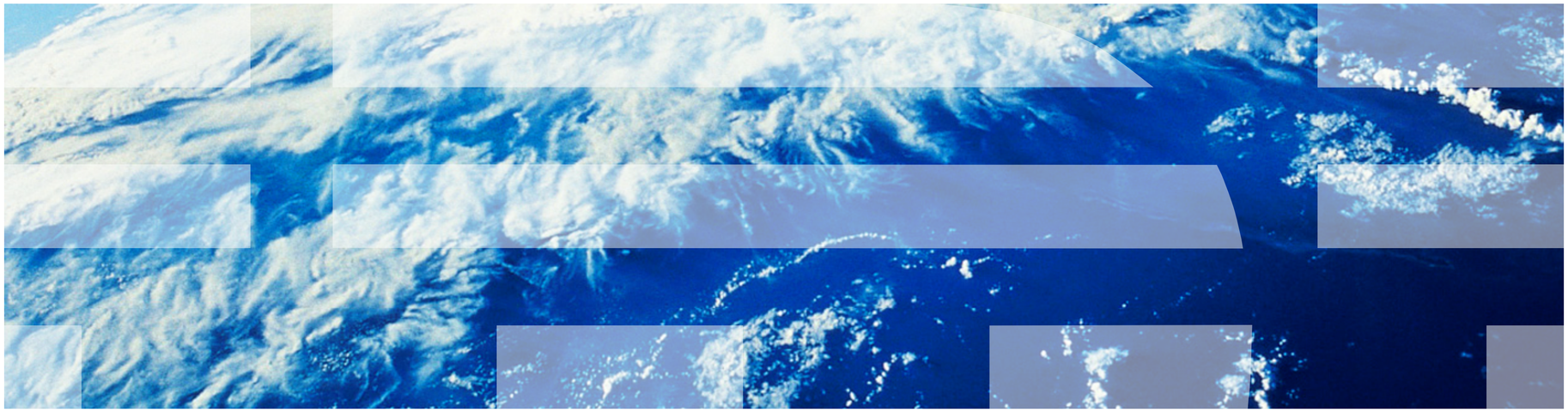
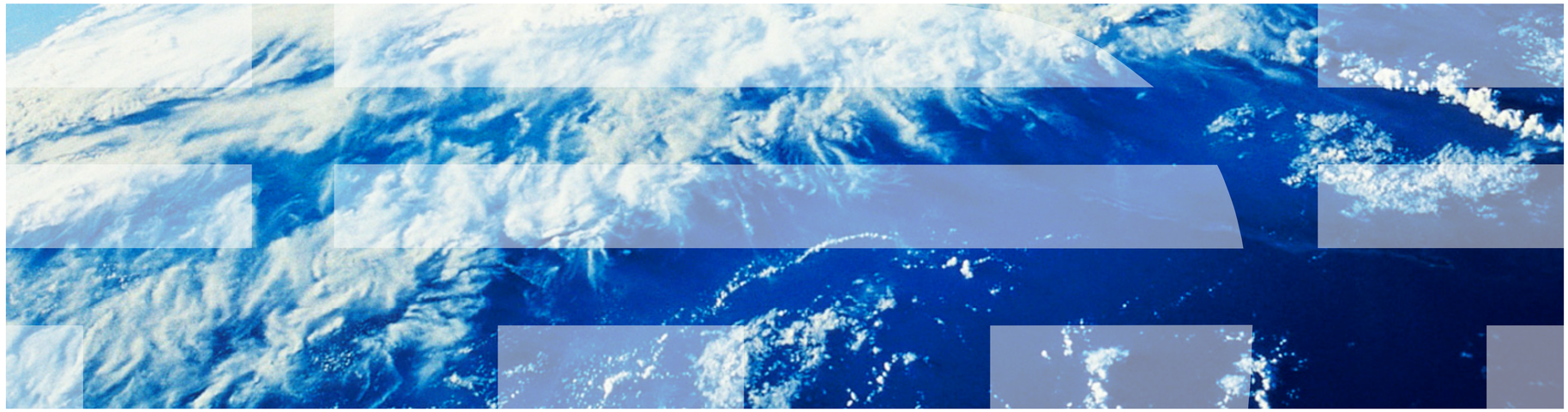
Lecture 1



Computer Systems for Data Science
Topic 1

Course Introduction

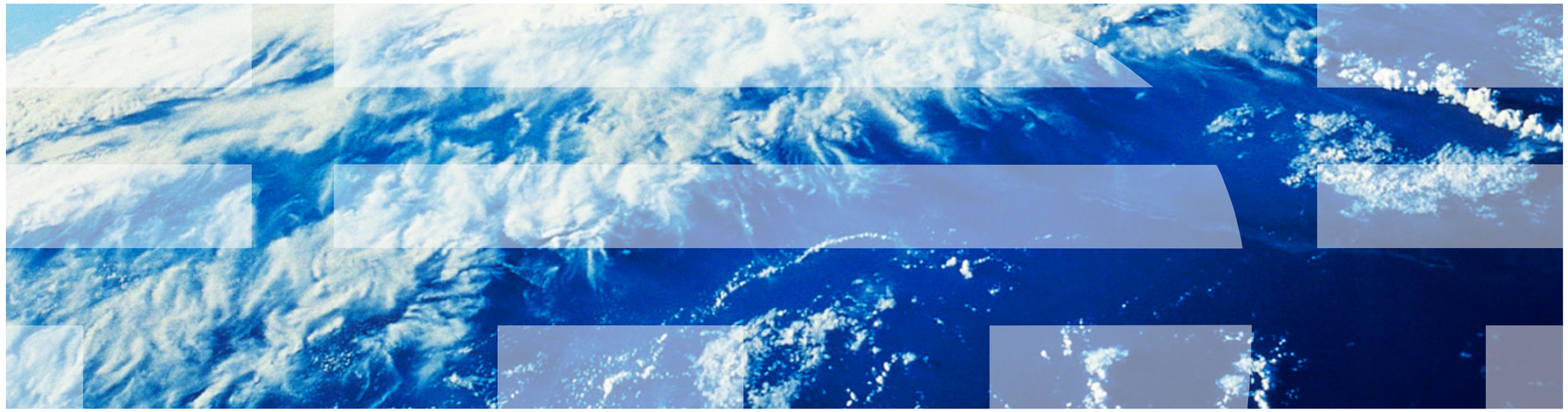
Systems concepts



Topic 1: 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

Course Instructors and TAs

- Instructor: Waqar Aqeel

Course Instructors and TAs

- Instructor: Waqar Aqeel
- Head TAs: Krishen and Anouksha

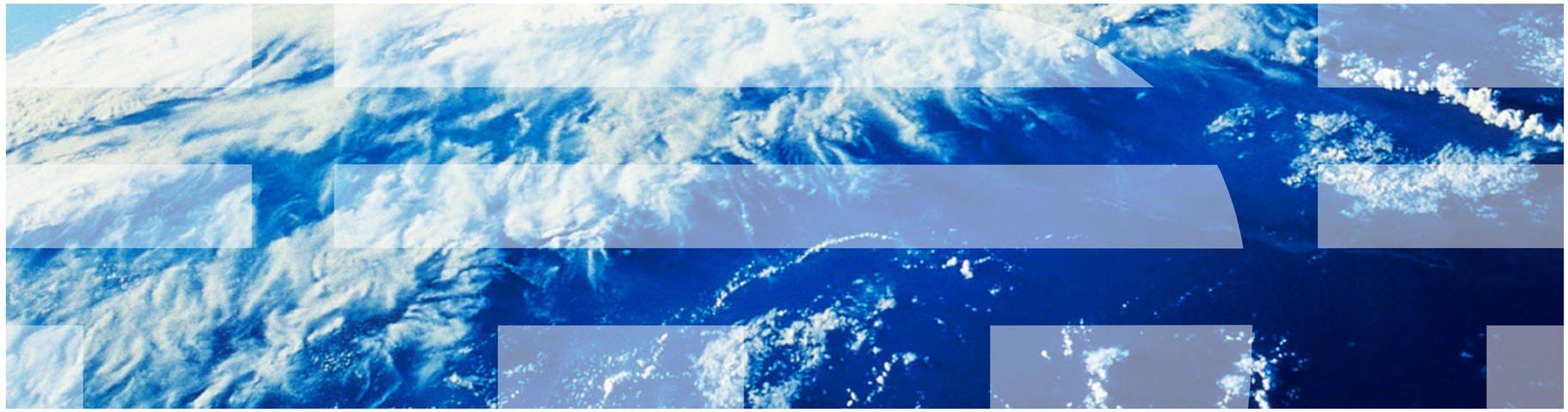
Course Instructors and TAs

- Instructor: Waqar Aqeel
- Head TAs: Krishen and Anouksha
- TAs: Anisha, Vaishnavi, Sushmita, Arya

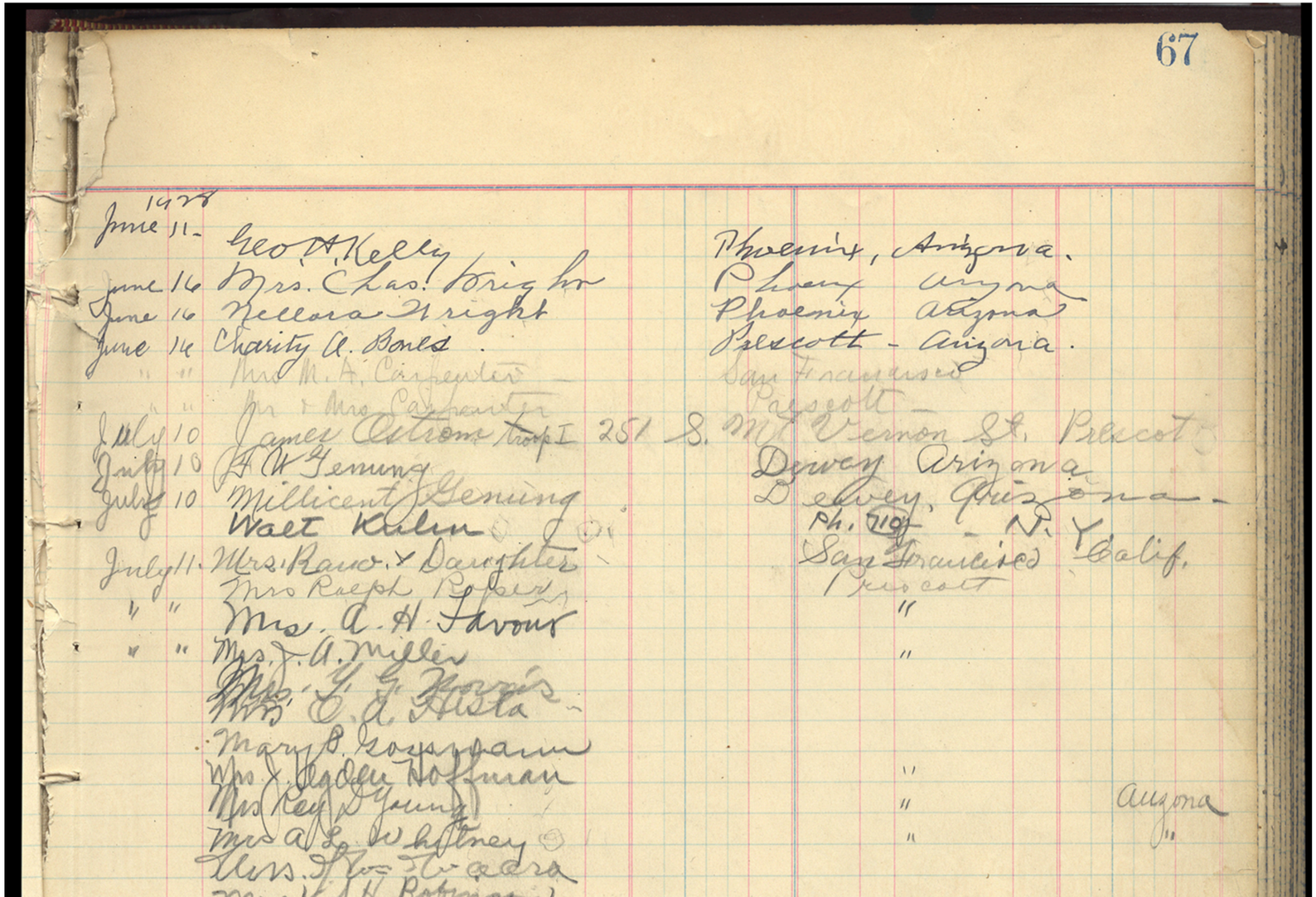
Course Instructors and TAs

- Instructor: Waqar Aqeel
- Head TAs: Krishen and Anouksha
- TAs: Anisha, Vaishnavi, Sushmita, Arya
- All CAs have experience in databases and systems

What is Data Science and Big Data?



This was a system for big data

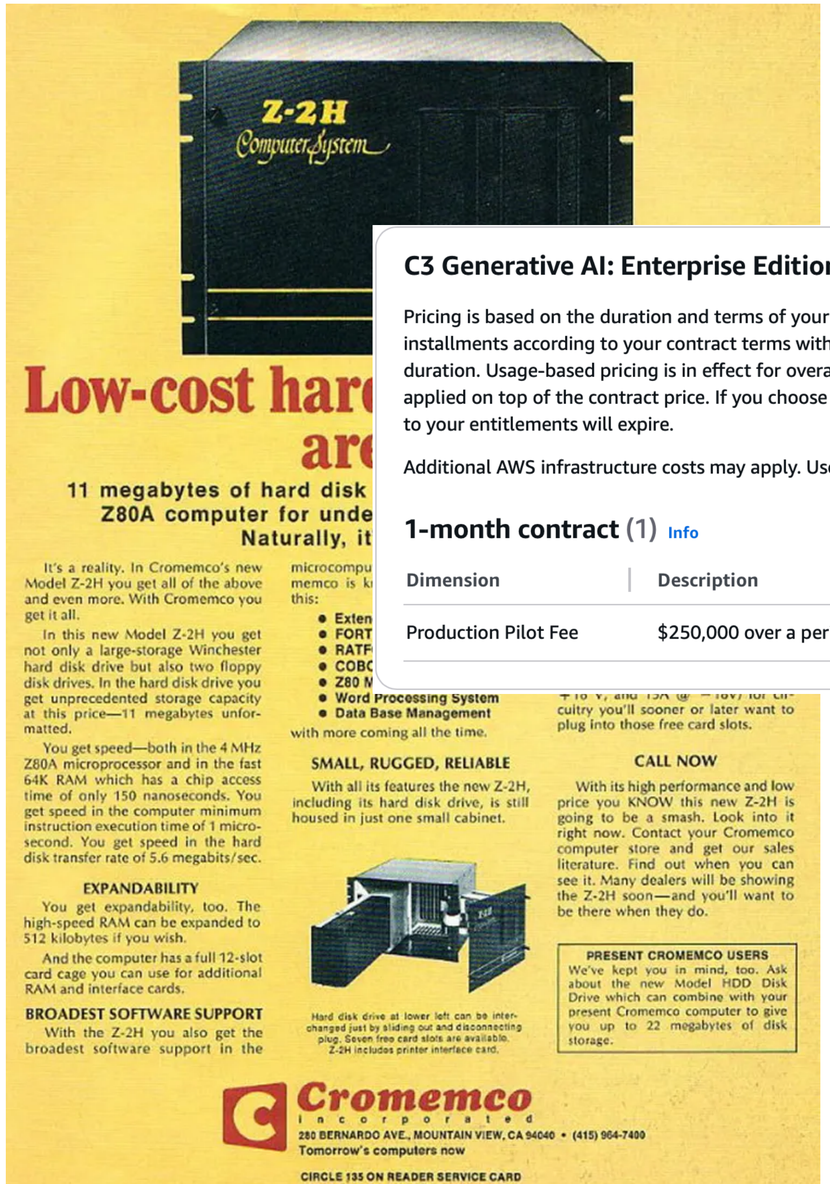


67

1928
June 11 Geo. A. Kelly — Phoenix, Arizona.
June 16 Mrs. Chas. Wright — Phoenix Arizona
June 16 Kellora Wright — Phoenix Arizona
June 16 Charity A. Bones — Prescott - Arizona.
" " Mrs M. A. Carpenter — San Francisco
" " Mr + Mrs Carpenter — Prescott
July 10 James Astrom Troop I — 251 S. Mt Vernon St. Prescot
July 10 A. H. Gemung — Dewey Arizona
July 10 Millicent L. Gemung — Dewey, Arizona
Walt Kahn — Ph. 7107 - N. Y.
July 11 Mrs. Rano + Daughter — San Francisco Calif.
Mrs Ralph Rogers — Prescott
" " Mrs. A. H. Favour — "
" " Mrs. J. A. Miller — "
Mrs. L. L. Harris
Mrs. E. A. Hasta
Mary B. Rosspaum
Mrs. Hayden Hoffman
Mrs Ray Young
Mrs A. L. Whitney — Arizona
Miss. Rose Rivera — "

Data science systems were expensive

Data science systems were expensive



They are still expensive!

C3 Generative AI: Enterprise Edition Info

View purchase options

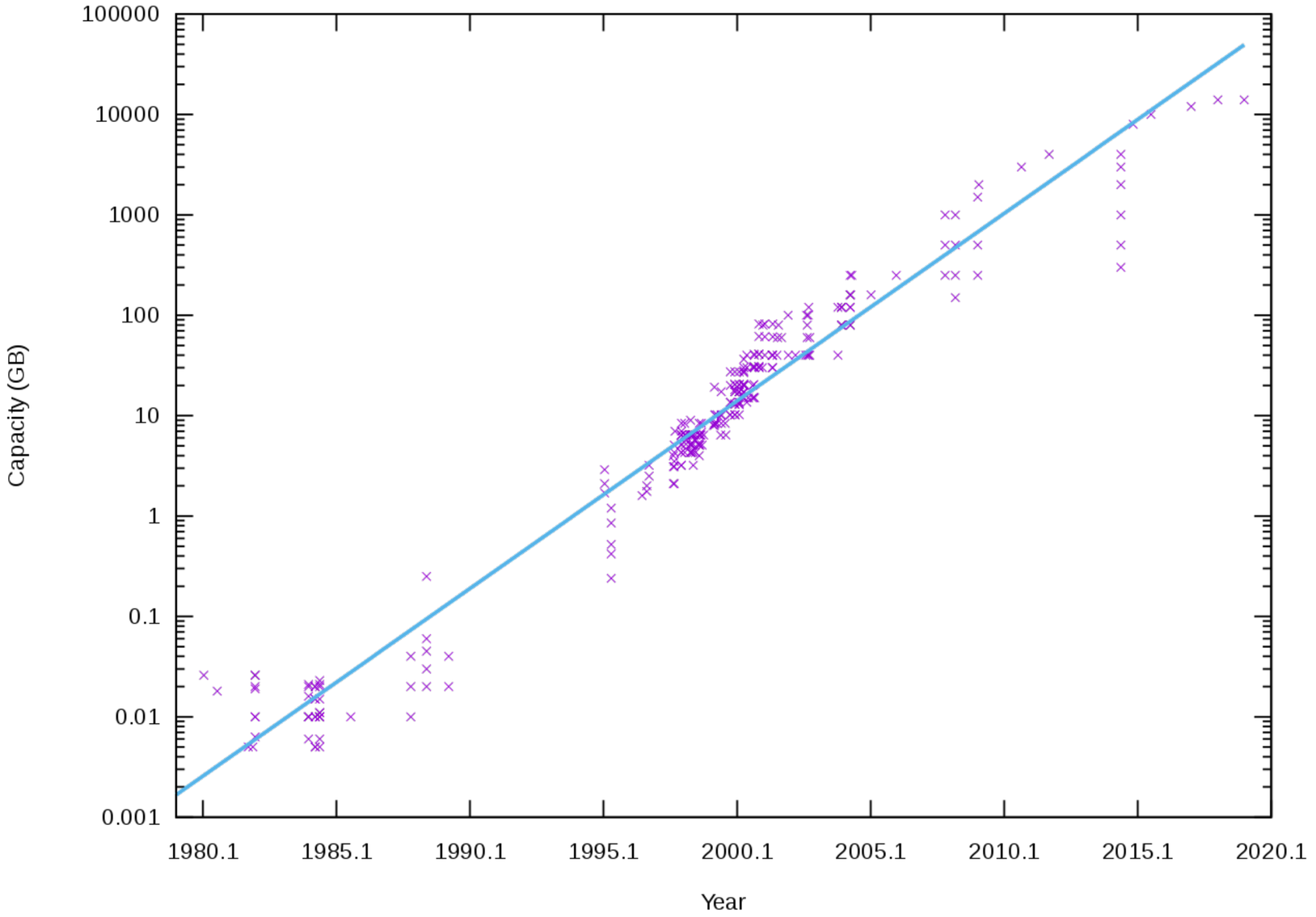
Pricing is based on the duration and terms of your contract with the vendor, and additional usage. You pay upfront or in installments according to your contract terms with the vendor. This entitles you to a specified quantity of use for the contract duration. Usage-based pricing is in effect for overages or additional usage not covered in the contract. These charges are applied on top of the contract price. If you choose not to renew or replace your contract before the contract end date, access to your entitlements will expire.

Additional AWS infrastructure costs may apply. Use the AWS Pricing Calculator to estimate your infrastructure costs.

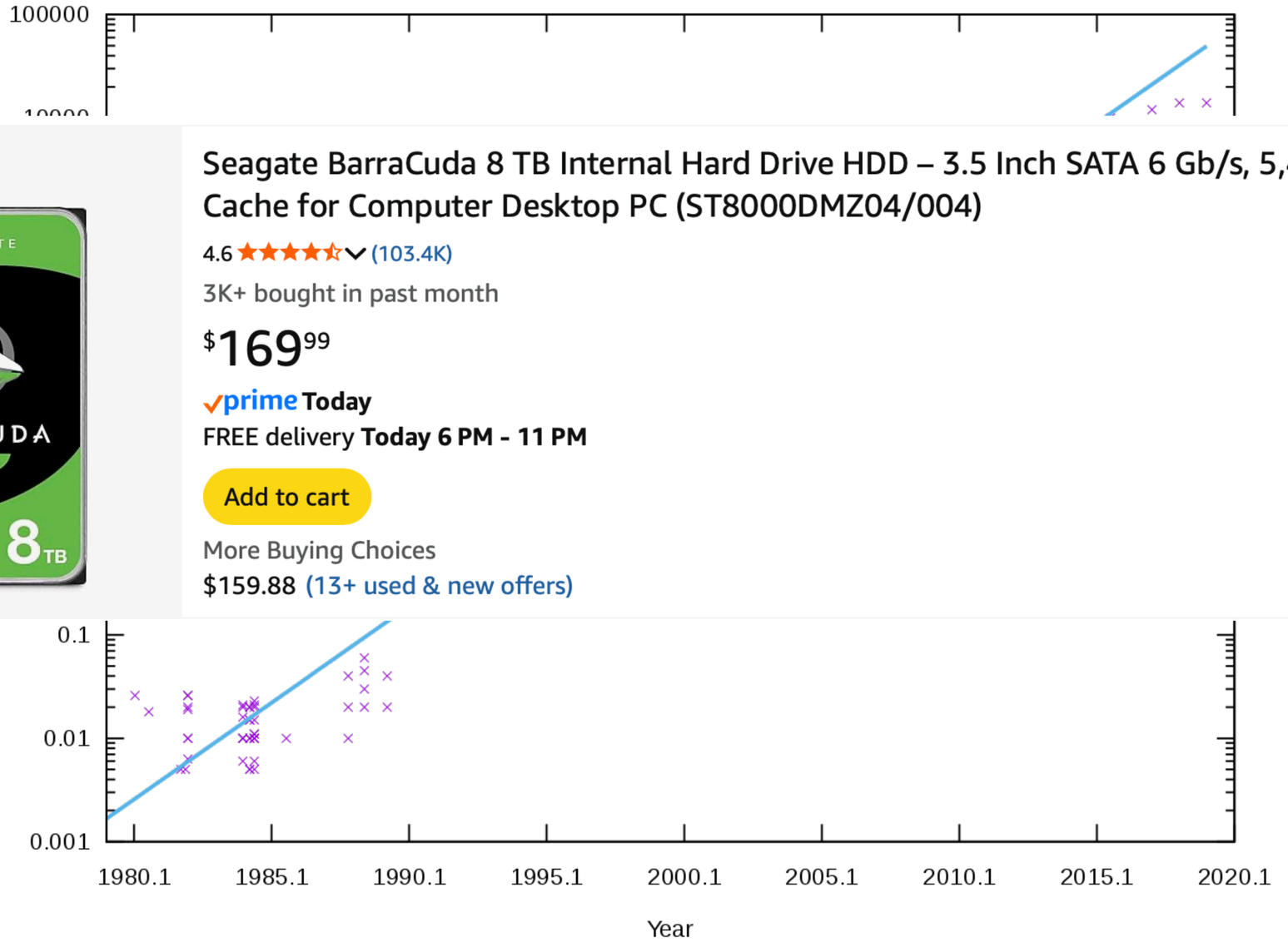
1-month contract (1) Info

Dimension	Description	Cost/month	Overage cost
Production Pilot Fee	$250,000 over a period of 3 months - required with all options	$250,000.00	$0.55/unit

Today: data is cheap



Today: data is cheap



Overall Pick

Seagate BarraCuda 8 TB Internal Hard Drive HDD – 3.5 Inch SATA 6 Gb/s, 5,400 RPM, 256 MB Cache for Computer Desktop PC (ST8000DMZ04/004)

4.6 ★★★★½ (103.4K)

3K+ bought in past month

$169.99

prime Today

FREE delivery **Today 6 PM - 11 PM**

Add to cart

More Buying Choices

$159.88 (13+ used & new offers)

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*

Where is data coming from?

- Physical devices
- Software logs
- Phones
- GPS/Cars
- Internet of *Things*
- Social media, website contents

What can we do with all this data?

- What video should I recommend to this user to view next?

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?

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?

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?

What can we do with all this data?

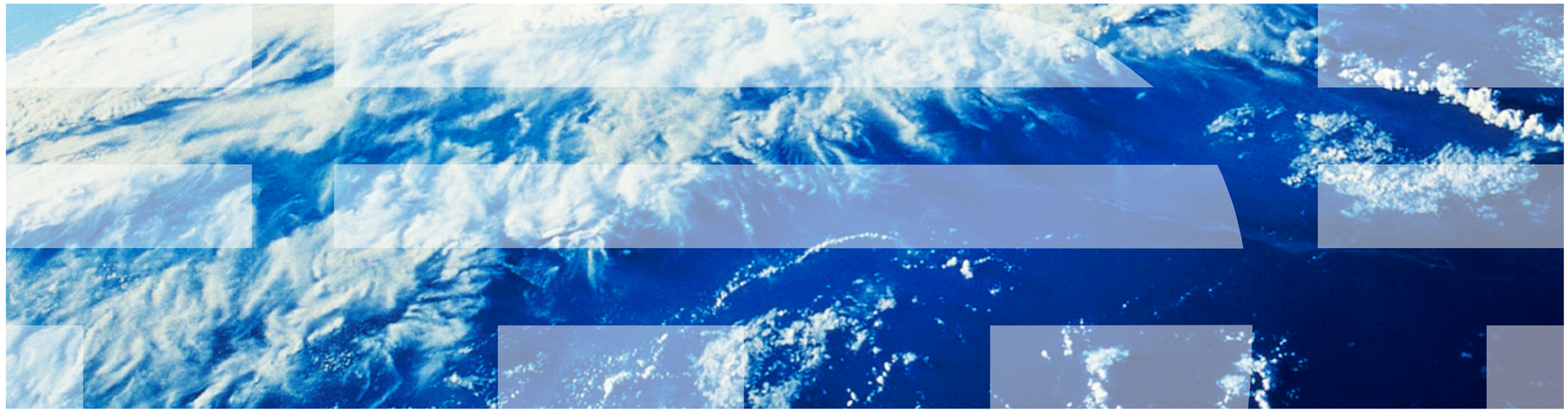
- What video should I recommend to this user to view next?
- Does this MRI image of a breast contain a tumor?
- Who is going to win the election?
- Which cities in the US will have high incidence of flu in 2 weeks?
- Is the object across from the car a pedestrian?

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?

What is this class about?



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

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

- Questions we **will** answer in this class:

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

- Questions we **will** answer in this class:

How are big data systems designed?

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?

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?

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?

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 do ML/AI systems work?

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 do ML/AI systems work?

How to ensure privacy/ security/quality?

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 do ML/AI systems work?

How to ensure privacy/ security/quality?

- Questions we **won't** answer in this class:

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 do ML/AI systems work?
  - How to ensure privacy/ security/quality?
- Questions we **won't** answer in this class:
  - What algorithm should we use?

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 do ML/AI systems work?
  - How to ensure privacy/ security/quality?
- Questions we **won't** answer in this class:
  - What algorithm should we use?
  - How to train my own ML models

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 do ML/AI systems work?
  - How to ensure privacy/ security/quality?
- Questions we **won't** answer in this class:
  - What algorithm should we use?
  - How to train my own ML models
  - How do we explain/ debug ML models?

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 do ML/AI systems work?
  - How to ensure privacy/ security/quality?
- Questions we **won't** answer in this class:
  - What algorithm should we use?
  - How to train my own ML models
  - How do we explain/ debug ML models?
  - How can data be visualized?

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 do ML/AI systems work?
  - How to ensure privacy/ security/quality?
- Questions we **won't** answer in this class:
  - What algorithm should we use?
  - How to train my own ML models
  - How do we explain/ debug ML models?
  - How can data be visualized?
  - What are the statistical/ mathematical foundations for data science?

Course Objectives

Course Objectives

- **Graduate-level course**

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)
  - **Systems for machine learning** (accelerators, distributed training/inference infrastructure)
  - **Basic security** for data scientists (encryption, privacy)

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)
  - **Systems for machine learning** (accelerators, distributed training/inference infrastructure)
  - **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 (BigQuery, RocksDB,…)

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)
  - **Systems for machine learning** (accelerators, distributed training/inference infrastructure)
  - **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 (BigQuery, RocksDB,…)
- 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, ML

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)
  - **Systems for machine learning** (accelerators, distributed training/inference infrastructure)
  - **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 (BigQuery, RocksDB,…)
- 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, ML
- **You come from diverse backgrounds:** Some of the content will be repetitive for students who have taken the classes above, like databases, systems, networks etc.

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)
  - **Systems for machine learning** (accelerators, distributed training/inference infrastructure)
  - **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 (BigQuery, RocksDB,…)
- 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, ML
- **You come from diverse backgrounds:** Some of the content will be repetitive for students who have taken the classes above, like databases, systems, networks etc.
- **Required background**
  - Programming experience with Python
  - Both programming assignments will be submitted in Python

Course Administration and Grading

Course Administration and Grading

- **All materials, assignments, etc. posted on course website**
  - Show locally, link to come

Course Administration and Grading

- **All materials, assignments, etc. posted on course website**
  - Show locally, link to come
- **Only one section**

Course Administration and Grading

- **All materials, assignments, etc. posted on course website**
  - Show locally, link to come
- **Only one section**
- **Announcement/Q&A will be posted on Ed**

Course Administration and Grading

- **All materials, assignments, etc. posted on course website**
  - Show locally, link to come
- **Only one section**
- **Announcement/Q&A will be posted on Ed**
- **Lecture Materials**
  - Lecture slides
  - No textbook

Course Administration and Grading

- **All materials, assignments, etc. posted on course website**
  - Show locally, link to come
- **Only one section**
- **Announcement/Q&A will be posted on Ed**
- **Lecture Materials**
  - Lecture slides
  - No textbook
- **Homework, assignments, exams**
  - Programming assignment 2: (5%)
  - Written assignment 2: systems and databases (5%)
  - Programming assignment 3: Indexing and filtering (10%)
  - Written assignment 4: distributed systems, ML, security (5%)
  - In-person midterm (25%)
  - In-person final exam (50%)

Course Administration and Grading

- **All materials, assignments, etc. posted on course website**
  - Show locally, link to come
- **Only one section**
- **Announcement/Q&A will be posted on Ed**
- **Lecture Materials**
  - Lecture slides
  - No textbook
- **Homework, assignments, exams**
  - Programming assignment 2: (5%)
  - Written assignment 2: systems and databases (5%)
  - Programming assignment 3: Indexing and filtering (10%)
  - Written assignment 4: distributed systems, ML, security (5%)
  - In-person midterm (25%)
  - In-person final exam (50%)
- **All assignments will be turned in online**

Course Administration and Grading

- **All materials, assignments, etc. posted on course website**
  - Show locally, link to come
- **Only one section**
- **Announcement/Q&A will be posted on Ed**
- **Lecture Materials**
  - Lecture slides
  - No textbook
- **Homework, assignments, exams**
  - Programming assignment 2: (5%)
  - Written assignment 2: systems and databases (5%)
  - Programming assignment 3: Indexing and filtering (10%)
  - Written assignment 4: distributed systems, ML, security (5%)
  - In-person midterm (25%)
  - In-person final exam (50%)
- **All assignments will be turned in online**
- **All classes streamed online (Zoom) and recorded (available on CourseWorks)**
  - No attendance required

Programming Assignments

Programming Assignments

- 2 programming assignments
  - Both done individually

Programming Assignments

- 2 programming assignments
  - Both done individually
- Programming assignments are in Python
  - Brush up on your Python if you are rusty: many resources online
    - Most commonly-used language for data scientists

Programming Assignments

- 2 programming assignments
  - Both done individually
- Programming assignments are in Python
  - Brush up on your Python if you are rusty: many resources online
    - Most commonly-used language for data scientists
- Programming assignment 1 done 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: Arya Shidore
    - But be careful not to spend too many!

Programming Assignments

- 2 programming assignments
  - Both done individually
- Programming assignments are in Python
  - Brush up on your Python if you are rusty: many resources online
    - Most commonly-used language for data scientists
- Programming assignment 1 done 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: Arya Shidore
    - But be careful not to spend too many!
- Programming assignment goals
  - Assignment 1: BigQuery
    - Learning to use SQL on a big data set
  - Assignment 3: Indexing and filtering data structures
    - Understanding how real-world data systems data structures work, strengthen Python skills

More logistics

- **Office hours:**
  - CAs will hold office hours every weekday over Google Meet
  - Course calendar will have the meeting link: all office hours will use the same link
- **Ed**
  - A CA is guaranteed to be available on Ed every weekday (when the school is open) from 9AM – 5PM. We will try to answer your questions as fast as possible
- **Submit your assignments on time!**
  - Homework submission will be on Gradescope
  - **If you do not submit your HW on time, your grade will be 0%**
  - We will give you **plenty of time** for the assignments, don't wait until the last minute!
  - You can resubmit homework as many times as you want, until the deadline

Tentative Contents and Syllabus

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

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
  - Core, Edge/Sateliite, PoP
  - The rise of AI data centers

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
  - Core, Edge/Sateliite, PoP
  - The rise of AI data centers
- Relational model and SQL
  - Relational model and SQL
  - SELECT, FROM, WHERE
  - GROUPBY
  - JOINs
  - Nested queries
  - Transactions
  - ACID
  - OLAP vs. OLTP, SQL vs. NoSQL
  - Logging

Tentative Contents and Syllabus

- Storage systems
  - The memory hierarchy
  - Storage technologies primer
  - Distributed file systems
  - Indexing
  - Filters
  - Caching
  - Storage engines
  - In-memory key-value stores

Tentative Contents and Syllabus

- Storage systems
  - The memory hierarchy
  - Storage technologies primer
  - Distributed file systems
  - Indexing
  - Filters
  - Caching
  - Storage engines
  - In-memory key-value stores
- Distributed online databases (OLTP)
  - 2 Phase Commit
  - Locking
  - Sharding
  - Fault tolerance
  - Replication and consensus

Tentative Contents and Syllabus

- Storage systems
  - The memory hierarchy
  - Storage technologies primer
  - Distributed file systems
  - Indexing
  - Filters
  - Caching
  - Storage engines
  - In-memory key-value stores
- Distributed online databases (OLTP)
  - 2 Phase Commit
  - Locking
  - Sharding
  - Fault tolerance
  - Replication and consensus
- Analytics (OLAP)
  - Mapreduce computing model
  - Stragglers
  - Lineage
  - Fault tolerance in distributed analytics: lineage
  - Streaming computing model

Tentative Contents and Syllabus

Tentative Contents and Syllabus

- Global serving infra
  - Layered load balancing
  - Multithreading
  - GPUs

Tentative Contents and Syllabus

- Global serving infra
  - Layered load balancing
  - Multithreading
  - GPUs
- Single-node ML
  - GPUs and ML accelerators
  - Kernels, ML compilation
  - ML single node bottlenecks
  - ML memory

Tentative Contents and Syllabus

- Global serving infra
  - Layered load balancing
  - Multithreading
  - GPUs
- Single-node ML
  - GPUs and ML accelerators
  - Kernels, ML compilation
  - ML single node bottlenecks
  - ML memory
- Distributed ML
  - ML network
  - Distributed training
  - Checkpointing
  - Inference systems challenges

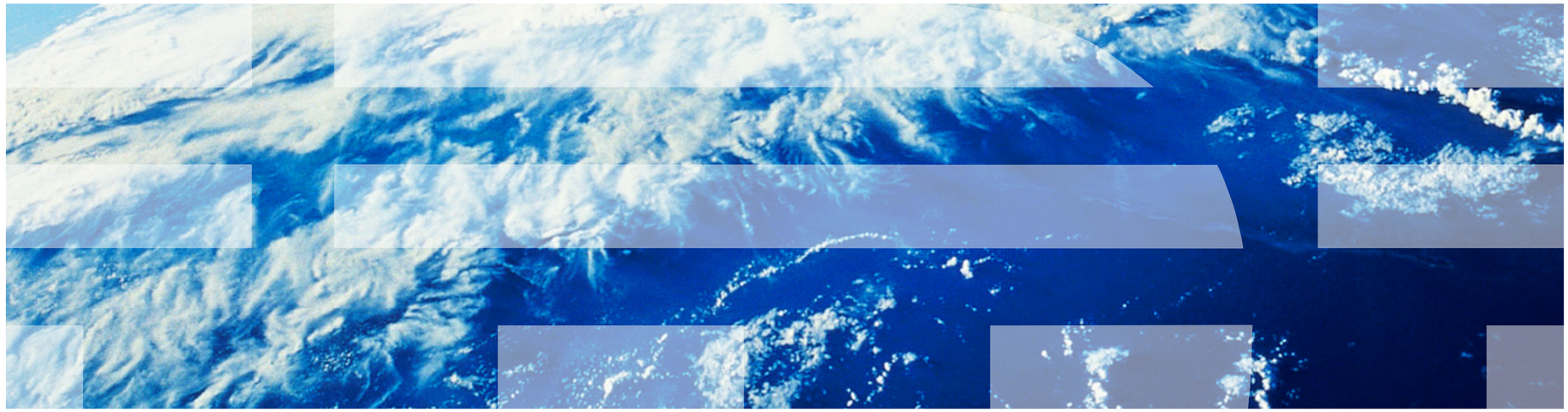
Tentative Contents and Syllabus

- Global serving infra
  - Layered load balancing
  - Multithreading
  - GPUs
- Single-node ML
  - GPUs and ML accelerators
  - Kernels, ML compilation
  - ML single node bottlenecks
  - ML memory
- Distributed ML
  - ML network
  - Distributed training
  - Checkpointing
  - Inference systems challenges
- Security and privacy
  - Security of big data systems
  - Privacy consideration
  - Data compliance and access control

Tentative Contents and Syllabus

- Global serving infra
  - Layered load balancing
  - Multithreading
  - GPUs
- Single-node ML
  - GPUs and ML accelerators
  - Kernels, ML compilation
  - ML single node bottlenecks
  - ML memory
- Distributed ML
  - ML network
  - Distributed training
  - Checkpointing
  - Inference systems challenges
- Security and privacy
  - Security of big data systems
  - Privacy consideration
  - Data compliance and access control
- Observability
  - Data monitoring
  - Production metrics as a big data system
  - Data quality

Performance Concepts and Rules of Thumb



Performance Evaluation

Performance Evaluation

- Metric: something we measure

Performance Evaluation

- Metric: something we measure
- Goal: evaluate how good/bad our computer system is performing

Performance Evaluation

- Metric: something we measure
- Goal: evaluate how good/bad our computer system is performing
- Examples:
  - Power consumed by our database
  - CPU cost of running a web backend
  - Average time it takes to render a user page
  - How many users can we support at the same time

Performance Evaluation

- Metric: something we measure
- Goal: evaluate how good/bad our computer system is performing
- Examples:
  - Power consumed by our database
  - CPU cost of running a web backend
  - 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

Performance Evaluation

- Metric: something we measure
- Goal: evaluate how good/bad our computer system is performing
- Examples:
  - Power consumed by our database
  - CPU cost of running a web backend
  - 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
- They are crucial for proving improvements, diagnosing regressions

Tradeoff: latency vs. throughput

Tradeoff: latency vs. throughput

- Pizza delivery example
  - Do you want your pizza to be fresh?
  - Do you want your pizza to be cheap?

Tradeoff: latency vs. throughput

- Pizza delivery example
  - Do you want your pizza to be fresh?
  - Do you want your pizza to be cheap?
- Why do these conflict?

Tradeoff: latency vs. throughput

- Pizza delivery example
  - Do you want your pizza to be fresh?
  - 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)

Tradeoff: latency vs. throughput

- Pizza delivery example
  - Do you want your pizza to be fresh?
  - 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)

Tradeoff: latency vs. throughput

- Pizza delivery example
  - Do you want your pizza to be fresh?
  - 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 (length of the pipe)

Tradeoff: latency vs. throughput

- Pizza delivery example
  - Do you want your pizza to be fresh?
  - 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 (length of the pipe)
- Throughput = number of tasks per unit time (width of the pipe)

Tradeoff: latency vs. throughput

- Pizza delivery example
  - Do you want your pizza to be fresh?
  - 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 (length of the pipe)
- Throughput = number of tasks per unit time (width of the pipe)

Tradeoff: latency vs. throughput

- Pizza delivery example
  - Do you want your pizza to be fresh?
  - 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 (length of the pipe)
- Throughput = number of tasks per unit time (width of the pipe)

- 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**?

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
- Depends on what YOU want

Definitions

Definitions

- Performance is in units of things-per-second
  - Bigger is better

Definitions

- Performance is in units of things-per-second
  - Bigger is better
- Response time of a system Y running Z
  - $\text{performance}(Y) = \dfrac{1}{execution\ time\ (Z\ on\ Y)}$

Definitions

- Performance is in units of things-per-second
  - Bigger is better
- Response time of a system Y running Z
  - $\text{performance}(Y) = \dfrac{1}{execution\ time\ (Z\ on\ Y)}$
- Throughput of system Y running many requests
  - $\text{performance}(Y) = \dfrac{number\ of\ requests}{unit\ time}$

Definitions

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

How do we improve performance?

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! (a pretty lousy answer)

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

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)
  - Who uses it? (business)

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)
  - Who uses it? (business)
- Speedup due to enhancement E:
  - $$Speedup(E) = \frac{Execution\ time\ without\ E}{Execution\ time\ with\ E} = \frac{Performance\ with\ E}{Performance\ without\ E}$$
  - $$ExTime_{new} = ExTime_{old} * \left[\left(1 - p\right) + \frac{p}{S} \right]$$
    - Explanation:
    - $\left(1 - p\right)$ 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
  - $$Speedup(E) = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{\left(1 - p\right) + \frac{p}{S}}$$

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)
  - Who uses it? (business)
- Speedup due to enhancement E:
  - $$Speedup(E) = \frac{Execution\ time\ without\ E}{Execution\ time\ with\ E} = \frac{Performance\ with\ E}{Performance\ without\ E}$$
  - $$ExTime_{new} = ExTime_{old} * \left[\left(1 - p\right) + \frac{p}{S} \right]$$
    - Explanation:
    - $\left(1 - p\right)$ 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
  - $$Speedup(E) = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{\left(1 - p\right) + \frac{p}{S}}$$
  - It's usually less formal. Most metrics either obviously matter or obviously don't.

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)
  - Who uses it? (business)
- Speedup due to enhancement E:
  - $Speedup(E) = \frac{Execution\ time\ without\ E}{Execution\ time\ with\ E} = \frac{Performance\ with\ E}{Performance\ without\ E}$
  - $ExTime_{new} = ExTime_{old} * \left[\left(1 - p\right) + \frac{p}{S} \right]$
    - Explanation:
    - $\left(1 - p\right)$ 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
  - $Speedup(E) = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{\left(1 - p\right) + \frac{p}{S}}$
- It's usually less formal. Most metrics either obviously matter or obviously don't.
- Customers are either complaining, or they are not

Amdahl’s law: example

Amdahl's law: example

- We built a new database that speeds up aggregate queries by 2x! Hurray!

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

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
- $$ExTime_{new} = ExTime_{old} * \left[(1 - p) + \frac{p}{S} \right]$$

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
- $ExTime_{new} = ExTime_{old} * \left[(1 - p) + \frac{p}{S} \right]$
- $ExTime_{new} = ExTime_{old} * \left[0.9 + \frac{0.1}{2} \right] = 0.95 * ExTime_{old}$

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
- $ExTime_{new} = ExTime_{old} * \left[(1 - p) + \frac{p}{S} \right]$
- $ExTime_{new} = ExTime_{old} * \left[0.9 + \frac{0.1}{2} \right] = 0.95 * ExTime_{old}$
- $Speedup_{total} = \frac{1}{0.95} = 1.053$ → only 5.3% overall speedup ☹

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
- $ExTime_{new} = ExTime_{old} * \left[(1 - p) + \frac{p}{S} \right]$
- $ExTime_{new} = ExTime_{old} * \left[0.9 + \frac{0.1}{2} \right] = 0.95 * ExTime_{old}$
- $Speedup_{total} = \frac{1}{0.95} = 1.053$ → only 5.3% overall speedup ☹
- Amdahl's law: speedup bounded by

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
- $ExTime_{new} = ExTime_{old} * \left[(1 - p) + \frac{p}{S} \right]$
- $ExTime_{new} = ExTime_{old} * \left[0.9 + \frac{0.1}{2} \right] = 0.95 * ExTime_{old}$
- $Speedup_{total} = \frac{1}{0.95} = 1.053$ → only 5.3% overall speedup ☹

- Amdahl's law: speedup bounded by

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

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
- $ExTime_{new} = ExTime_{old} * \left[(1 - p) + \frac{p}{S} \right]$
- $ExTime_{new} = ExTi$ [illegible] $0.1]$
- $Speedup_{total} = \frac{1}{0.95}$ [illegible]

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

- Amdahl's law: speedup bounded by

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

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
- $ExTime_{new} = ExTime_{old} * \left[(1 - p) + \frac{p}{S} \right]$
- $ExTime_{new} = ExTi$ [illegible] $0.1]$
- $Speedup_{total} = \frac{1}{0.95}$ [illegible]

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

- 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:

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
- $ExTime_{new} = ExTime_{old} * \left[(1 - p) + \frac{p}{S} \right]$
- $ExTime_{new} = ExTi$ [illegible] $0.1]$
- $Speedup_{total} = \frac{1}{0.95}$ [illegible]

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

- 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:
- $Speedup_{optimal} = \frac{1}{0.9} = 1.111$

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
- $ExTime_{new} = ExTime_{old} * \left[(1 - p) + \frac{p}{S} \right]$
- $ExTime_{new} = ExTi$ [illegible] $0.1]$
- $Speedup_{total} = \frac{1}{0.95}$ [illegible]

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

- 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:
- $Speedup_{optimal} = \frac{1}{0.9} = 1.111$
- More useful for parallel programming bottlenecks, but can be adapted here.

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

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

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

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

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: 20,000ns, 20us

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: 20,000ns, 20us
- Read a small object within the same network in a data center: 100,000ns, 100us

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: 20,000ns, 20us
- 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

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: 20,000ns, 20us
- 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

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

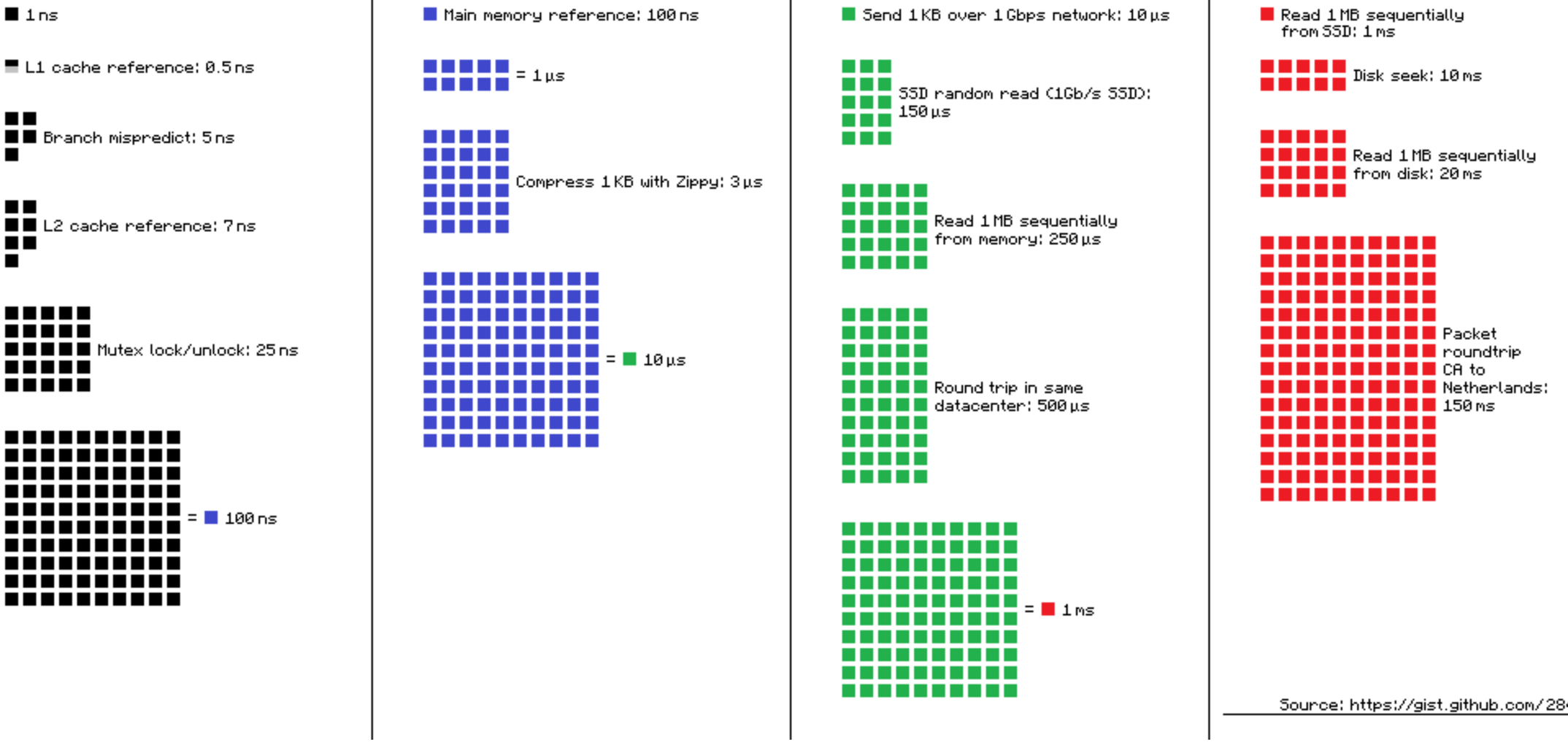
- CPU cache access: 1ns
- Memory access: 100ns
- Read a small object from a random location on a local flash drive: 20,000ns, 20us
- Read a small object within the same network in a data center: 100,000ns, 100us
- Run a SQL query on a flash database: 1,000,000ns, 1ms
- Read a small random object from magnetic disk: 10,000,000ns, 10ms
- Run a SQL query on a disk database: 20,000,000ns, 20ms

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: 20,000ns, 20us
- 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: 30,000,000us, 30ms
  - Bounded by the speed of light! Roundtrip light speed from NYC to Beijing is ~150ms

Latency Numbers Every Programmer Should Know



Source: https://gist.github.com/2841832

How can we use these numbers? A database example

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

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:

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 +

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 +

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)

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

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

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

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

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
- → Since 72% requests go to the database and it's so slow, its latency dominates the total latency

Disk vs. Flash, Cost vs. Performance

Disk vs. Flash, Cost vs. Performance

- Your app needs a cloud database that runs SQL queries

Disk vs. Flash, Cost vs. Performance

- Your app needs a cloud database that runs SQL queries
- You are considering running the database on two types of storage devices: flash vs. magnetic disk
  - You received some quotes from database company, and flash database is 2X more expensive, but 10X faster

Disk vs. Flash, Cost vs. Performance

- Your app needs a cloud database that runs SQL queries
- You are considering running the database on two types of storage devices: flash vs. magnetic disk
  - You received some quotes from database company, and flash database is 2X more expensive, but 10X faster
- Your users don’t notice page loading times, as long as they are under 300,000,000ns (300ms)

Disk vs. Flash, Cost vs. Performance

- Your app needs a cloud database that runs SQL queries
- You are considering running the database on two types of storage devices: flash vs. magnetic disk
  - You received some quotes from database company, and flash database is 2X more expensive, but 10X faster
- Your users don't notice page loading times, as long as they are under 300,000,000ns (300ms)
- You measured: Internet roundtrip (100ms), disk DB access (10ms), flash DB access (1ms)

Disk vs. Flash, Cost vs. Performance

- Your app needs a cloud database that runs SQL queries
- You are considering running the database on two types of storage devices: flash vs. magnetic disk
  - You received some quotes from database company, and flash database is 2X more expensive, but 10X faster
- Your users don't notice page loading times, as long as they are under 300,000,000ns (300ms)
- You measured: Internet roundtrip (100ms), disk DB access (10ms), flash DB access (1ms)
- Scenario 1: Your user queries involve only a single database access in the cloud (over the Internet)
  - Latency with flash database: 101ms
  - **Latency with disk database: 110ms**

Disk vs. Flash, Cost vs. Performance

- Your app needs a cloud database that runs SQL queries
- You are considering running the database on two types of storage devices: flash vs. magnetic disk
  - You received some quotes from database company, and flash database is 2X more expensive, but 10X faster
- Your users don't notice page loading times, as long as they are under 300,000,000ns (300ms)
- You measured: Internet roundtrip (100ms), disk DB access (10ms), flash DB access (1ms)
- Scenario 1: Your user queries involve only 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 a single user query, and then it can return a response
  - **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

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 once ~ 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

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 once ~ 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)

Identifying performance bottlenecks

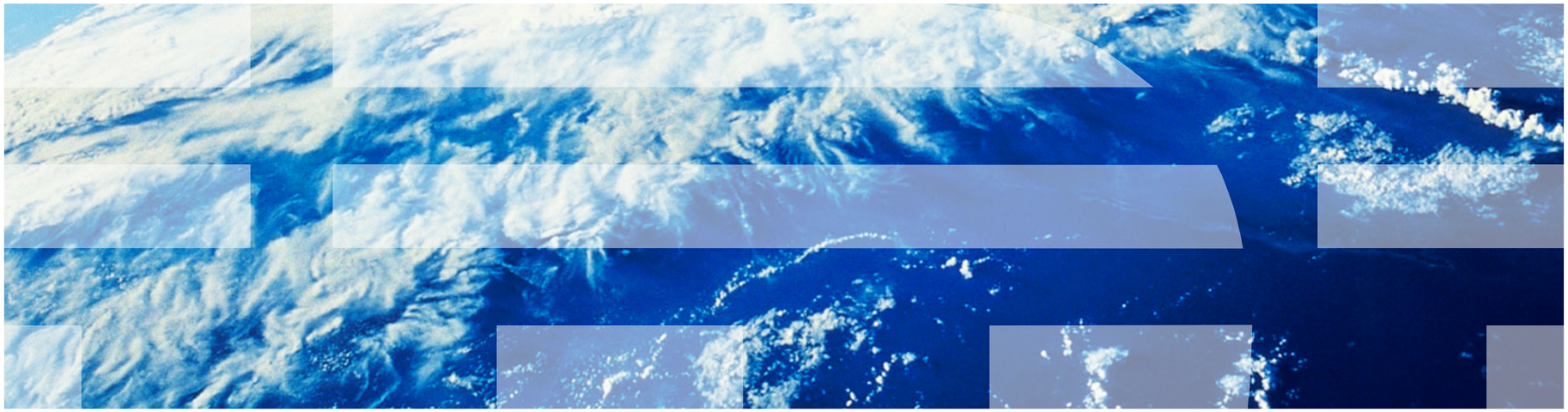
- My application is seeing an average latency of 200ms, where is the bottleneck?
- A few guiding questions:
  1. What systems does the web page need to access? Which networks does it need to traverse?
  2. Start from the most common case + highest latency
- Example:
  - Application needs to go through the Internet once ~ 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)

Summary

- Latency and throughput: two important metrics, sometimes correlate, but often do not
- Amdahl's law: optimize the common case
- Computer systems almost always involve a performance vs. cost trade off

Adapted from Mendel Rosenblum and Jeff Dean

The Infrastructure of Big Data



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

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

- # docs: tens of millions to tens of billions ~1000X

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

- # docs: tens of millions to tens of billions ~1000X
- Queries processed/day: ~1000X

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

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

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

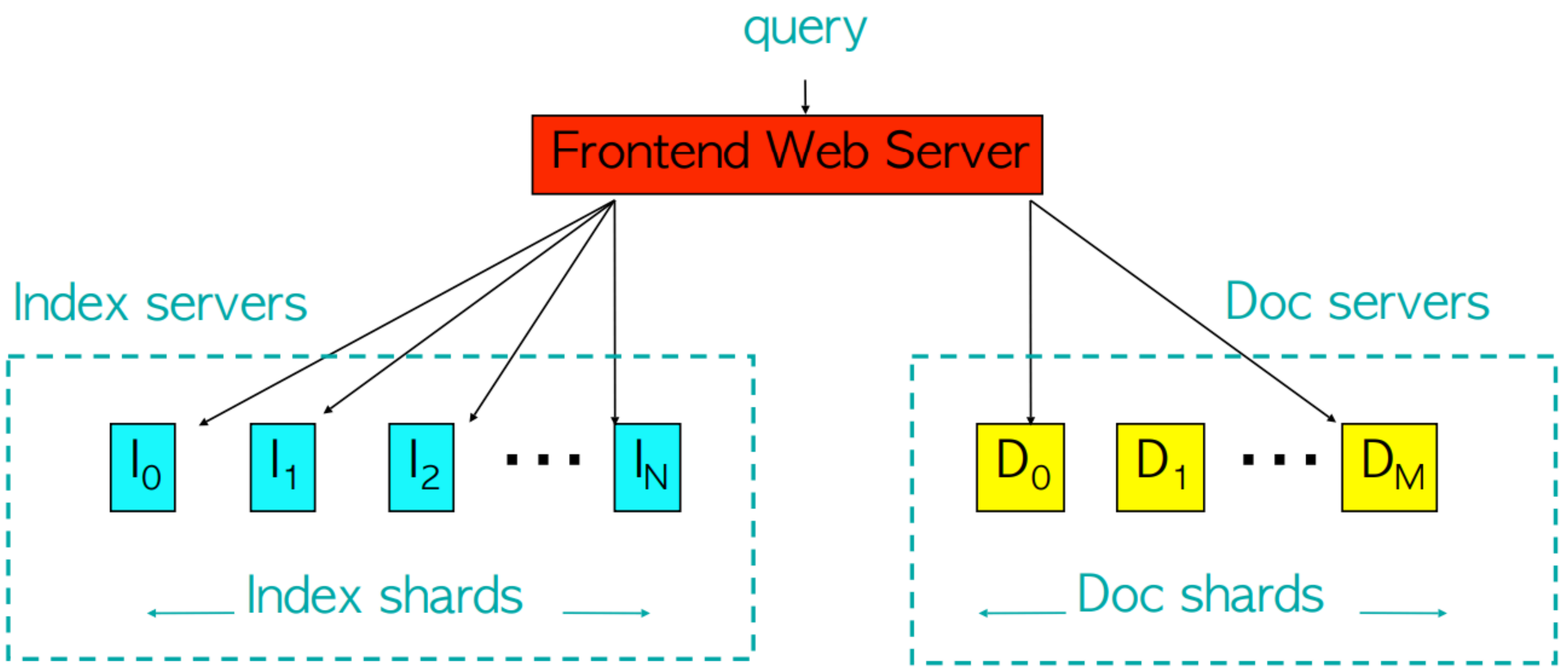
Motivating example: Google web search (1999 vs. 2010)

- # 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?

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

Evolution of data centers

- 1960's, 1970's: a few very large time-shared computers

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.

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.
- 2000-2020:
  - Data centers contain large numbers of nearly identical machines
  - Geographically spread around the world
  - Individual applications can use thousands of machines simultaneously

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.
- 2000-2020:
  - Data centers contain large numbers of nearly identical machines
  - Geographically spread around the world
  - Individual applications can use thousands of machines simultaneously
- 2020’s-today:
  - Accelerated construction of AI-specific datacenters
  - Clusters of datacenters in the same region to train massive models

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.
- 2000-2020:
  - Data centers contain large numbers of nearly identical machines
  - Geographically spread around the world
  - Individual applications can use thousands of machines simultaneously
- 2020’s-today:
  - Accelerated construction of AI-specific datacenters
  - Clusters of datacenters in the same region to train massive models

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.
- 2000-2020:
  - Data centers contain large numbers of nearly identical machines
  - Geographically spread around the world
  - Individual applications can use thousands of machines simultaneously
- 2020’s-today:
  - Accelerated construction of AI-specific datacenters
  - Clusters of datacenters in the same region to train massive models
- Companies consider data center technology a trade-secret, especially in the age of AI
  - Limited public discussion of the state of the art from industry leaders

Power is the biggest constraint

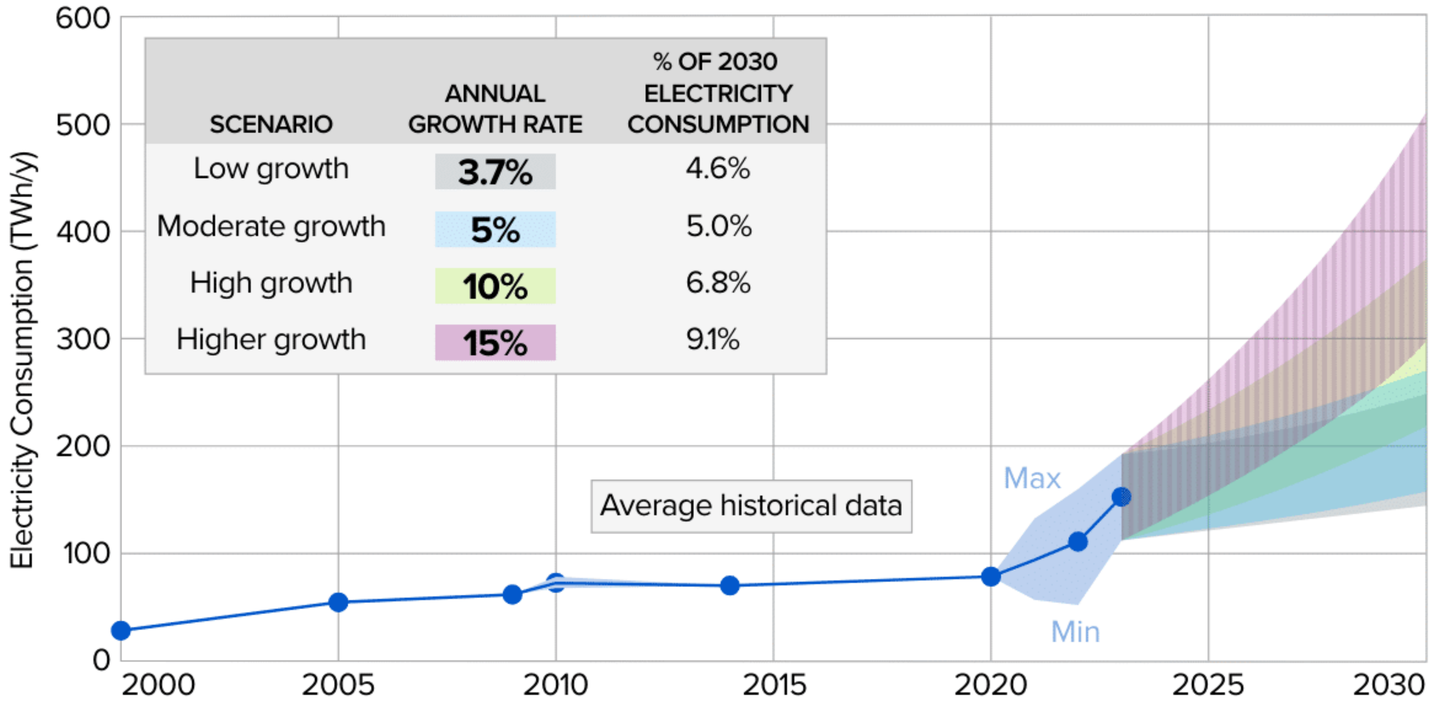


Figure ES-1. *Projections of potential electricity consumption by U.S. data centers: 2023–2030 . % of 2030 electricity consumption projections assume that all other (non-data center) load increases at 1% annually.*

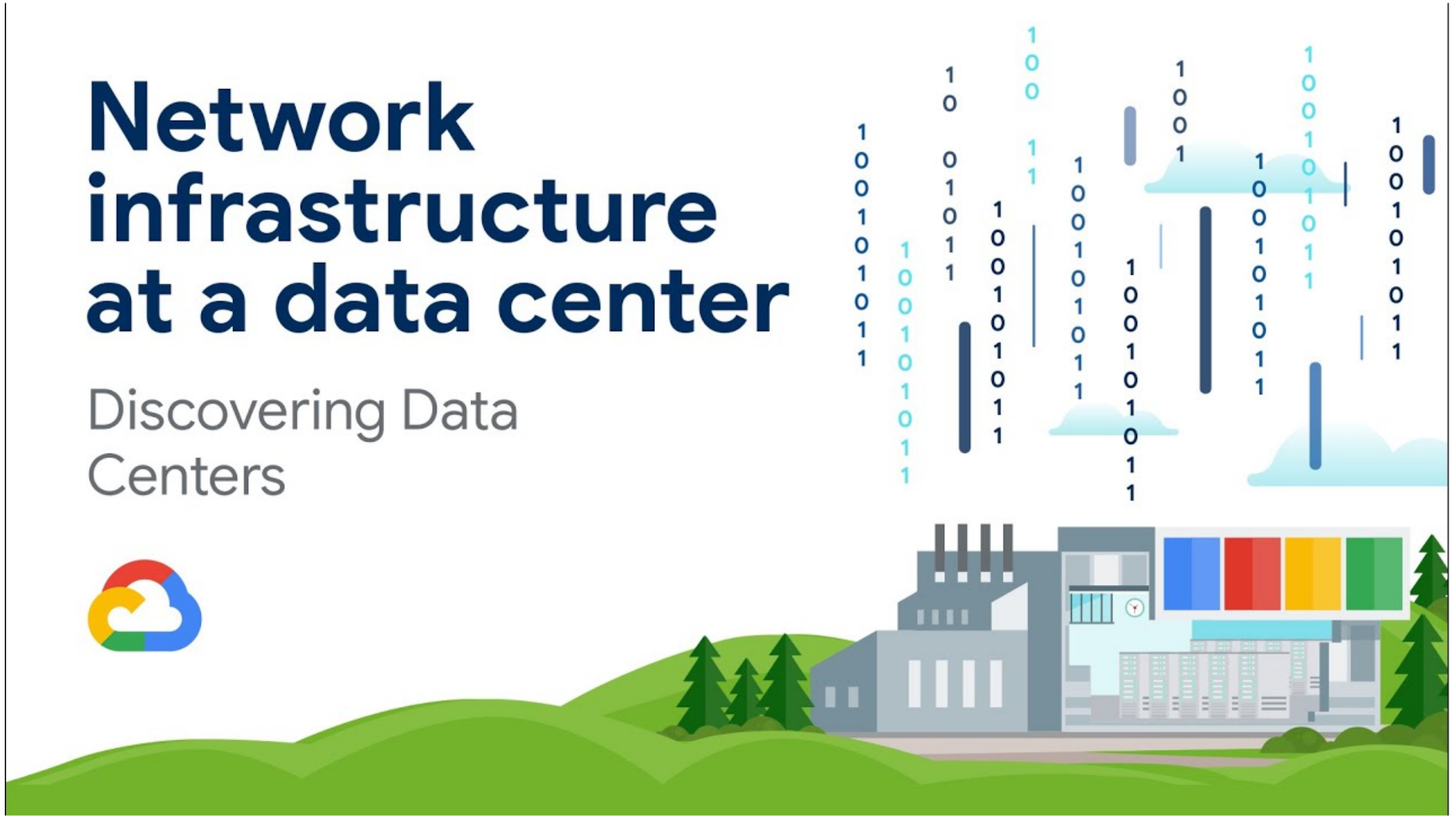
NATIONAL

Three Mile Island nuclear plant will reopen to power Microsoft data centers

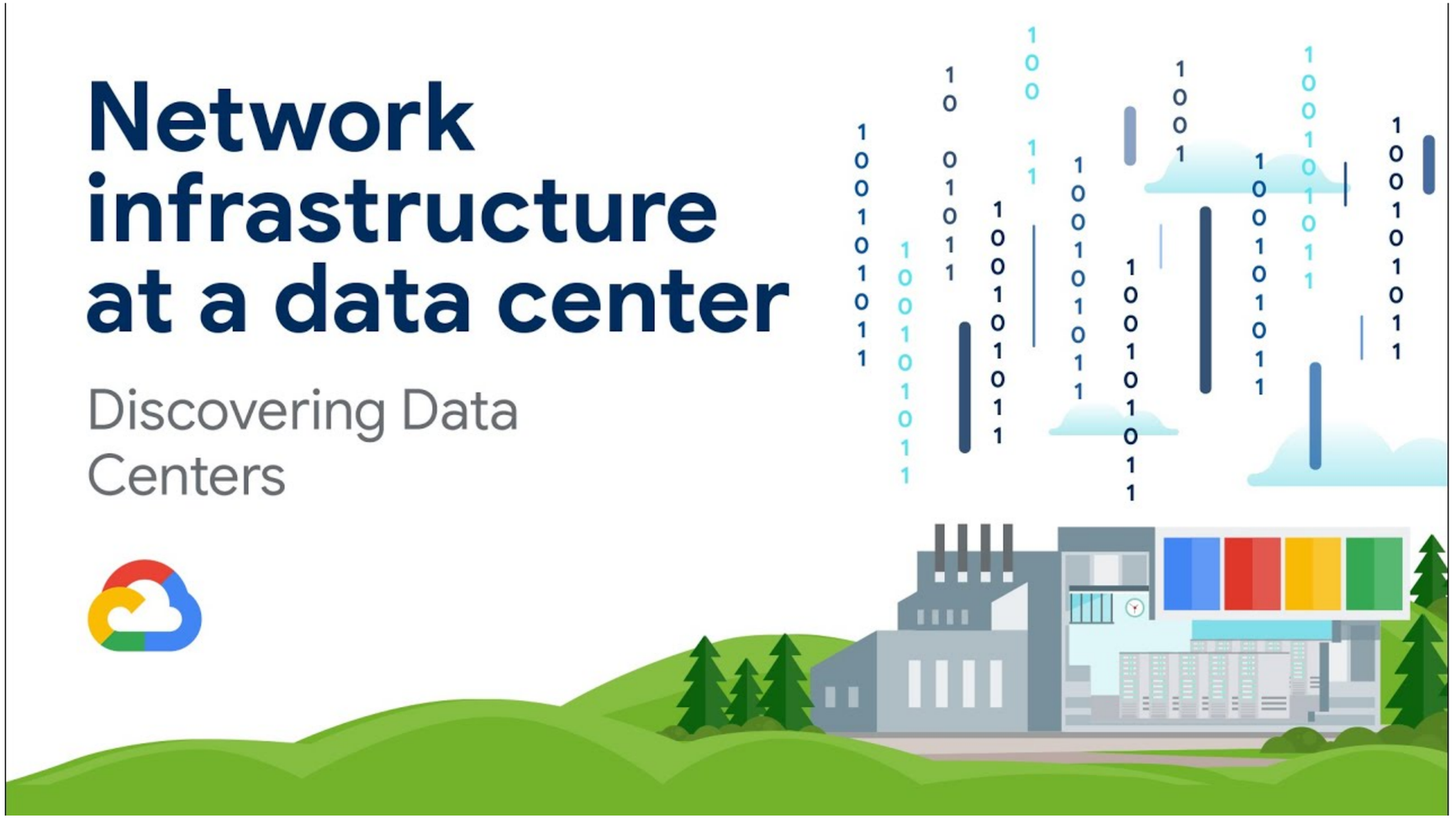
Google emissions jump 48% in five years due to AI data center boom

Water and electricity use soar to record highs

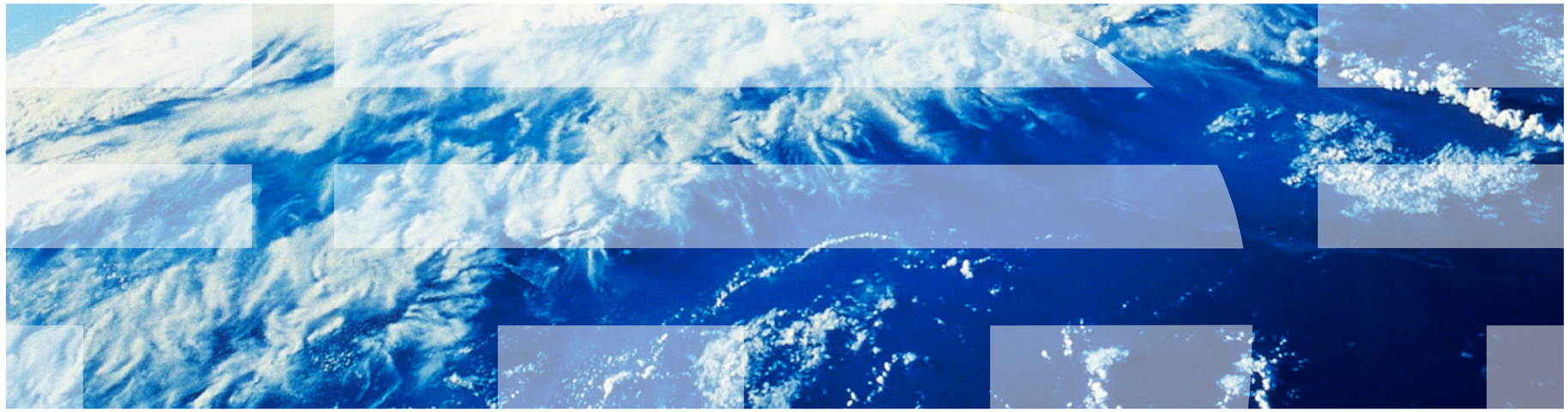
Core, Edge/Satellite, PoP



Core, Edge/Satellite, PoP



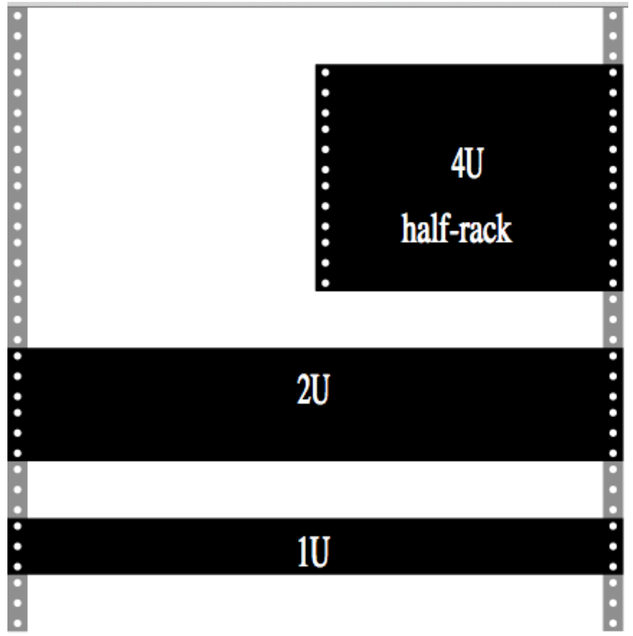
Datacenter building blocks



Rack

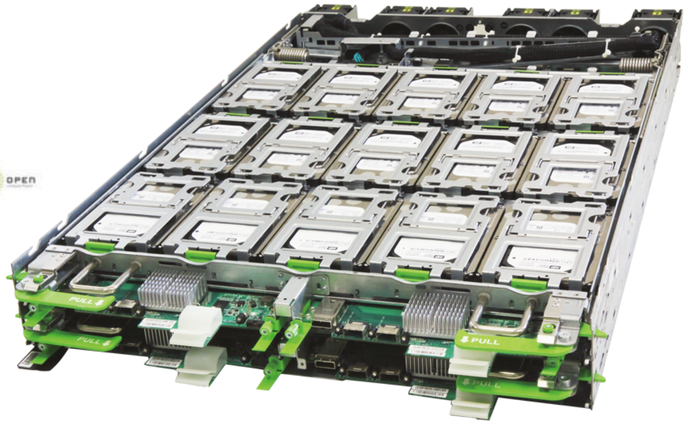
Rack

- 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
  - 128-192 cores
  - DRAM: 256-512 GB
- Typical storage: 2U
  - 30 drives
- Typical Network: 1U
  - 72 100Gb/s



Project Stargate

Announcing The Stargate Project

Announcing The Stargate Project

2 min. read · View original

The Stargate Project is a new company which intends to invest $500 billion over the next four years building new AI infrastructure for OpenAI in the United States. We will begin deploying $100 billion immediately. This infrastructure will secure American leadership in AI, create hundreds of thousands of American jobs, and generate massive economic benefit for the entire world. This project will not only support the re-industrialization of the United States but also provide a strategic capability to protect the national security of America and its allies.

The initial equity funders in Stargate are SoftBank, OpenAI, Oracle, and MGX. SoftBank and OpenAI are the lead partners for Stargate, with SoftBank having financial responsibility and OpenAI having operational responsibility. Masayoshi Son will be the chairman.