CSC 2024 Data Technologies

Introduction and Exercises

Dr. **Andreas-Joachim Peters**
CERN IT-SD

CERN

Why are Data Technologies important?

CERN

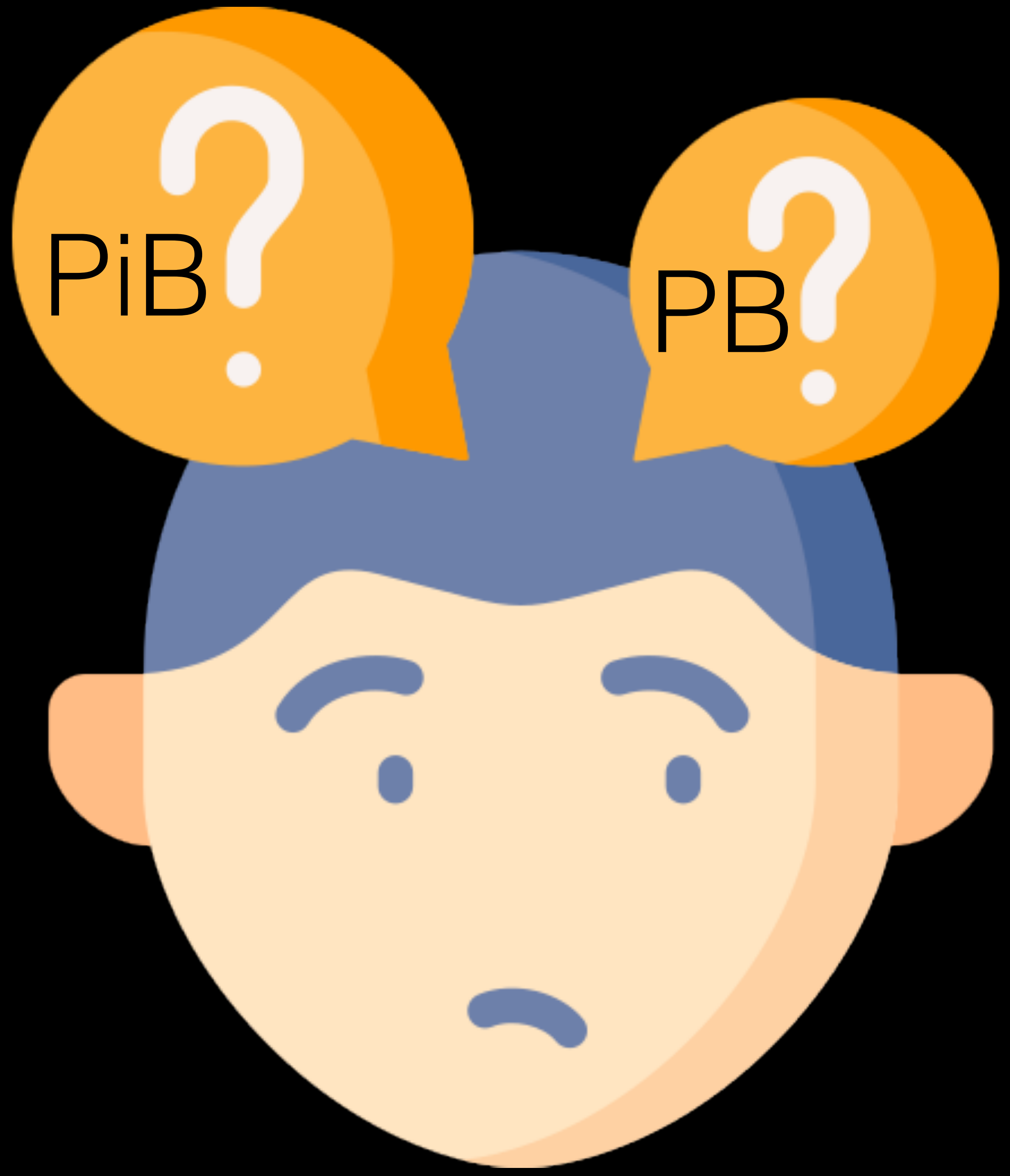
Accélérateur de science

.. or why is that part of this school?

Dataflow & Storage
ALICE LHC Experiment
CERN Computer Center
CERN Experimental Site
Worldwide LHC
Computing GRID
10-40 GB/s
shared
1 PB
TAPE
EOS
SDD
10-40 GB/s
150 PB
Learn core technology in
Exercise 1
10+ GB
50-200 GB/s
Processing
Analysis
CERN
Cloud
Learn core technology in
Exercise 2
EOS
48h Storage
Fallback Buffer
Disk
14 PB
100-150 GB/s
100-250 GB/s
260 Nodes
+
2k x GPUs
5 Tbit/s
Experiment
28 Tbit/s
3h Storage
Realtime Buffer
Flash
100-250 GB/s
SSDs
1 PB

SI vs. IEC units …

Storage vendors always use SI units



Prefixes for binary multiples

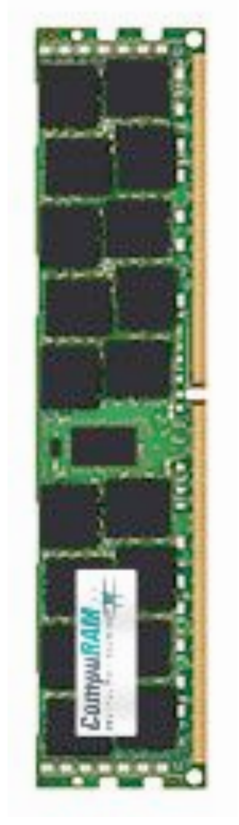
Factor	Name	Symbol	Origin	Derivation
2^{10}	kibi	Ki	kilobinary: $(2^{10})^1$	kilo: $(10^3)^1$
2^{20}	mebi	Mi	megabinary: $(2^{10})^2$	mega: $(10^3)^2$
2^{30}	gibi	Gi	gigabinary: $(2^{10})^3$	giga: $(10^3)^3$
2^{40}	tebi	Ti	terabinary: $(2^{10})^4$	tera: $(10^3)^4$
2^{50}	pebi	Pi	petabinary: $(2^{10})^5$	peta: $(10^3)^5$
2^{60}	exbi	Ei	exabinary: $(2^{10})^6$	exa: $(10^3)^6$

Examples and comparisons with SI prefixes

one **kibibit**	1 Kibit = 2^{10} bit = **1024 bit**
one **kilobit**	1 kbit = 10^3 bit = **1000 bit**
one **byte**	1 B = 2^3 bit = **8 bit**
one **mebibyte**	1 MiB = 2^{20} B = **1 048 576 B**
one **megabyte**	1 MB = 10^6 B = **1 000 000 B**
one **gibibyte**	1 GiB = 2^{30} B = **1 073 741 824 B**
one **gigabyte**	1 GB = 10^9 B = **1 000 000 000 B**

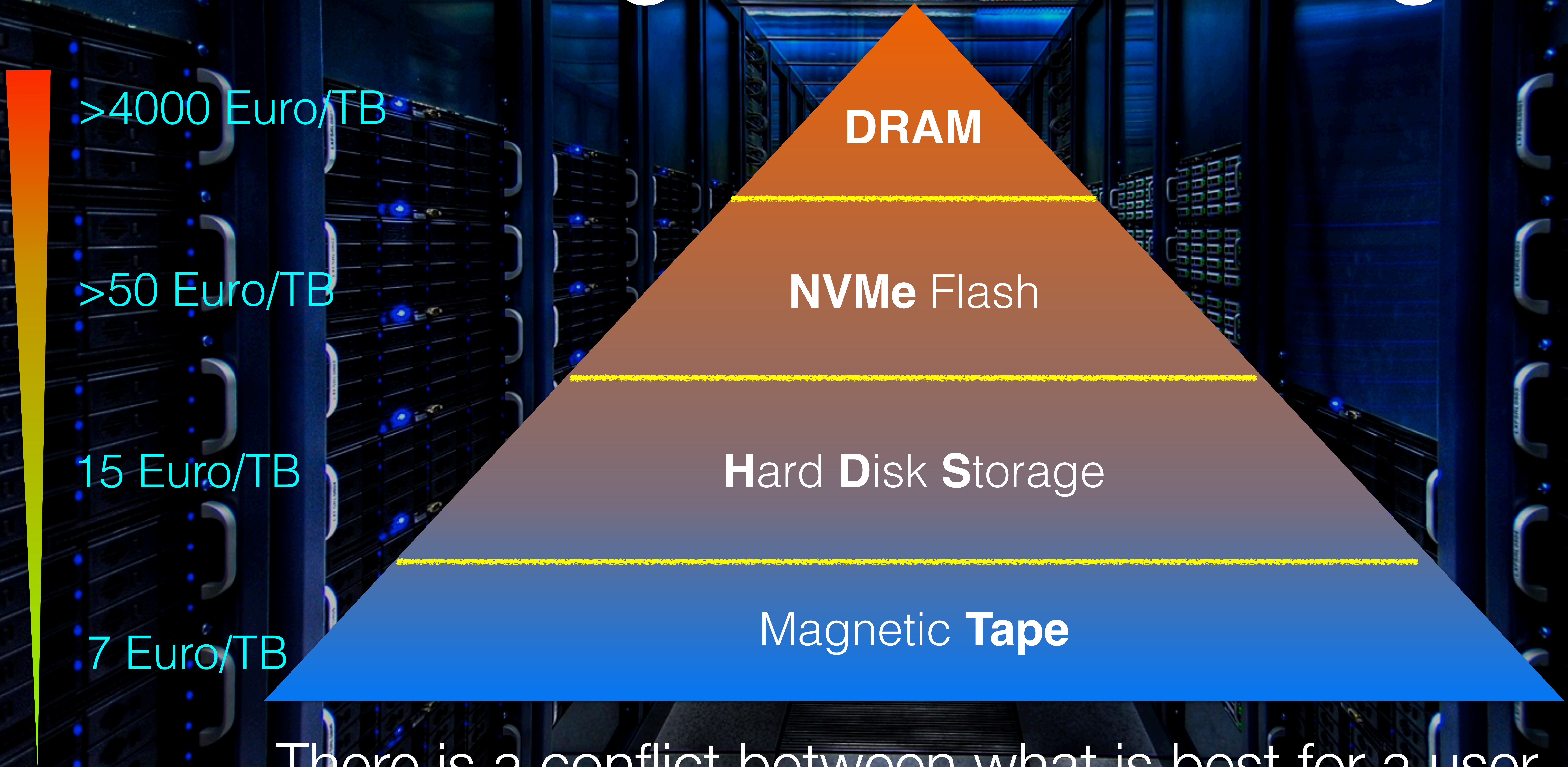
System of Units (SI)			Binary Numeral				% Difference
Factor	Name	Symbol	Factor	Name	Symbol	# of Bytes	
10^3	kilobyte	KB	2^{10}	kibibyte	KiB	1,024	2.4%
10^6	megabyte	MB	2^{20}	mebibyte	MiB	1,048,576	4.9%
10^9	gigabyte	GB	2^{30}	gibibyte	GiB	1,073,741,824	7.4%
10^{12}	terabyte	TB	2^{40}	tebibyte	TiB	1,099,511,627,776	10.0%
10^{15}	petabyte	PB	2^{50}	pebibyte	PiB	1,125,899,906,842,624	12.6%
10^{18}	exabyte	EB	2^{60}	exbibyte	EiB	1,152,921,504,606,846,976	15.3%
10^{21}	zettabyte	ZB	2^{70}	zebibyte	ZiB	1,180,591,620,717,411,303,424	18.1%
10^{24}	yottabyte	YB	2^{80}	yobibyte	YiB	1,208,925,819,614,629,174,706,176	20.9%

Storage Media Types in CERN CC



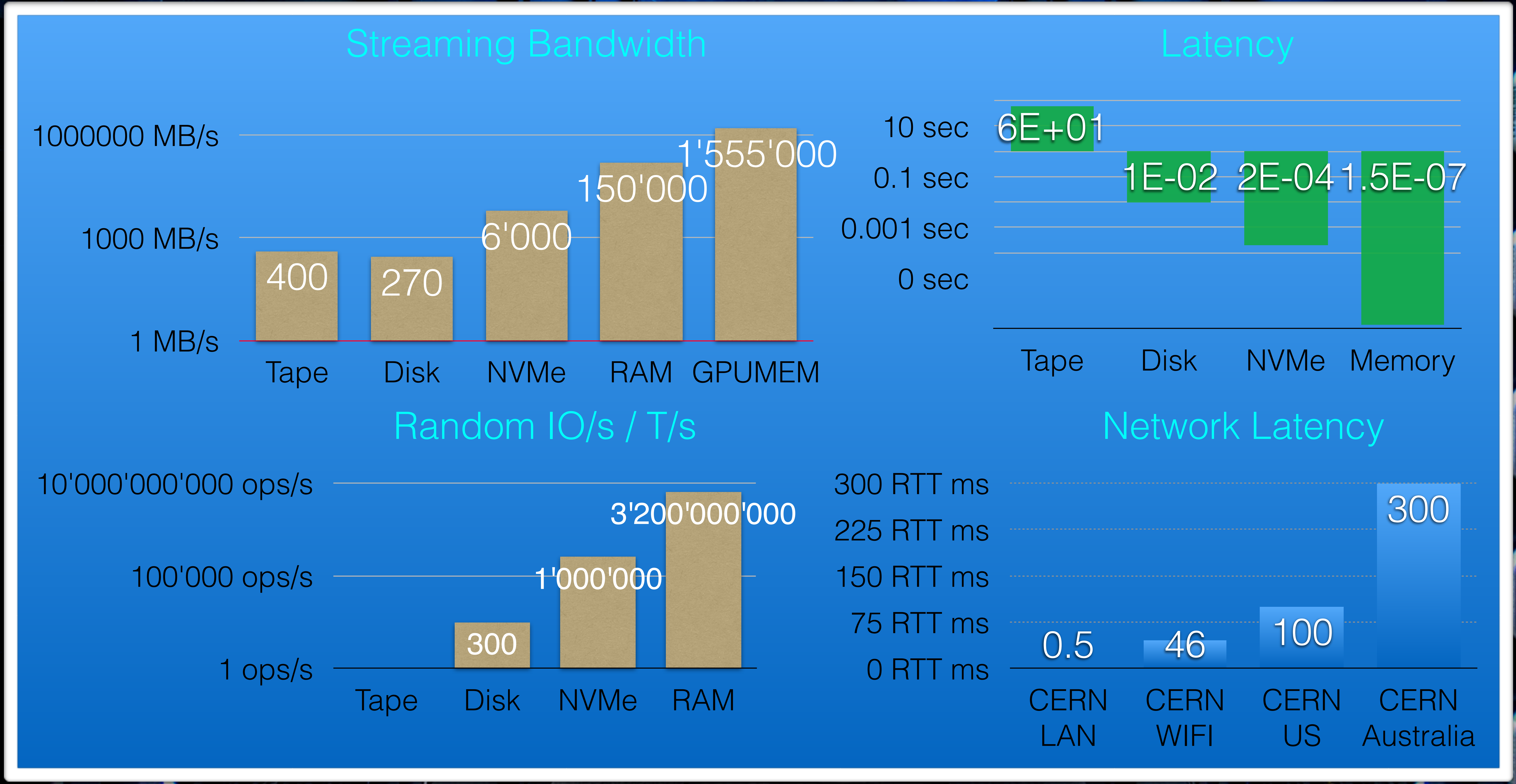
Memory	NVMe	Enterprise HDD	Tapes
		100k	**30k**
~PB	few PB	~1 EB	0.6 EB

Storage Media Pricing



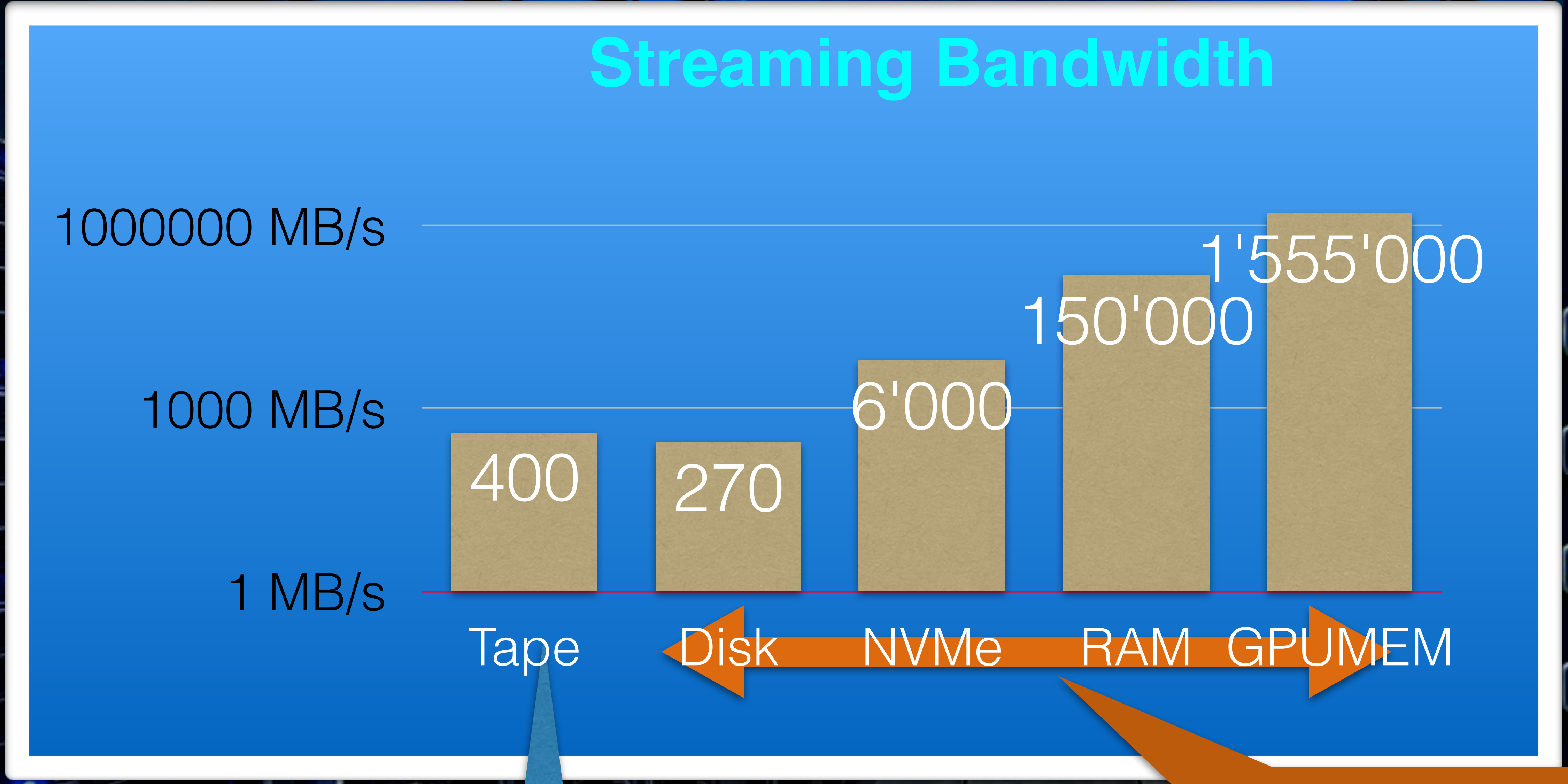
There is a conflict between what is best for a user and what it costs

Storage Media Characteristics



Disclaimer:
- numbers are indicative for enterprise devices
- not always symmetric for RO,WO, RW

Storage Media Characteristics

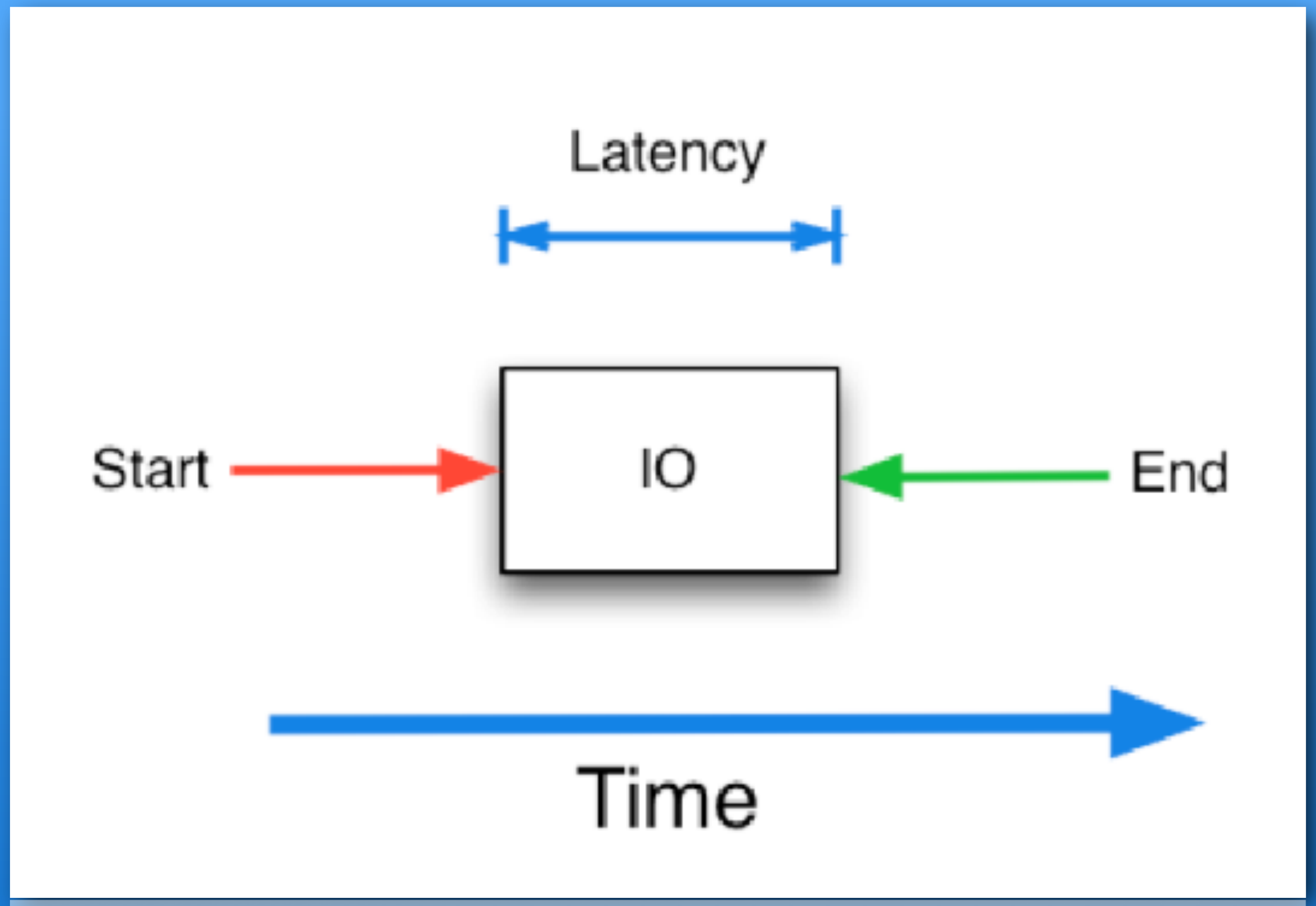


Tape Bandwidth is decoupled from media:
you need more bandwidth
you have to pay for more tape drives!
Tape volume is cheap - bandwidth is not!

Disk/Flash/Memory Bandwidth is
coupled to the media:
performance/capacity ratio:
you buy more space, you increase
your bandwidth to data …

Optimizations used in IO systems

IO Language



Bandwidth = IO volume / time

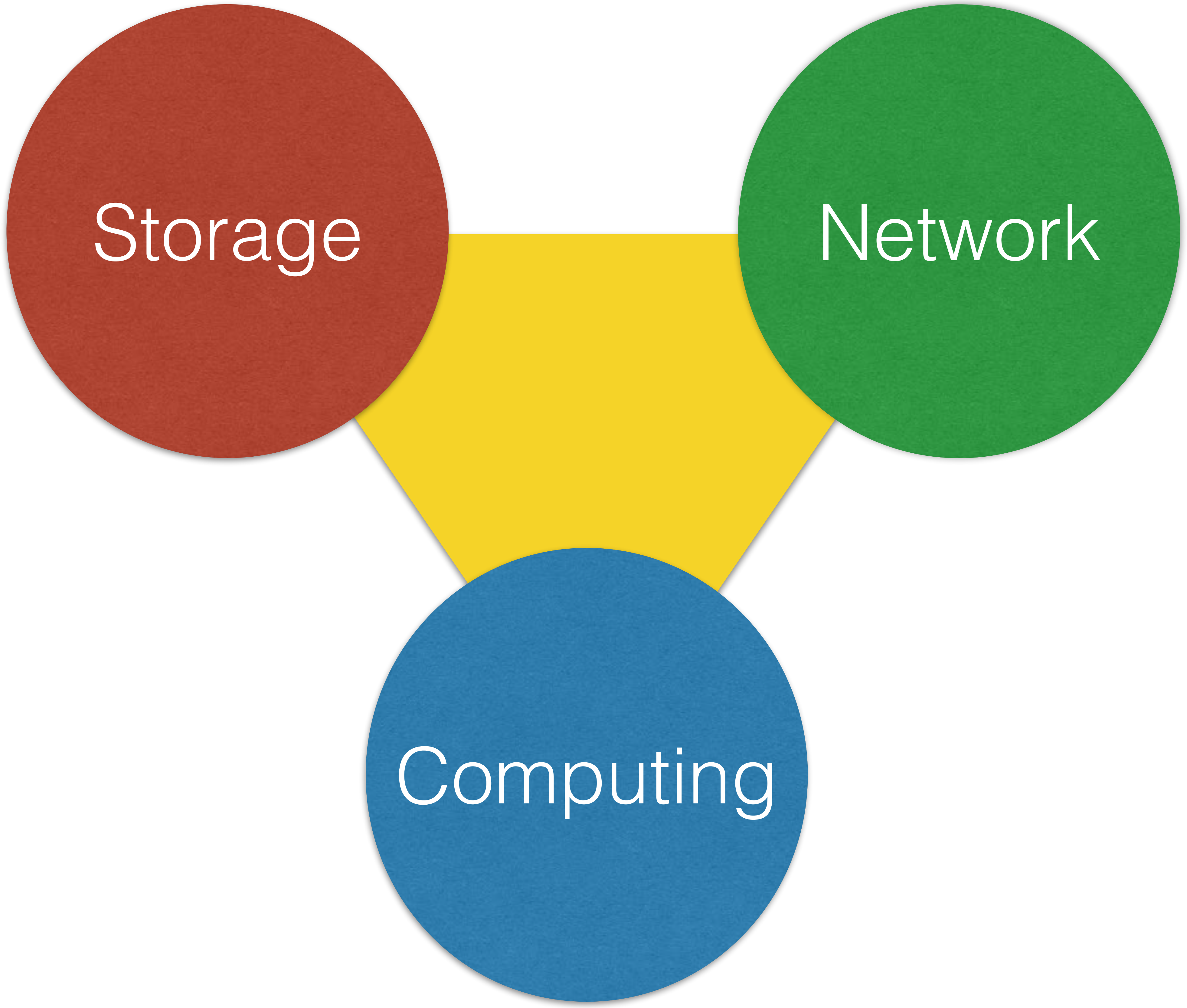
IOPS = IO operations / time

Latency = time / IO operation

Blocksize = payload / operation

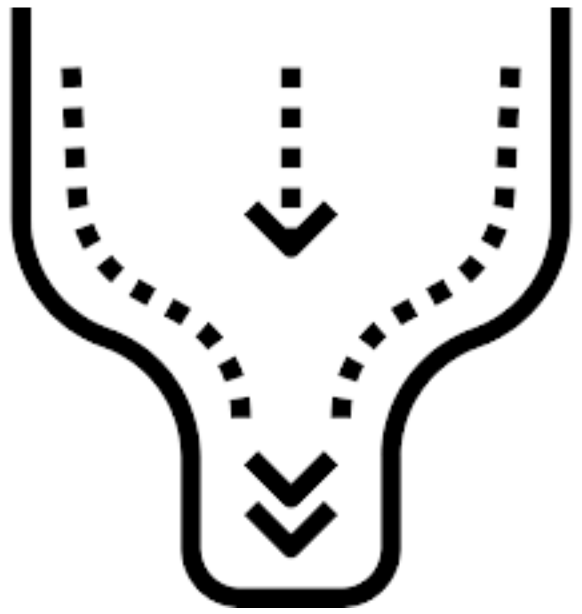
- Bandwidth
- IOPS
- Latency
- Blocksize

IO foundation



Building blocks for an IO system …

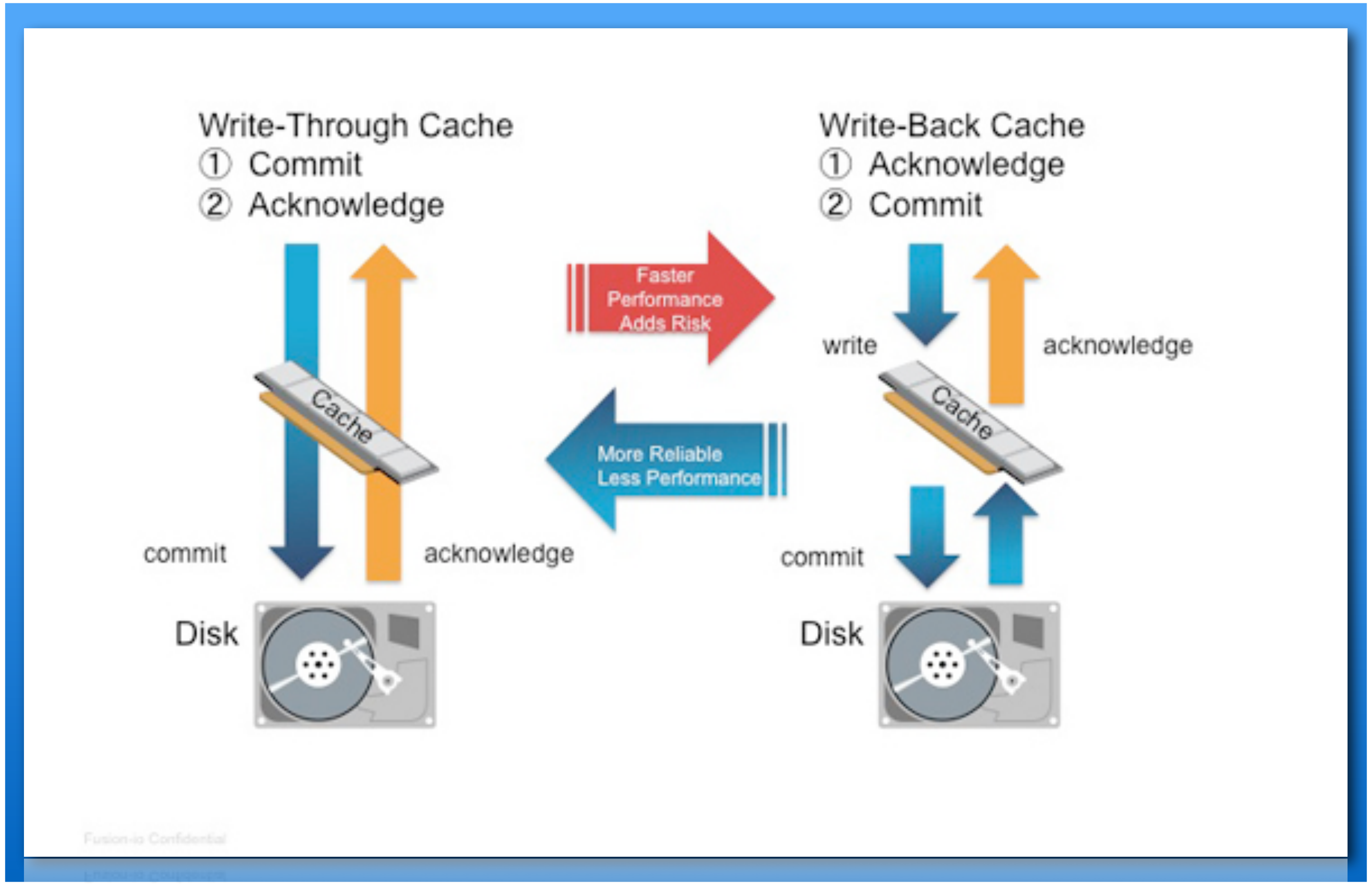
= Bottlenecks for an IO system …



Caching Strategies

Which strategy is best for low latency?

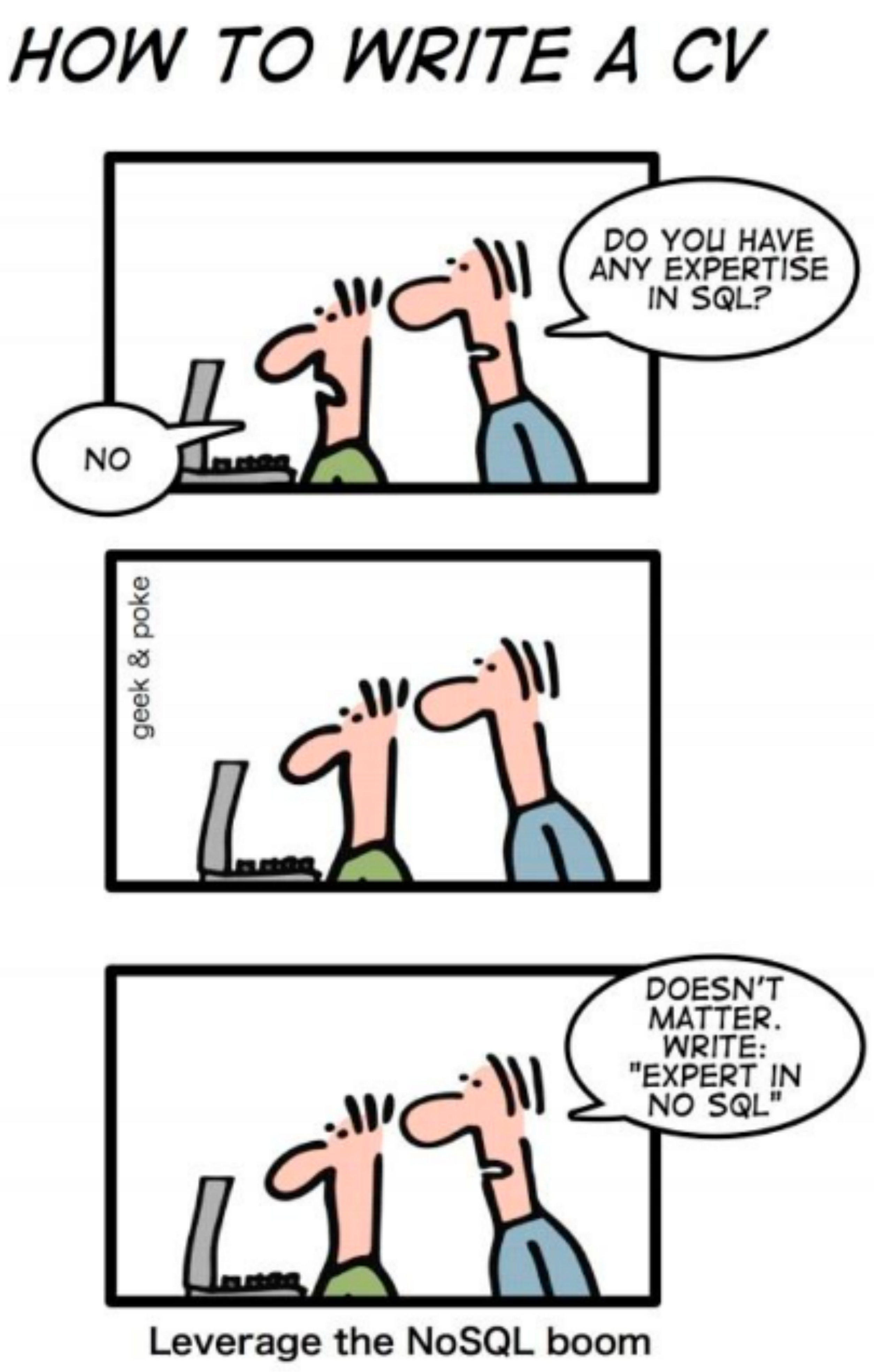
What is the latency difference when reading?



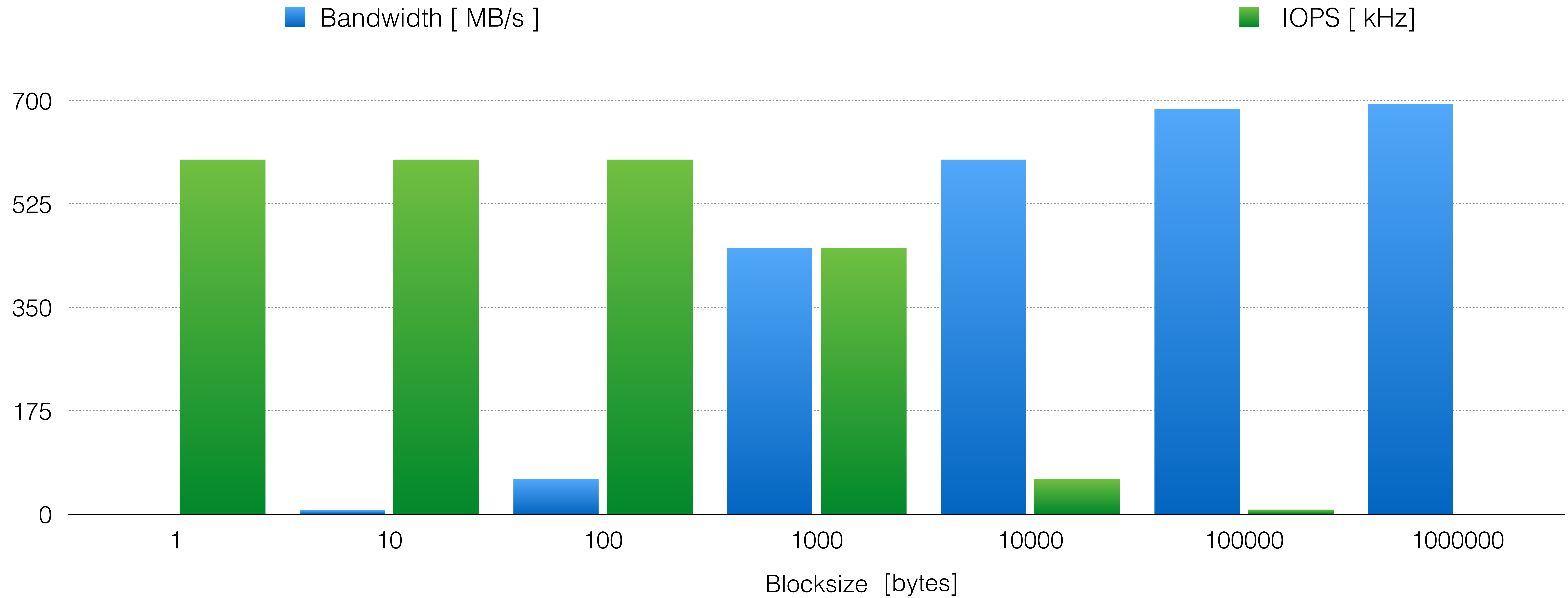
What is the danger when using a write-back cache?

Data Stores & APIs

- File Systems : *hierarchical namespace (tree structure)*
  POSIX `open, read, write, close file`
- Object Storage / NOSQL KV stores : *flat namespace*
  REST `get|put|delete|list object, sets, maps`
- Relational Databases
  SQL `select from, insert, delete from table`



Impact of **Blocksize** in IO systems



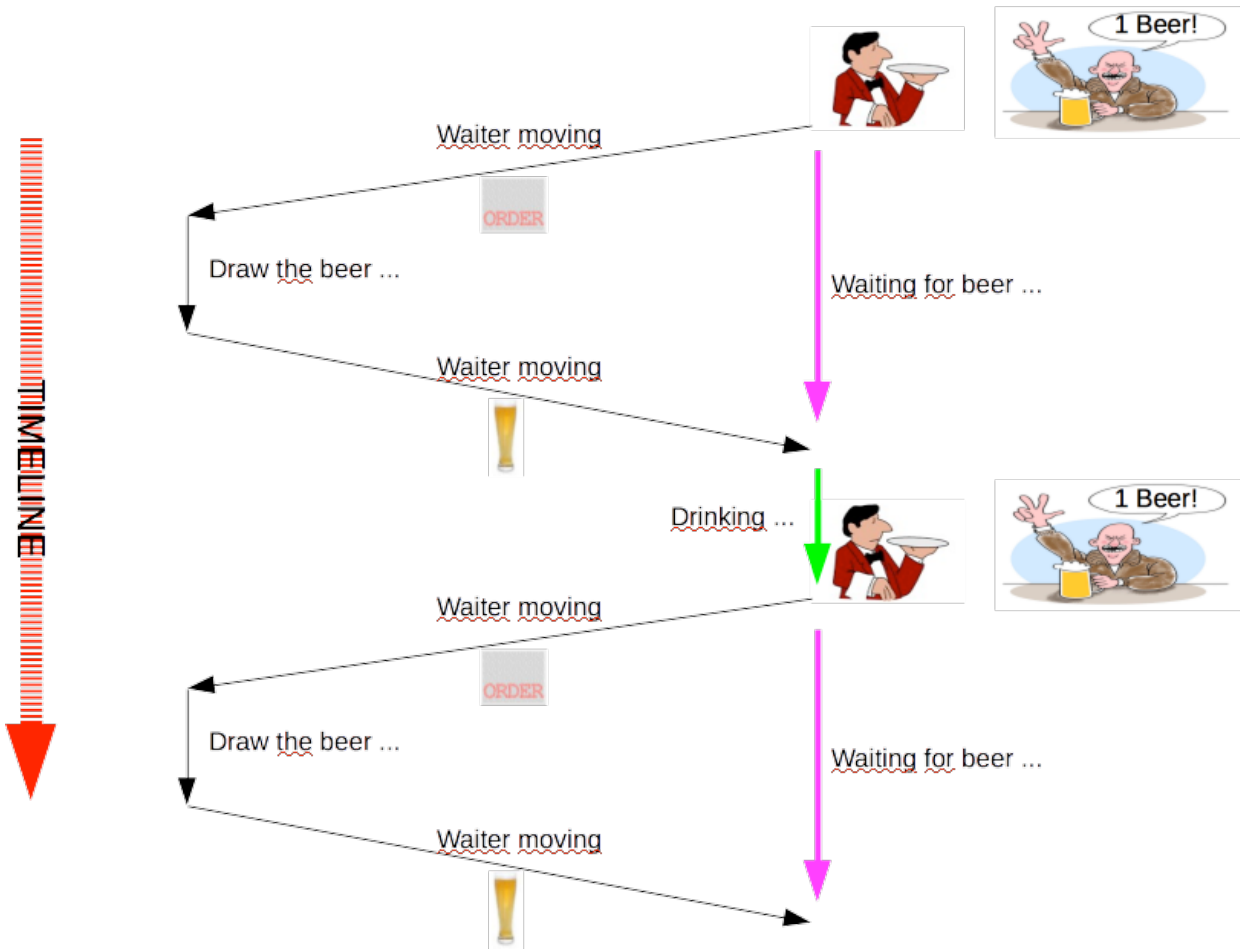
Which bandwidth bottleneck can you see?
Which IOPS bottleneck can you see?

Impact of **Latency**

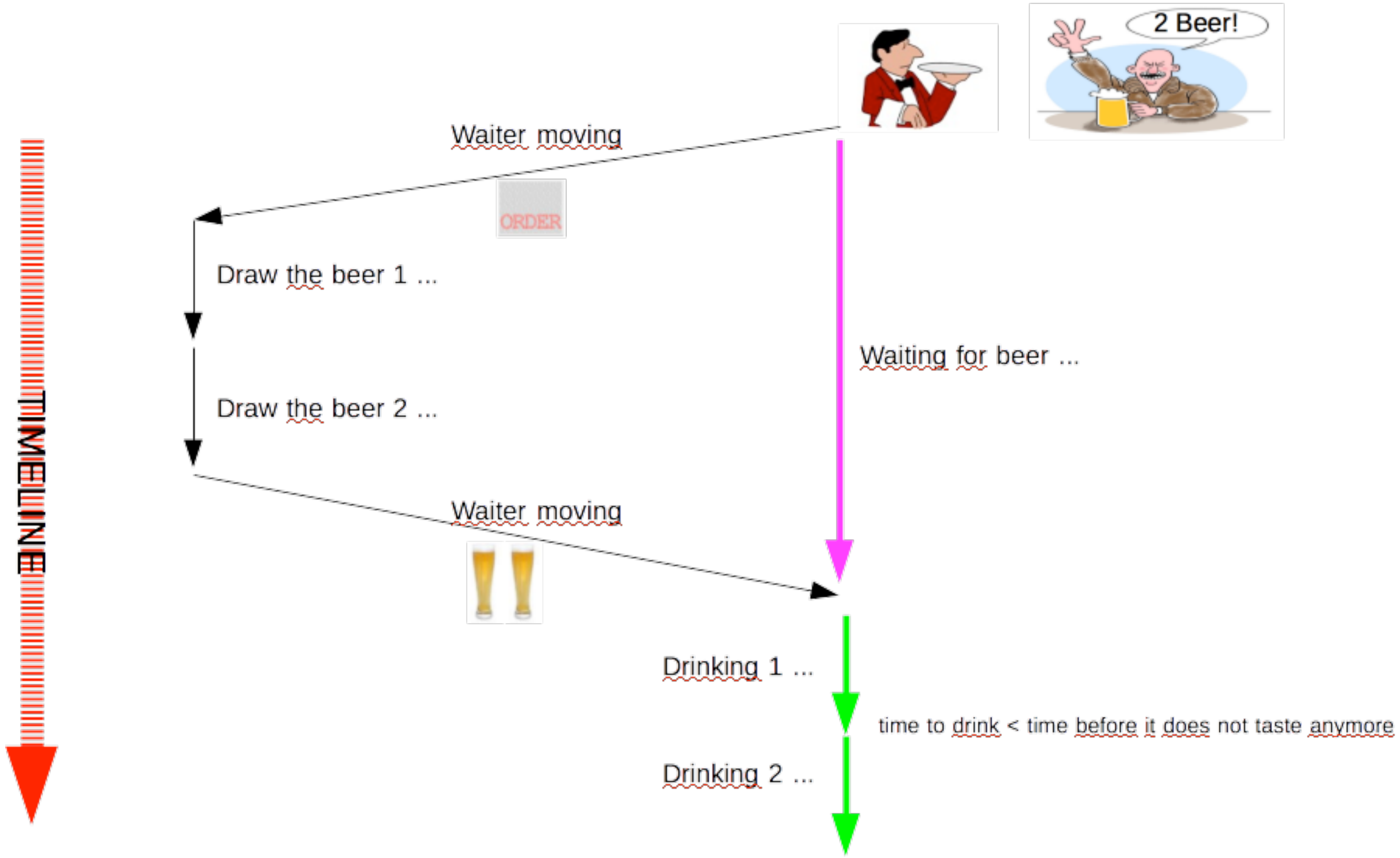
in analysis use cases

- **ROOT** as an example for an analysis application issues thousands of small read requests to iterate over data structures stored inside ROOT format files
  - if such an application reads files from a remote storage system latency has a big impact on the IO efficiency of the application:
    `100k x 10ms latency create 100s transport time` ☹️
  - ROOT implements various techniques to compensate latency

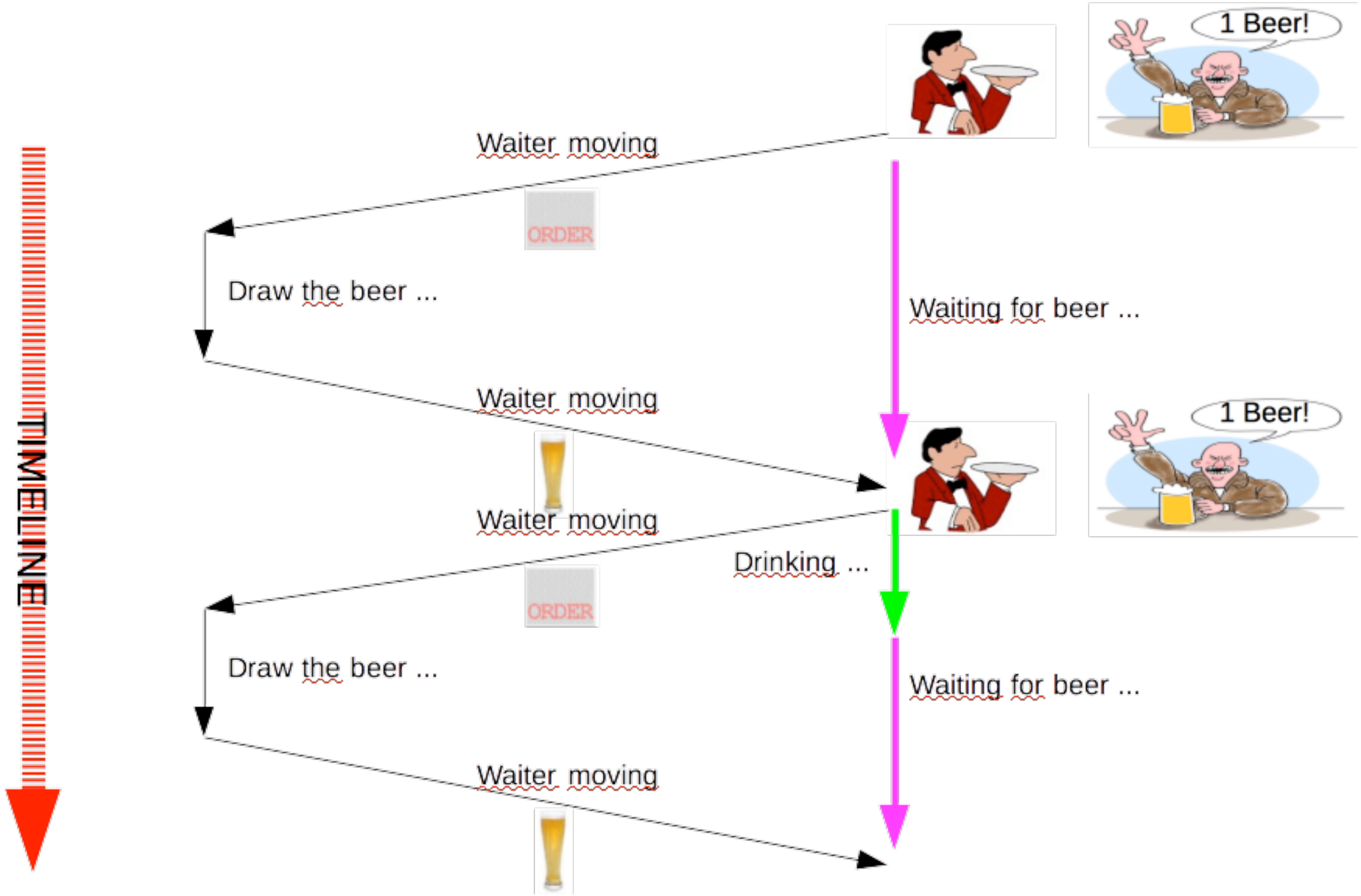
Analogy: ordering beer with a high-latency waiter - uncompensated



Analogy: ordering beer with a high-latency waiter - pre-fetching



Analogy: ordering beer with a high-latency waiter - asynchronous pre-fetching



Exercises Overview

0 IO Systems

- characteristics, measurement and debugging tools

1st hour today

https://cern.ch/cscdt0

1 Redundancy Technology

- **RAID** technology

2nd hour today

https://cern.ch/cscdt1

- **RAIN** / Erasure Encoded Storage Systems **EC**

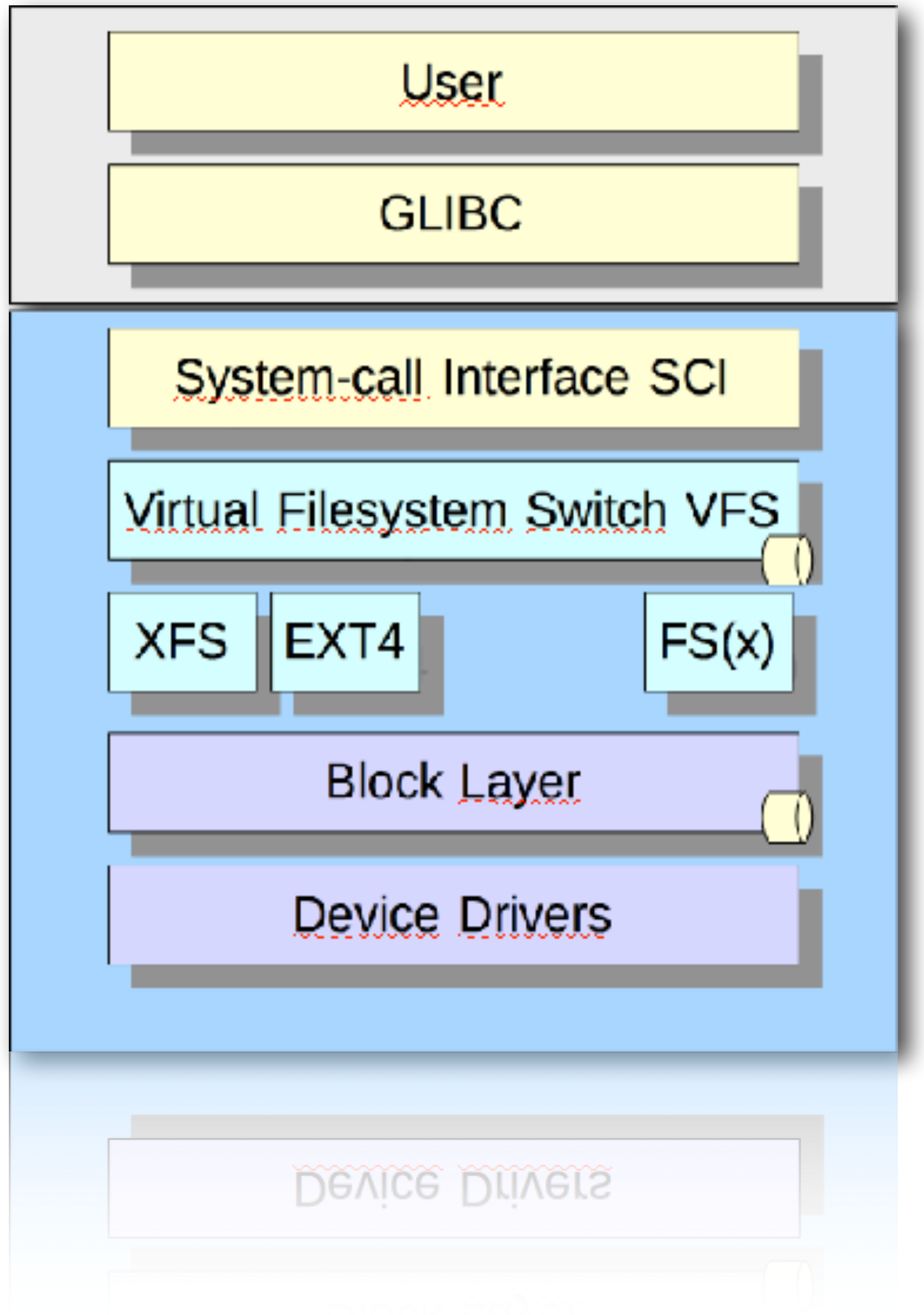
2 Cloud Storage Technology

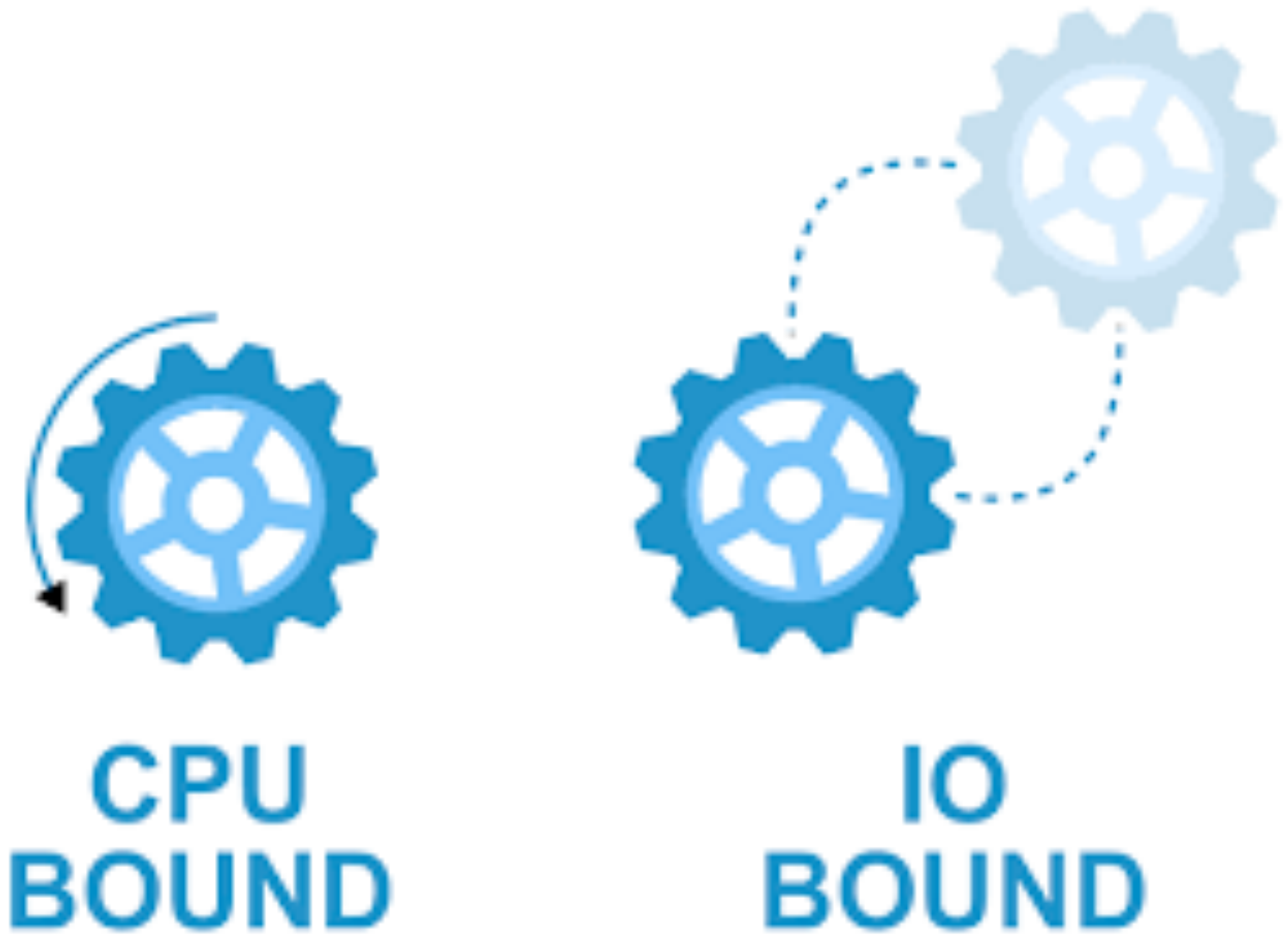
3rd + 4th hour tomorrow

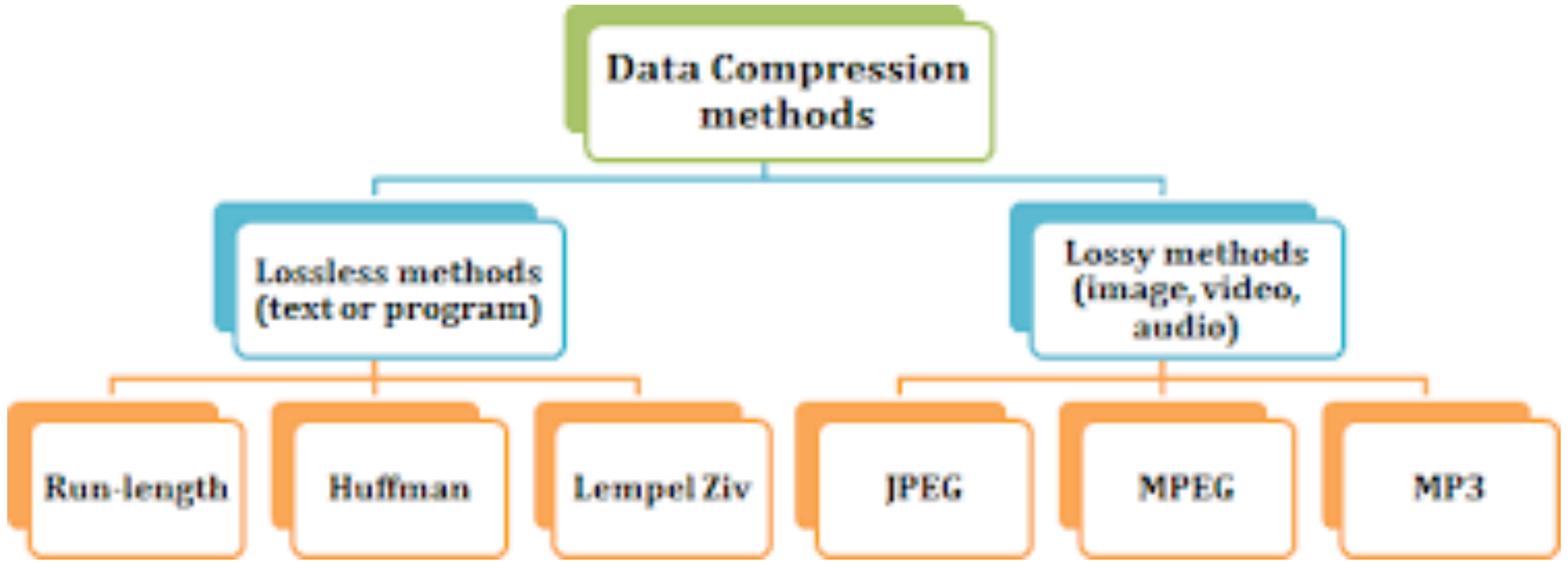
https://cern.ch/cscdt2

- **Scalability, Replication, Namespace, Placement toy MonteCarlo**

Tutorial - Exercise 0







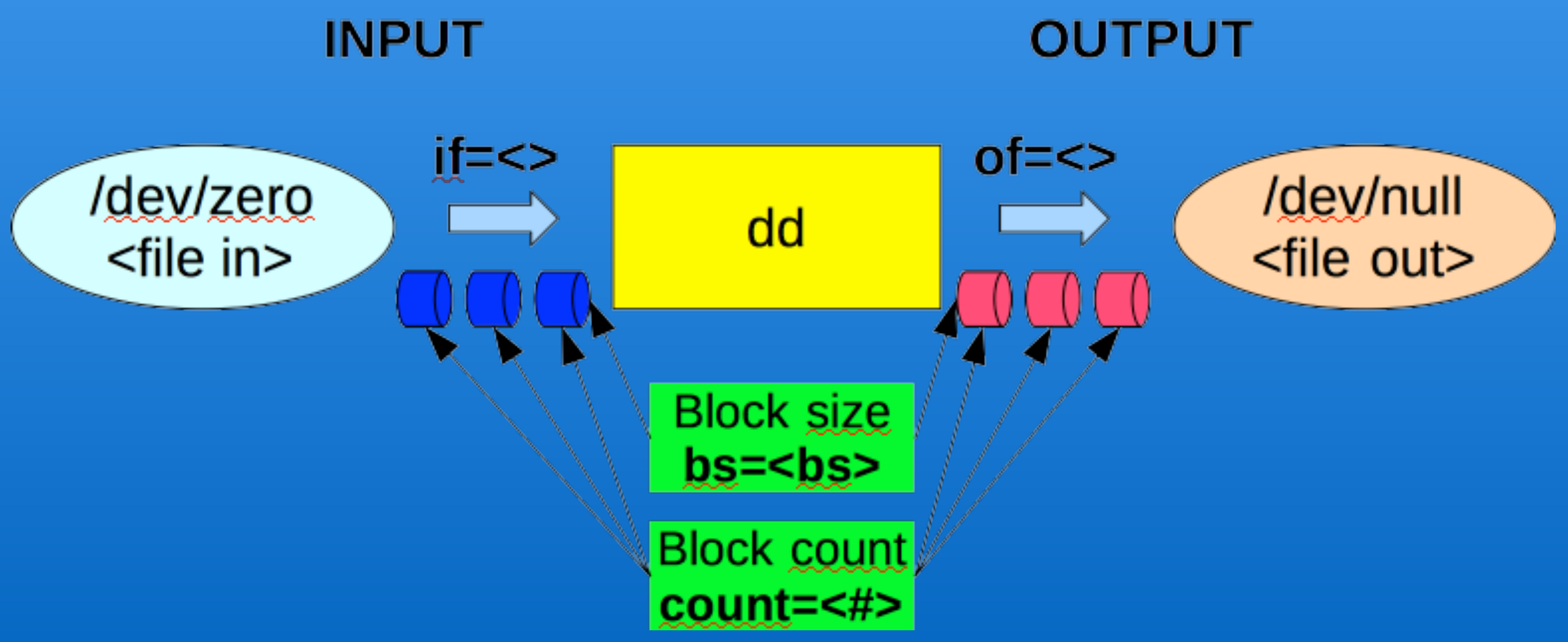
Useful Linux Command

Realtime, CPU, System Time Measurement

time <program>

Copy/Block IO tool

dd



Useful Linux Commands

- Trace system calls of a program

  `strace <program> [program args]`

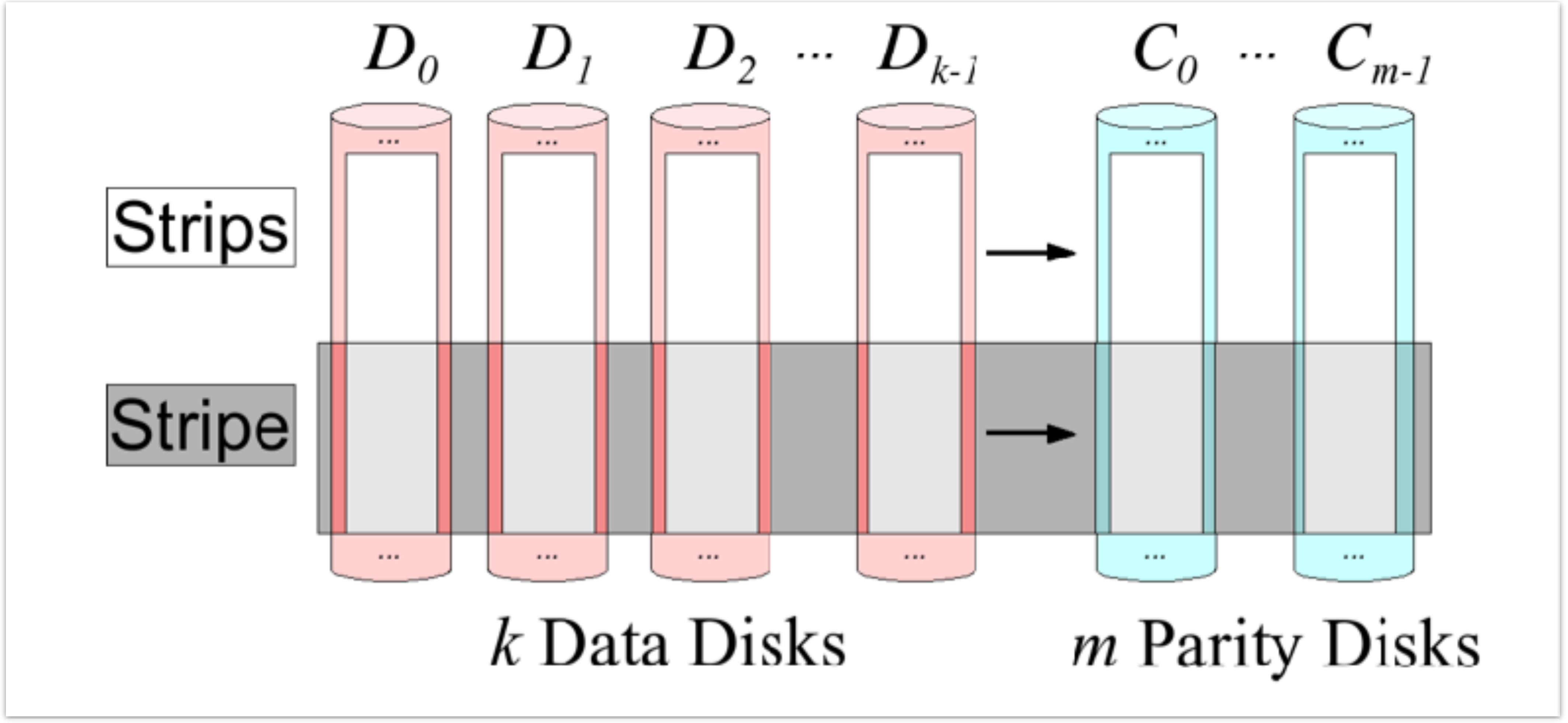
  `-c` count sys calls

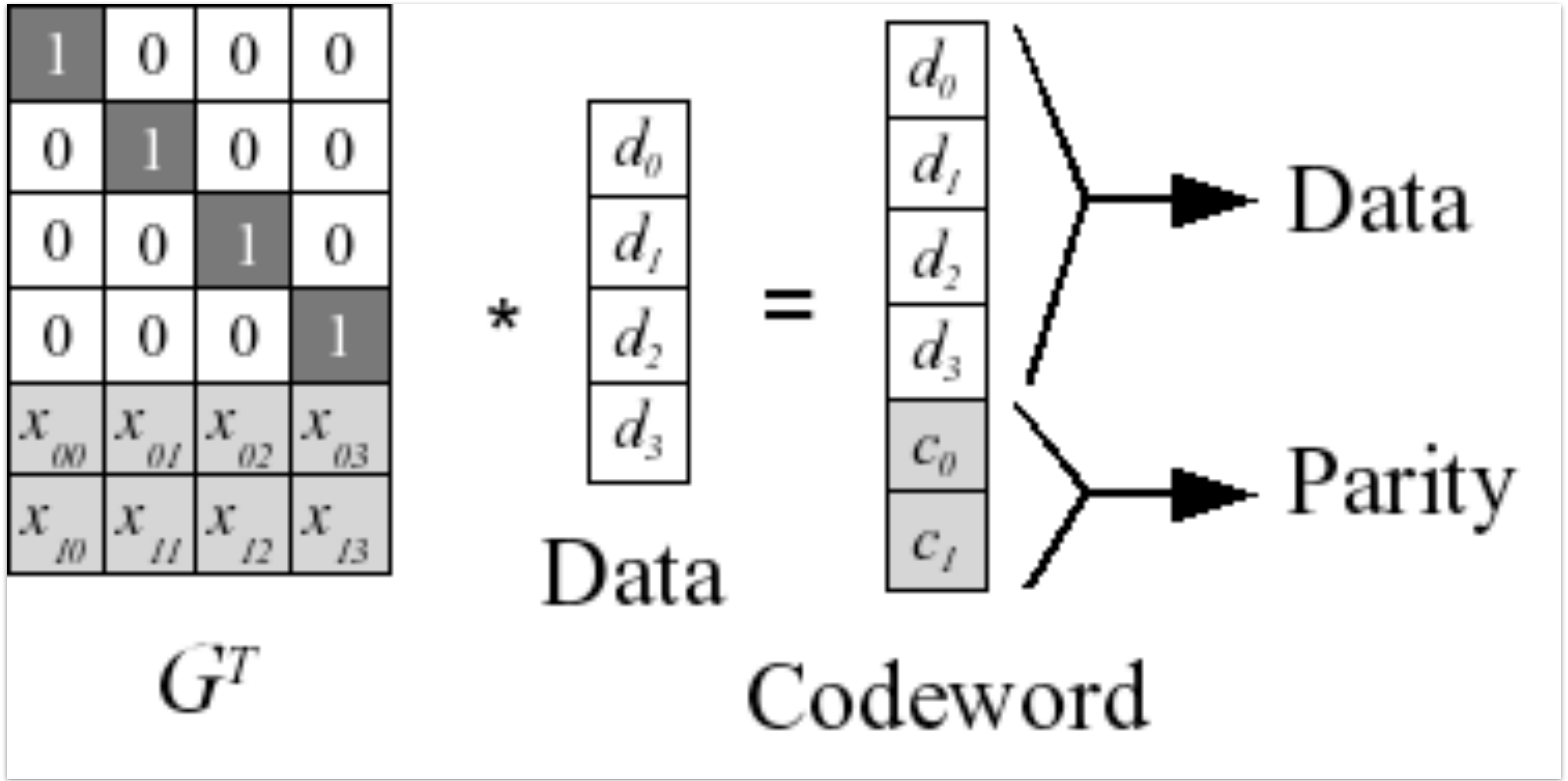
  `-ttt` show high time resolution

  → `man strace` !

Tutorial - Exercise 1

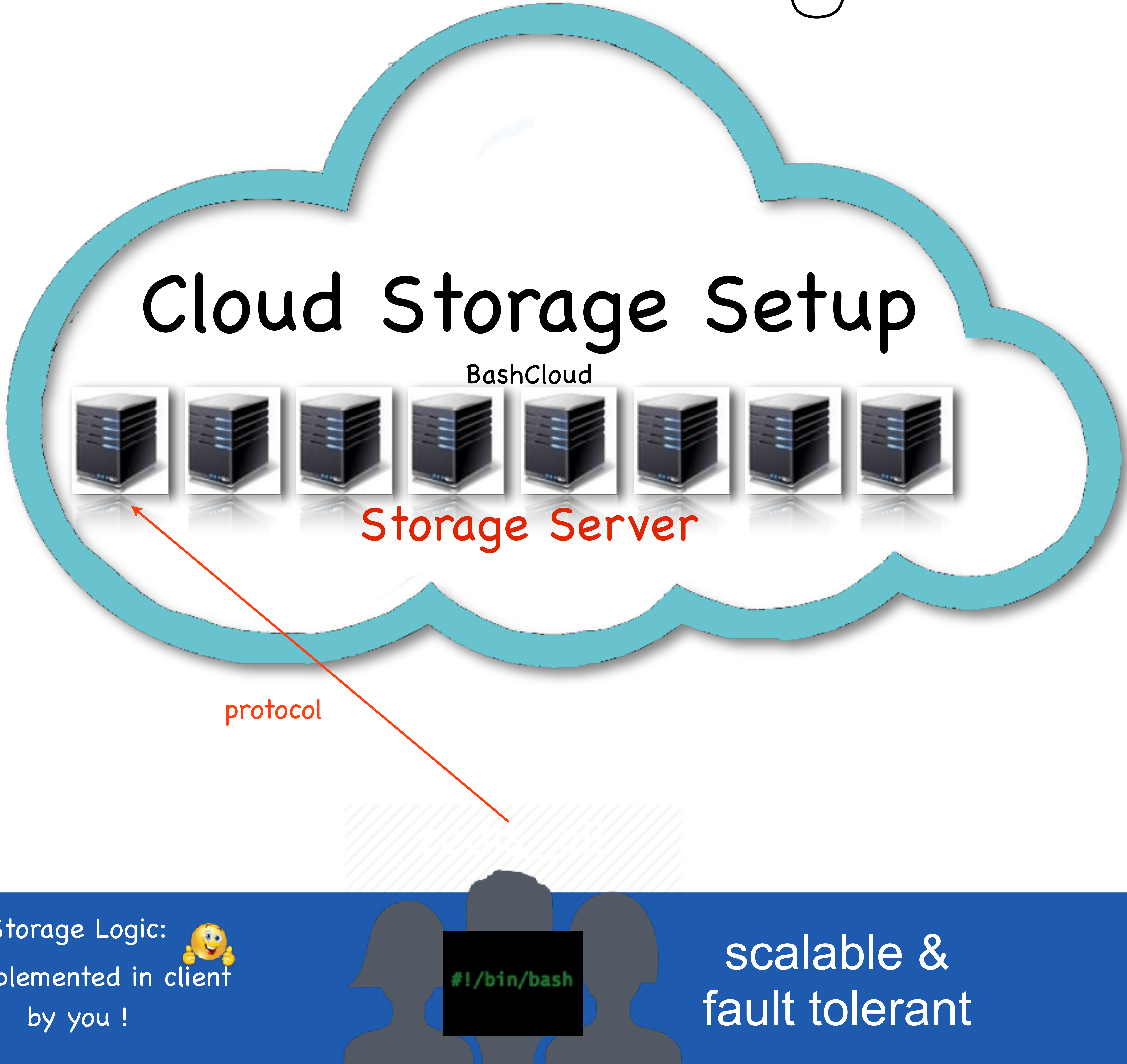
Redundancy RAID/RAIN





Tutorial - Exercise 2

Cloud Storage



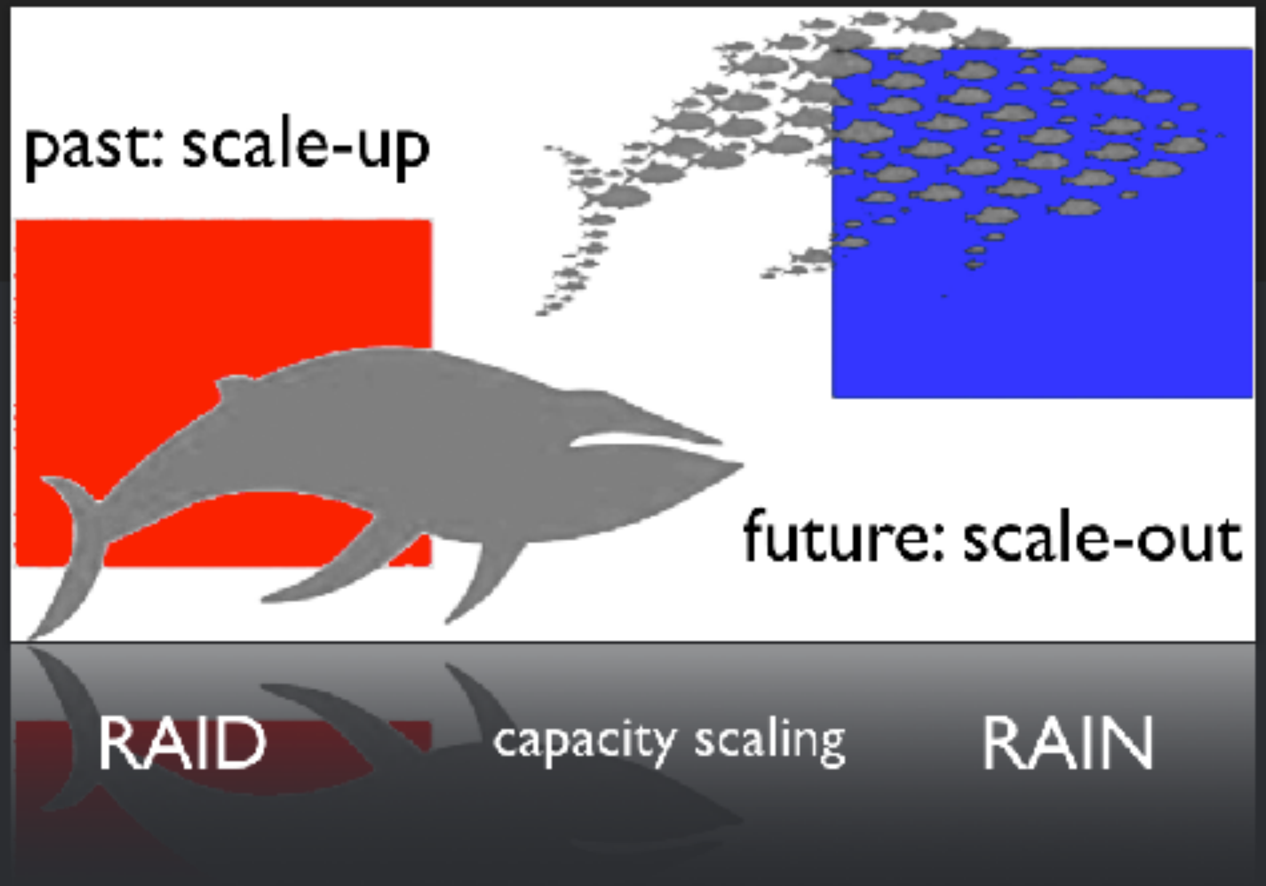
Storage Logic:
→ implemented in client
by you !

scalable &
fault tolerant

Cloud Storage= Scale-Out Storage

'Big Data - The Solution'

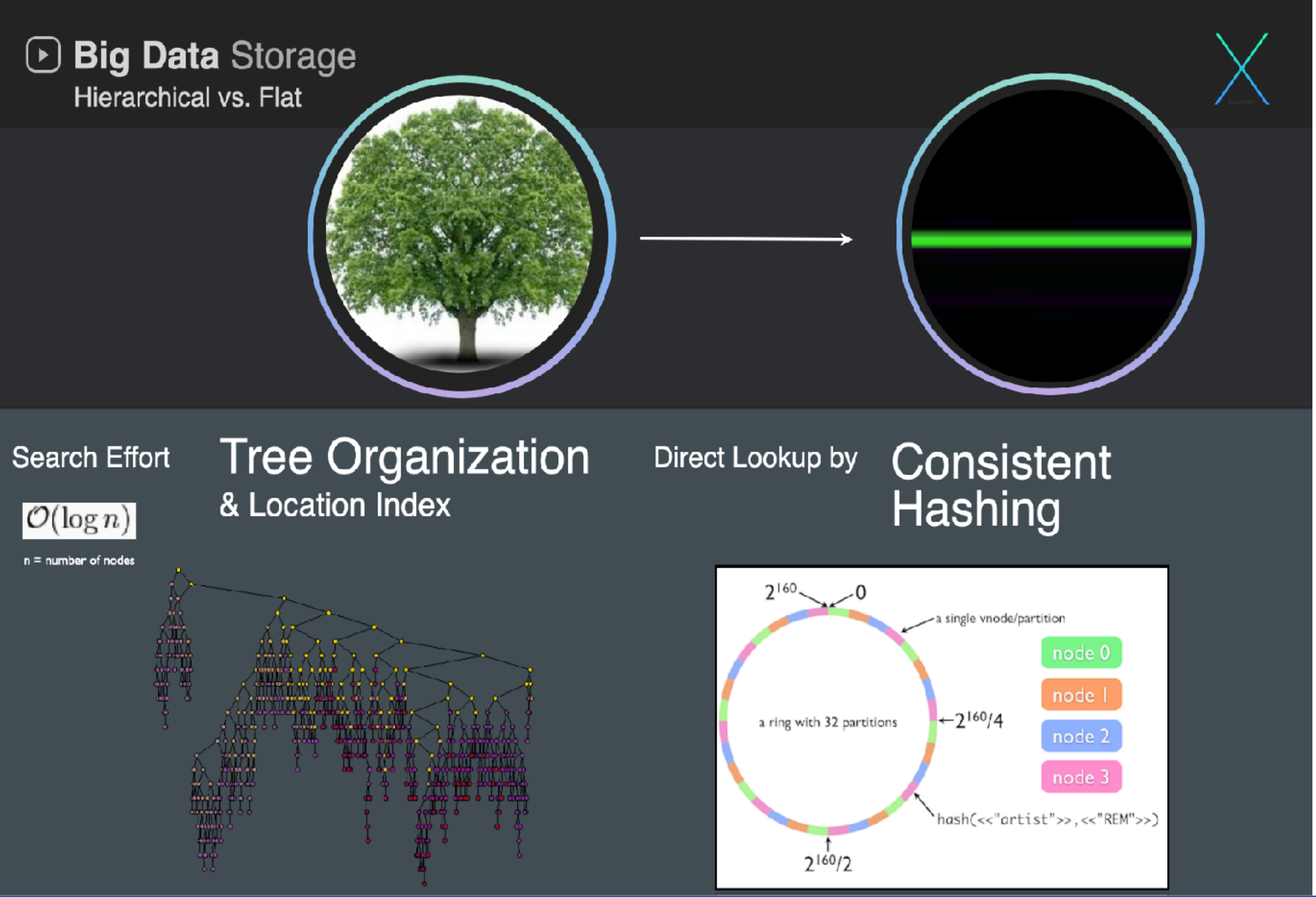
From Scale-Up to Scale-Out Storage



- structured data with central view & strong consistency
- predictable access
- SPOF
- slow growth rate

- best for unstructured data
- unpredictable growth rates
- MPOF resistant

many storage systems like HDFS use hybrid approaches with scale-up for meta-data and scale-out for data path



How to find
a file or on object?

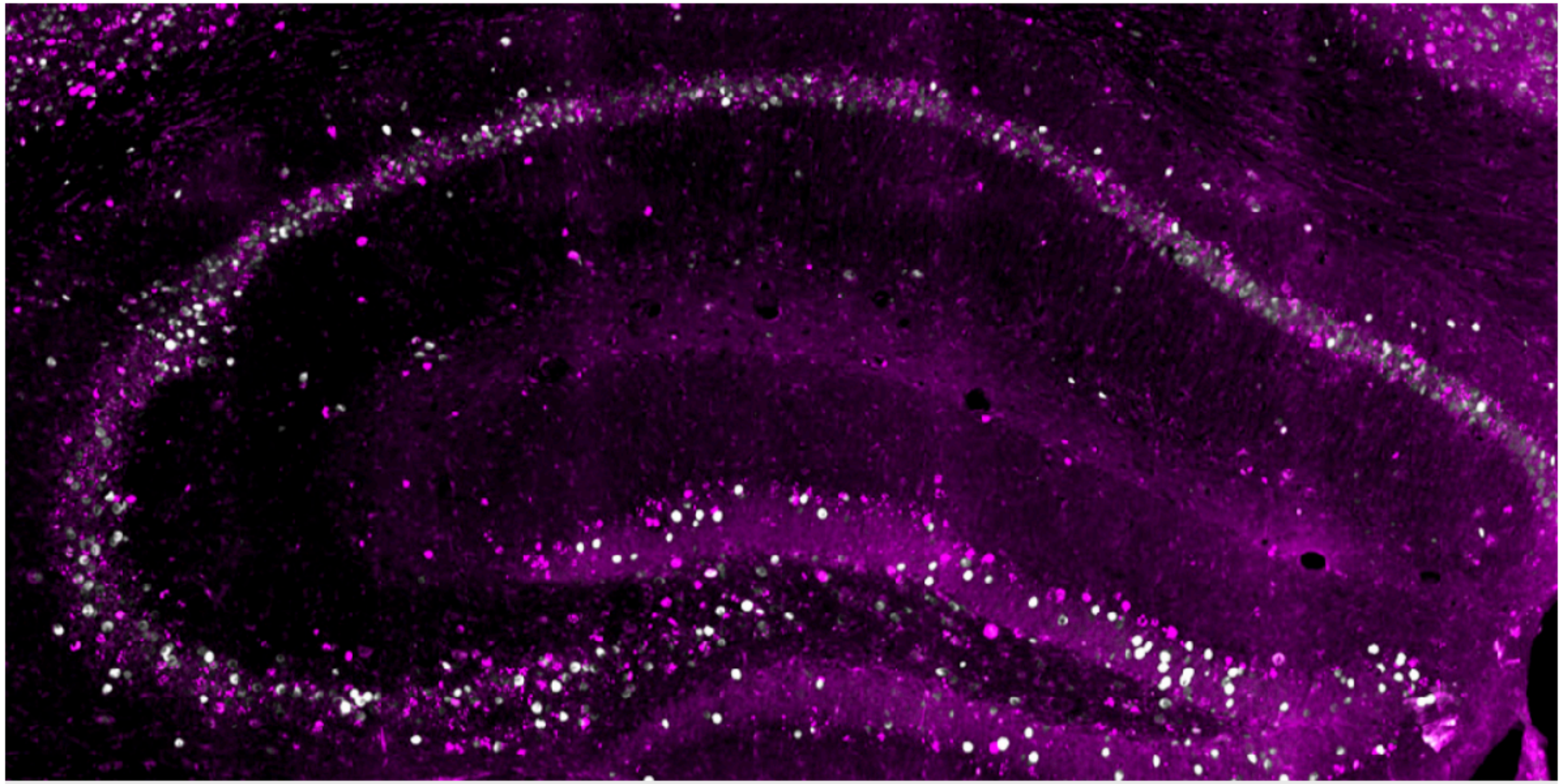
Filesytem:
tree search

Object Storage:
consistent hashing

Cloud / Object Storage

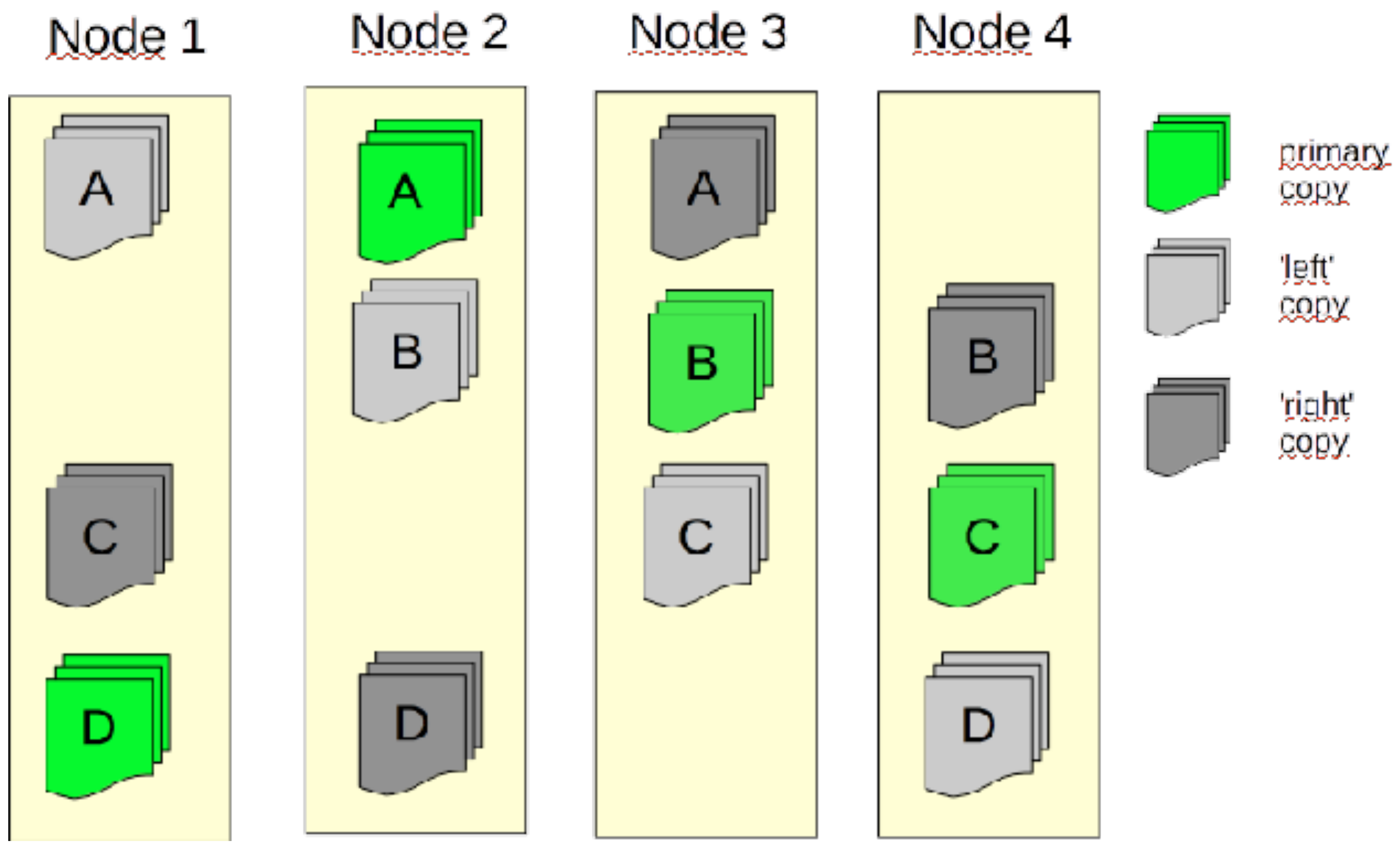
- basic principles for the exercise
  - sharding: files are placed and located using a distributed hash table (DHT)
    - the DHT can be changed to change the storage configuration
    - files are located computing the SHA1 checksum of their filename in hex representation
  - files get replicated to each neighbouring node e.g. every file has 3 copies
  - files can be listed using a 'bucket'

The brain creates three copies for a single memory



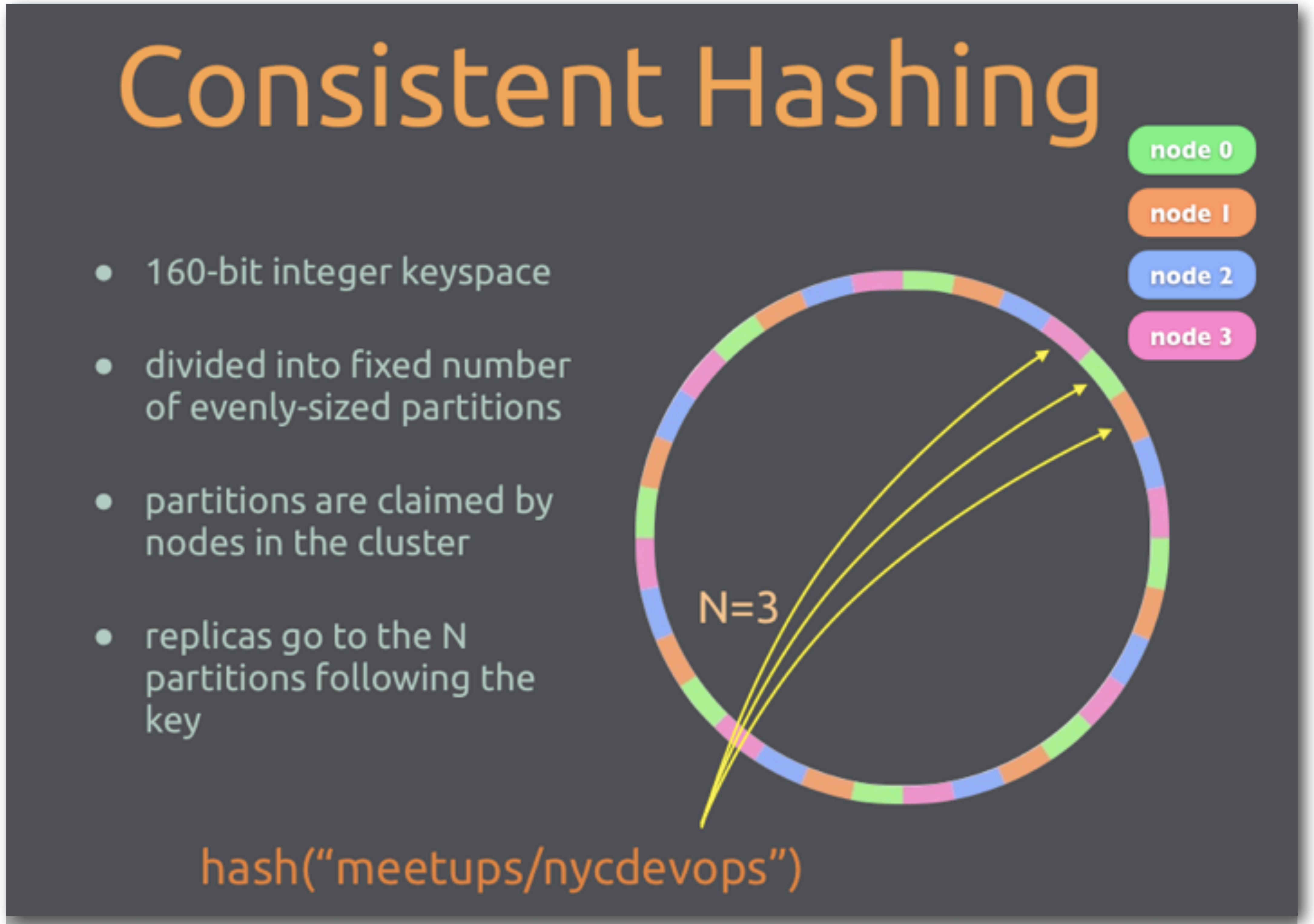
Cross-section through the hippocampus of a mouse: Early-born neurons (magenta) create a long-persisting copy of a memory. (Image: University of Basel, Biozentrum)

Locating files with consistent hashing



File Location Table = „Recipe to find a file by name“

Hash Value	Node Name
A	2
B	3
C	4
D	1



Cloud Storage Buckets

flat namespaces

- buckets are represented by a set of file names e.g.

```
ls /
1.jpg
2.jpg
3.jpg
```

- a set is more suitable than a list because it does not allow duplicated file names

- one can also shard buckets for scalability purposes - we don't do this

    - to list a directory one combines the listing of all participating servers

Basic Ingredients of Cloud Storage

- Objects:
  K-V Store API
  UPLOAD DOWNLOAD DELETE LIST

- Collections:
  SET API
  ADD DELETE LIST MEMBERS

- Scalability:
  Sharding of Objects and Collections

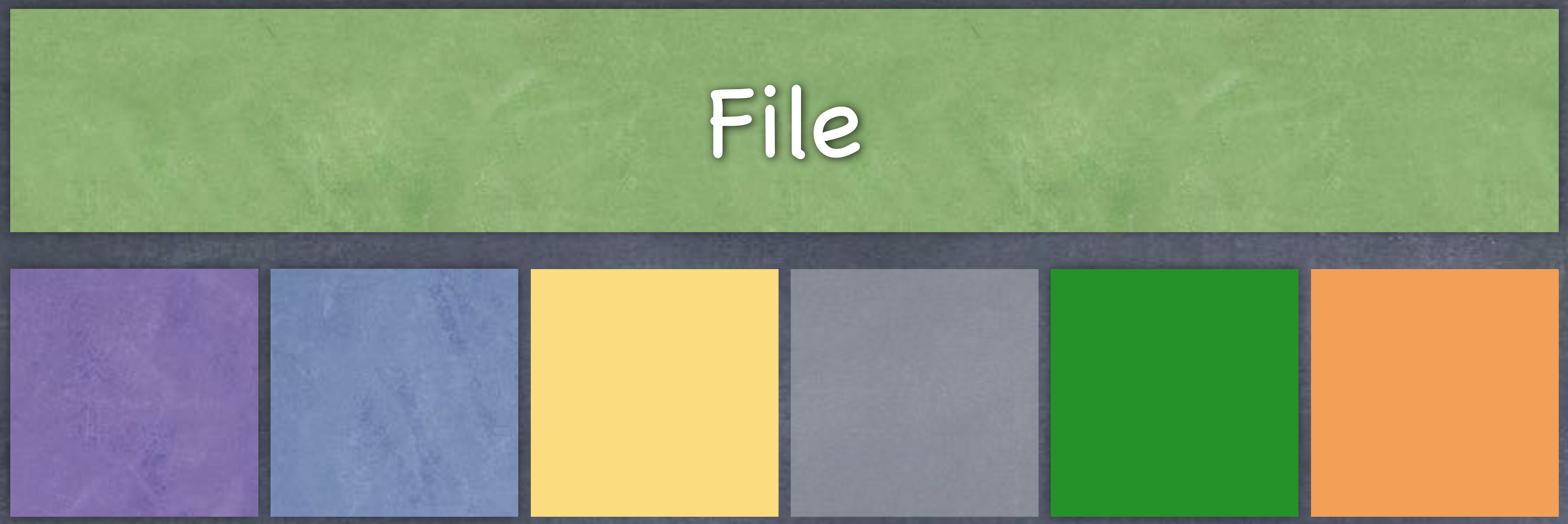
- Redundancy:
  Replication & Erasure Encoding
  (RS Encoding)

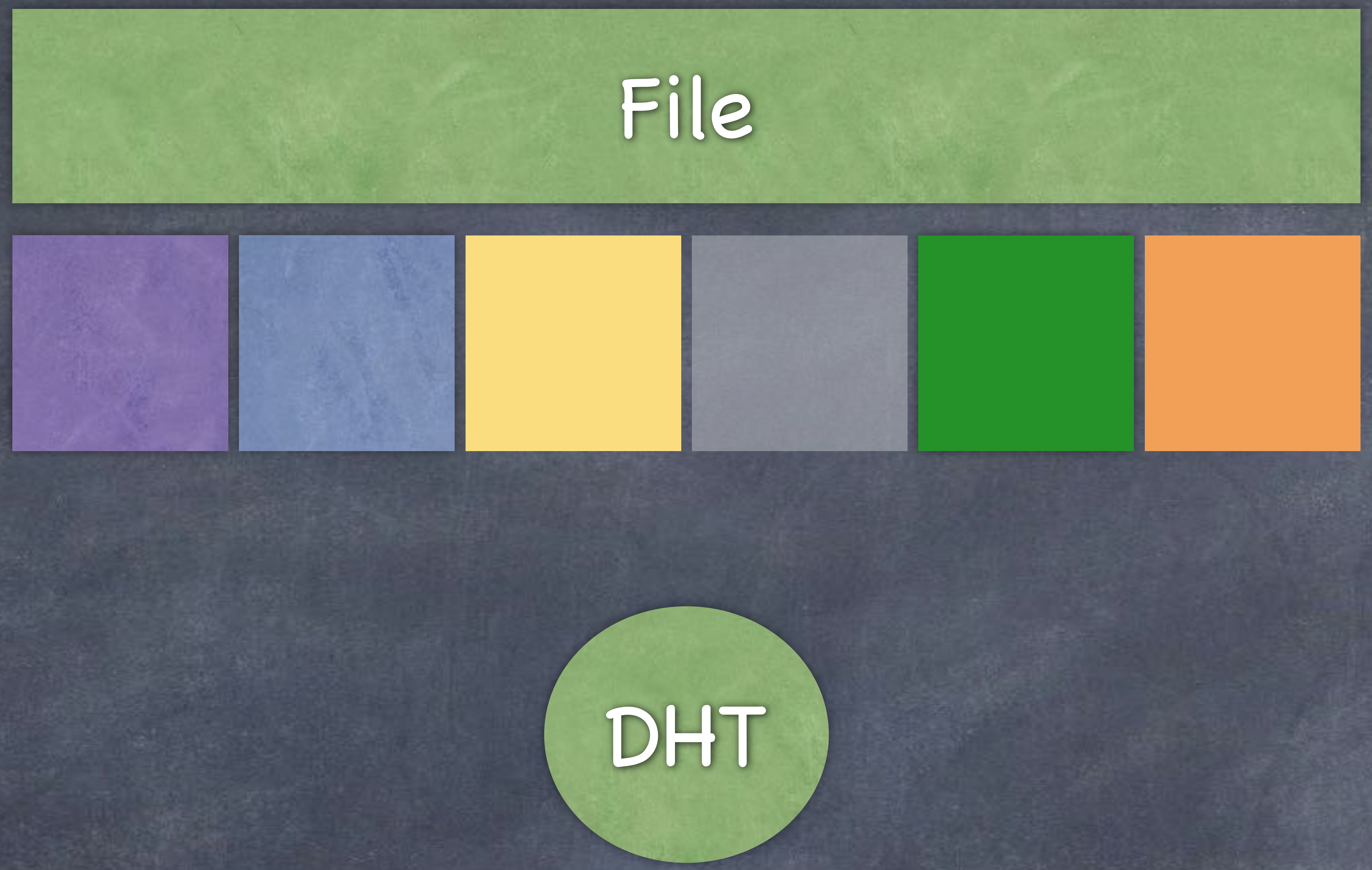
Exercises

1st hour	https://cern.ch/cscdt0
2nd hour	https://cern.ch/cscdt1
3rd + 4th hour today	https://cern.ch/cscdt2

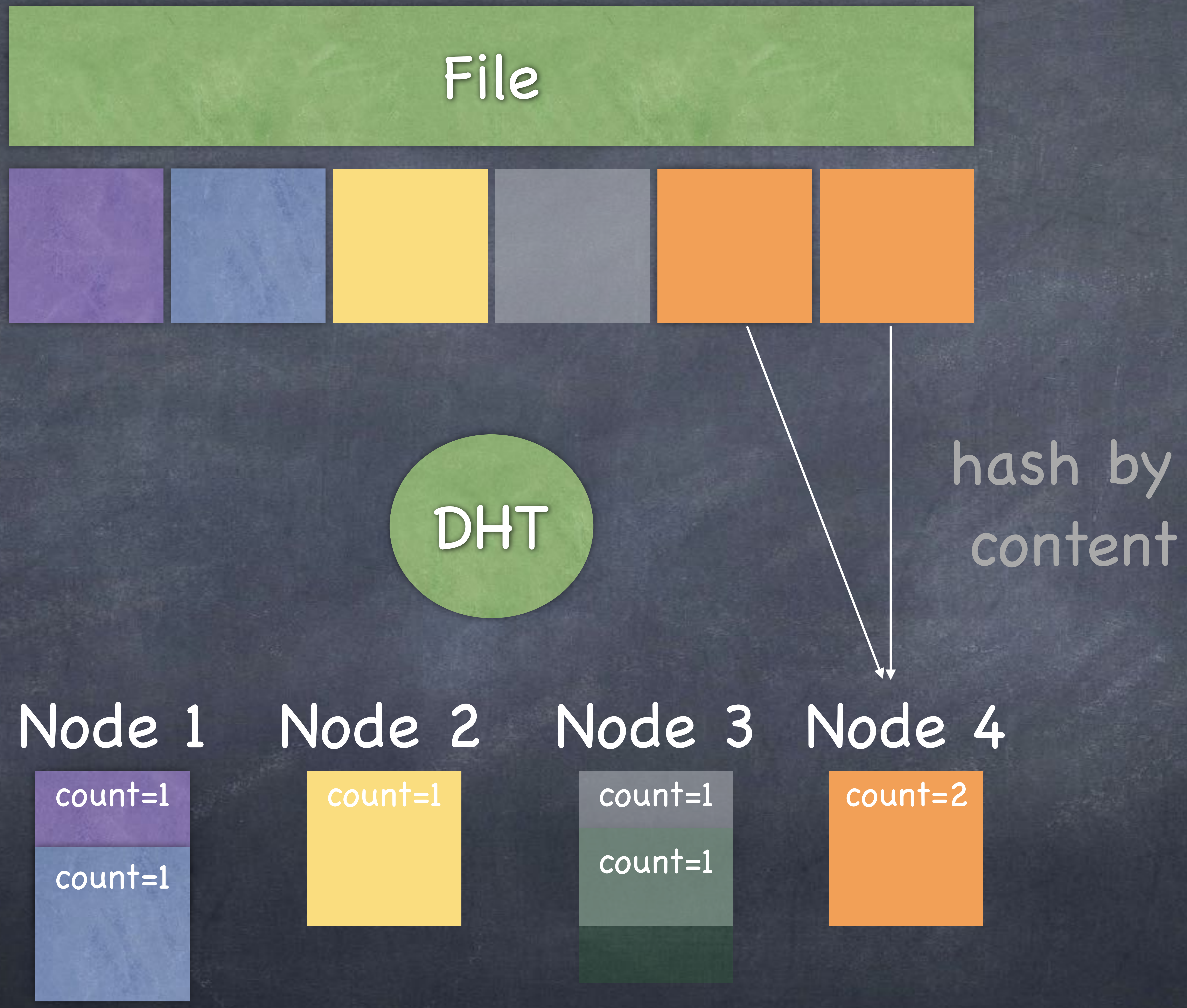
THANK YOU

File


File


File
DHT

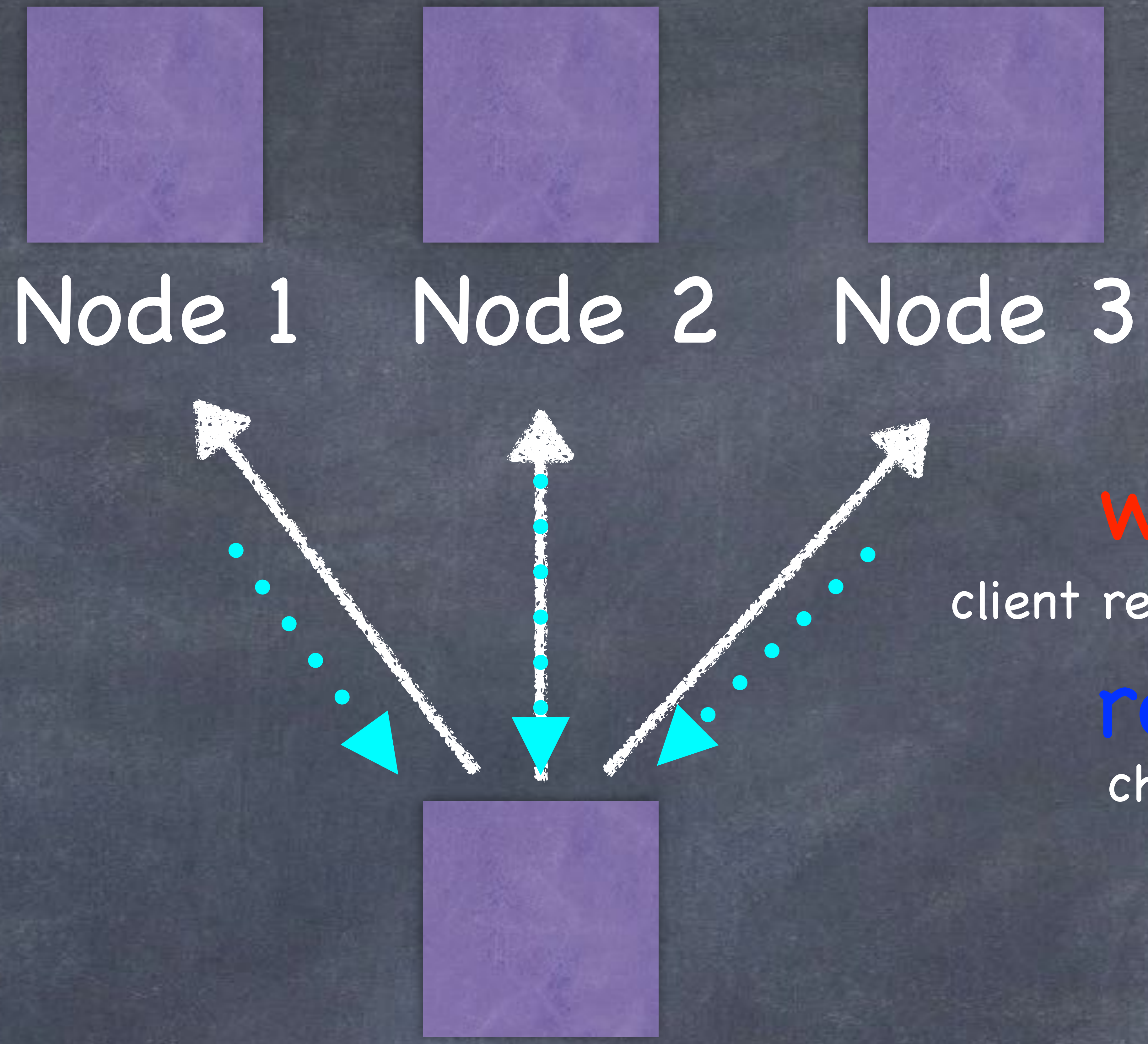

File.0
File.1
File.2
File.3
File.4
File.5
DHT
hash by name
Node 1
Node 2
Node 3
Node 4



Allows data de-duplication

Replication Schemes

client side



write path

client replicates to three nodes

read path

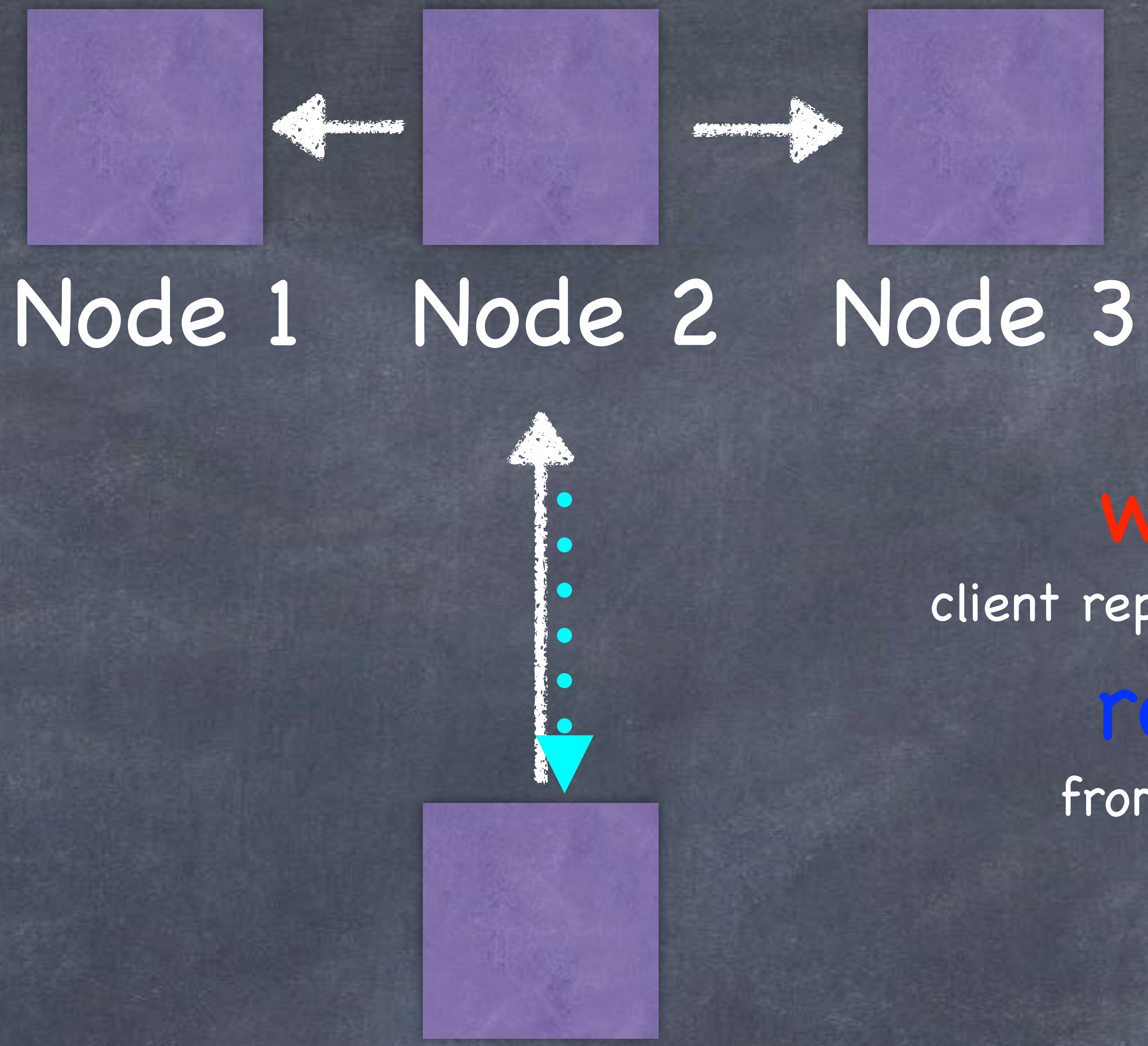
choose random node

According to the CAP theorem, is that an CA, CP or AP system?

C = Consistency A = Availability P = Partition Tolerance

Replication Schemes

server side



write path

client replicates trough one node

read path

from write entry node

According to the CAP theorem, is that an CA, CP or AP system?

C = Consistency A = Availability P = Partition Tolerance

Take Home Messages
IO Systems

- difference & relation of **bandwidth** and **IOPS** (IO Operations Per Second)
  - they are computed with **realtime measurements**
- **dd, strace & time** are simple tools helping to investigate IO bottlenecks
- understanding **IO bound** & **CPU bound** applications
  - impact of latency on CPU bound applications (they become IO bound)

Take Home Messages

Redundancy Algorithms

- Definition of Parity = XOR operation
- **Parity** allows to identify **data corruption**
  - parity alone does not locate corrupted data - you need checksumming
- Evolution **RAID** (scale-up) to **RAIN** (scale-out) system (disk redundancy => disk+node redundancy) => similar evolution as monolithic application towards micro service architecture
  - **RAIN** technology is built on top of your network!

Take Home Messages
Cloud Storage

- Cloud storage **locates objects** using a distributed hash table **DHT**
- Cloud storage uses the concept of a **bucket** to provide a fast/scalable listing (index)
- Cloud storage uses replication or erasure encoding to provide data high-availability
- Cloud storage uses **DHTs** based on the **consistent hashing principle** to minimise data movements when servers are added or removed