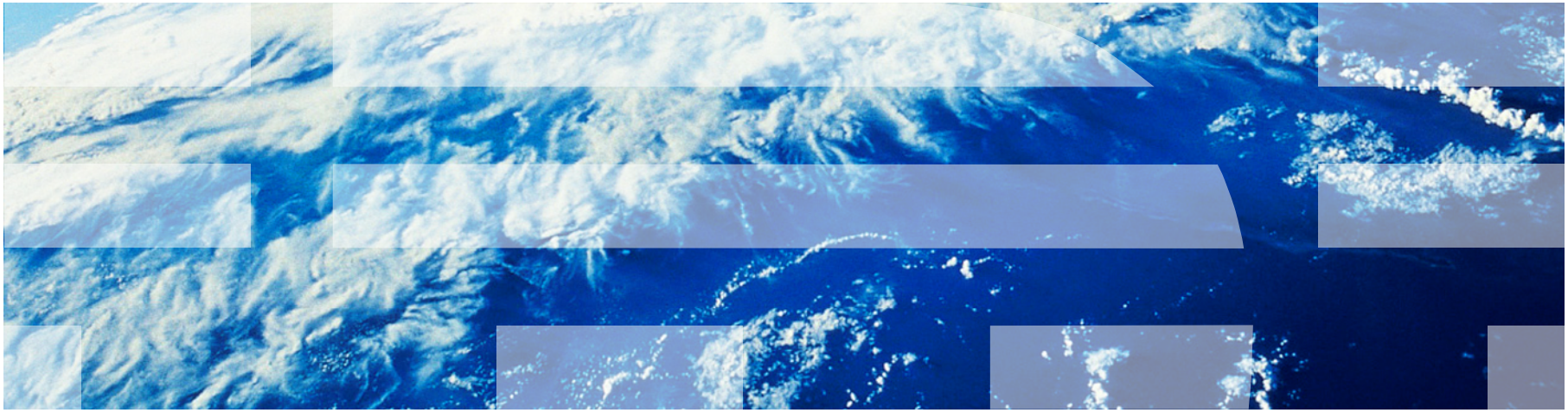
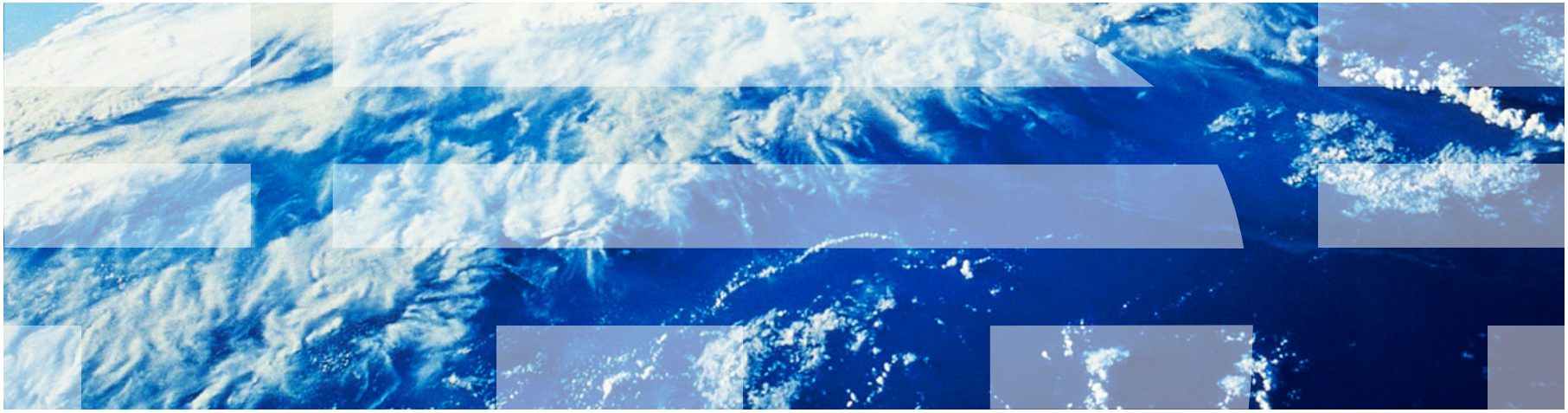
Lecture 1



Computer Systems for Data Science
Topic 1

Course Introduction

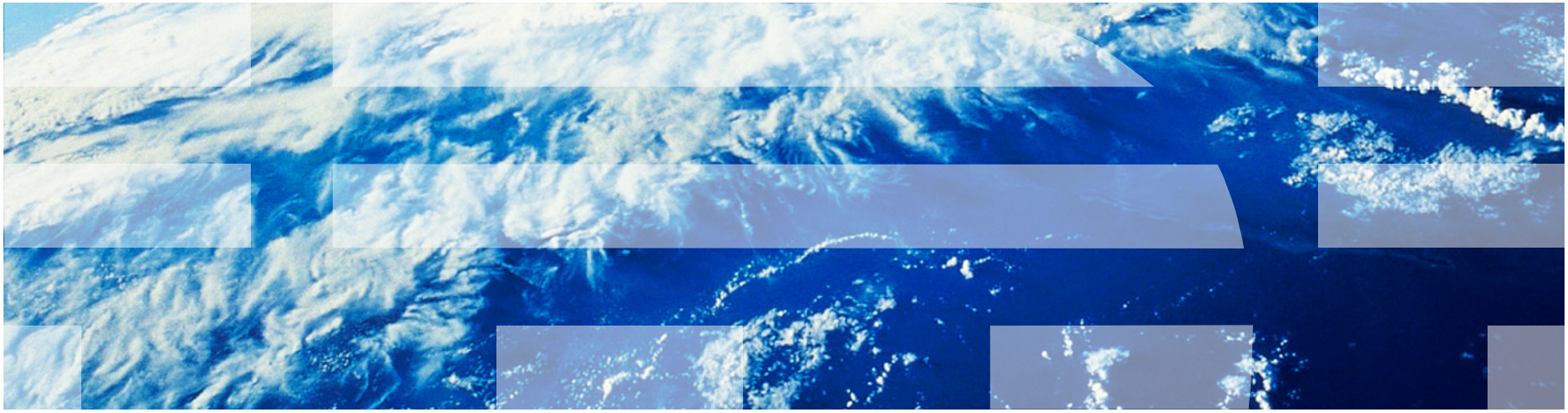
Systems concepts



Agenda

- Intro to instructors
- High-level overview
  - What is data science and big data?
  - Class goals and why should you care?
- Class logistics
  - How the class is going to work?
- Performance and systems rules of thumb
- Intro to datacenters

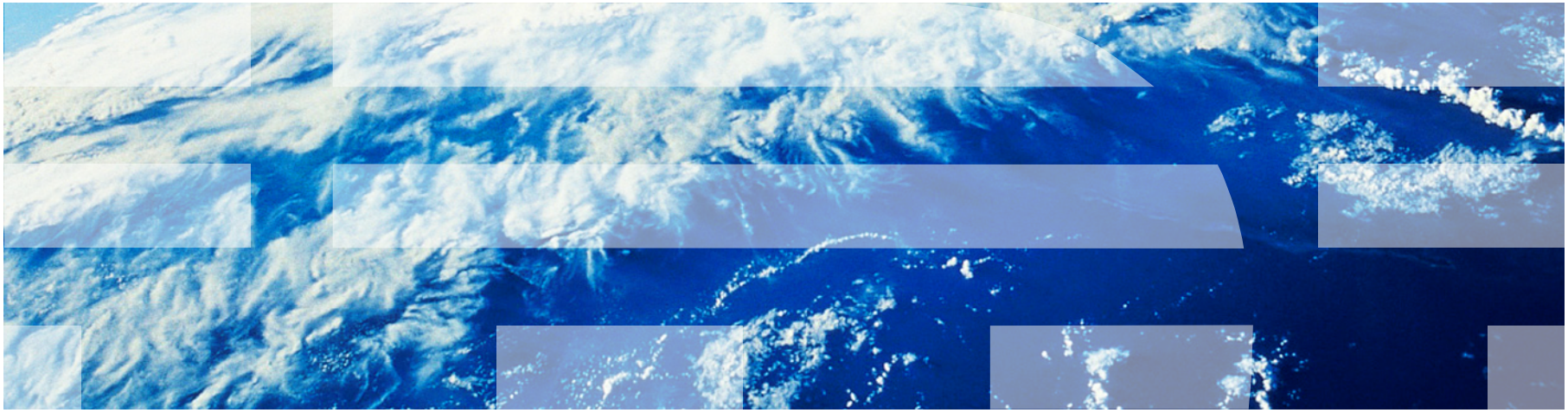
Who Are We?



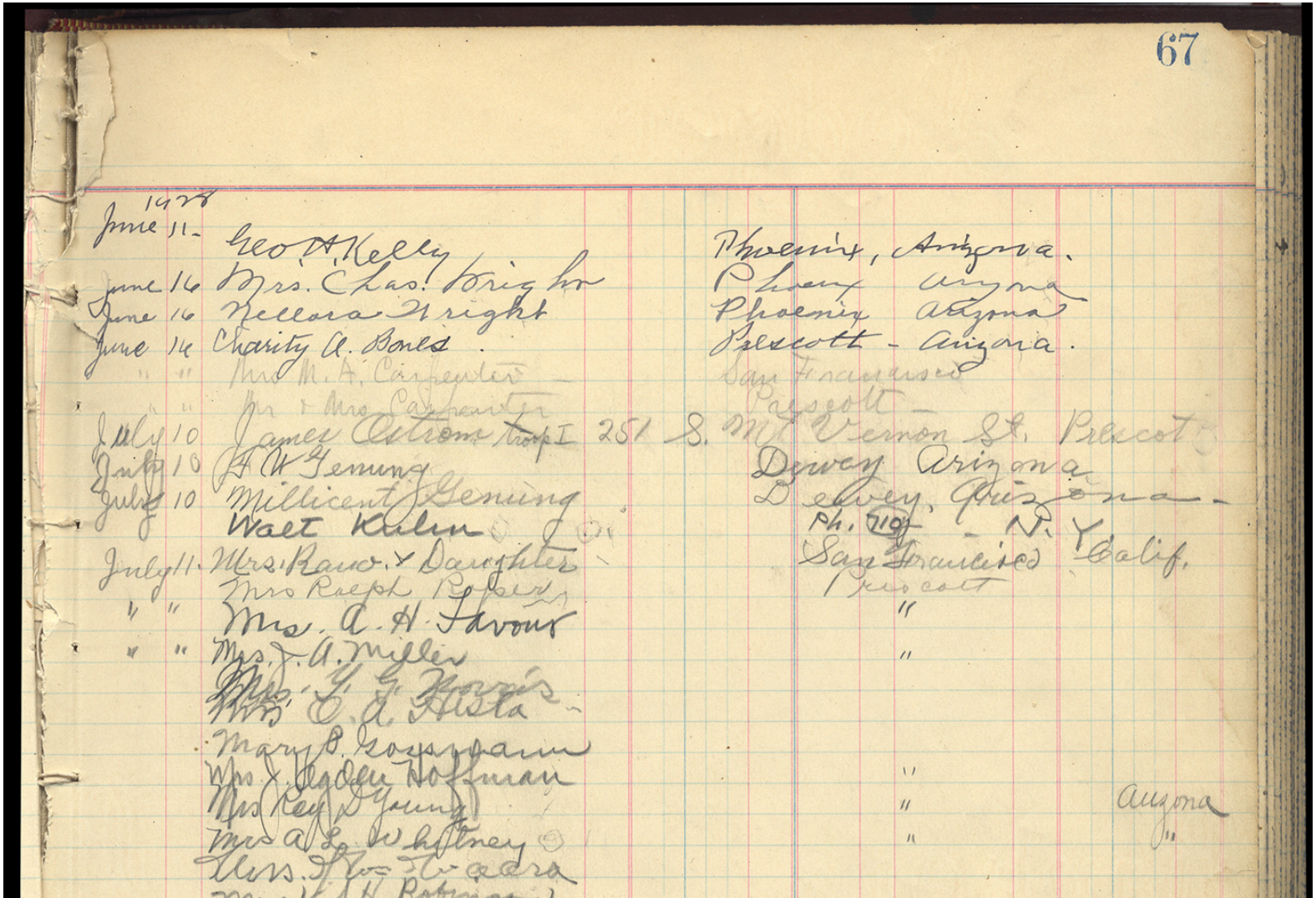
Course Instructors and TAs

- Instructors: Asaf Cidon and Sambit Sahu
- Head IAs: Rahul Chaudhari and Shantanu Jain
- IAs: Aashish Arora, Harshitha Malireddi, Joy Parikh, Ruchika Goel, Sai Karthik Ammanamanchi, Wei Hao, Zhejian Jin, Koushik Roy, Suvansh Dutta, Manisha Rajkumar
- All IAs have experience in databases and systems
  - Plus Shantanu and Rahul helped create the course homework

What is Data Science and Big Data?



This was a system for big data



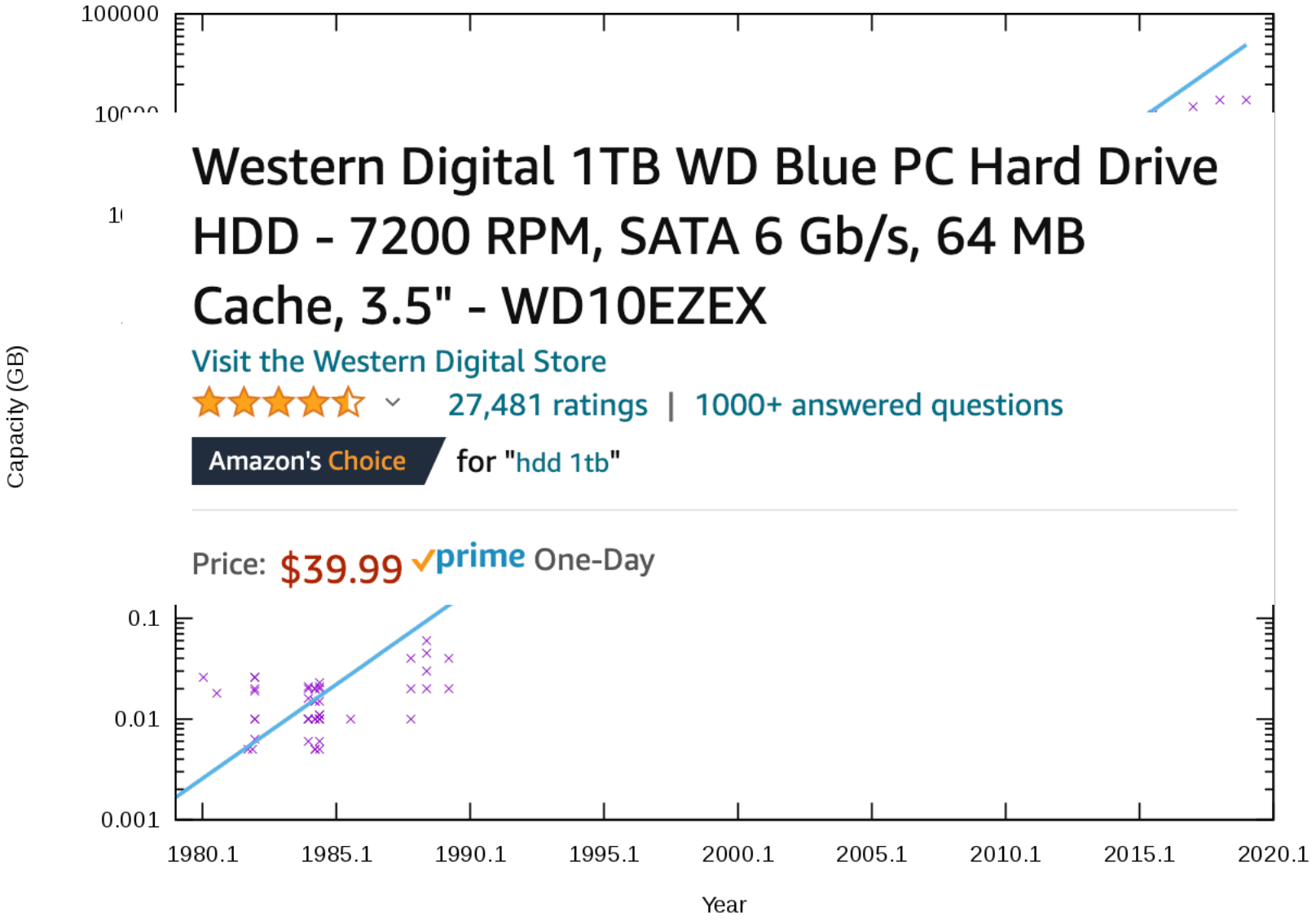
67

1928

June 11 Geo. A. Kelly — Phoenix, Arizona.
June 16 Mrs. Chas. Bright — Phoenix Arizona
June 16 Nellora Wright — Phoenix Arizona
June 14 Charity A. Bowles — Prescott - Arizona.
" " Mrs M. A. Carpenter — San Francisco
" " Mr & Mrs Carpenter — Prescott
July 10 James Astrom Troop I 251 S. Mt Vernon St. Prescott
July 10 L A Gening — Dewey Arizona
July 10 Millicent J Gening — Dewey, Arizona
Walt Kuhn — Ph. 7109 - N. Y.
July 11 Mrs. Rand & Daughter — San Francisco Calif.
Mrs Ruth Roberts — Prescott
" " Mrs. A. H. Favour — "
" " Mrs. J. A. Miller — "
Mrs. E. G. Harris
Mrs. E. A. Hista
Mary B. Rozzgrann
Mrs. Hadley Hoffman
Mrs Ray D Young — "
Mrs A. L. Whitney — " Arizona
Mrs. Ina Tweedy — "

Data science systems were expensive

Today: data is cheap



Where is data coming from?

- Physical devices

Where is data coming from?

- Physical devices
- Software logs

Where is data coming from?

- Physical devices
- Software logs
- Phones

Where is data coming from?

- Physical devices
- Software logs
- Phones
- GPS/Cars

Where is data coming from?

- Physical devices
- Software logs
- Phones
- GPS/Cars
- Internet of *Things*

What can we do with all this data?

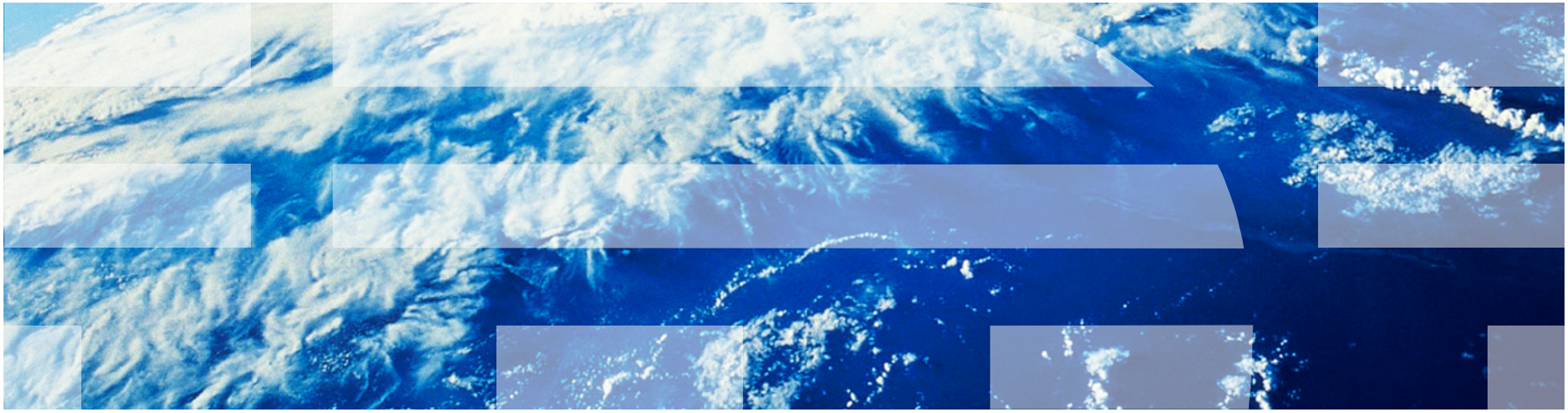
- What video should I recommend to this user to view next?
- Does this MRI image of a breast contain a tumor?
- Who is going to win the election?
- Which cities in the US will have high incidence of flu in 2 weeks?
- Is the object across from the car a pedestrian?

What is big data?

- "**Extremely large data sets** that may be analyzed computationally to reveal patterns, trends, and associations, especially relating to human behavior and interactions" – Oxford Dictionary

- What's an extremely large data set?
  - Fits on a single machine?
  - Fits on 10 machines?

Ok… But what is this class about?



Our focus in this class: **Computer Systems** for Data Science

- Questions we **will** answer in this class:
  - How are big data systems designed?
  - How to store the data?
  - How to query/analyze the data?
  - How do we ensure uptime/availability to the data?
  - How to ensure privacy/security/quality?
- Questions we **won't** answer in this class:
  - What algorithm should we use?
  - How to use ML for big data?
  - How do we explain/debug ML models?
  - How can data be visualized?
  - What are the statistical/mathematical foundations for data science?

Course Objectives

- **Graduate-level course**
- **Broad overview of cloud systems that are used in data science**
  - **Database** related topics (DBMS, SQL, NoSQL, data lakes/warehouses)
  - **Computer systems** foundations (throughput vs. latency, scalability vs. performance)
  - **Distributed systems** for data scientists (sharding, fault tolerance)
  - **Basic security** for data scientists (encryption, privacy)
- Throughout the class we will focus on how **commonly used and modern** cloud-based big data systems work (Spark, Tensorflow, Hadoop file system,…)
- The class will give a **broad and hopefully practical** introduction to these topics geared towards data scientists, but **does not replace** core CS/EE classes like OS, databases, distributed systems, security, architecture
- **You come from diverse backgrounds:** Some of the content will be repetitive for students who have taken the classes above
- **Required background**
  - Programming experience with Python or equivalent
  - Programming assignment 2 will be in pairs in Python, so we will make sure that all pairs have at least one student who knows Python

Course Administration and Grading

- **All materials, assignments, etc. posted on course website**
  - https://csee-4121-2022.github.io/
- **Announcement will be posted on Piazza, courseworks**
- **Lecture Materials**
  - Lecture slides
  - No textbook (new, fast moving field)
- **Homework, assignments, exams**
  - Programming assignment 1: BigQuery (20%), alone
  - Written assignment 1: systems and databases (10%), alone
  - Midterm (15%)
  - Programming assignment 2: Spark (20%), **in pairs**
  - Written assignment 2: distributed systems and security (10%), alone
  - Final exam (25%)
- **All assignments, midterm and final will be turned in online**
- **All classes streamed online (Zoom) and recorded (available on CourseWorks)**
  - No attendance required
  - If you want to watch other section you can do that over Zoom/recording, but not in person

Programming Assignments

- 2 programming assignments
  - Assignment 1 is solo, assignment 2 is in pairs
- Programming assignments are in Python
  - If you don't know Python
    - Pair with someone who does
    - A useful language to learn!
- Programming assignments in Google Cloud (GCP)
  - Goal: familiarize yourself with working in public cloud environment
    - AWS / Azure / GCP are similar
    - Many systems and deployment details are hidden / automated (but we won't ignore them!)
    - We will be focusing on systems-level problems, not on algorithms
  - We will provide GCP credits, if you run out contact us
    - If you reach $10 of credits or less, please contact: TBD
    - But be careful not to spend too many!
- Programming assignment goals
  - Assignment 1: BigQuery
    - Learning to use SQL on a big data set
  - Assignment 2: DataProc + Spark + HDFS + Streaming
    - Understanding Spark, Streaming systems concepts

More logistics

- **Office hours:**
  - CAs will hold office hours every day over Zoom
  - We will announce the Zoom link: all office hours will use the same Zoom link
- **Piazza**
  - A CA is guaranteed to be available on Piazza every weekday (when the school is open) from 9AM – 5PM. We will try to answer your questions within 1 hour during those time windows
  - We will cannot guarantee a fast response when questions are answered not in those times windows
- **Late days**
  - You have 3 “free” late days you can choose to use on any programming assignment
    - For programming HW 2, both of the team members need to use a late day for it to count
  - If you’ve used up all your late days and you submit your assignment late:
    - 1st late day results in 5% point reduction
    - 2nd late day results in 10% point reduction
    - 3rd late day results in 20% point reduction
    - Beyond that the assignment will get a score of 0

Tentative Contents and Syllabus

- Computer systems and performance rules of thumb
  - Latency vs. throughput
  - Amdahl's law
  - Back-of-the-envelope systems math
  - Performance bottlenecks
- Data centers
  - What is a data center?
  - Data center failures
  - Achieving reliability with smart software
- Databases
  - Relational model and SQL
  - SELECT, FROM, WHERE
  - GROUPBY
  - JOINs
  - Nested queries
  - Transactions
  - ACID
  - OLAP vs. OLTP, SQL vs. NoSQL
  - Indexing
  - Logging
  - **System highlight: BigQuery, MySQL**
- Storage and distributed file systems
  - Storage technologies primer
  - Distributed file systems
  - **System highlight: Hadoop File System (HDFS), amazon S3/Google Cloud Storage**

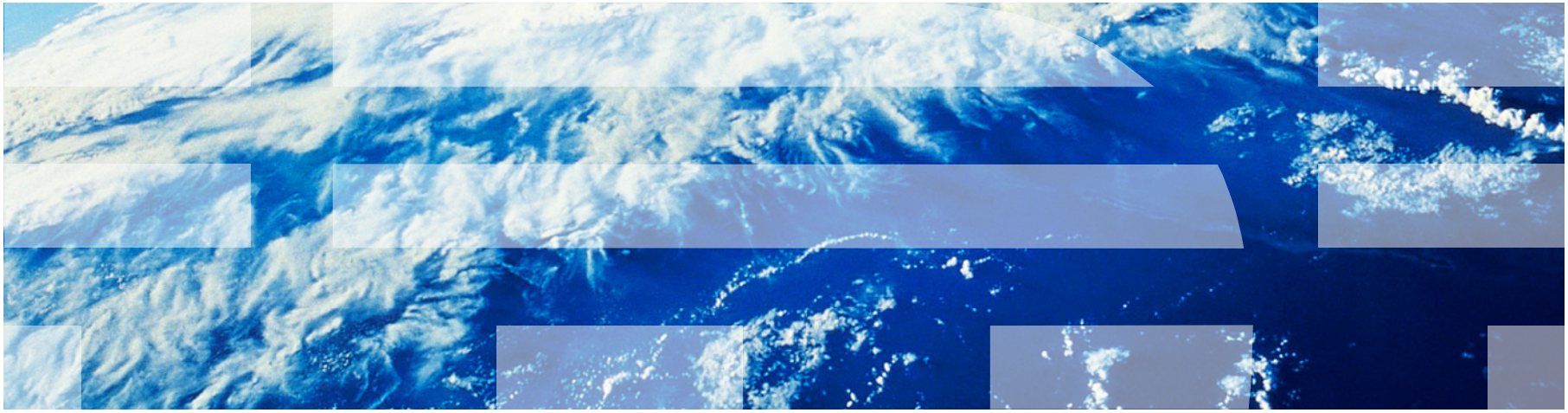
Tentative Contents and Syllabus

- Distributed systems
  - 2 Phase Commit
  - Locking
  - Sharding
  - Fault tolerance
  - Replication and consensus
- Mapreduce
  - Mapreduce computing model
  - Stragglers
  - Importance of 99th latency
  - Strategies to mitigate tail latency and increase availability
- Distributed analytics and streaming
  - Resilient Distributed Dataframes (RDD)
  - Fault tolerance in distributed analytics: lineage
  - Streaming computing model
  - **Systems highlight: Spark, Google Dataproc, Spark streaming**
- Caching
  - Performance benefits
  - When to use a cache? (hint: almost everywhere ☺)
  - Consistency and performance considerations
  - Eviction policies
  - **Systems highlight: Memcached and Redis**

Tentative Contents and Syllabus

- Tensorflow and pipelines
  - Tensorflow programming model
  - ML hardware trends
  - ML pipelines
  - Data validation and data quality
  - **Systems highlight: Tensorflow and Tensorflow Extended**
- Security and privacy
  - Security of big data systems
  - Privacy consideration
  - Data compliance and access control
- Data Quality
  - Data pipelines
  - Data quality
  - Importance of data observability

Performance Concepts and Rules of Thumb



Performance Evaluation

- Metric: something we measure
- Goal: evaluate how good/bad our computer system is performing
- Examples:
  - Power consumed by our database
  - Cost of running our web application
  - Average time it takes to render a user page
  - How many users can we support at the same time
- Metrics allow us to compare two computer systems

Tradeoff: latency vs. throughput

- Pizza delivery example
  - Do you want your pizza hot?
  - Do you want your pizza to be cheap?
- Why do these conflict?
- Two different strategies for pizza company
  - Often we have a requirement for both (I want my pizza to be delivered in X time as cheaply as possible)

- Latency = execution time for a single task
- Throughput = number of tasks per unit time

- A more relevant example:
  - Latency requirement: Assuming cars drive at 65mph, so self driving car needs to recognize an object in 0.1 seconds
  - Throughput requirement: Object recognition system needs to process 1 million object recognition tasks every second to support 10,000 cars simultaneously

Latency vs. Throughput is often a trade off

Plane	DC to Paris	Speed	Passengers	Throughput (pmph)
Boeing 747	6.5 hours	610 mph	470	286,700
Concorde	3 hours	1350 mph	132	178,200

- Which plane has higher **performance**?

- Time to do the task (execution time)
  - **Latency**, execution time, response time
- Tasks per day, hour, week, sec (performance)
  - **Throughput**, bandwidth, operations per second

Definitions

- Performance is in units of things-per-second
  - Bigger is better
- Response time of a system Y running Z
  - $\text{performance}(Y) = \frac{1}{execution\ time\ (Z\ on\ Y)}$
- Throughput of system Y running many requests
  - $\text{performance}(Y) = \frac{number\ of\ requests}{unit\ time}$
- "System X is n times faster than Y" means:
  - $n = \frac{performance(X)}{performance(Y)}$

How do we improve performance?

- Suppose we have a database that processes two types of queries:
  - Query A finishes in 100 seconds
  - Query B finishes in 2 seconds

- We want better performance
  - Which query should we improve?

- The answer: it depends!

Speedup

- Make a change to the system
- Measure how much faster/slower it is

- $Speedup = \frac{Execution\ time\ before\ change}{Execution\ time\ after\ change}$

Speedup when we know details about the change

- Performance improvement depends on:
  - How good is the enhancement? (factor S)
  - How often is it used? (factor p)
- Speedup due to enhancement E:
  - $Speedup(E) = \frac{Execution\ time\ without\ E}{Execution\ time\ with\ E} = \frac{Performance\ with\ E}{Performance\ without\ E}$
  - $ExTime_{new} = ExTime_{old} * \left[(1-p) + \frac{p}{S}\right]$
    - Explanation:
    - $(1-p)$ is the fraction of operations that are not affected by E
    - $\frac{p}{S}$ is the fraction of operations that are affected by E, with the enhancement factor
  - $Speedup(E) = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{(1-p)+\frac{p}{S}}$

Amdahl's law: example

- We built a new database that speeds up aggregate queries by 2x! Hurray!
- But… only 10% of queries are aggregate queries
- $ExTime_{new} = ExTime_{old} * \left[(1-p) + \frac{p}{s}\right]$
- $ExTime_{new} = E$

Amdahl's law in simple terms:
Make the common case fast!

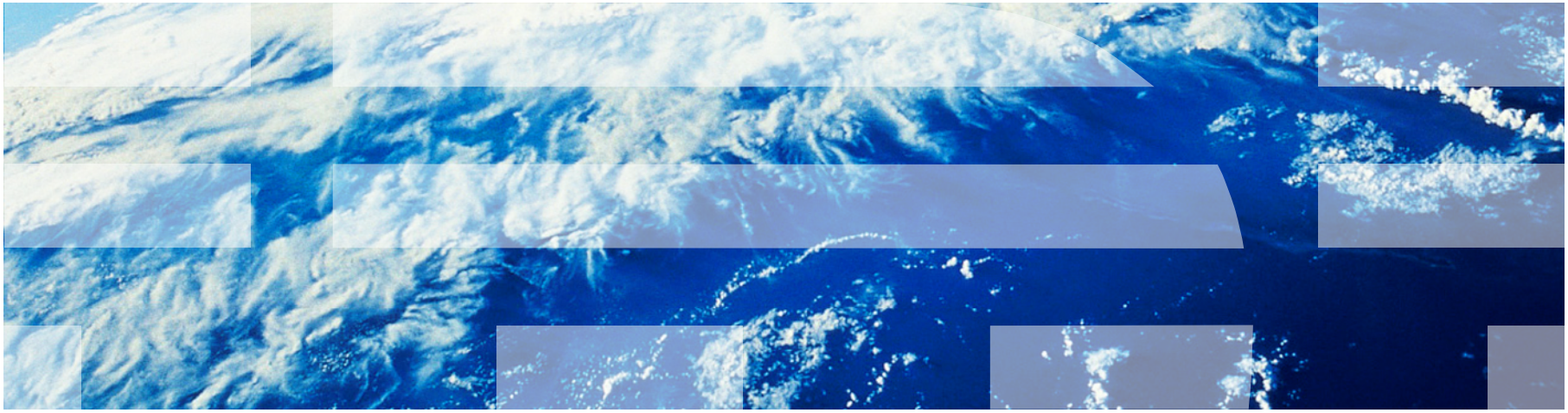
- $Speedup_{total} = \frac{1}{0.95} = 1.053$ → only 5.3% overall speedup ☹

- Amdahl's law: speedup bounded by

$$\frac{1}{fraction\ of\ time\ not\ enhanced}$$

- Even if aggregated queries could be completed in zero time, our **maximum** speedup would be:
- $Speedup_{optimal} = \frac{1}{0.9} = 1.111$

Lecture 2



Useful back-of-the-envelope latency numbers (all rough estimates)

- Time measurements:
  - Nanosecond (ns): 1/1,000,000,000 second
  - Microsecond (us): 1/1,000,000 second
  - Millisecond (ms): 1/1000 second

- CPU cache access: 1ns
- Memory access: 100ns
- Read a small object from a random location on a local flash drive: 50,000ns, 50us
- Read a small object within the same network in a data center: 100,000ns, 100us
- Run a SQL query on a flash database: 1,000,000ns, 1ms
- Read a small random object from magnetic disk: 10,000,000ns, 10ms
- Run a SQL query on a disk database: 20,000,000ns, 20ms
- Roundtrip time over the internet: 100,000,000ns, 100ms
  - Bounded by the speed of light! Roundtrip light speed from NYC to Beijing is ~150ms

Quick recap

- What we covered last time?
  - Motivation, logistics etc etc
  - Latency vs. throughput
    - Pizza hot vs. cheap
  - Amdahl's law
    - Speed up the common case
  - Back of the envelope numbers
    - CPU cache << memory << flash ~ local network << hard disk << packet over the Internet

- Agenda for today
  - How to use back of the envelope numbers
  - What is the cloud?
  - Start of topic 2: relational model

How can we use these numbers? A database example

- Scenario:
  - A user application running in the cloud needs to read a small object (e.g., lookup the student's name using their CUID).
  - It first checks if the object is already saved locally, either in the CPU cache or in memory:
    - 10% chance it's in the CPU cache
    - If not, 20% chance it's in memory
  - If not saved locally, it fetches it from a database from within the same network

- Compute exepcted latency:
- Prob(CPU) * cache_latency + Prob(not in CPU) * (Prob (memory) * memory_latency + Prob (not in memory) * database_latency)
- 0.1 * cache latency + 0.9 * (0.2 * memory latency + 0.8 * (database latency))
  = 0.1ns + 18ns + 0.72 * database latency
- Remote database latency = network latency + database latency = 1,100,000ns
- Total average latency = 792,018ns or 790us
- Total average latency ~= 0.72 * not in memory latency = 792,000ns
- → If there is a problem, it's probably caused by the slowest component (database)

Disk vs. Flash, Cost vs. Performance

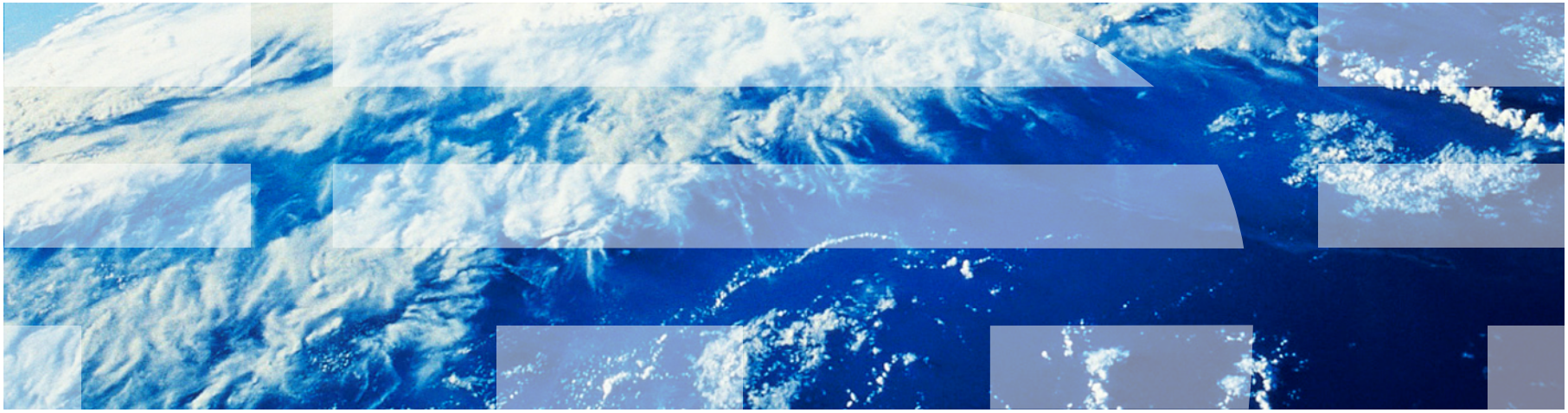
- Acme runs a COVID prediction service
- They have an app that displays a graph on the geographic spread of COVID, which requires running a SQL query on their database stored in the cloud
- Acme is considering running their database on flash vs. magnetic disk
  - They received some quotes from database company, and flash database is 2X more expensive, but 10X faster
- An Acme user study shows that users don't notice page loading times, as long as they are under 300,000,000ns (300ms)
- Acme measured: Internet roundtrip (100ms), disk DB access (10ms), flash DB access (1ms)
- Scenario 1: To compute the graph, we only need a single database access in the cloud (over the Internet)
  - Latency with flash database: 101ms
  - **Latency with disk database: 110ms**
- Scenario 2: The app requires getting an initial response from the cloud database, then a user input, and then another cloud database request
  - Latency with flash database: 202ms
  - **Latency with disk database: 220ms**
- Scenario 3: The app requires 20 sequential databases accesses within the cloud to compute the graph, and then it can return
  - **Latency with flash database: 120ms**
  - Latency with disk database: 300ms

Identifying performance bottlenecks

- My application is seeing an average latency of 200ms, where is the bottleneck?
- A few guiding questions:
  1. What systems does the web page need to access? Which networks does it need to traverse?
  2. Start from the most common case + highest latency
- Example:
  - Application needs to go through the Internet ~ 1 * 100ms
  - Hits a server that first checks if the request is saved on memory cache in the cloud ~ 0.2 * 100us
  - If not (80% of the time), goes over the network and accesses a single disk database ~ 0.8 * 10ms
- Guess 1: Internet slowdown (highest latency)
- Guess 2: database slowdown (second highest latency)

Adapted from Mendel Rosenblum and Jeff Dean

The Infrastructure of Big Data

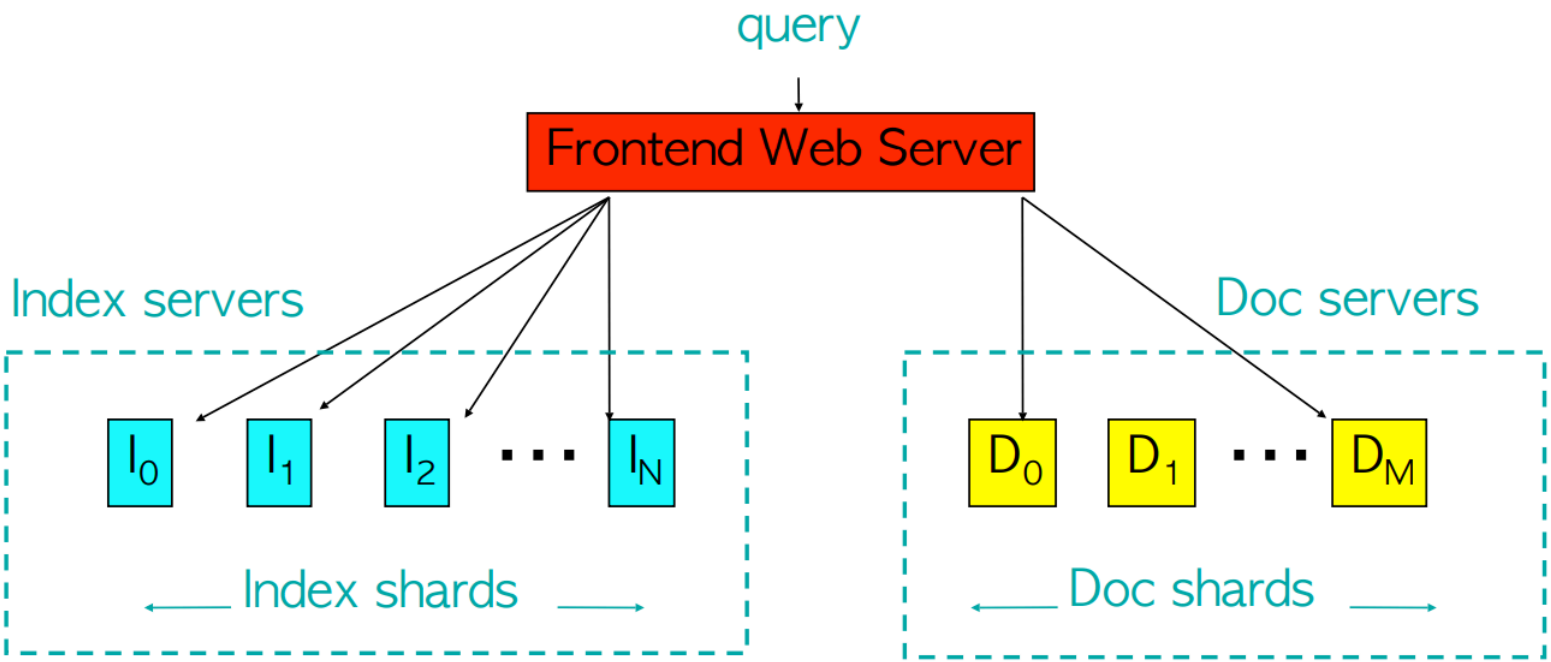


Motivating example: Google web search (1999 vs. 2010)

# docs: tens of millions to tens of billions	~1000X
Queries processed/day:	~1000X
Per doc info in index:	~3X
Update latency: months to tens of seconds	~50000X
Average query latency: 1 seconds to 0.2 seconds	~5X
More machines * faster machines:	~1000X

Google Circa 1997 (definitely not big data)

Google infrastructure circa 1997 could fit in a single room



Scaling up

- What happens when a server doesn't fit in a single room?
- What happens if we need 1000X more servers?

- The cloud to the rescue!
  - Also known as… **data centers**

Evolution of data centers

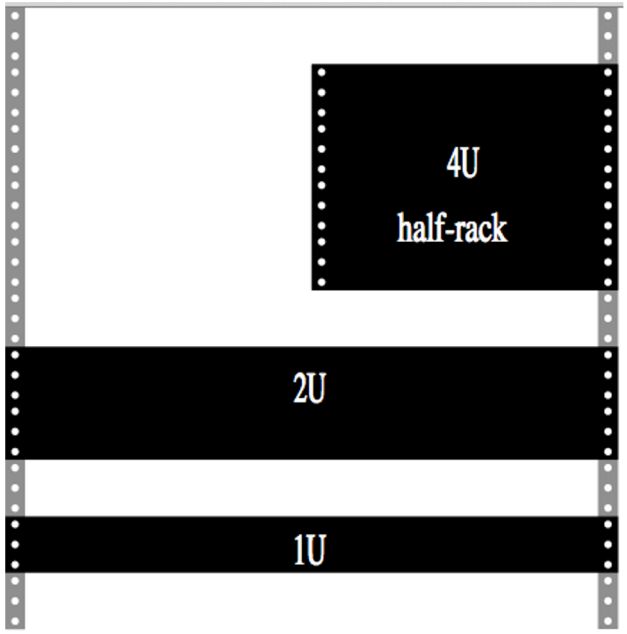
- 1960's, 1970's: a few very large time-shared computers
- 1980's, 1990's: heterogeneous collection of lots of smaller machines.
- Today and into the future:
  - Data centers contain large numbers of nearly identical machines
  - Geographically spread around the world
  - Individual applications can use thousands of machines simultaneously
- Companies consider data center technology a trade-secret
  - Limited public discussion of the state of the art from industry leaders

Typical specs for a data center today

- 15-40 megawatts power (Limiting factor)
- 50,000-200,000 servers
- $1B construction cost
- Onsite staff (security, administration): 15

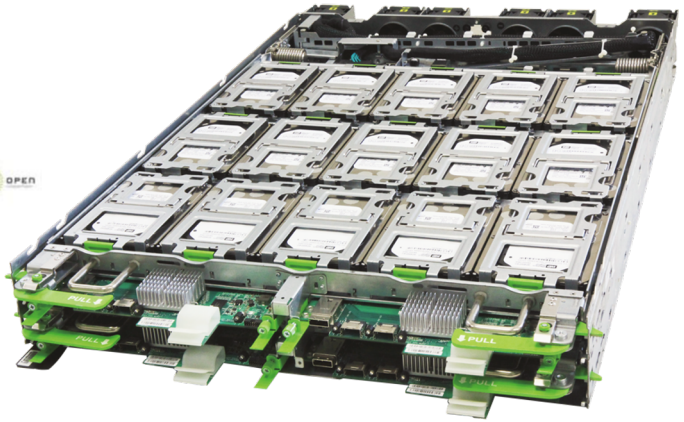
Rack

- Typically is 19 or 23 inches wide
- Typically 42 U
  - U or RU is a Rack Unit - 1.75 inches
- Slots:



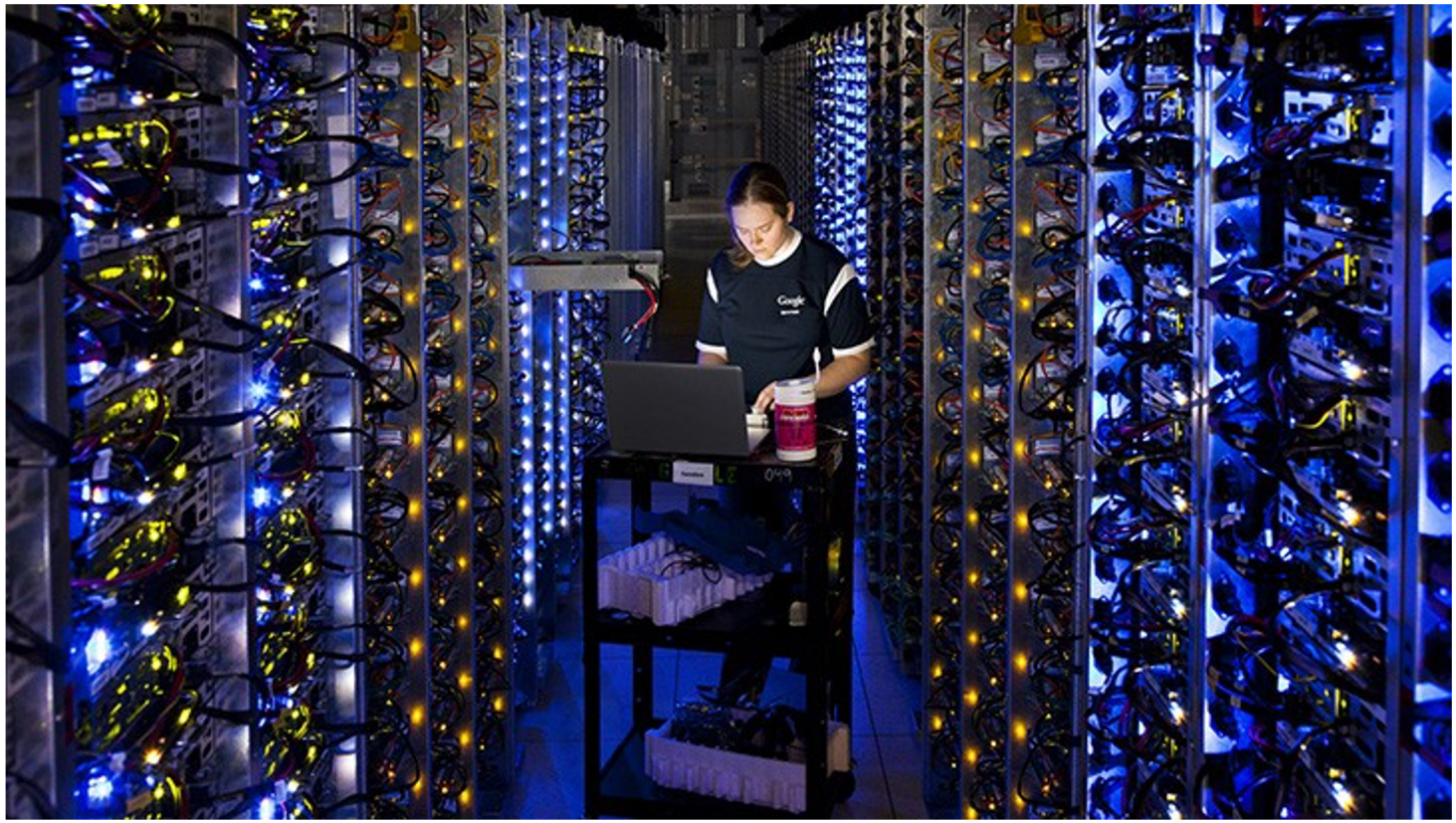
Rack Slots

- Slots hold power distribution, servers, storage, networking equipment
- Typical server: 2U
  - 8-128 cores
  - DRAM: 32-512 GB
- Typical storage: 2U
  - 30 drives
- Typical Network: 1U
  - 72 10GB



Row/Cluster

- 30+ racks

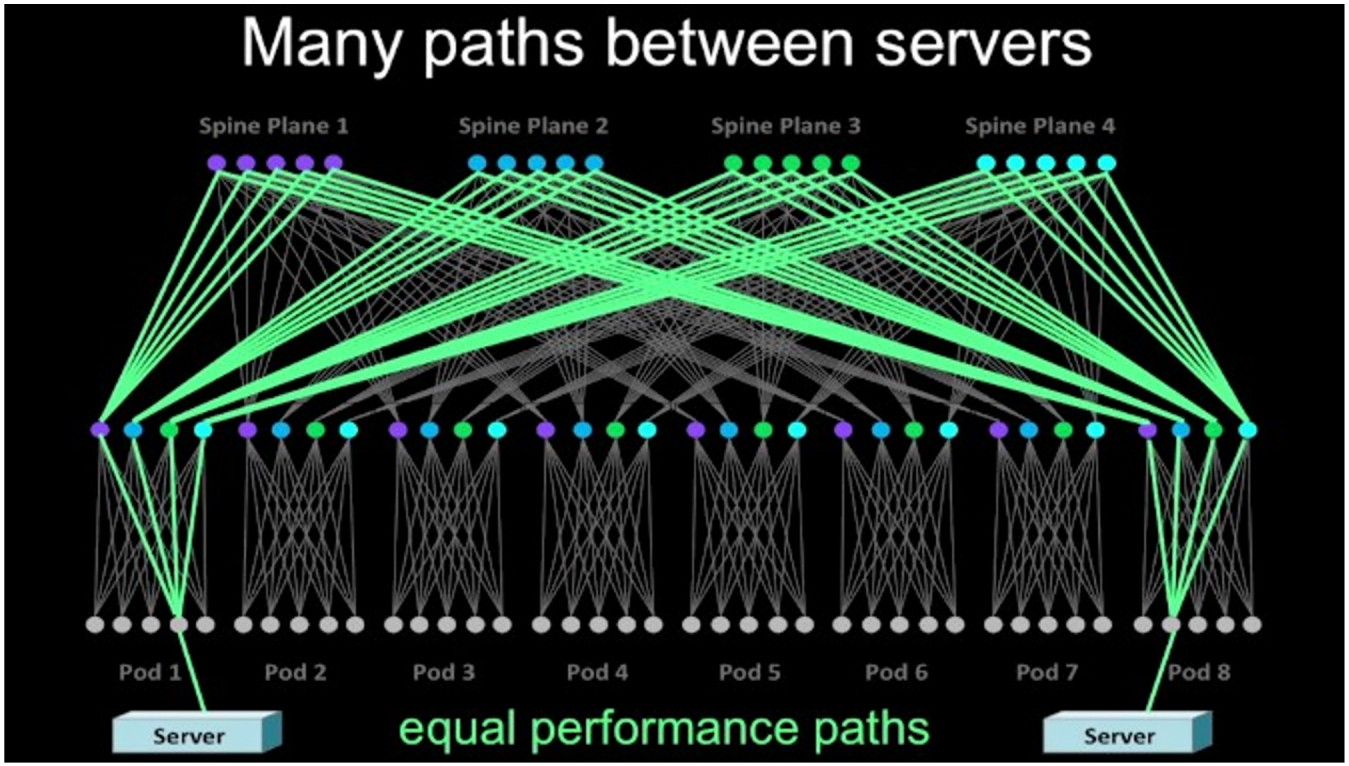


Networking - Switch locations

- Top-of-rack switch
  - Connecting machines in rack
  - Multiple links going to end-of-row routers
- End-of-row router
  - Aggregate row of machines
  - Multiple links going to core routers
- Core router
  - Multiple core routers

- Each of these have different latencies, throughput

Multipath routing



Ideal: "full bisection bandwidth"

- Would like network where everyone has a private channel to everyone else
  - (cross-bar topology)
  - Why is this useful?

- In practice today:
  - Assumes applications have locality to rack or row but this is hard to achieve in practice.

Power Usage Effectiveness (PUE)

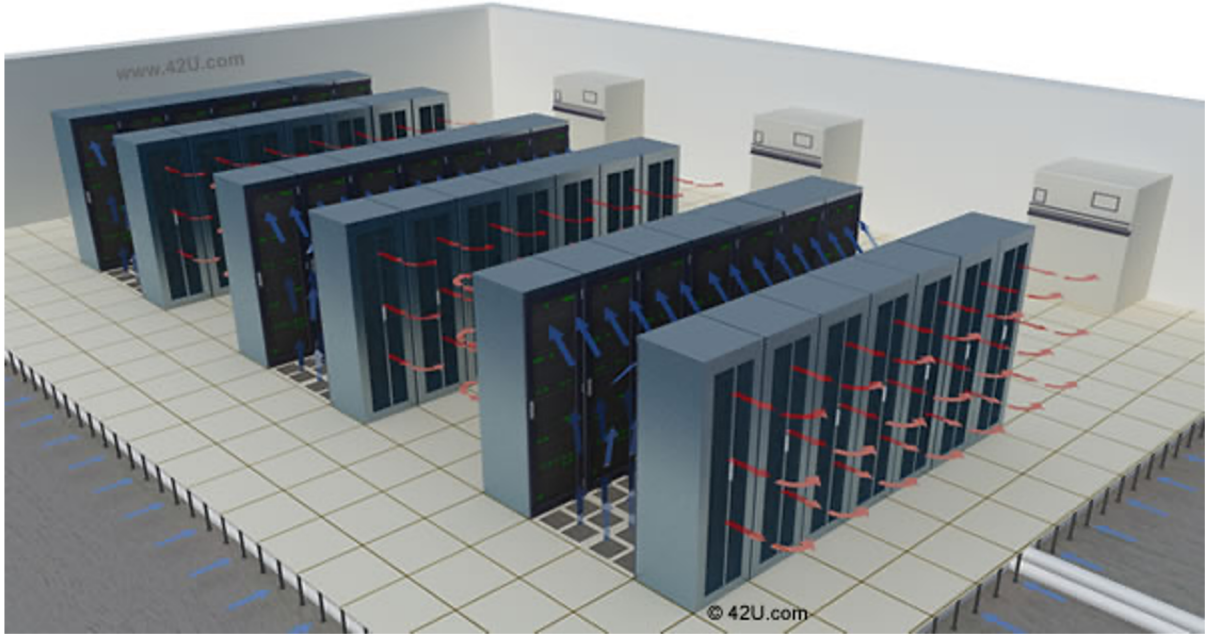
- Early data centers built with off-the-shelf components
  - Standard servers
  - HVAC unit designs from malls
- Inefficient: Early data centers had PUE of 1.7-2.0

$$\text{PUE ratio} = \frac{\text{Total Facility Power}}{\text{Server/Network Power}}$$

- Best-published number (Facebook): 1.07 (no air-conditioning!)
- Power is about 25% of monthly operating cost
  - And is a limiting factor in how large the datacenter can be

Energy Efficient Data Centers

- Better power distribution - Fewer transformers
- Better cooling - use environment (air/water) rather than air conditioning
  - Bring in outside air
  - Evaporate some water
- IT Equipment range
  - OK up to +115°F



Backup Power

- Massive amount of batteries to tolerate short glitches in power
  - Just need long enough for backup generators to startup
- How do glitches occur?
  - Thunder, earthquake, power loss from power company, cyber attack, …
- Massive collections of backup generators
- Huge fuel tanks to provide fuel for the generators
- Fuel replenishment transportation network (e.g. fuel trucks)

Fault Tolerance

- At the scale of new data centers, things are breaking constantly
- Every aspect of the data center must be able to tolerate failures
- Solution: Redundancy
  - Multiple independent copies of all data
  - Multiple independent network connections
  - Multiple copies of every services

Failures in first year for a new data center (Jeff Dean)

~thousands of hard drive failures

~1000 individual machine failures

~dozens of minor 30-second blips for DNS

~3 router failures (have to immediately pull traffic for an hour)

~12 router reloads (takes out DNS and external VIPs for a couple minutes)

~8 network maintenances (4 might cause ~30-minute random connectivity losses)

~5 racks go wonky (40-80 machines see 50% packet loss)

~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)

~1 network rewiring (rolling ~5% of machines down over 2-day span)

~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hours)

~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hours to come back)

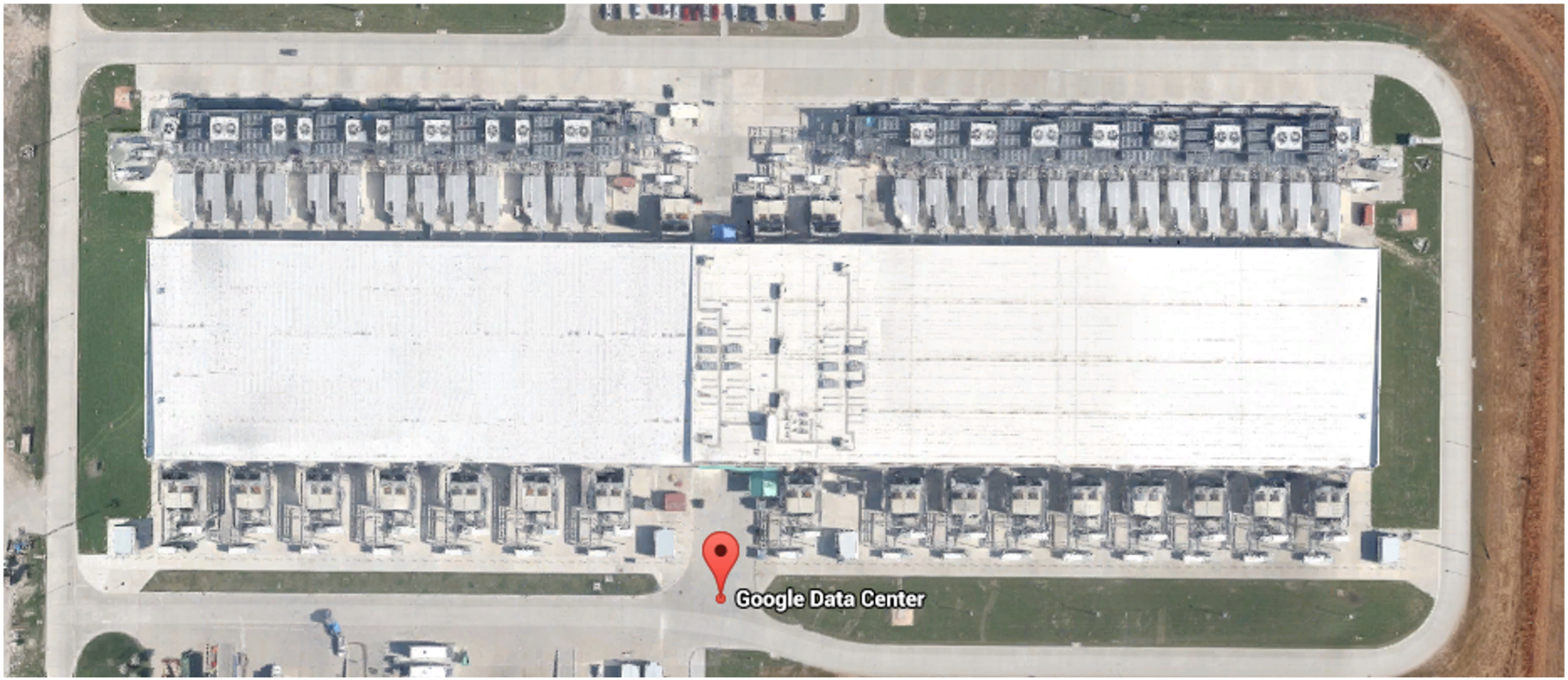
~0.5 overheating (power down most machines in <5 mins, ~1-2 days to recover)

→ Reliability must come from software!

How to choose where to place a data center

- Plentiful, inexpensive electricity
  - Examples - Oregon: Hydroelectric; Iowa: Wind
- Good network connections
  - Access to the Internet backbone
- Inexpensive land
- Geographically near users
  - Speed of light latency
  - Country laws (e.g. Our citizen's data must be kept in our county.)
- Available labor pool

Google Data Center - Council Bluffs, Iowa, USA



Google data center pictures: Council Bluffs