

# Data Management in Large-Scale Distributed Systems

## Introduction

Thomas Ropars

`thomas.ropars@univ-grenoble-alpes.fr`

`http://tropars.github.io/`

2018

# Teaching staff

- Thomas Ropars  
(`thomas.ropars@univ-grenoble-alpes.fr`)
- Yves Denneulin (`yves.denneulin@grenoble-inp.fr`)
- Francieli Zanon Boito (`francieli.zanon-boito@inria.fr`)

# Organization of the course

## 18 hours

- 12 hours of lectures
- 6 hours of practical sessions

## Grading

- Graded Lab (30% of the final grade)
- Written exam (70% of the final grade)

## Covered topics

- The challenges of Big Data and distributed data processing
- The Map/Reduce programming model
- Batch and stream processing systems
- Distributed databases
- Performance of distributed data processing

# Overview of this lecture

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

# Agenda

The challenges of Big Data

Distributed and Parallel Systems

Cloud Computing

Running at scale

# References

- Coursera – *Big Data*, University of California San Diego
- The lecture notes of V. Leroy
- The lecture notes of R. Lachaize
- Designing Data-Intensive Applications by Martin Kleppmann

# The data deluge

Many sources of data



# The data deluge

## Many sources of data

- Sensors
- Social media
- Scientific experiments
- Industry activity
- Etc.

## Some numbers

- Every 2 days, we create as much information as we did since 2013<sup>1</sup>
- 40K search queries on Google every second<sup>2</sup>
- 30M messages posted on Facebook every minute
- 6.1 Billions of smartphone users by 2020 (and 50 Billions connected devices)
- 570 new web sites every minute
- Largest database: 3.2 Trillions rows (AT&T)
- 40 TB of data every second during an experiment at the Large Hadron Collider

---

<sup>1</sup><https://www.slideshare.net/BernardMarr/big-data-25-facts>

<sup>2</sup><https://www.newgenapps.com/blog/>

# Hardware capacity

## Storage

- All the music of the world stored for 500\$
- Large Amazon EC2 instance: 768GB of RAM, 3.6TB of SSD

## Computing resources

- Google data-centers: more than 2.5M servers (2016)
- Amazon capacity increase each day = size of Amazon in 2005

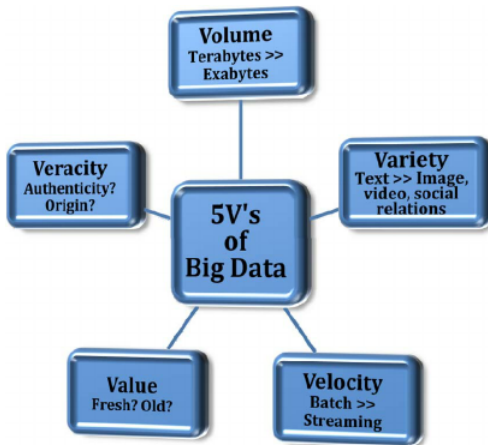
**Huge opportunities for storing and processing data**

# Big data challenges: The V's

source: Big Data for Modern Industry: Challenges and Trends

# Big data challenges: The V's

source: Big Data for Modern Industry: Challenges and Trends



# 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

# Big data challenges: The V's

In this course we are going to deal with:

- **Volume**
- **Velocity**
- **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?

# Agenda

The challenges of Big Data

Distributed and Parallel Systems

Cloud Computing

Running at scale



# Motivation

The solution to process large amount of data:

**Using large amount of resources**

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

# 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




# 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

# 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

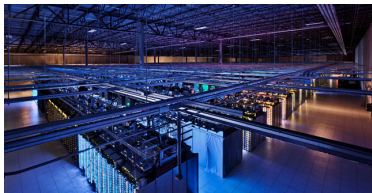


Figure: Google Data-center



Figure: Amazon Data-center



Figure: Barcelona Supercomputing Center



# Distributed computing: Definition

A **distributed computing system** is a system including several computational entities where:

- Each entity has its own local memory
- All entities communicate by message passing over a network

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

# Distributed computing: Challenges<sup>1</sup>

---

<sup>1</sup>Read Chapter 1 of *Designing Data-Intensive Applications* for further details

# Distributed computing: Challenges<sup>1</sup>

## 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?

---

<sup>1</sup>Read Chapter 1 of *Designing Data-Intensive Applications* for further details

# 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?

# A warning about distributed computing

*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 than 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.)

# Agenda

The challenges of Big Data

Distributed and Parallel Systems

Cloud Computing

Running at scale

# Where to find computing resources?

## Cloud computing

- A service provider gives access to computing resources through an internet connection.

## Pros and Cons

# Where to find computing resources?

## Cloud computing

- A service provider gives access to computing resources through an internet connection.

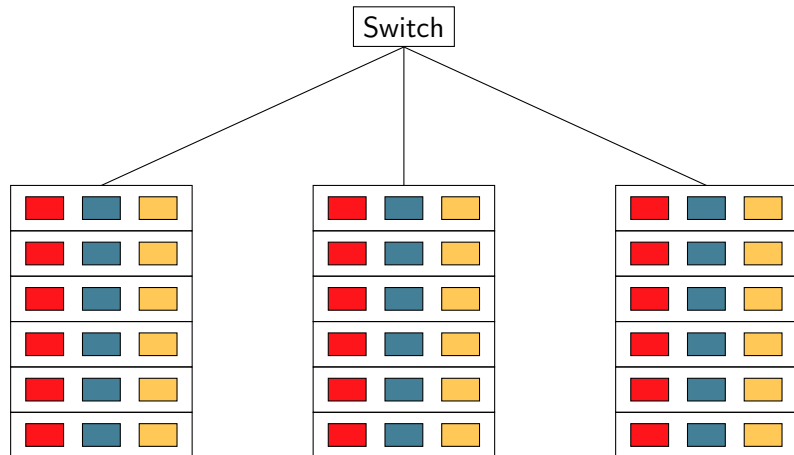
## Pros and Cons


- 😊 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"
- 😞 Performance variability
  - ▶ Physical resources shared with other users





# Architecture of a data center

Simplified



 : storage

 : memory

 : processor

# 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

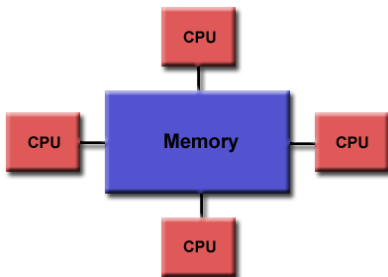
# A hybrid system

Two paradigms for communicating between computing entities:

- Shared memory
- Message passing

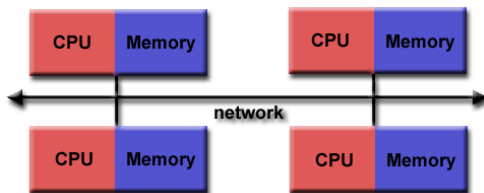
# Shared memory

- 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



# Agenda

The challenges of Big Data

Distributed and Parallel Systems

Cloud Computing

Running at scale

# Running at scale

## How to distribute data?

- Partitioning
- Replication

# Running at scale

## How to distribute data?

- Partitioning
- Replication

## Replication

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

## Partitioning

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



# Replication

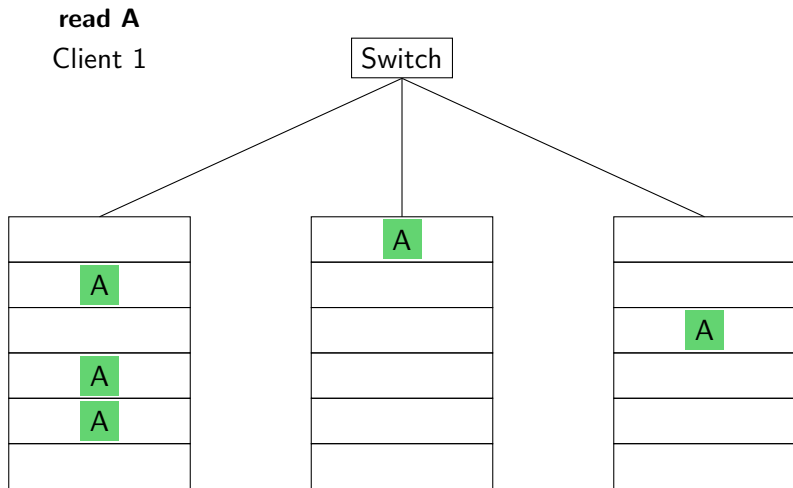
## Purposes

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

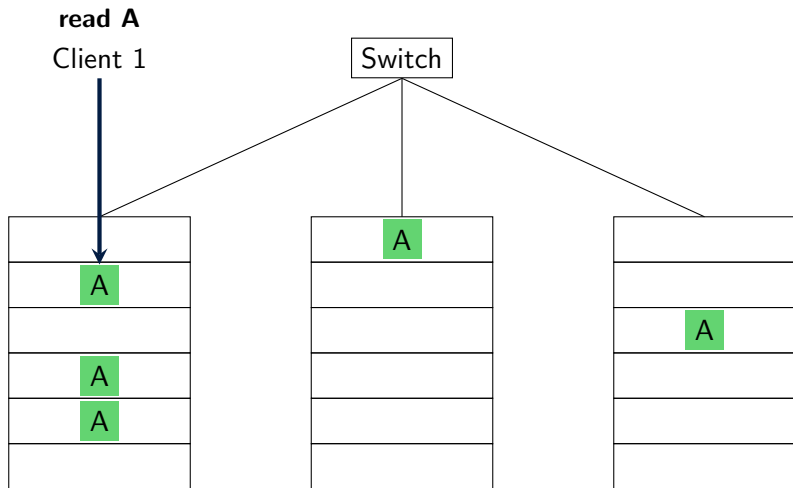
## Challenges

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

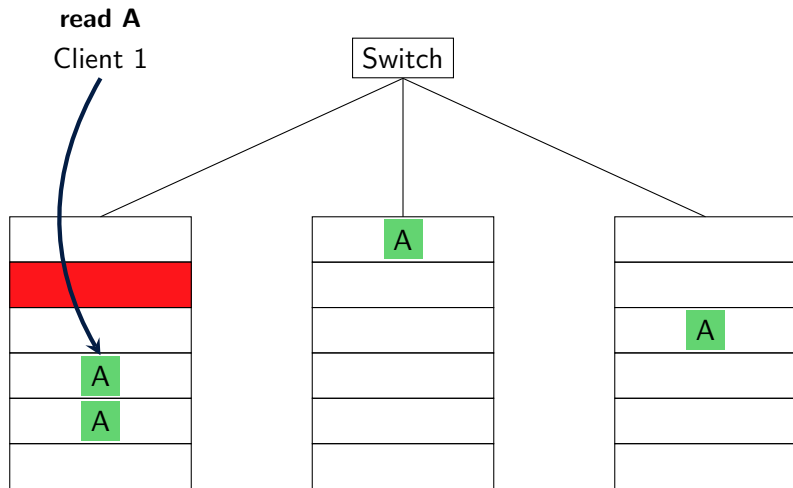
# Replication



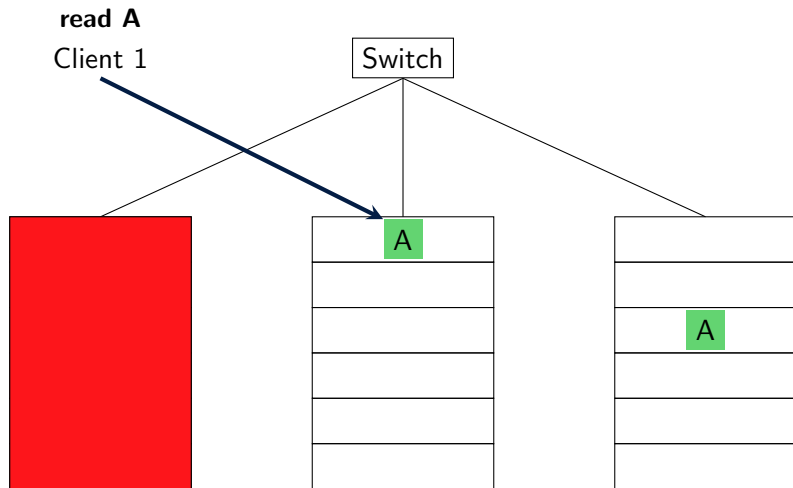
# Replication



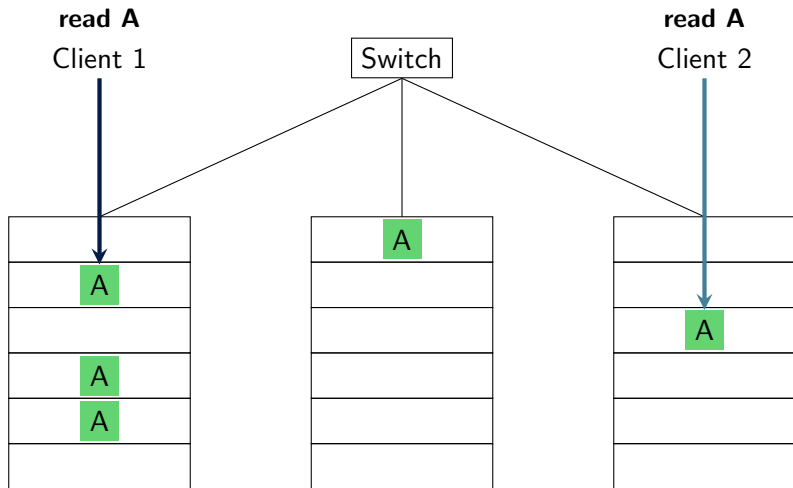
# Replication



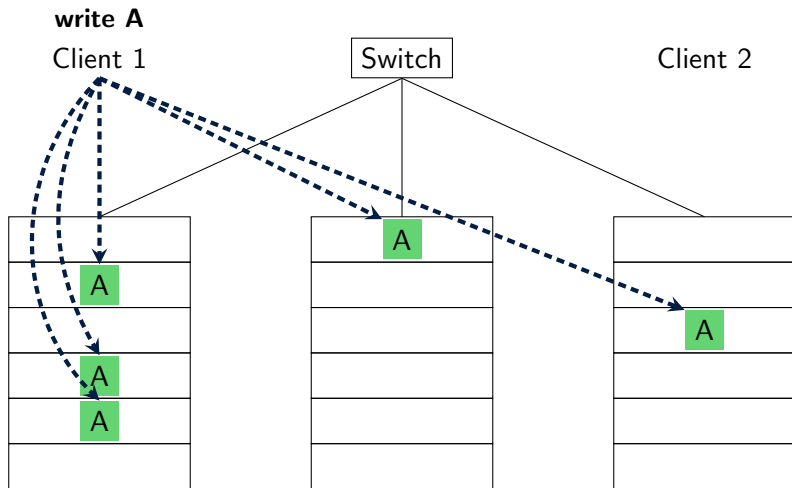
# Replication



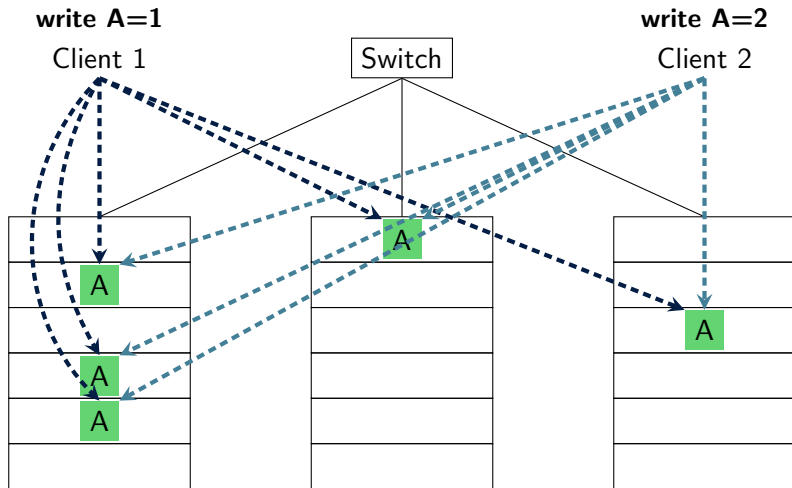
# Replication



# Replication

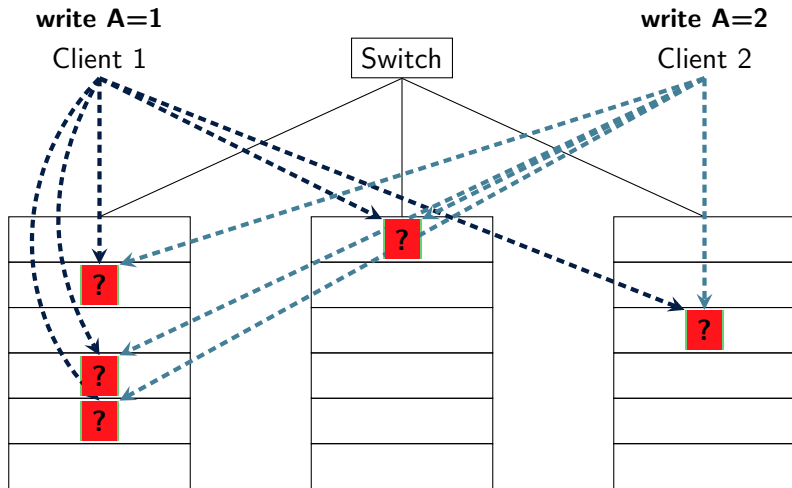


# Replication





# Replication



# Partitioning

## Sharding

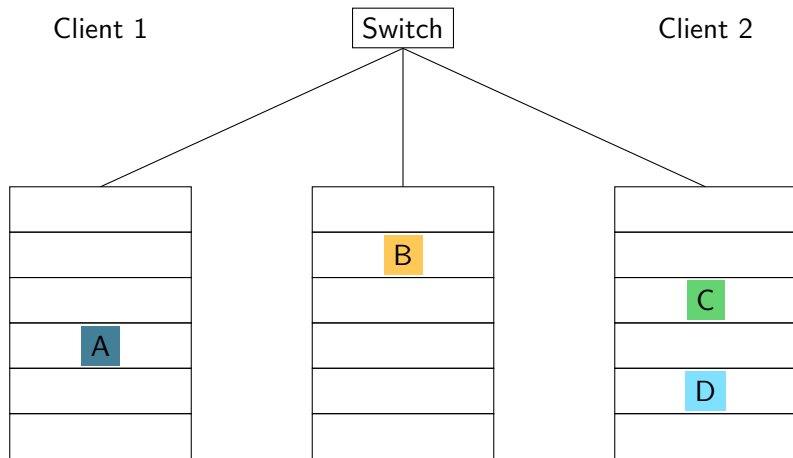
### Purposes

- Performance
  - ▶ Distributing the load over several nodes

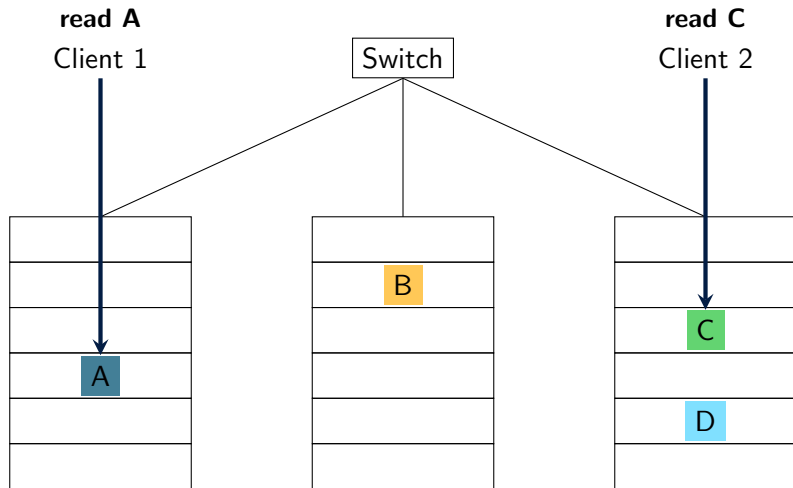
### Challenges

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

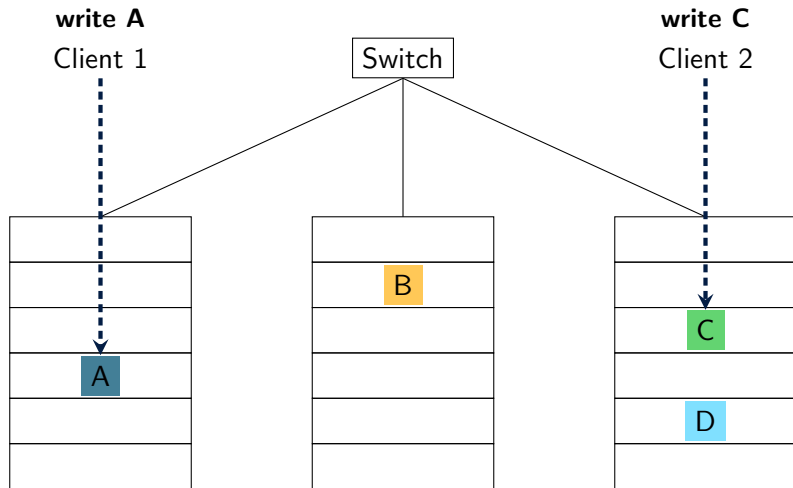
# Partitioning



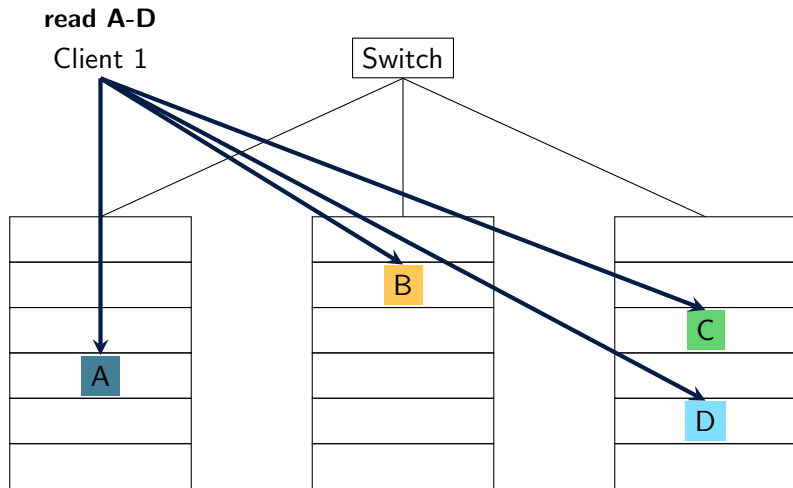
# Partitioning



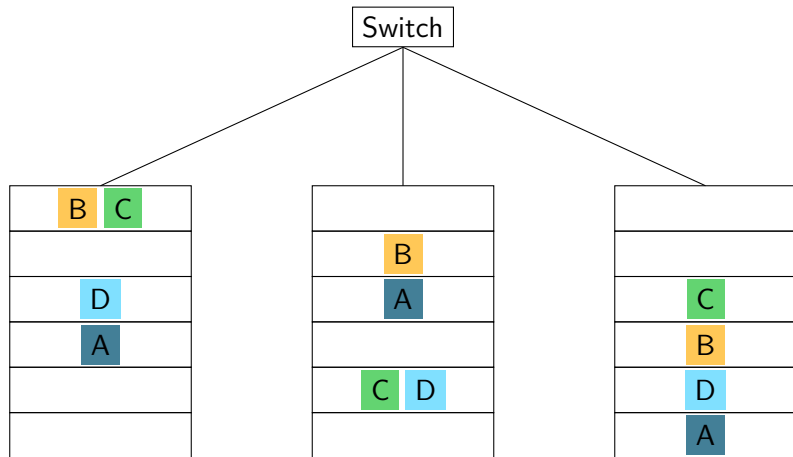
# Partitioning



# Partitioning



# Partitioning + Replication



# More references

## Mandatory reading

- *Big data and its technical challenges*, by Jagadish et al, CACM 2014.

## Suggested reading

- Chapter 1 of *Designing Data-Intensive Applications* by Martin Kleppmann
- The Netflix Simian Army<sup>1</sup>

---

<sup>1</sup><https://medium.com/netflix-techblog/the-netflix-simian-army-16e57fbab116>