

# Distributed Data Processing Environments

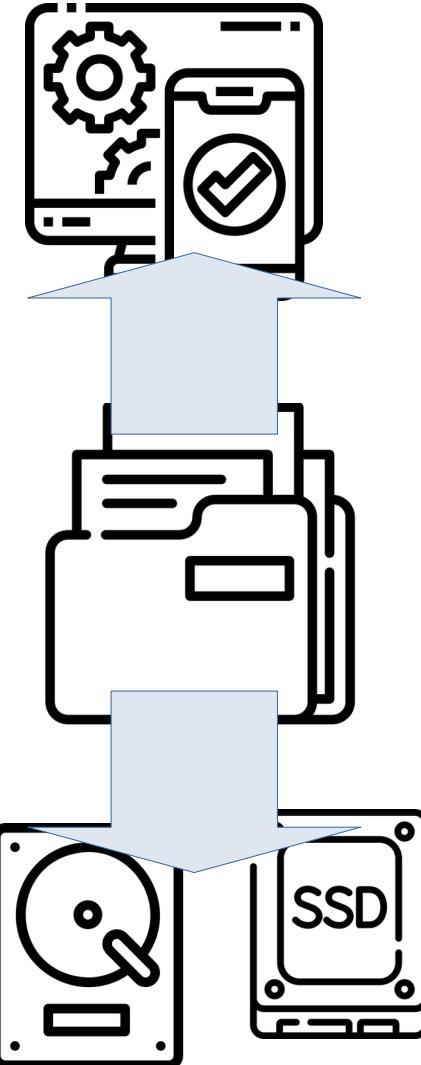
José Orlando Pereira

Departamento de Informática  
Universidade do Minho



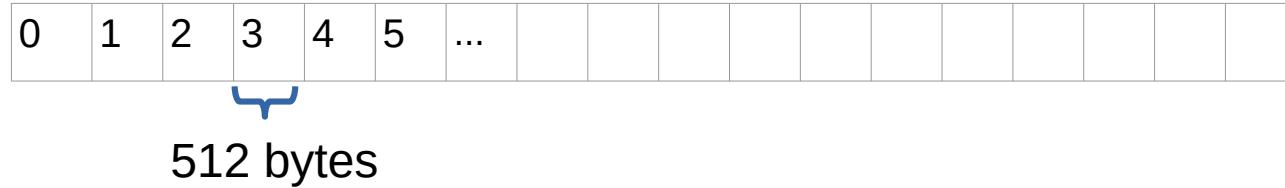
# Storage stack

- What is stored in files?
- How are files stored?



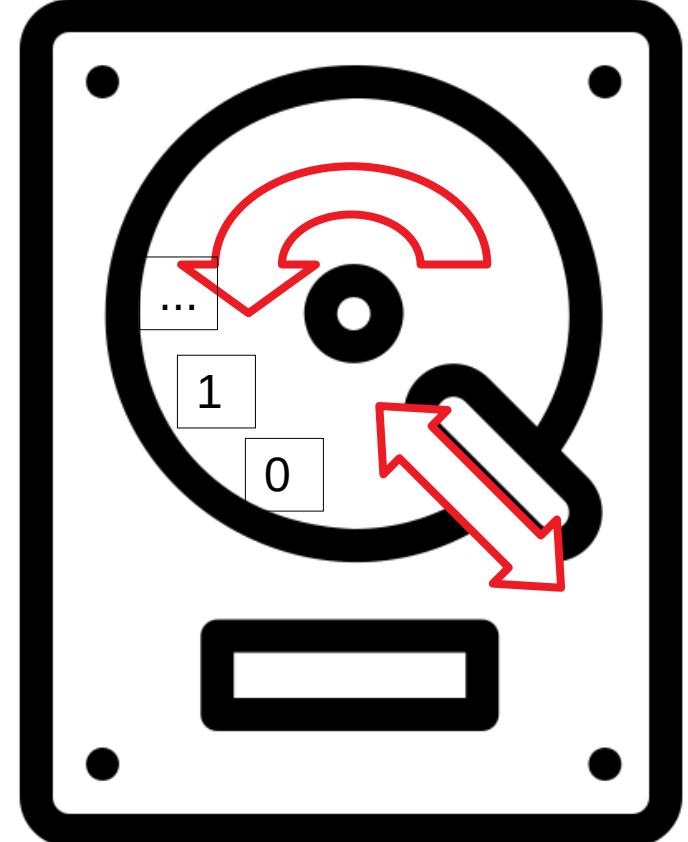
# Logical disk

- Sequence of fixed size sectors
  - e.g. 512 bytes
- Available operations:
  - Read sectors(s)
  - Write sectors(s)



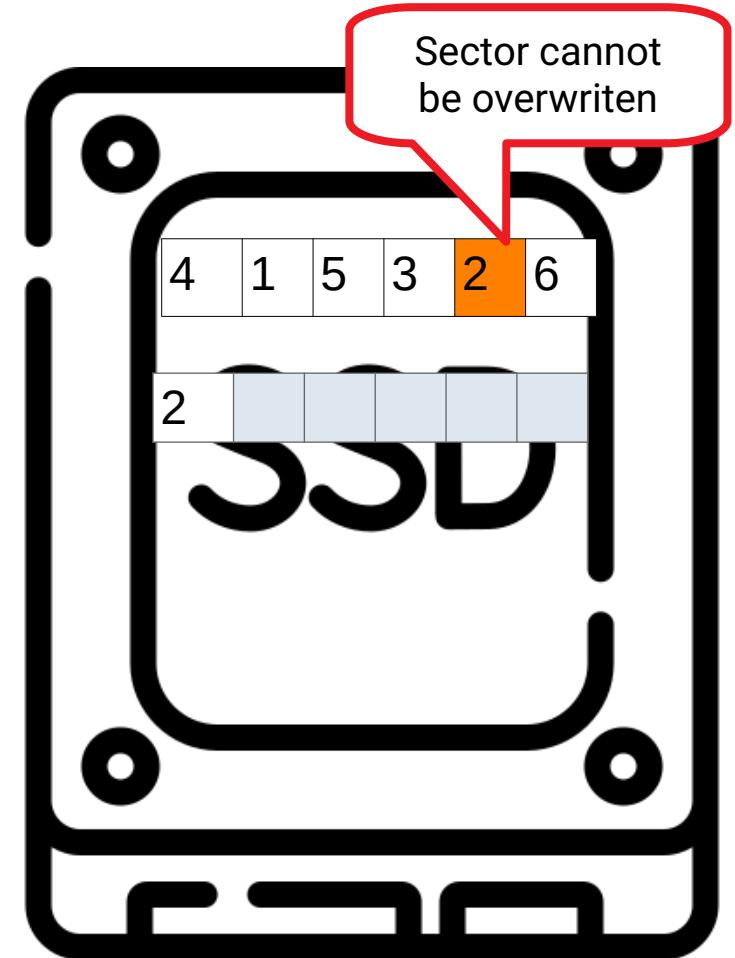
# Hard disks (HD)

- Stack of multiple disk platters
- Mechanically seek a sector by:
  - Moving heads
  - Waiting for rotation until desired sector
- Read/write contiguous sectors in a single rotation
- Cost and performance:
  - Low cost
  - High latency for random access
  - Higher heat generation and sensitivity
- Typical failure modes:
  - Individual sectors
  - Mechanical failure



# Solid state disks (SSD)

- Collection of FLASH memory chips
- Cannot overwrite single sector:
  - Erase a full block
  - Relocate sectors to an empty block on write/overwrite
- Read directly
- Benefits from TRIM operation:
  - Proactively erase blocks that are unused before they are overwritten
- Performance and host:
  - Low latency and high throughput
  - Higher cost and lower capacity
- Failure modes:
  - FLASH wear

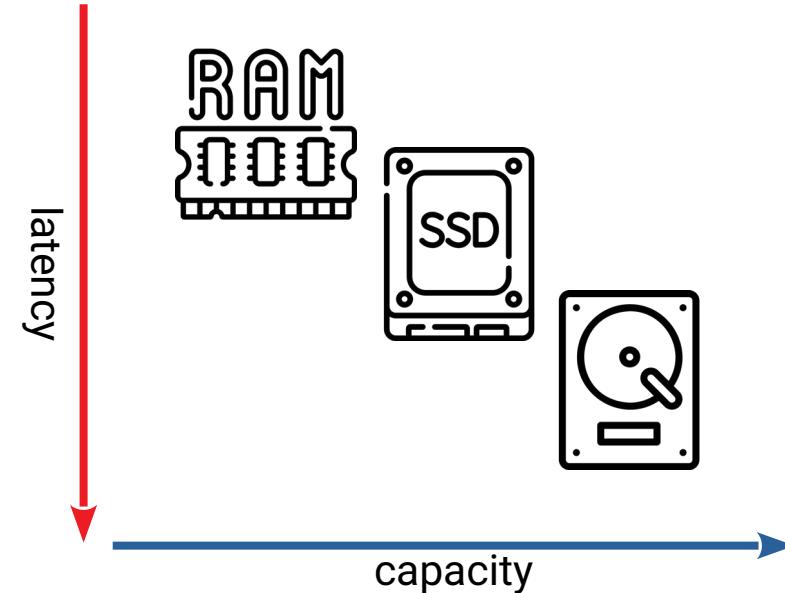


# Challenges

- Capacity
  - Not enough
  - Too much
  - Changing demand
- Performance
  - High latency
  - Low throughput
- Reliability

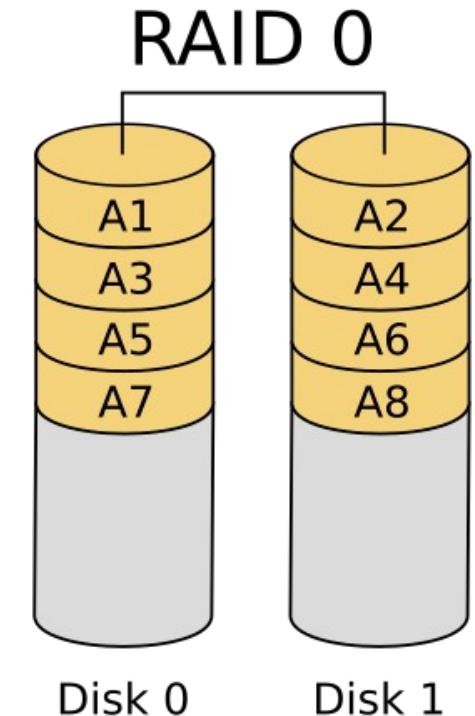
# Caching and tiering

- Keep a subset of data in a smaller but high performance storage medium
- Requires access locality: revisiting recently used data
- Capacity 
- Reliability 
- Latency 
- Throughput 



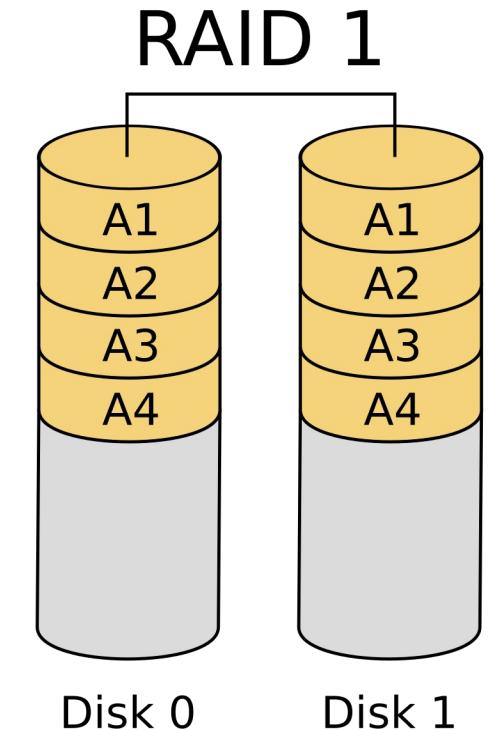
# RAID

- **Redundant Array of Inexpensive Disks**
- Level 0: Interleave sectors over more than one disk
- Capacity = 
- Reliability 
- Latency 
- Throughput 



# RAID

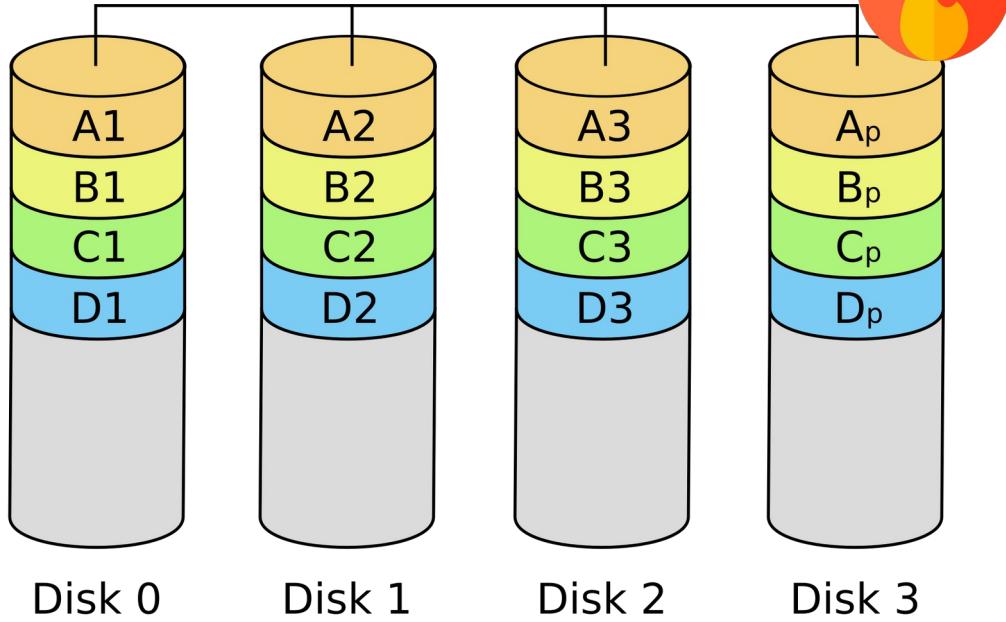
- Level 1: Duplicate sectors over more than one disk
- Capacity 
- Reliability 
- Latency 
- Throughput
  - Reading 
  - Writing 



# RAID

- Level 4: Store parity in a dedicated disk
- Capacity 
- Reliability 
- Latency 
- Throughput
  - Reading 
  - Writing 
    - Must read others in same stripe
    - Always writes to the same disk

## RAID 4

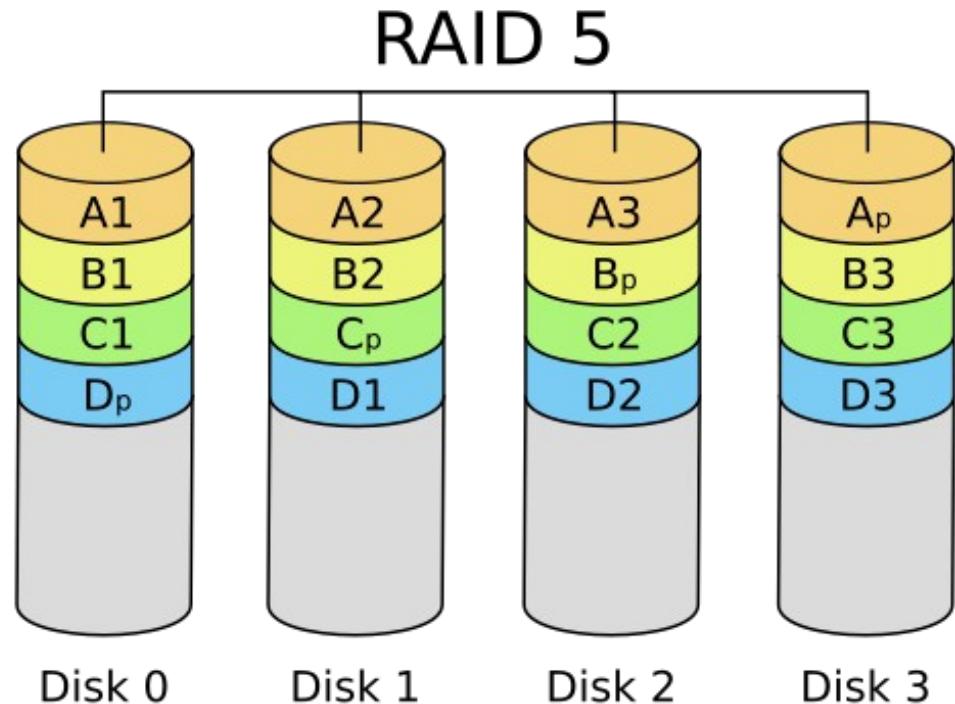


When writing:  $A_p = A_1 + A_2 + A_3$   
When reading:  
- normally, read  $A_1$   
- upon failure:  $A_1 = A_p - (A_2 + A_3)$

# RAID

- Level 5: Interleave parity in all disks

- Capacity 
- Reliability 
- Latency 
- Throughput
  - Reading 
  - Writing  



With effective caching, as no single disk is always updated

# Volume management

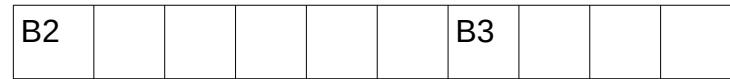
- Arbitrarily map physical disk extents to logical disks (volumes)
  - Allow dynamic reconfiguration

Logical disks

A	0	1	2	3	4	5
---	---	---	---	---	---	---

B	0	1	2	3
---	---	---	---	---

C	0	1	2	3	4
---	---	---	---	---	---



Physical disks

# Volume management

- Can be provided over the network
  - Access “slices” of a large disk stored in a remote server
- Often combined with caching, tiering, and RAID
- Provided by cloud IaaS

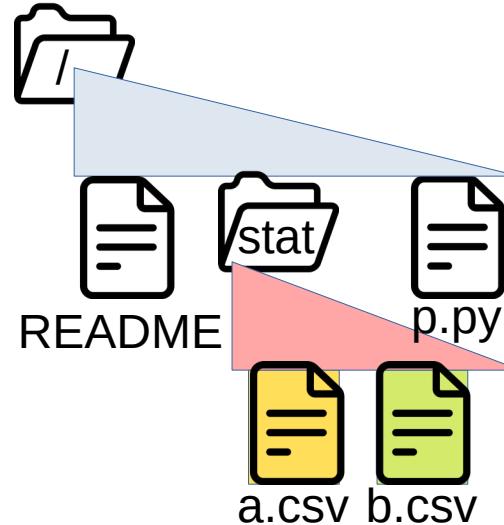


# Summary

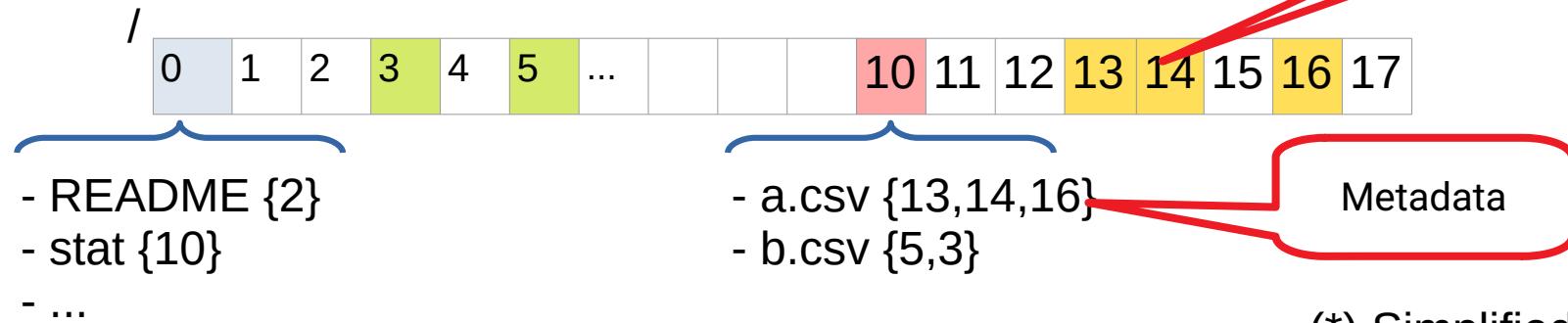
- Main challenges solved by arbitrary combinations of:
  - Caching
  - RAID levels
  - Volume management
- Problem:
  - Sequences of fixed size blocks are not a good abstraction for applications and users

# File systems

Logical organization  
of data



Physical layout in filesystem(\*)



(\*) Simplified view.

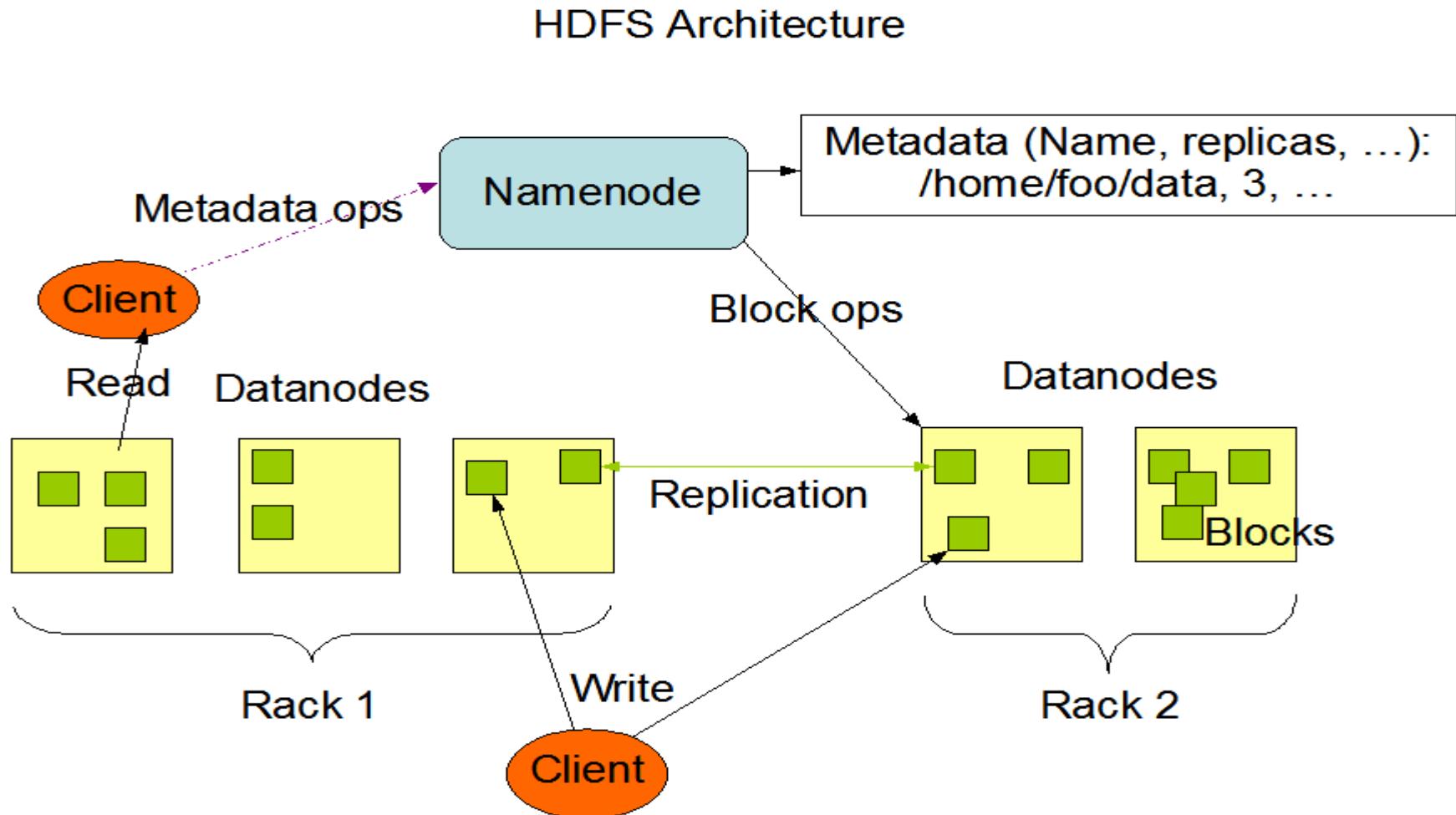
# File systems

- Additional functionality:
  - Access control, compression, snapshotting, ...
- Sometimes include volume management functionality (zfs, btrfs, ...)
- Can be “mounted”:
  - The folder hierarchy of a file system stored in a separate can be made to appear under a folder in a different file system

# Scale

- What if data does not fit in disks attached to a single server?
  - Capacity
  - Throughput
- Design principles and goals for cloud storage:
  - Simplified abstraction (no hierarchy, append only, ...)
  - Separation of data and metadata
  - Cope with new failure modes: entire servers, DCs, ...
  - Decentralization and parallelism

# Hadoop Distributed File System



# Cloud object storage

<b>Capacity and throughput</b>	Amazon S3 holds more than <b>280 trillion objects</b> and averages over <b>100 million requests per second</b>
<b>Events</b>	Every day, Amazon S3 sends over <b>125 billion event notifications</b> to serverless applications
<b>Replication</b>	Customers use Amazon S3 Replication to <b>move more than 100 PB</b> of data per week
<b>Cold Storage Retrieval</b>	Every day, customers <b>restore more than 1PB</b> from the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes
<b>Data Integrity Checks</b>	Amazon S3 performs over <b>4 billion checksum computations per second</b>
<b>Cost Optimization</b>	On average, customers using Amazon S3 Storage Lens advanced metrics and recommendations have obtained <b>cost savings 6x greater</b> than the Storage Lens cost in the first six months of using it.
<b>Flexibility</b>	<b>Hundreds of thousands of data lakes</b> are built on Amazon S3

Fonte: "Building and operating a pretty big storage system called S3," All Things Distributed, July 27, 2023.

<https://www.allthingsdistributed.com/2023/07/building-and-operating-a-pretty-big-storage-system.html>

# Conclusion

- Cloud services encapsulate storage management concerns in two main abstractions
  - Networked volumes:
    - for program and working area storage
  - Object storage:
    - for long term data archival (*data lakes*)