu
  - Office hours: TBD

  - Make Li
  - Email: ufs7eg@virginia.edu
  - Office hours: TBD

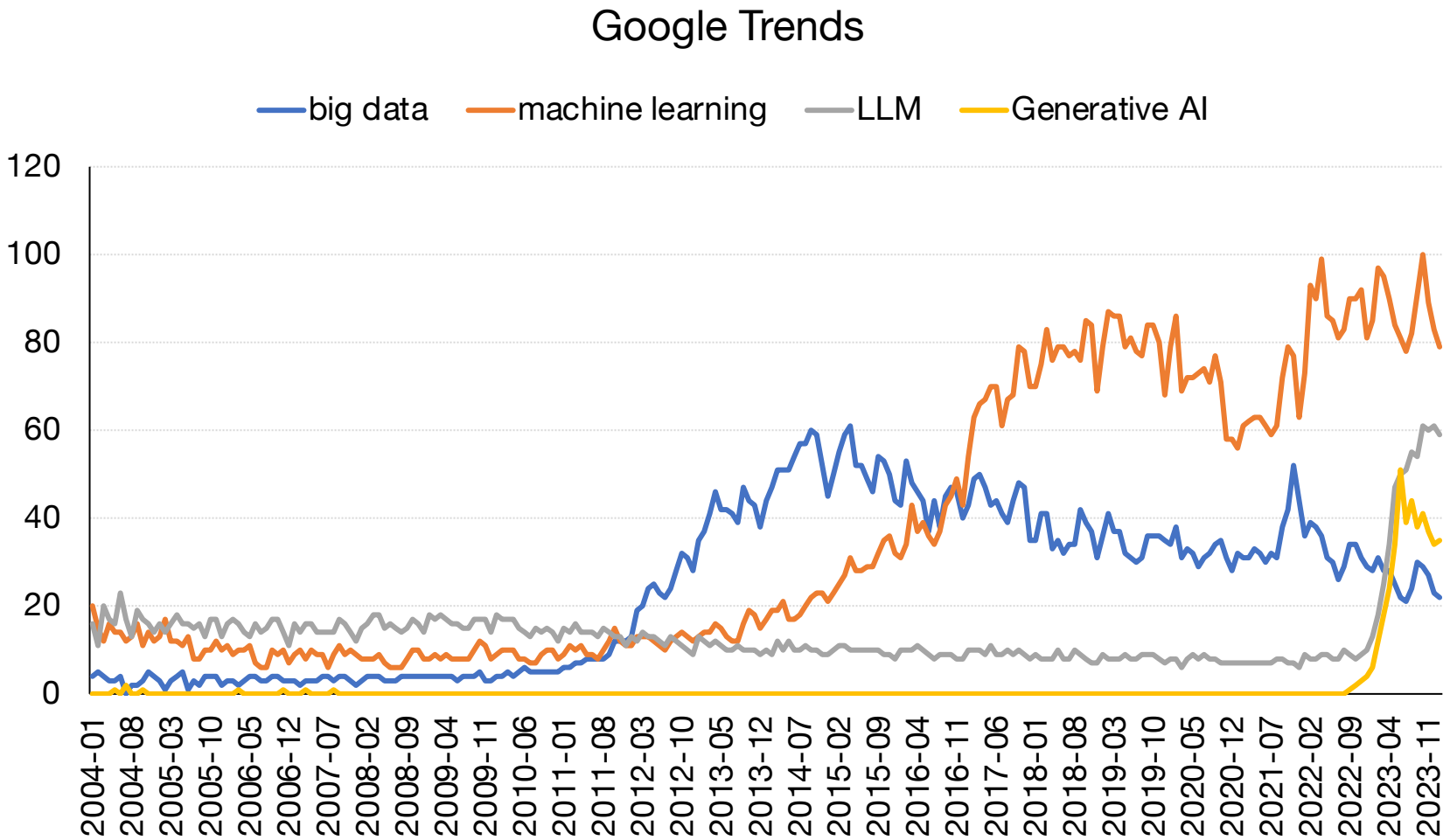
Course staff and getting help

- Discussion, questions: Ed
  - https://edstem.org/us/dashboard
  - Alternative place to ask questions about assignments, materials, and projects
  - 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

- What is this course about?
- Why are we studying Big Data Systems?
- What will you do in this course?

A brief history about Big Data



Google circa 1997

Everything is about data

"… Storage space must be used efficiently to store indices and, optionally, the documents themselves. The indexing system must process hundreds of gigabytes of data efficiently…"

"The system… downloading the last 11 million pages in just 63 hours… The sorter can be run completely in parallel; using four machines, the whole process of sorting takes about 24 hours…"

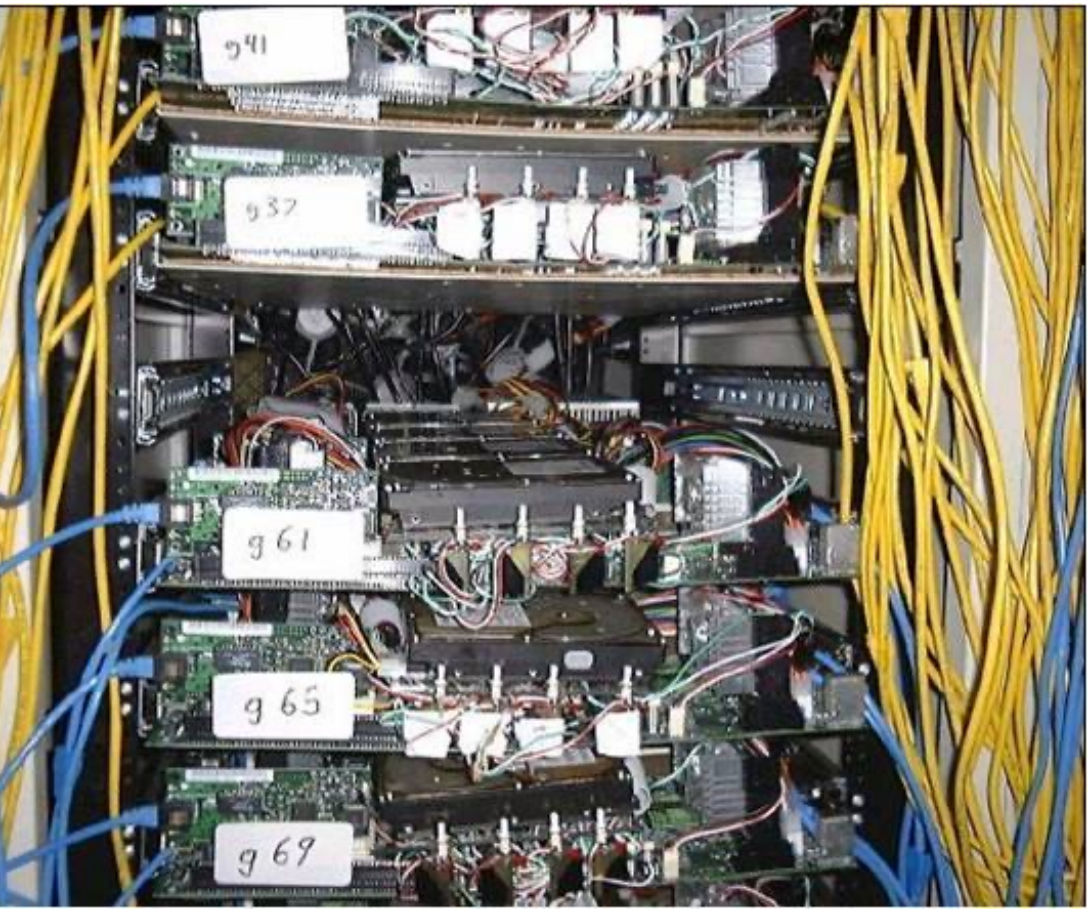
The anatomy of a large-scale hypertextual Web search engine [1]

Sergey Brin [2], Lawrence Page [*,2]

Computer Science Department, Stanford University, Stanford, CA 94305, USA

Abstract

Google circa 2000



Commodity CPUs

Lots of disks

Low bandwidth network

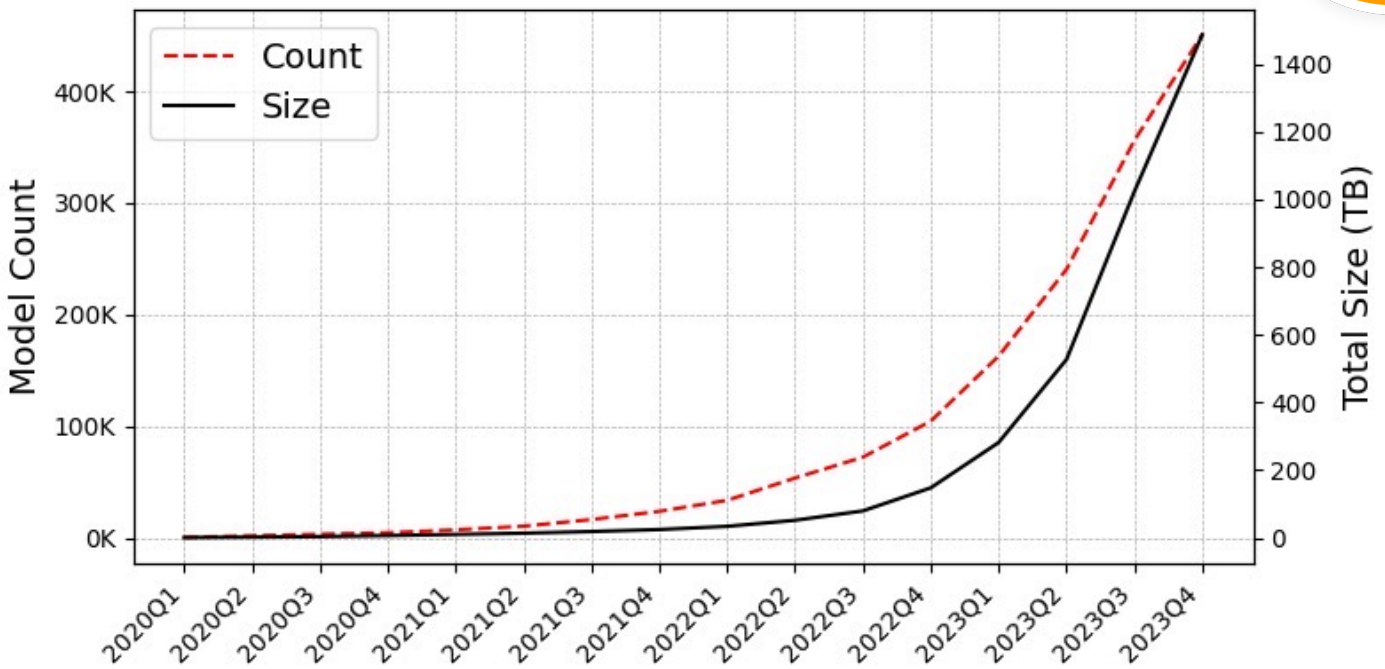
Cheap!


Google


A Google Datacenter in Hamina, Finland

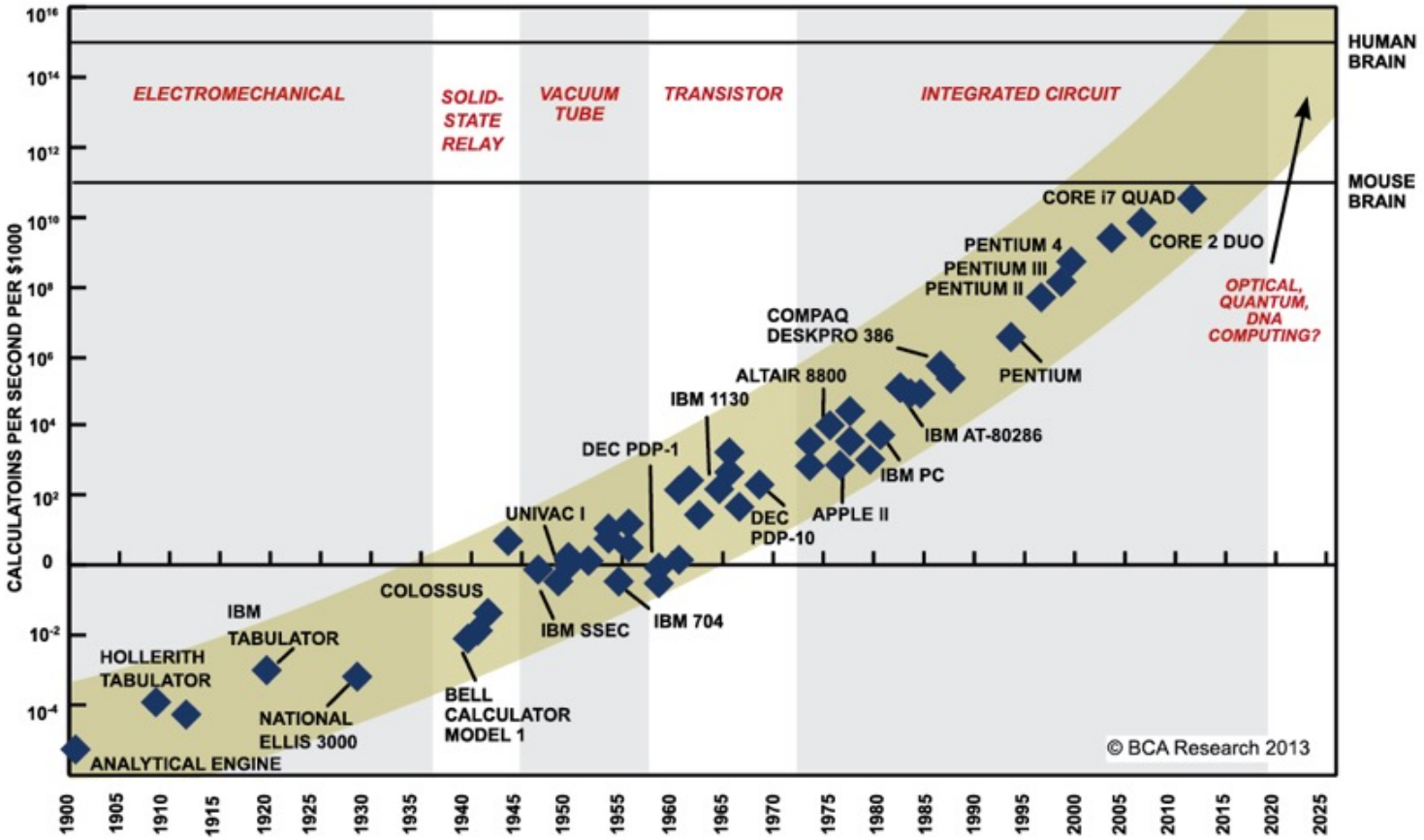
Data explosion

- Facebook's (now Meta) daily logs: 60 TB
- Google web index: 10+ PB
- Hugging Face: Stored model datasets **exponentially** increasing



Exciting time in big data systems

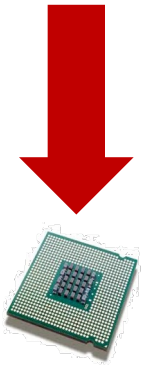
Moore's law ending → many challenges



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.

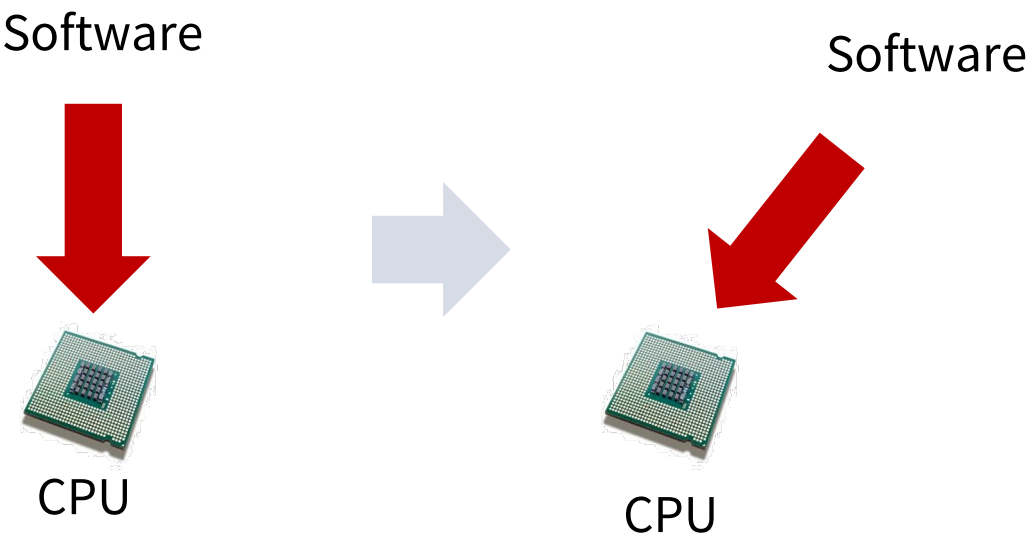
Increased complexity – Computation

Software

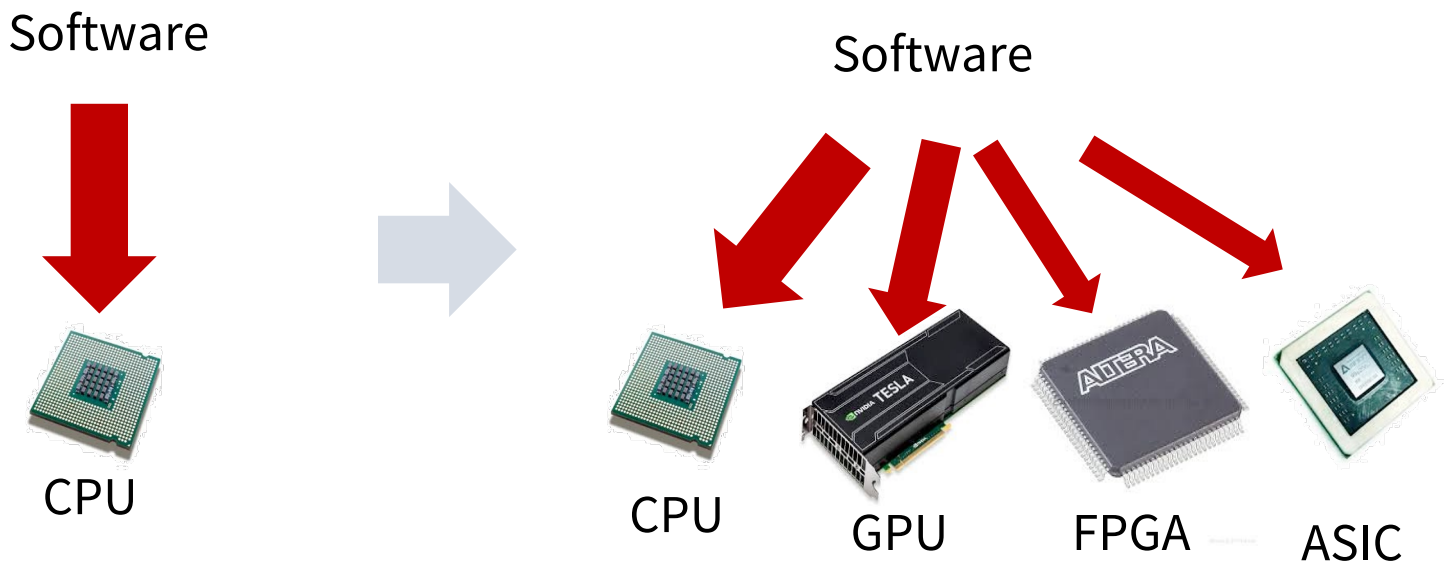


CPU

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

2023

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

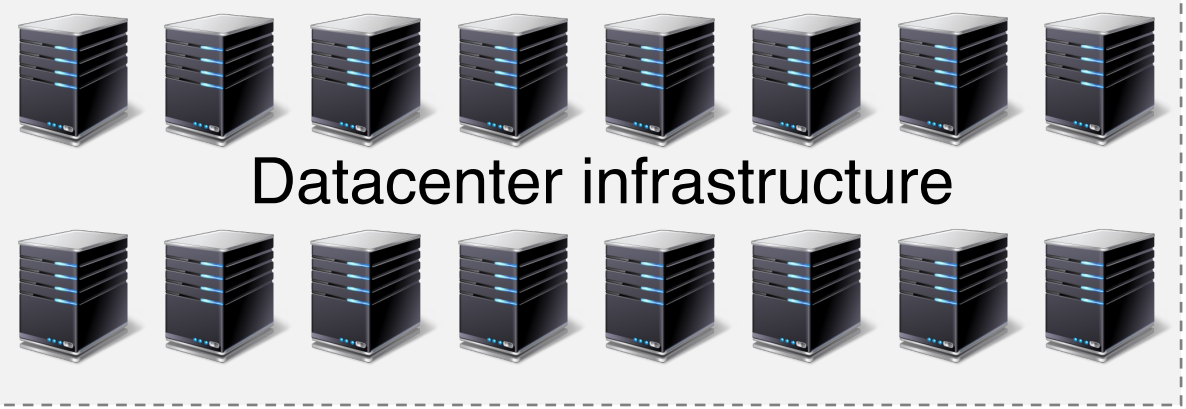
n1-standard-1, ns1-standard-2, ns1-standard-4, ns1-standard-8, ns1-standard-16, ns1highmem-2, ns1-highmem-4, ns1-highmem-8, n1-highcpu-2, n1-highcpu-4, n1-highcpu-8, n1-highcpu-16, n1-highcpu-32, f1-micro, g1-small…

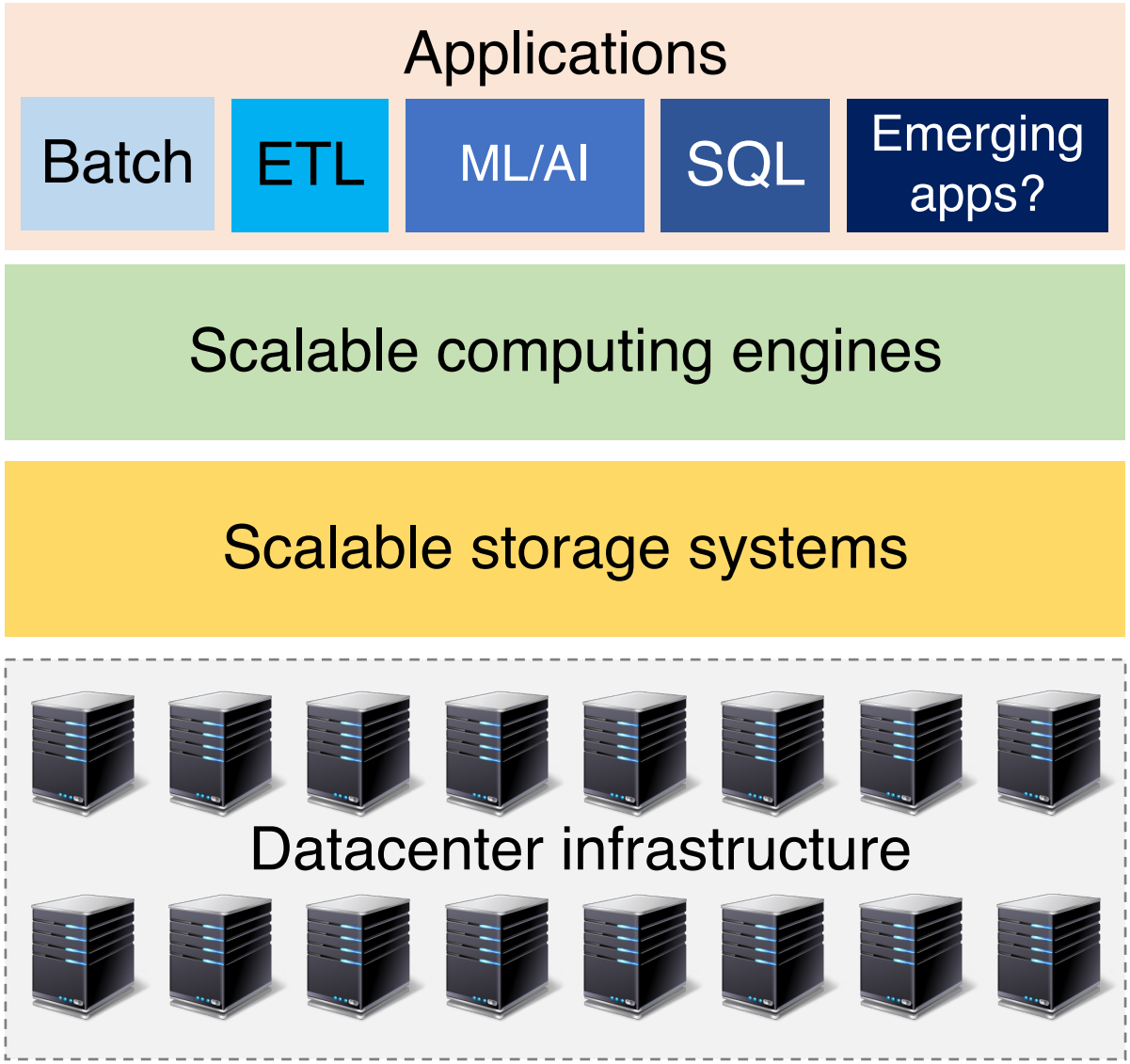
Google Cloud

But how do we program this to tackle the challenges of big data?

Scalable computing engines

Scalable storage systems




Applications
Batch
ETL
ML/AI
SQL
Emerging apps?
Scalable computing engines
Scalable storage systems
Datacenter infrastructure

Course syllabus

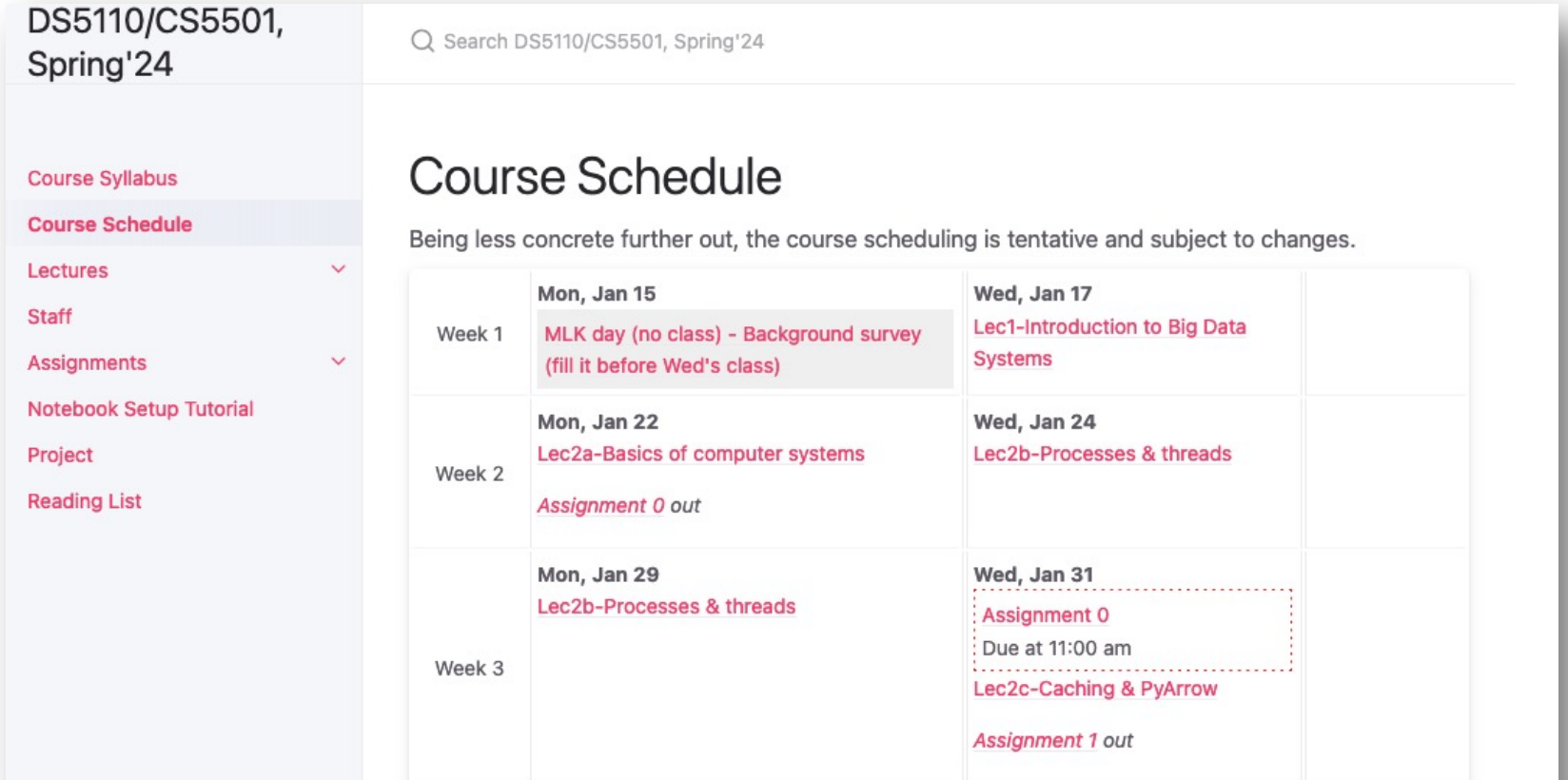
Big picture course goals

- Learn about some of the most influential works in big data systems
- Explain the design and architecture of big data systems
- Read and evaluate some seminal papers
- Develop and deploy applications on open-source big data frameworks
- Design and report some data systems ideas

Schedule (tentative)

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

https://tddg.github.io/ds5110-cs5501-spring24/



Course format: Lectures

- Lecture (+ some discussion + some demos)
  - Slides available on course website (night before or morning on the same day)
- First 3 weeks: Basics of computer systems
  - Mostly from textbook
- Week 3-6: Python analytics, MapReduce, Spark
- Week 7: Midterm exam
- Week 9-14: ML systems, cloud computing, cloud storage systems, and datacenter computing
- Some lectures have required readings
  - Review forms to help drive discussion

Course format: Review forms

- Goal: read and help with better understanding
- Review form will be posted on Ed few days before lecture
  - Around 7 review forms (tentative)
  - You need to fill out review form **by 11am** same day of lecture
  - Review form will cover required reading with a strong focus on stimulating a fruitful discussion

- No late submission will be accepted (you can use two wildcards)
- Contact instructor for exceptions in severe circumstances only

How to read a paper: GFS

The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

Google*

ABSTRACT

We have designed and implemented the Google File System, a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.

While sharing many of the same goals as previous distributed file systems, our design has been driven by observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system assumptions. This has led us to reexamine traditional choices and explore radically different design points.

The file system has successfully met our storage needs. It is widely deployed within Google as the storage platform for the generation and processing of data used by our service as well as research and development efforts that require large data sets. The largest cluster to date provides hundreds of terabytes of storage across thousands of disks on over a thousand machines, and it is concurrently accessed

1. INTRODUCTION

We have designed and implemented the Google File System (GFS) to meet the rapidly growing demands of Google's data processing needs. GFS shares many of the same goals as previous distributed file systems such as performance, scalability, reliability, and availability. However, its design has been driven by key observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system design assumptions. We have reexamined traditional choices and explored radically different points in the design space.

First, component failures are the norm rather than the exception. The file system consists of hundreds or even thousands of storage machines built from inexpensive commodity parts and is accessed by a comparable number of client machines. The quantity and quality of the components virtually guarantee that some are not functional at any given time and some will not recover from their current failures. We have seen problems caused by application

How to read a paper: Summary

- Start your reading early
- Repeat, give time between iterations
- 1st pass: Read abstract, introduction, section headings, conclusion
- 2nd pass: Read required sections, make notes
- Iterate… (if your schedule fits)

- Some key points (examples):
  - What is the **problem** being solved?
  - What are the main contributions? What is the design?
  - What workloads, setups were considered in evaluation?
  - How do they compare to prior work?
  - What parts of the claims are adequately backed up?

Course format: Discussion

- Your participation is very important
  - As an indicator of how well you've prepared

- Review form examples
  - Description of the problem or assumptions made
  - Experimental setup and what the results mean?
  - …

Textbooks?

- Papers, documentations, blog articles (required or optional) 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
  - "**Designing Data-Intensive Applications (1st edition)**" by Martin Kleppmann (can be accessed via UVA library)

Assignments

- Four programming assignments in Python on AWS
  - **Assignment 0:** Using AWS Academy, EC2, and Linux shell
  - **Assignment 1:** Parallelizing Python processing with Dask
  - **Assignment 2:** A tour of Apache HDFS and Spark
  - **Assignment 3:** A deeper dive with Ray

- All assignments are individual

- Short coding-based assignments
  - Gain hands-on experience with popular big data tools
  - Preparation for the course project

Course project

- Goal: Explore some new ideas or implementation in (big) data systems
  - Define the problem
  - Execute the research
  - Write up and present your research

- Two project styles
  - **Data analysis:** Work towards analyzing large, real-world dataset of interest using big data tools
  - **System implementation:** Work towards open-source contribution to big data tools

Course project steps

- I will distribute a list of project ideas (around Week 4)
  - You can either choose one or come up with your own
  - We will meet together to discuss
- Pick your teammates: a team of up to 4 students
  - A team of one is OK, but a team of 3-4 is recommended
- Milestones (tentative)
  - Project bid + team composition due Friday, Feb 23
  - Project checkpoint 1 due Friday, Mar 22
  - Project checkpoint 2 due Friday, Apr 12
  - Final project presentation on Wed, Apr 24 + Mon, Apr 16
  - Final project everything due Wed, May 1

Some project examples from last year

- Comparing Cylon/Dask/Modin/Spark frameworks
- Measuring market sentiment from Reddit using OpenAI
- ChatGPT tweat analysis
- Embarrassingly parallel time series modeling using AWS Lambda
- …

Grading

- Assignments (35% total)
  - Assignment 0 (5%)
  - Assignment 1 (10%)
  - Assignment 2 (10%)
  - Assignment 3 (10%)
- Reading review forms (5%)
- Quizzes (10%)
- Midterm exam (10%): open-book, open-note
- Project (40%)

FAQ: Why take this course?

- **Interesting** – hard problems, powerful solutions
- **Used by real systems** – driven by the rise of large businesses (e.g., Google, Amazon)
- **Active research area** – lots of progress + big unsolved problems
- **Hands-on** – you'll build something by the end of the semester