Large-Scale Data Management and Distributed Systems

I. Introduction

Vania Marangozova

Vania.Marangozova@imag.fr

2023-2024

About me

- CV
  - Associate professor since 2004
  - Habilitation for Directing Research since 2015

- Research
  - Resource Optimisation in large-scale Distributed Platformes – clouds
  - Microservices: Energy consumption and scaling

Organization of the course

- 2 complementary topics
    - Distributed algorithms (T. Ropars) – 18 hours
    - Data management (V. Marangozova) – 18 hours
- Data Management
    - 12 hours of lectures
    - 6 hours of practical sessions
- Grading
    - Graded Lab (25% of the final grade)
    - Written exam (75% of the final grade)

Covered Topics

- The challenges of Big Data and distributed data processing
- Processing large amount of data
    - Batch and stream processing systems
- Distributed (NoSQL) databases
- Underlying Design Principles

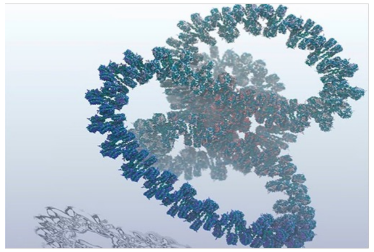
This lecture

- Introduction to the Big Data challenges
- Challenges of distributed computing
- Introduction to Cloud Computing
- Scalability techniques

The Challenges of Big Data

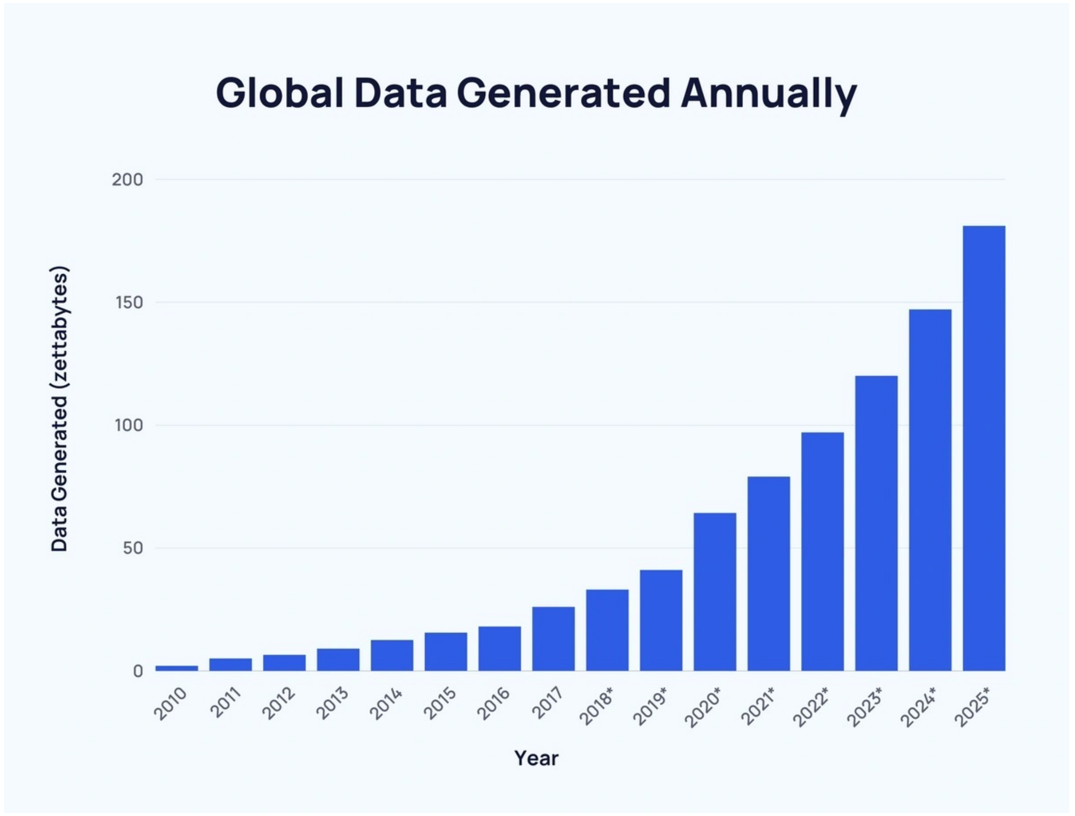
The Data Deluge

- All activities become digitalized
- Various sources of data
  - Sensors
  - Social media
  - Scientific experiments • Industry activity
  - Etc.



Some Numbers

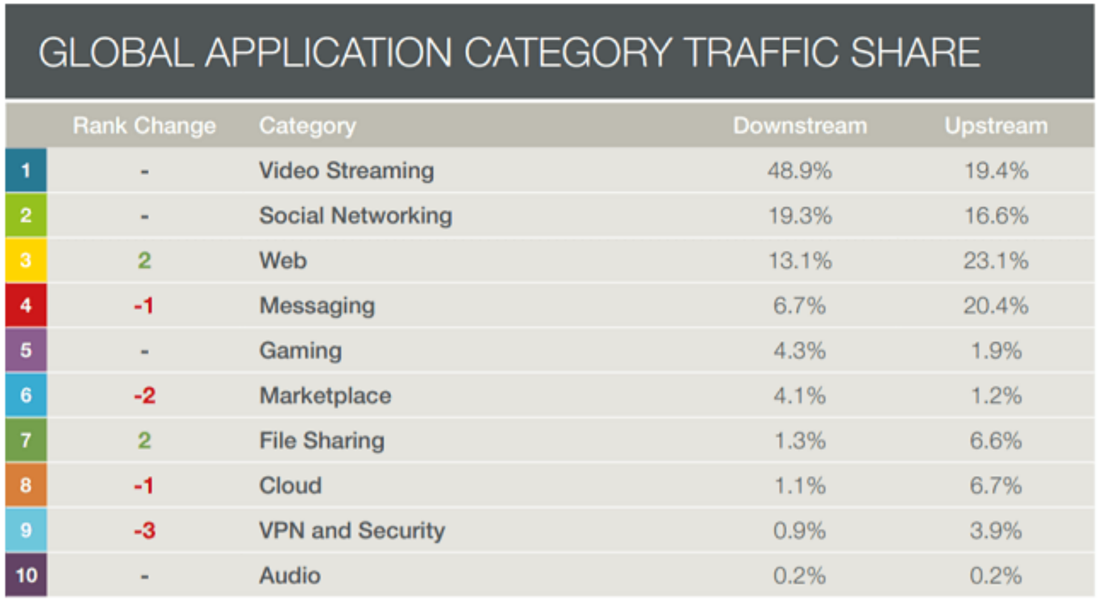
- Every day we create 328.77 million terabytes
  - This year we will produce 13 times more than what we produced in 2013
- COVID has brought +56% of data explosion (2020)
- ChatGPT gets x1,000,000 queries daily
  - Its monthly cost is estimated at $3 million
  - https://www.demandsage.com/chatgpt-statistics/
- ~99,000 Google search queries/second
  - https://seo.ai/



https://explodingtopics.com/blog/data-generated-per-day
Consulted 15/11/2023

Numbers continued...

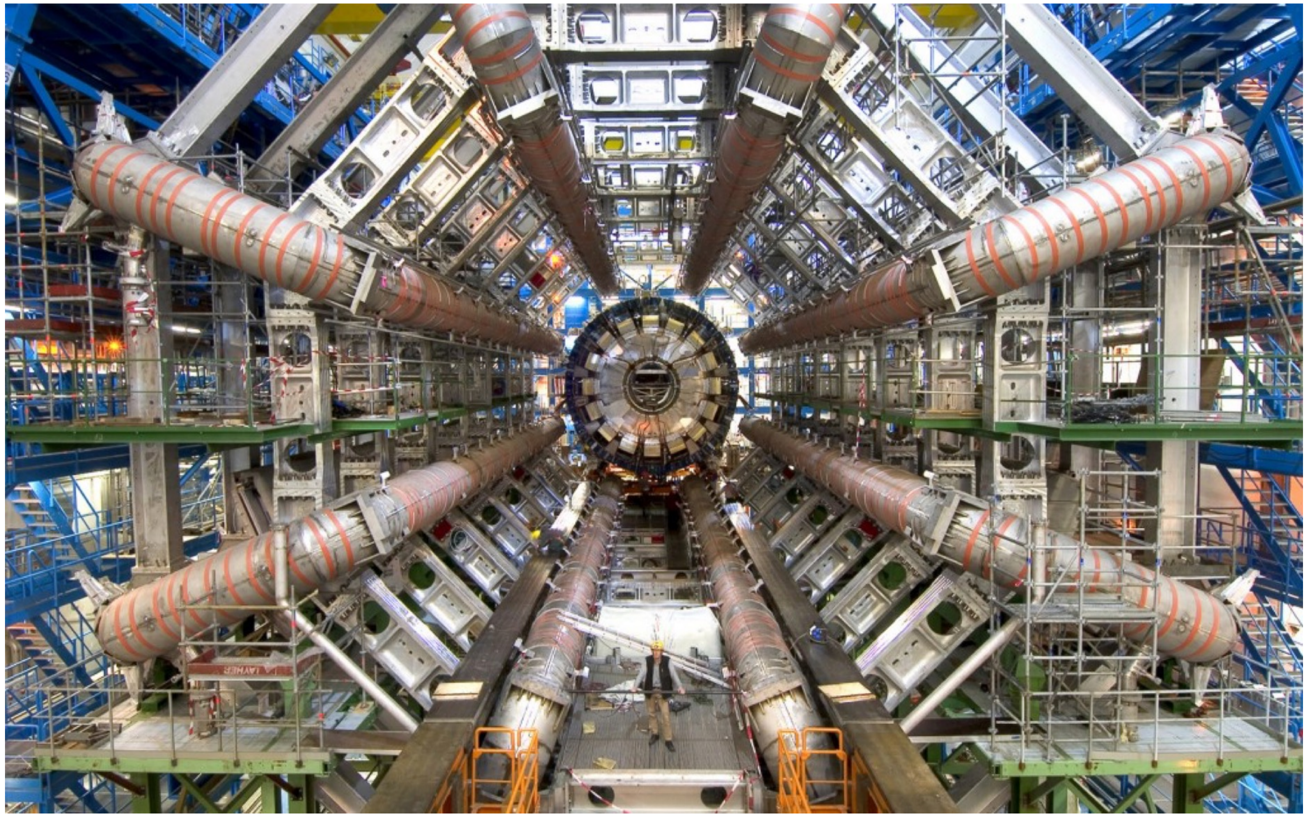
- Video streaming accounted for nearly half of all downstream internet traffic
  - https://bloggingwizard.com/live-streaming-statistics/
- The number of devices connected to IP networks
  is more than three times the global population
  - Statista & CISCO report
- 14 billion connected IoT devices
  - https://explodingtopics.com/blog/iot-stats
- A new website is built every 3 seconds
  - https://www.forbes.com/advisor/business/software/website-statistics/



GLOBAL APPLICATION CATEGORY TRAFFIC SHARE

	Rank Change	Category	Downstream	Upstream
1	-	Video Streaming	48.9%	19.4%
2	-	Social Networking	19.3%	16.6%
3	2	Web	13.1%	23.1%
4	-1	Messaging	6.7%	20.4%
5	-	Gaming	4.3%	1.9%
6	-2	Marketplace	4.1%	1.2%
7	2	File Sharing	1.3%	6.6%
8	-1	Cloud	1.1%	6.7%
9	-3	VPN and Security	0.9%	3.9%
10	-	Audio	0.2%	0.2%

40 TB of data every second during an experiment at the Large Hadron Collider



Hardware Capacity

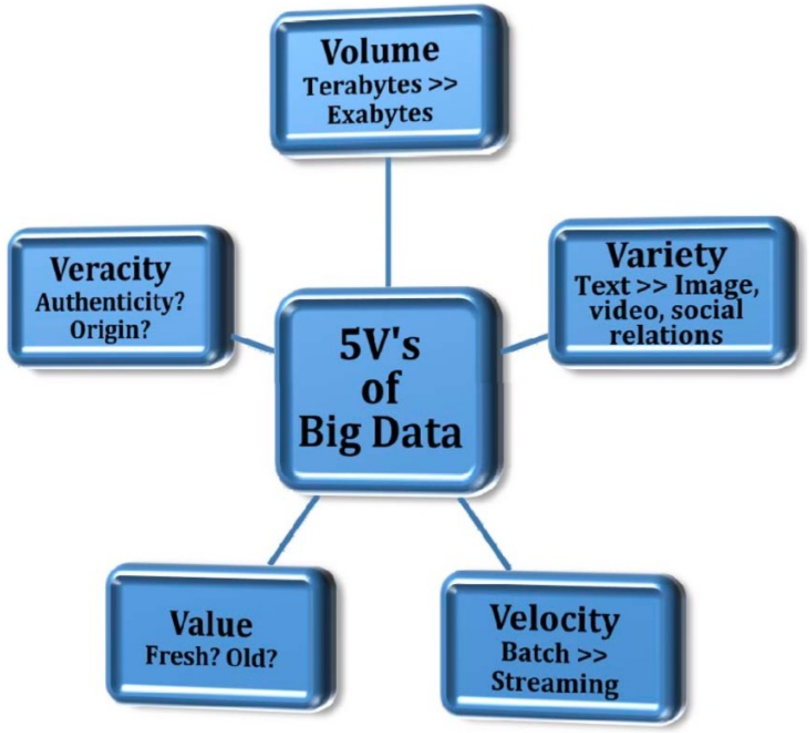
- **To produce data**
- **To store data**
  - All the music of the world stored for $~ 500
  - Amazon EC2 instances: RAM up to 24TB, disk x10x16TB
- **To compute on data**
  - Google data-centers: more than 2.5M servers (2016), secret, "x10x1000" servers
  - Amazon capacity increases each day, the offer changes each year
- Not to forget generative AI which is a disruptive technology

Huge opportunities for storing and processing data

Big data challenges: The V's

Source : Big Data for Modern Industry: Challenges and Trends

https://ieeexplore.ieee.org/document/7067026



Big data challenges: The V's

- Volume: Amount of data generated
- Variety: all kinds of data are generated (text, image, voice,time series, etc.)
- Velocity: Rate at which data are produced and should be processed
- Veracity: Noise/anomalies in data, truthfulness
- Value: How do we extract/learn valuable knowledge from the data

In this course

- Volume, Velocity
  - will see Variety

Questions to be answered:

- How to build a system and algorithms that can process huge amount of data?
- How to build a system and algorithms that can process data in a timely manner?
- (Bonus questions) How to build software that can deal with the variety of data?

Distributed and Parallel Systems

Motivation

- Big data = big computation
  - Needs quite a lot of resources !

- Distribution: when it does not fit in a single machine
- Paralellization: when the computation takes a lot of time

- Note that:
  - Different strategies can be used to leverage these resources
  - Using large amount of resources **presents new challenges**

Increasing the processing power and the storage capacity

- Goals
  - Increasing the amount of data that can be processed (weak scaling)
  - Decreasing the time needed to process a given amount of data (strong scaling)

- Two solutions
  - Scaling up
  - Scaling out

Vertical scaling (scaling up)

- Idea: increase the processing power by adding resources to existing nodes
  - Upgrade the processor (more cores, higher frequency)
  - Increase memory volume
  - Increase storage volume

- Pros and Cons
  - ☺ Performance improvement without modifying the application
  - ☹ Limited scalability (capabilities of the hardware, cf The end of Moore's law)
  - ☹ Expensive (non linear costs)

Horizontal scaling (scaling out)

- Idea: increase the processing power by adding more nodes to the system
  - Cluster of commodity servers

- Pros and Cons
  - ☹ Often requires modifying applications
  - ☺ Less expensive (nodes can be turned off when not needed)
  - ☺ Infinite scalability

The solution studied in this course

Large Scale Infrastructures

Figure: Google Data-center

Figure: Amazon Data-center

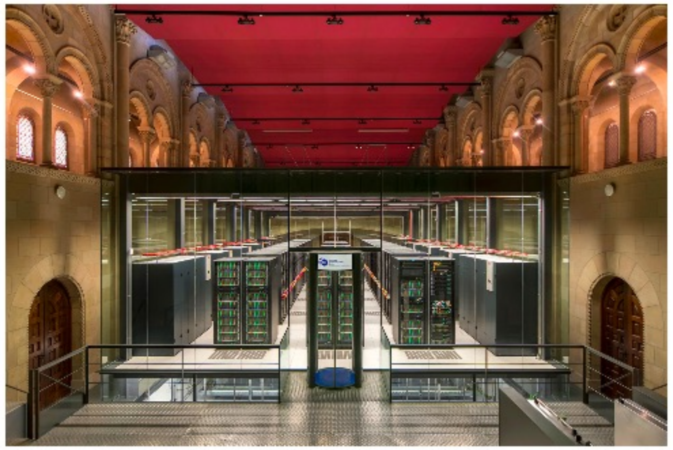


Figure: Barcelona Supercomputing Center

Distributed Computing: Definition

"A system in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages"

G.Coulouris, J. Dollimore, T. Kindberg.
Distributed Systems: Concepts and Design (4th Edition).

- Network: latency & failures
- No global memory = no global state: each entity has its own local memory
- Coordination
- Message passing

- Each entity of the system is called a **node**.

Distributed computing: Challenges

- Scalability
  - How to take advantage of a large number of distributed resources?
- Performance
  - How to take full advantage of the available resources?
  - Moving data is costly: how to maximize the ratio between computation and communication?
  - How to ensure that the latency of requests processing remains below some upper bound?

Distributed computing: Challenges

- Fault tolerance
  - The more resources, the higher the probability of failure
  - MTBF (Mean Time Between Failures)
    - MTBF of one server = 3 years
    - MTBF of 1000 servers $\simeq$ 19 hours (beware: over-simplified computation)
  - How to ensure computation completion?
  - How to ensure that results are correct?
- Programmability
  - How to provide programming models that hide the complexity of distributed computing? (while remaining efficient)
  - What high level services should be made available to ease life of programmers?

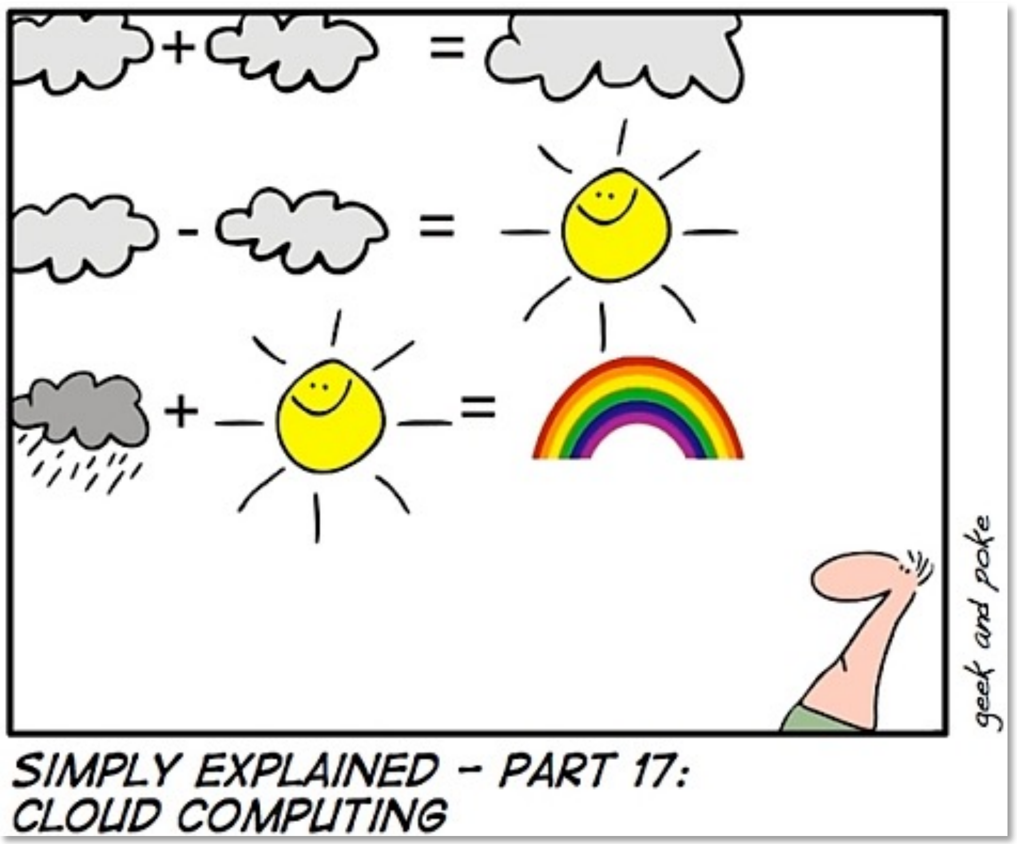
The Problem of Distributed Computing

…the same as parallel computing… allows **great** things but creates **huge** problems!

You can have a second computer once you've shown you know how to use the first one. (P. Braham)

- Horizontal scaling is very popular.
  - But not always the most efficient solution (both in time and cost)
- Examples
  - Processing a few 10s of GB of data is often more efficient on a single machine that on a cluster of machines
  - Sometimes a single threaded program outperforms a cluster of machines (F. McSherry et al. "Scalability? But at what COST!". 2015., http://www.frankmcsherry.org/graph/scalability/cost/2015/01/15/COST.html)

Cloud Computing



Cloud Computing Definitions

Business Perspective

UCBerkeley R&DLabs:

"Cloud computing has the following characteristics:

(1) The illusion of infinite computing resources...

(2) The elimination of an up-front commitment by Cloud users...

(3) The ability to pay for use...as needed..."

Cloud Computing Definitions

Technical Perspective

NIST (National Institute of Standards and Technology)

- **On-demand self-service.** A consumer can ask for more resources: computation, network, storage, ...
- **Broad network access.**
- **Resource pooling:** resources (virtual and physical) are dynamically assigned and serve multiple consumers
- **Rapid elasticity.** Capabilities can be elastically provisioned and released
- **Measured service.** "pay as you go"

Pros and Cons

☺ Pay only for the resources you use

☺ Get access to large amount of resources

- Amazon Web Services features millions of servers

☹ Volatility

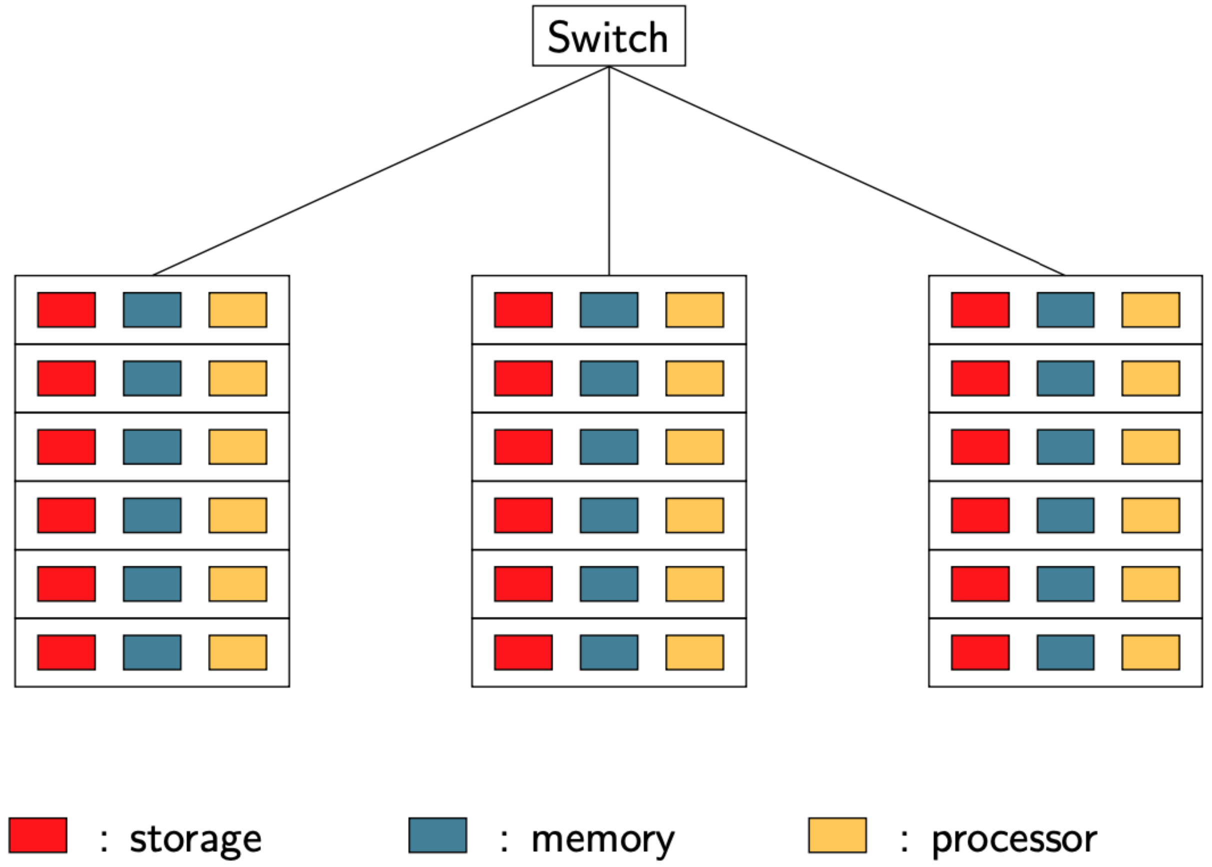
- Low control on the resources
- Example: Access to resources based on bidding
- See "The Netflix Simian Army" (https://netflixtechblog.com/the-netflix-simian-army-16e57fbab116)

☹ Performance variability

- Physical resources shared with other users

Architecture of a Data Center

Simplified



Architecture of a Data Center

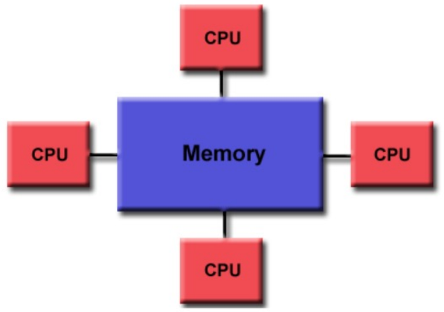
- A shared-nothing architecture
  - Horizontal scaling
  - No specific hardware

- A hierarchical infrastructure
  - Resources clustered in racks
  - Communication inside a rack is more efficient than between racks
  - Resources can even be geographically distributed over several datacenters

Communication

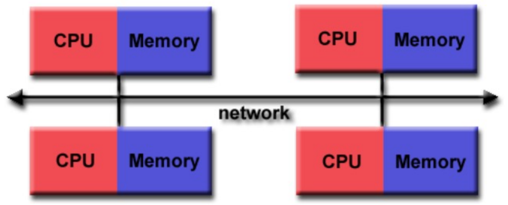
Shared Memory

- Entities share a global memory
- Communication by reading and writing to the globally shared memory
- Communication between threads inside one node



Message Passing

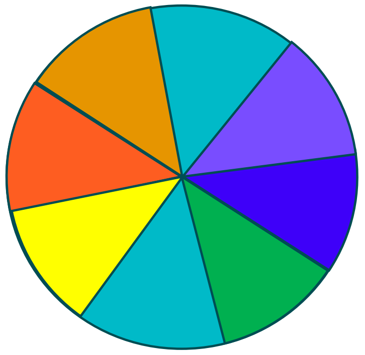
- Entities have their own private memory
- Communication by sending/receiving messages over a network
- Communication between nodes



How to Distribute Data ?

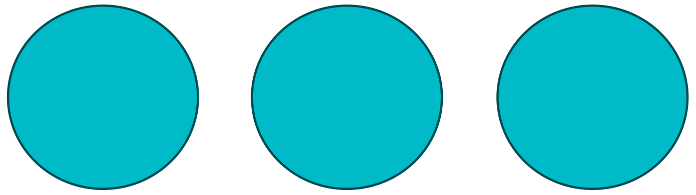
Partitioning

- Splitting the data into partitions
- Partitions are assigned to different nodes
- Main goal: Performance
    - Partitions can be processed in parallel



Replication

- Several nodes host a copy of the data
- Main goal: Fault tolerance
    - No data lost if one node crashes

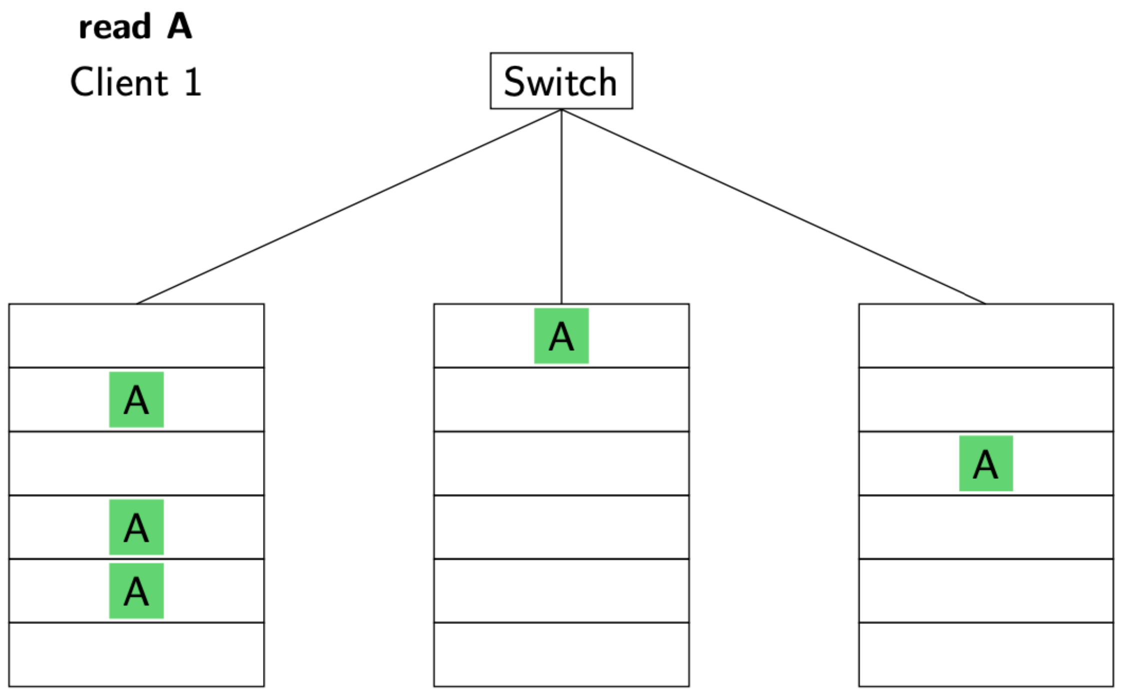


Replication

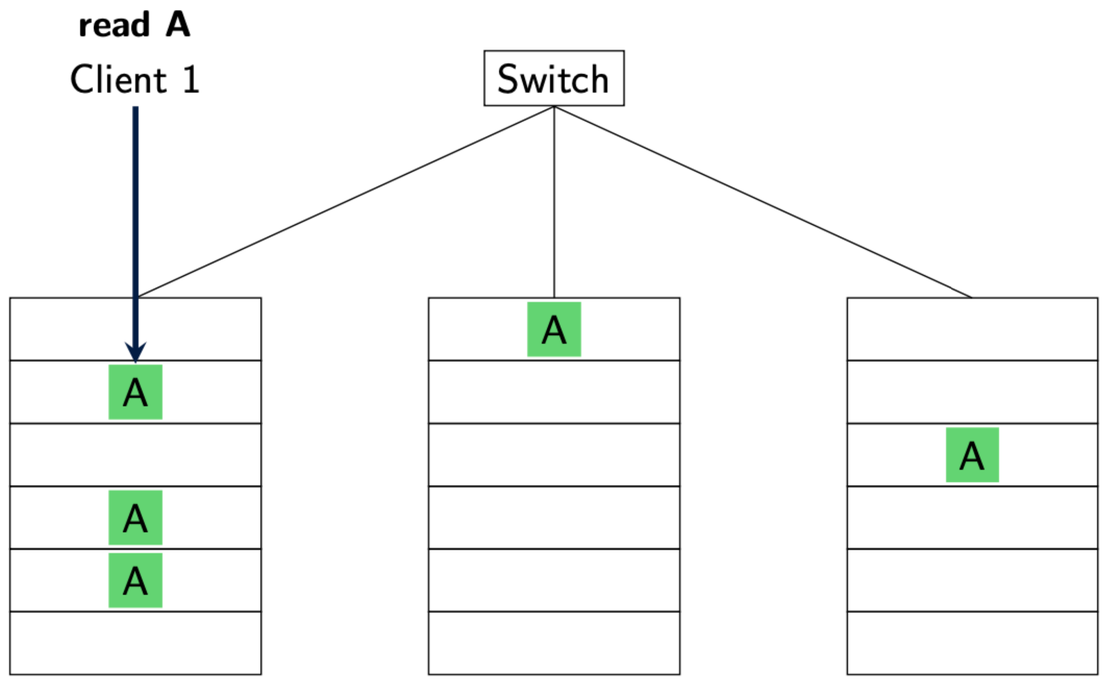
- Purposes
  - Continuing to serve requests when parts of the system fail
  - Keep data close to the users
  - Having multiple servers able to answer read requests

- Challenges
  - How to handle operations that modify data? (write operations)
    - Consistency (Consensus in a distributed system is a very difficult problem)
    - Performance

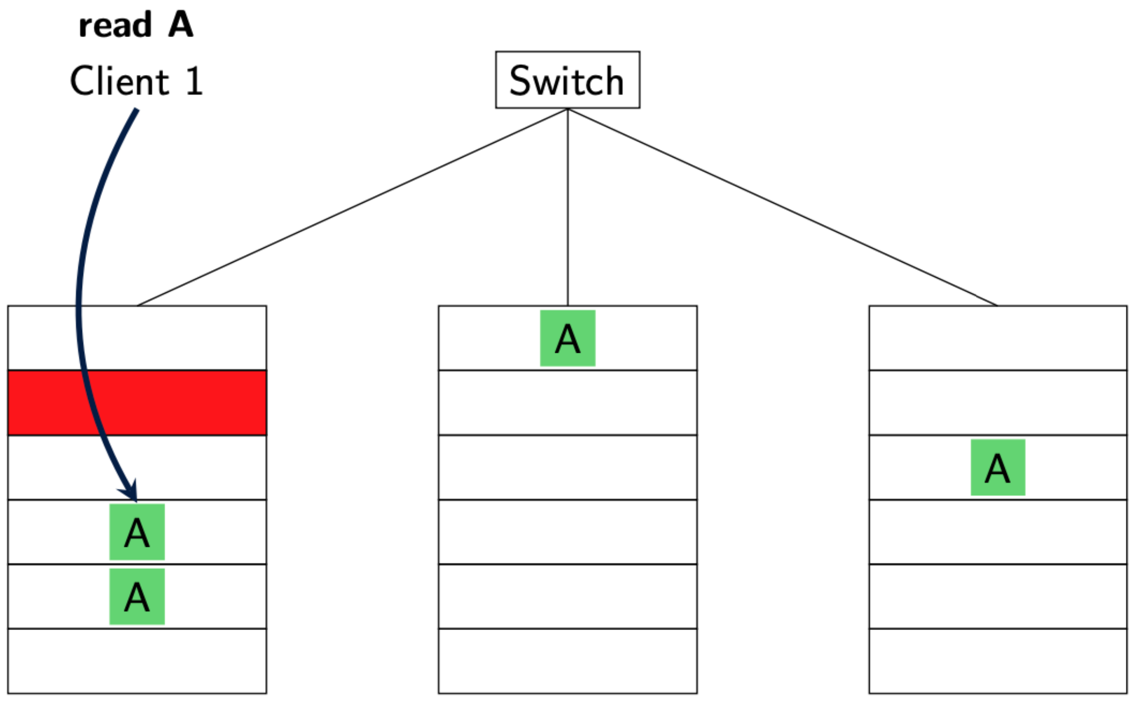
Replication: read



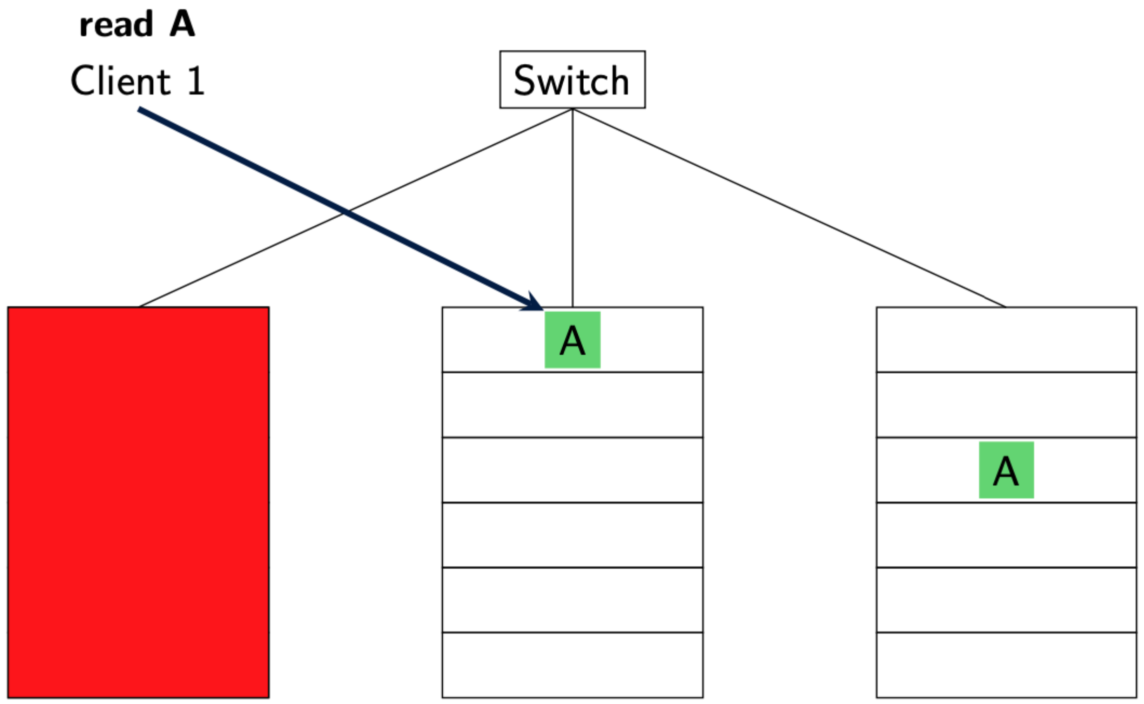
Replication: read the closest



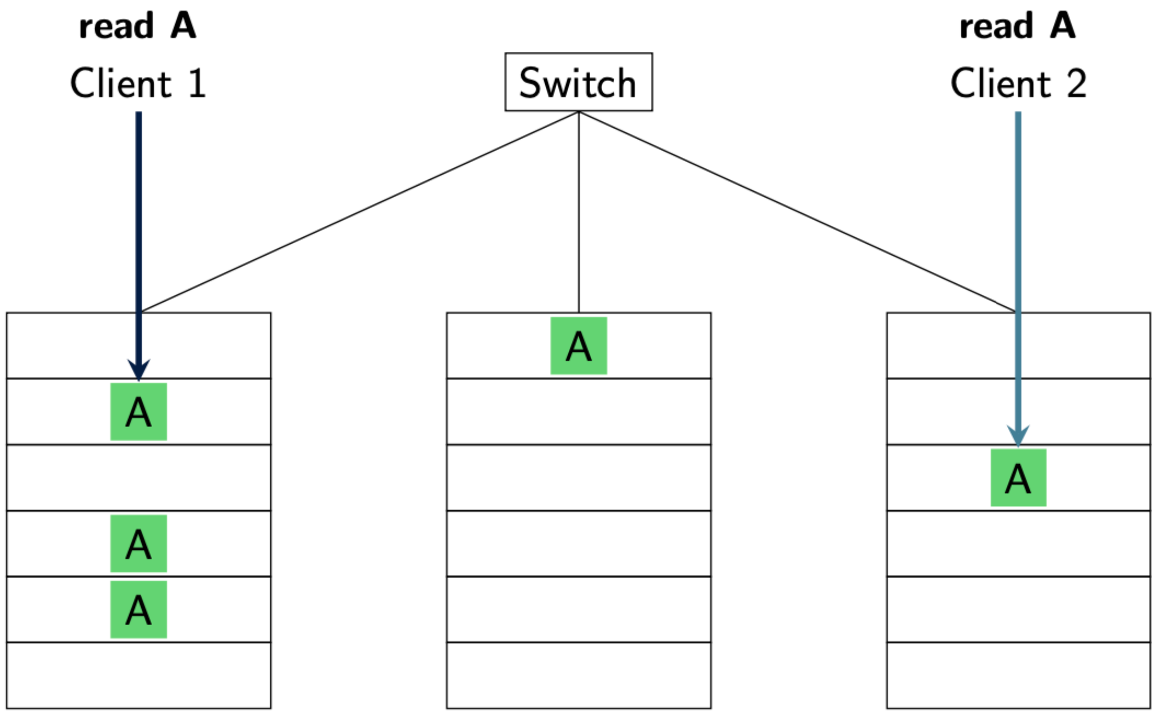
Replication: read the closest replica



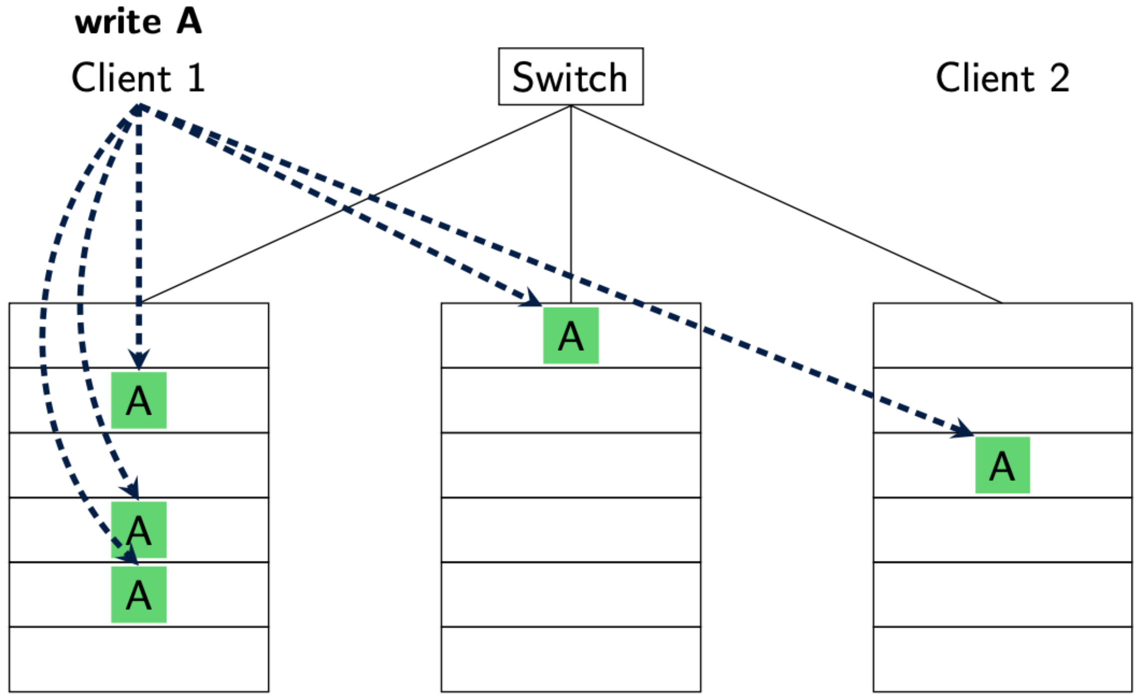
Replication: if the closest crashes



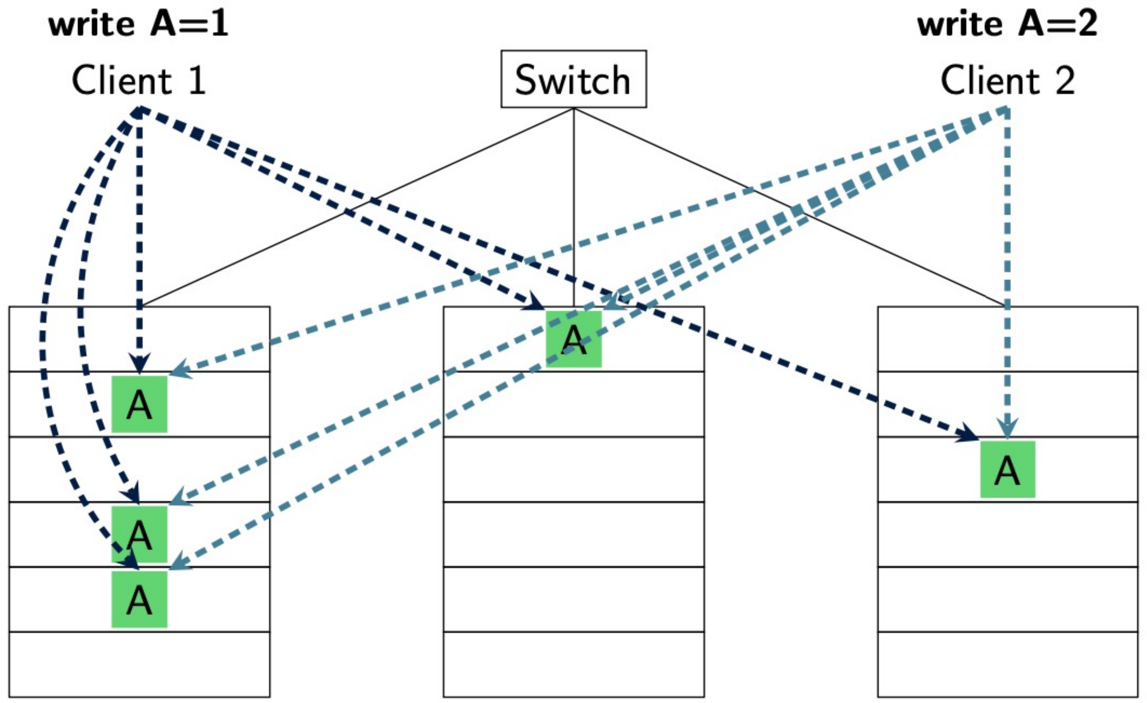
Replication: parallel reads



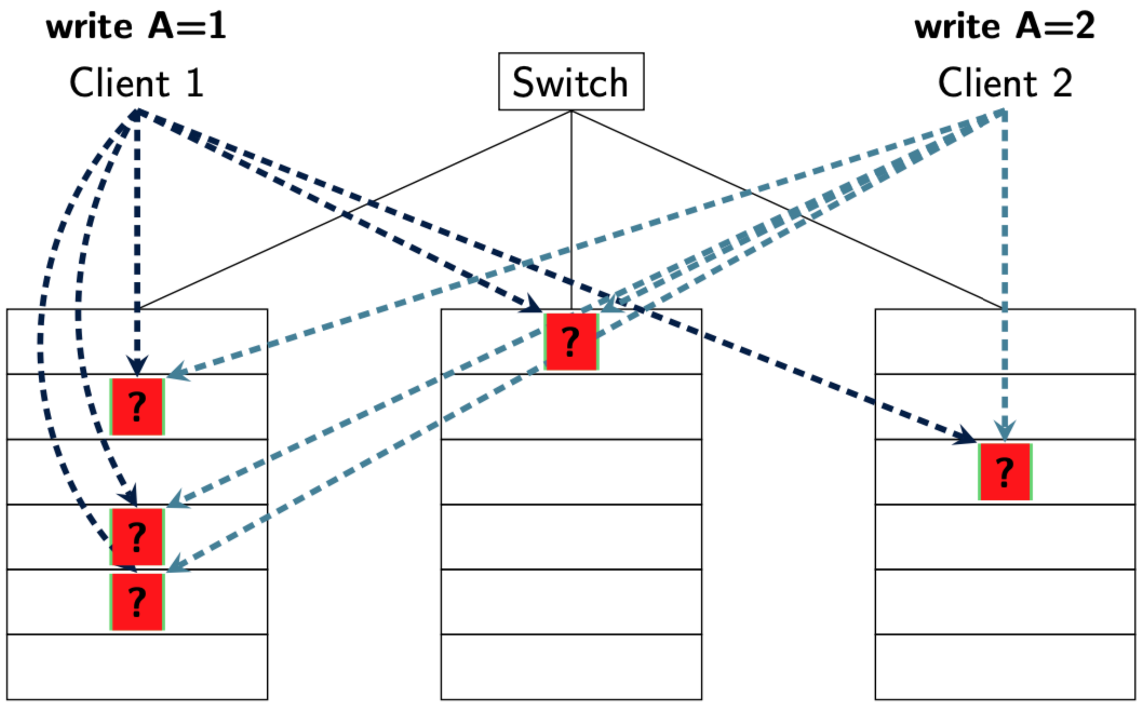
Replication: write



Replication: parallel writes



Replication: parallel writes

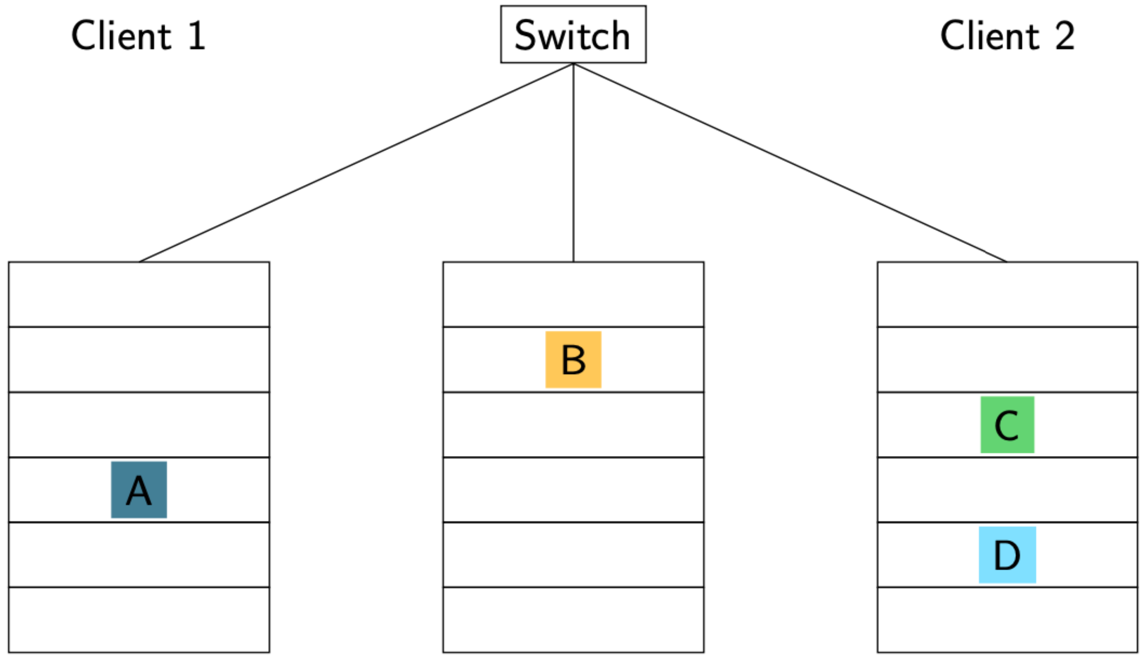


Partitioning

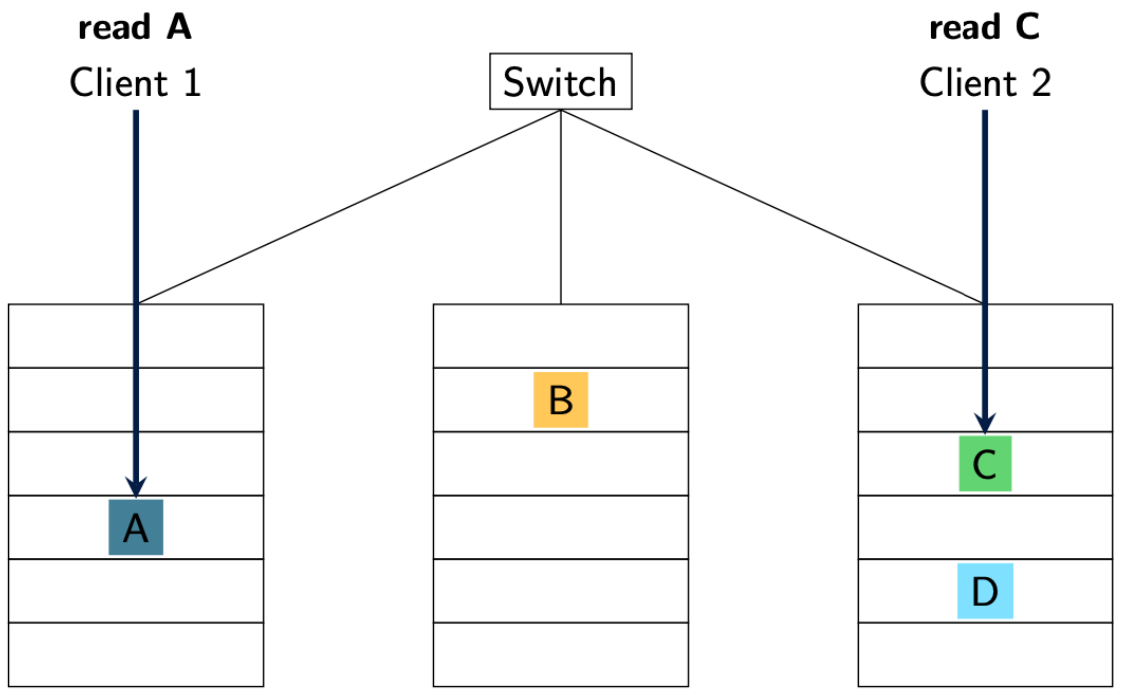
- Purposes
  - Performance
    - Distributing the load over several nodes

- Challenges
  - How to partition the data?
  - Evenly distributed load (even for skewed workloads)
  - Range queries

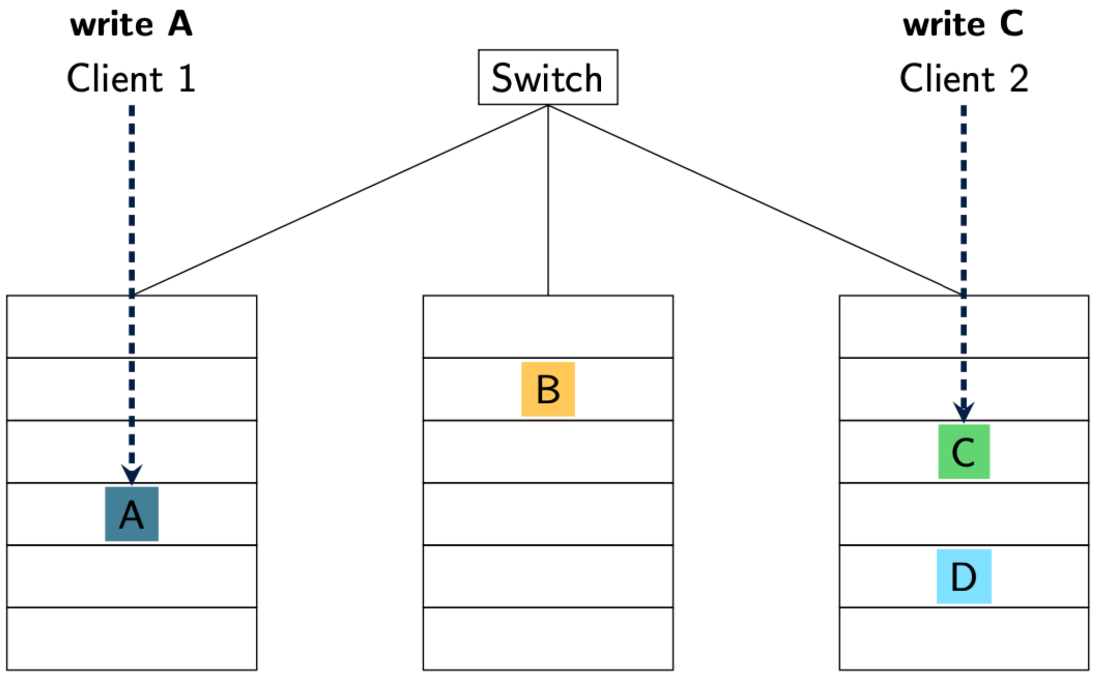
Partitioning



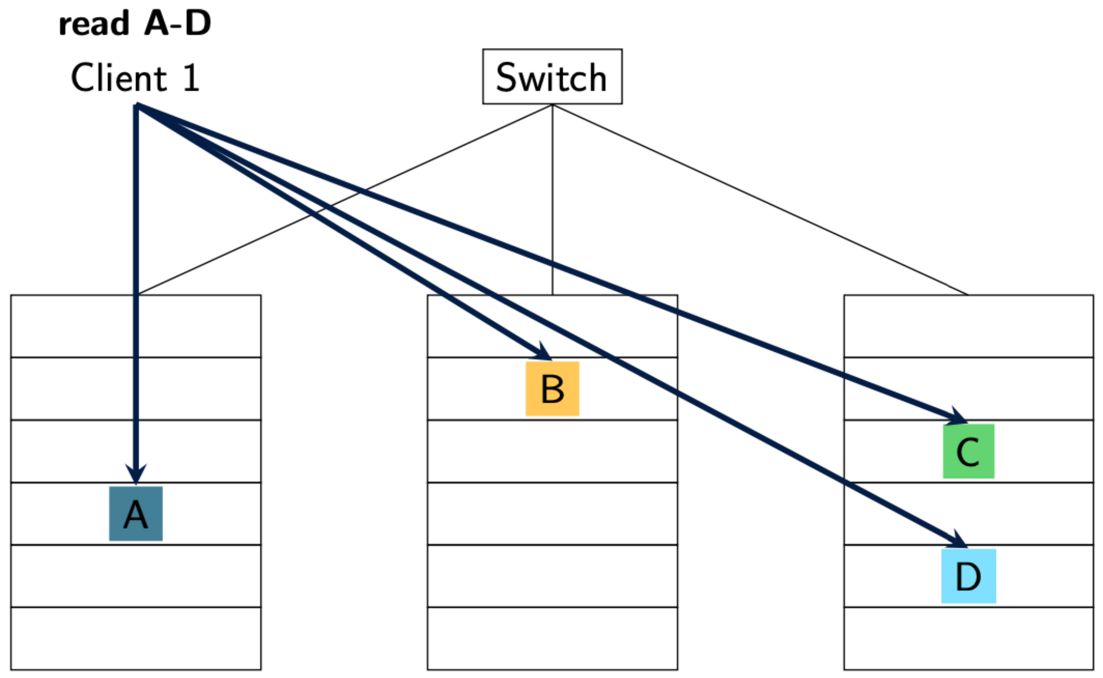
Partitioning: parallel reads



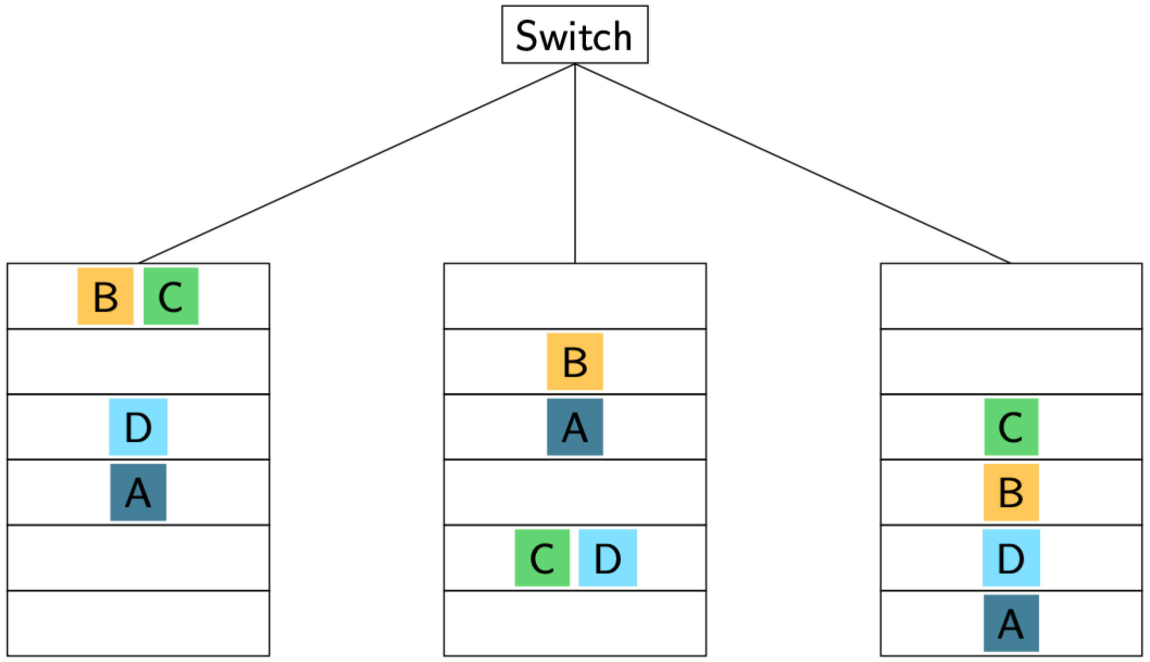
Partitioning: parallel writes



Partitioning: reading the whole data



Partitioning+Replication


Switch
B
C
D
A
B
A
C
D
C
B
D
A