CSC 2023 Data Technologies

Introduction and Exercises

Andreas-Joachim Peters
CERN IT-SD

Contents of this Lecture

- Data Technologies? Why?
- Future Technologies
- Present Technologies
- Data Formats & Access Patterns
- Optimizations used in IO systems
- Introduction to the Exercises

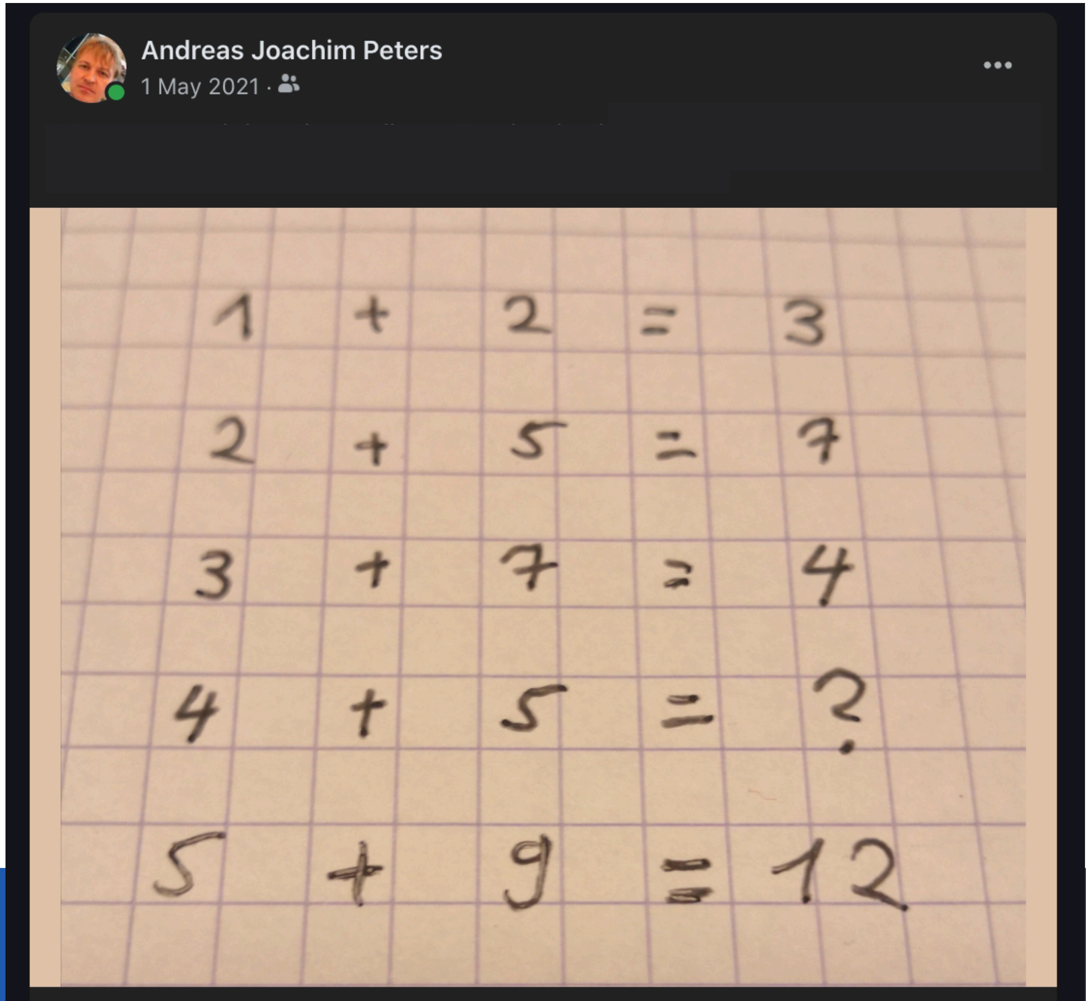
Why are Data Technologies important?

Accélérateur de science

.. or why is that part of this school?

.. because of ..

Social Media Puzzles



Nobody knew the answer!!!

“Best" answer:

1+2=3
2+5=7
~~3+7=4~~
4+5=9
~~5+9=12~~

Do you know the answer?
Don’t spoil the exercise!

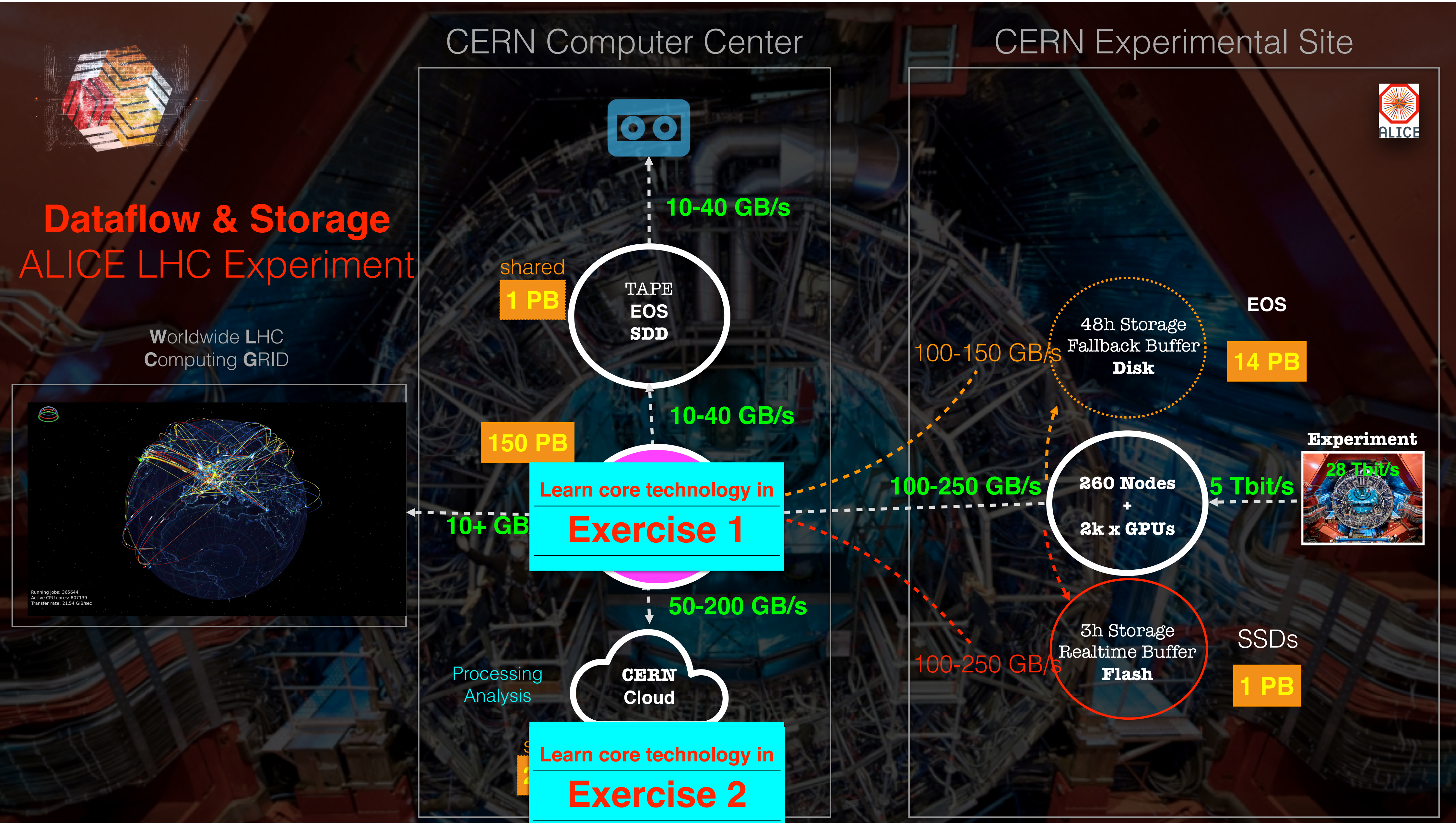
Scan Me!

Which mathematical object is that?

Why this odd picture to tell something to our phone?

.. but actually these ..

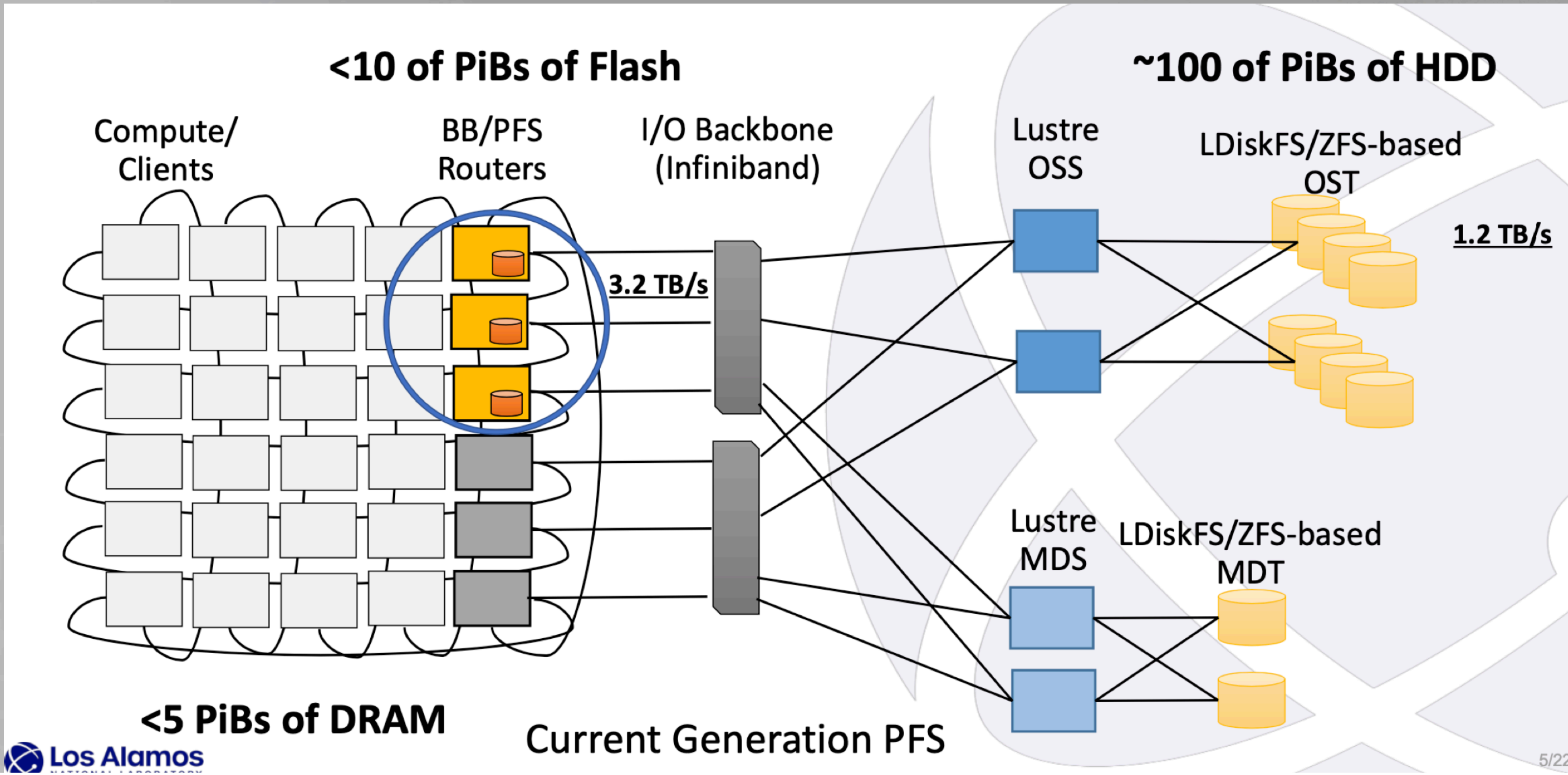

CERN

Dataflow & Storage
ALICE LHC Experiment
Worldwide LHC
Computing GRID
CERN Computer Center
CERN Experimental Site
ALICE
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


LOS ALAMOS
PROJECT
MAIN GATE
PASSES MUST BE
PRESENTED TO
GUARDS
SECURE
PASSES
HERE
POST
No. 1

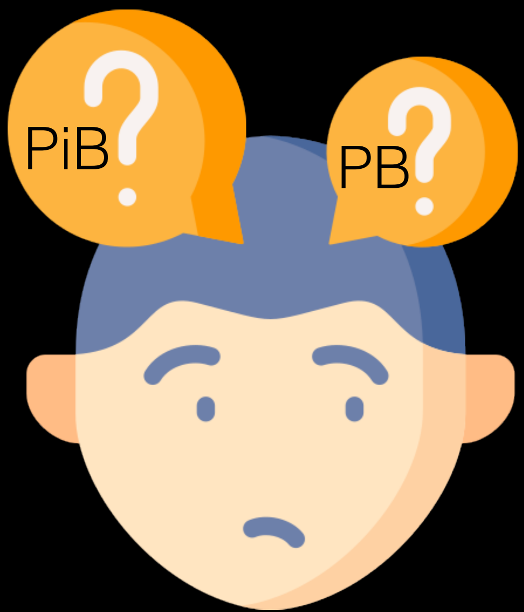
Lustre Storage System



Side Note

SI vs. **IEC** units …

Storage vendors always use SI units



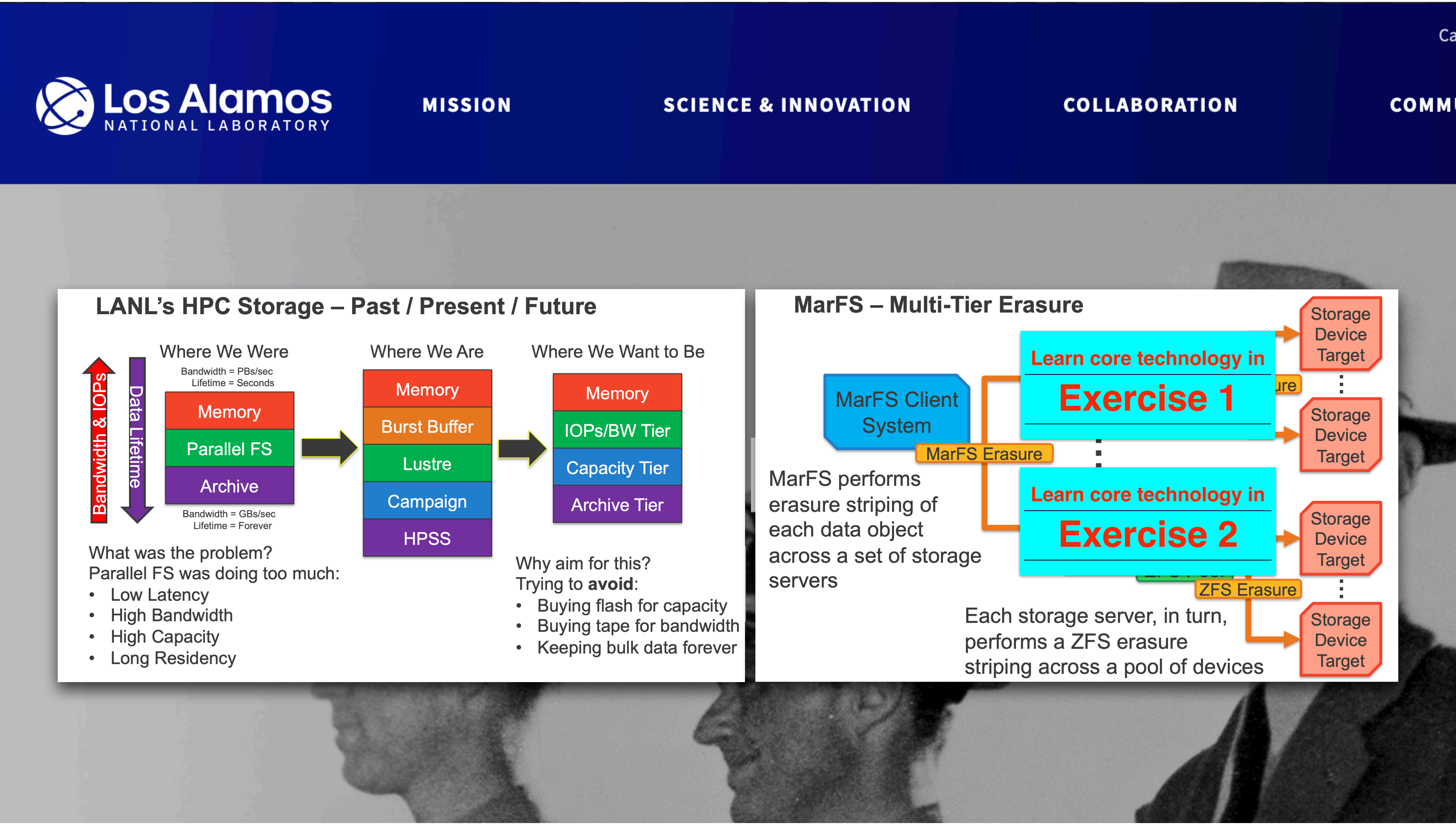
Prefixes for binary multiples

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%


Los Alamos
NATIONAL LABORATORY
MISSION
SCIENCE & INNOVATION
COLLABORATION
LANL's HPC Storage – Past / Present / Future
Where We Were
Bandwidth = PBs/sec
Lifetime = Seconds
Bandwidth & IOPs
Data Lifetime
Memory
Parallel FS
Archive
Bandwidth = GBs/sec
Lifetime = Forever
Where We Are
Memory
Burst Buffer
Lustre
Campaign
HPSS
Where We Want to Be
Memory
IOPs/BW Tier
Capacity Tier
Archive Tier
What was the problem?
Parallel FS was doing too much:
• Low Latency
• High Bandwidth
• High Capacity
• Long Residency
Why aim for this?
Trying to avoid:
• Buying flash for capacity
• Buying tape for bandwidth
• Keeping bulk data forever
MarFS – Multi-Tier Erasure
MarFS Client System
MarFS Erasure
Learn core technology in
Exercise 1
Learn core technology in
Exercise 2
ZFS Erasure
Storage Device Target
Storage Device Target
Storage Device Target
Storage Device Target
MarFS performs erasure striping of each data object across a set of storage servers
Each storage server, in turn, performs a ZFS erasure striping across a pool of devices

Data is our main asset !

- keep it **safe**
- provide **fast** acces
- keep it **small**


The FUTURE
of Data Storage
Gear 7-38 1:2.75
s/n: 3941501
MT-450919
Teeth: 40
Trans.Gear 1:3.25
STATOR (s3x51n)
s/n: 3941501
MT-450919
Op.T 10-150
Winding F: 0.99
n.c. 24
Prototype Phase
Power
Torque

Technologies

Daily Storage

12 JUL 2023

2023 Trends in Storage

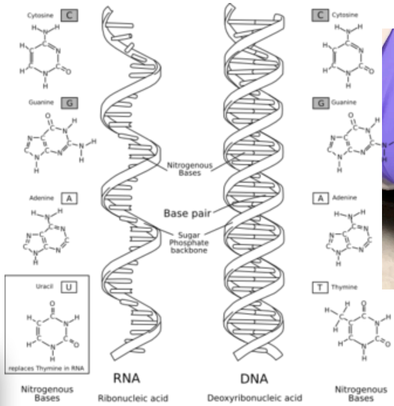
By CERN SCHOOL OF COMPUTING

DNA Storage, Crystals, Block Chain Structures, Multi Cloud Storage, Helium Hard Drives, SMR Drives 26 TB Capacity, 900 GB/s GPU Memory Bandwidth, 30 TB Laptop SSDs, 32 & 64 TB Enterprise SSDs and 2 TB USB flash drives!

Technologies

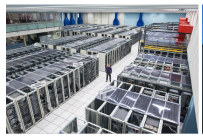
DNA Storage - 4-bit encoding

nucleobases AGCT



215 PB /g

1959 Initial idea by physicist Richard P. Feynman
2019 16 GB of storage in DNA (all Wikipedia)
2020 bit-wise random read access
2021 18 Mbit/s writing
2023 First rack-sized systems in development
- relies heavily on erasure coding

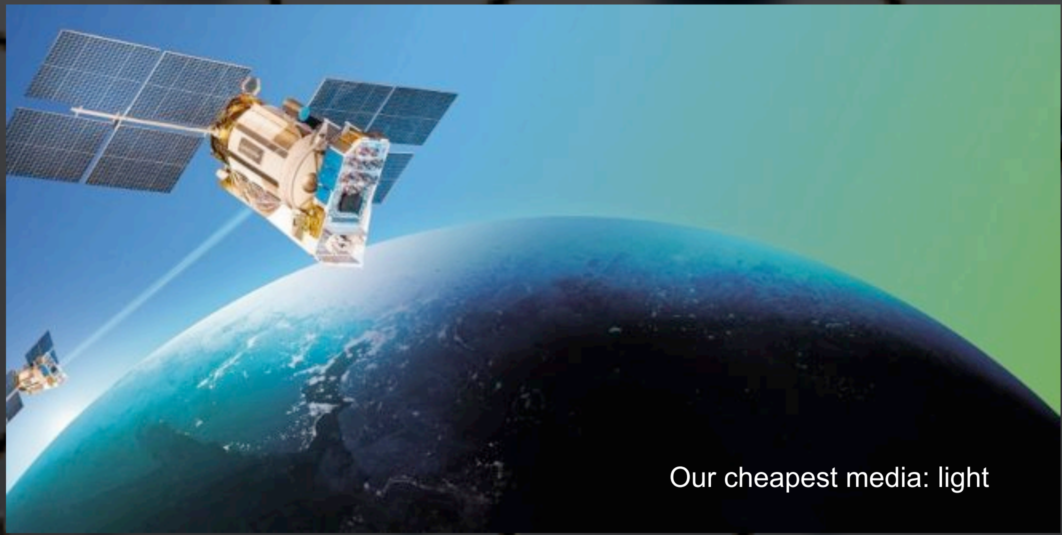


CERN CC = 5g

Exabytes in Space … 2021

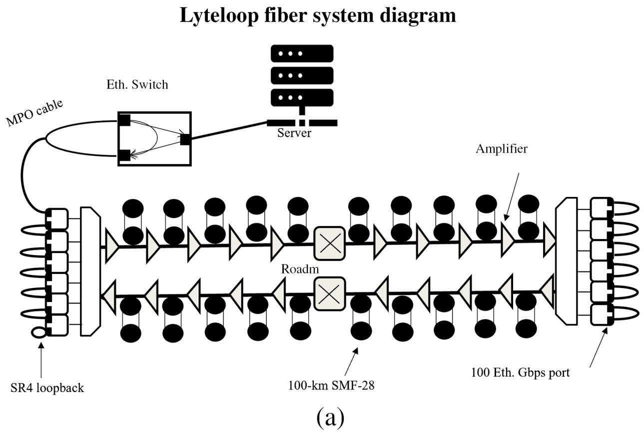
https://www.lyteloop.com/

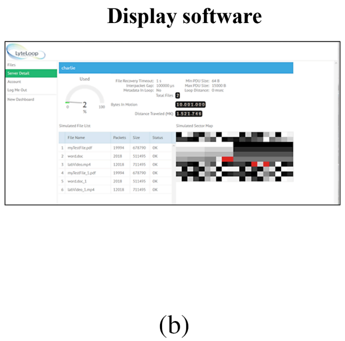
LyteLoop™



Cloud **is today** - Space **is tomorrow!**

LyteLoop ...





- **2021**: demonstrated storing 1.5 GB capacity in 2000km of optical fibre
- **2021**: Startup received 40M $ Funding
- **2022**: internet domain disappeared ...

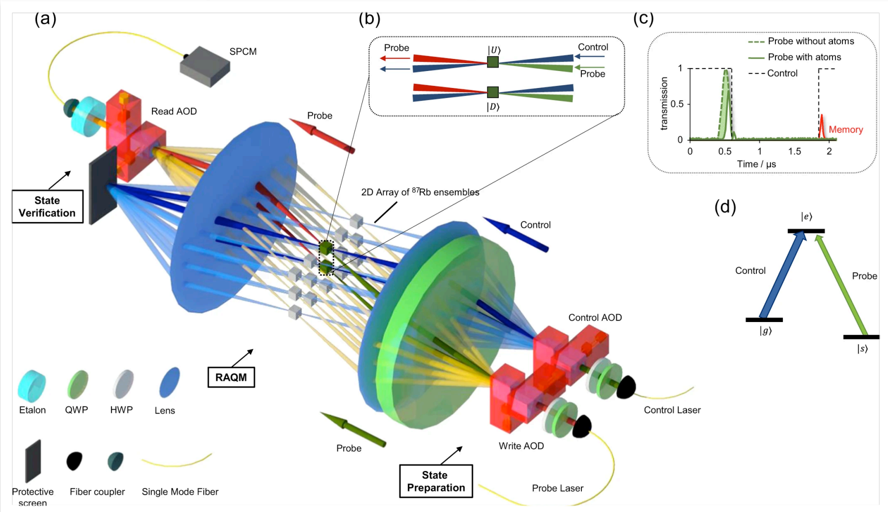
.. let’s try again ..

~~Space~~ **Quantum**

Cloud **is today -** Quantum **is tomorrow!**

Quantum Memory

105 qbit random access memory

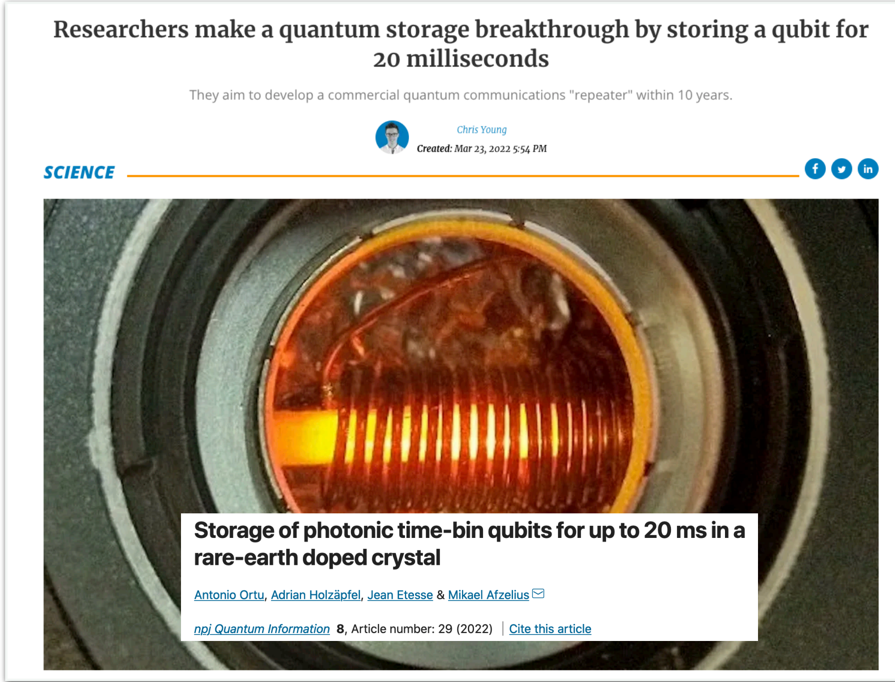


2019: magneto optical trap qbits

Experimental realization of 105-qubit random access quantum memory

N. Jiang, Y.-F. Pu, W. Chang, C. Li, S. Zhang & L.-M. Duan

npj Quantum Information **5**, Article number: 28 (2019) | Cite this article


Researchers make a quantum storage breakthrough by storing a qubit for 20 milliseconds

They aim to develop a commercial quantum communications "repeater" within 10 years.

Chris Young

Created: Mar 23, 2022 5:54 PM

SCIENCE

Storage of photonic time-bin qubits for up to 20 ms in a rare-earth doped crystal

Antonio Ortu, Adrian Holzäpfel, Jean Etesse & Mikael Afzelius

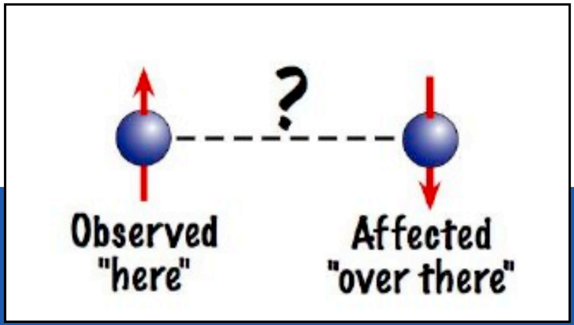
npj Quantum Information **8**, Article number: 29 (2022) | Cite this article

2022: crystal qbits

100 qbits can hold more states than all hard drives on earth $\mathbf{2^{100}}$ …

Quantum Memory

- Bummer: **Hoelvo's Bound**
  - The amount of data that can be retrieved from n qubits cannot be any larger than the amount of data that could be retrieved from n bits
    - pity: back from all hard disks on earth to **100 bits** …
  - **Reasons**: no cloning theorem, collapse upon measurement and complexity with entanglement



Quantum Memory Problem

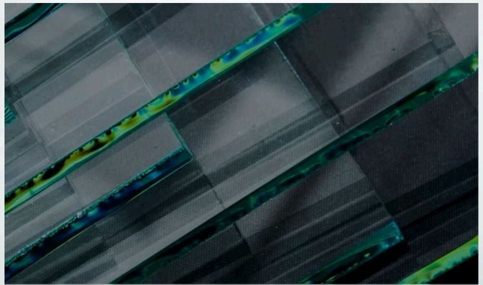
explained for the layman

a traffic jam in Beijing …

This car can't get out from the middle of this traffic jam. An entangled qubit in the middle of a complicated state has a comparable problem.

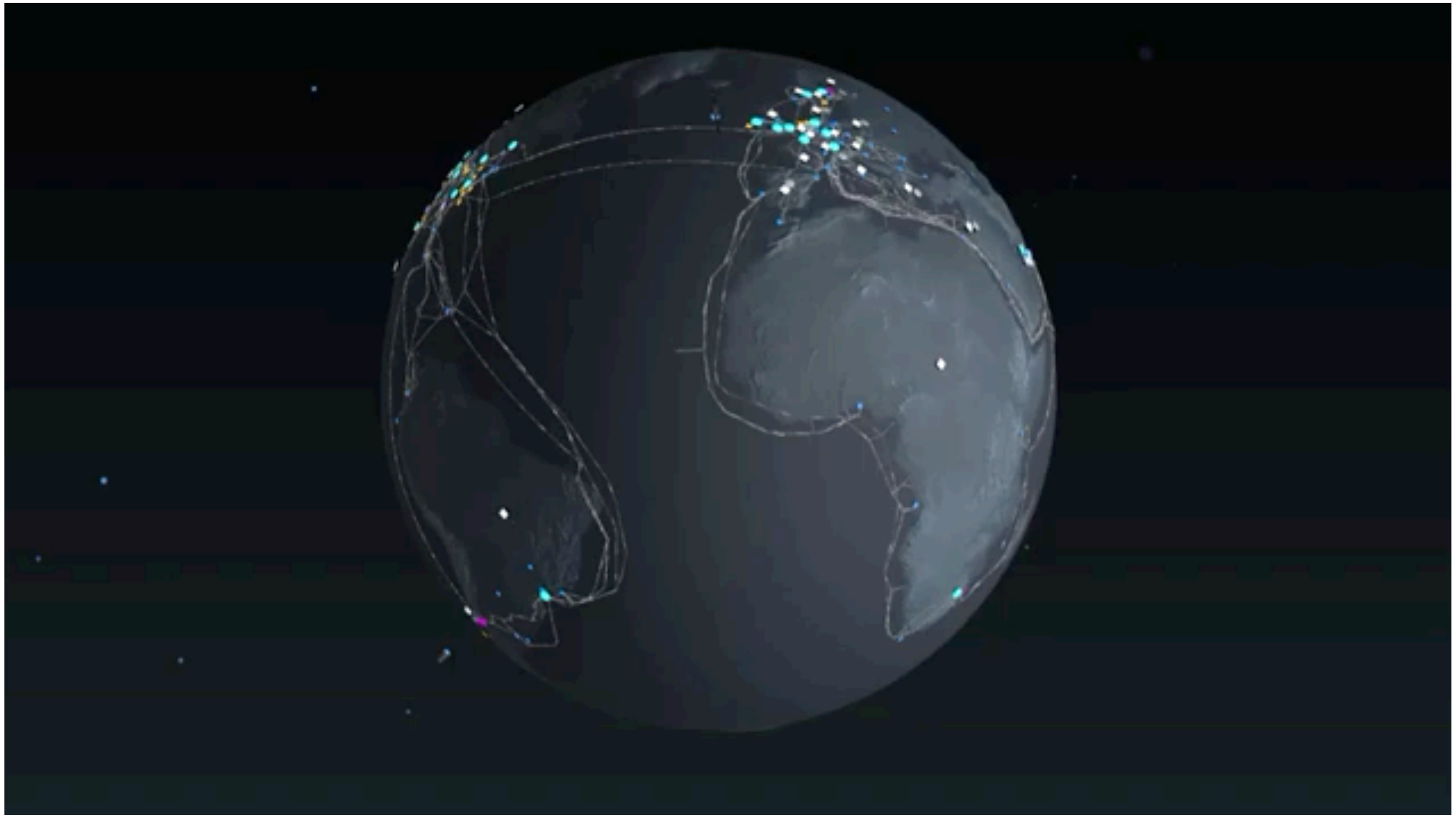
Doug Finke - SDC 2021

It's not (yet) **SPACE**, **QUANTUM** or **DNA** … what comes next in storage?

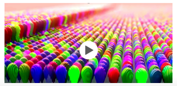


GLASS!

Crystal Storage - Project Silica



Back to today ...

CERN CC ... not as fancy ...   

New CERN Data Center (Prevessin)

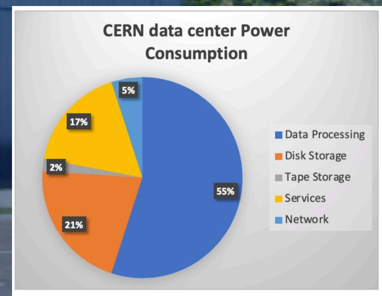
- will be online end of 2023
- **Meyrin** centre hosts most of **STORAGE**, **Prevessin** centre focus is **CPU**
- second computer centre allows to implement better business continuity for mission critical services

Meyrin DC PUE 1.5 - new Prevessin DC PUE 1.1

4MW installation - capable of 12MW

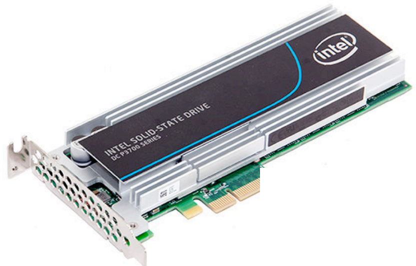
CERN 1.25 TWh per year

Computing < 5%



CERN Data Center (Meyrin)

Storage Media Types in CERN CC



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

EOS Disk Storage Server

JBODs with XFS filesystems

- Profiting from ***economy of scale***
  - minimise price per TB
- System Unit:
  - 1-2 CPUs: 8-16 physical cores 64-256GB RAM
  - disk-tray of 24 x **4-6-10-12-14-18** TB HDDs

- Running different generations
  - 2 trays per system unit - 48 disks
  - **4 trays per system unit - 96 disks**
  - 8 trays per system unit - 192 disks
  - 10/25/40/**100 GE** ethernet

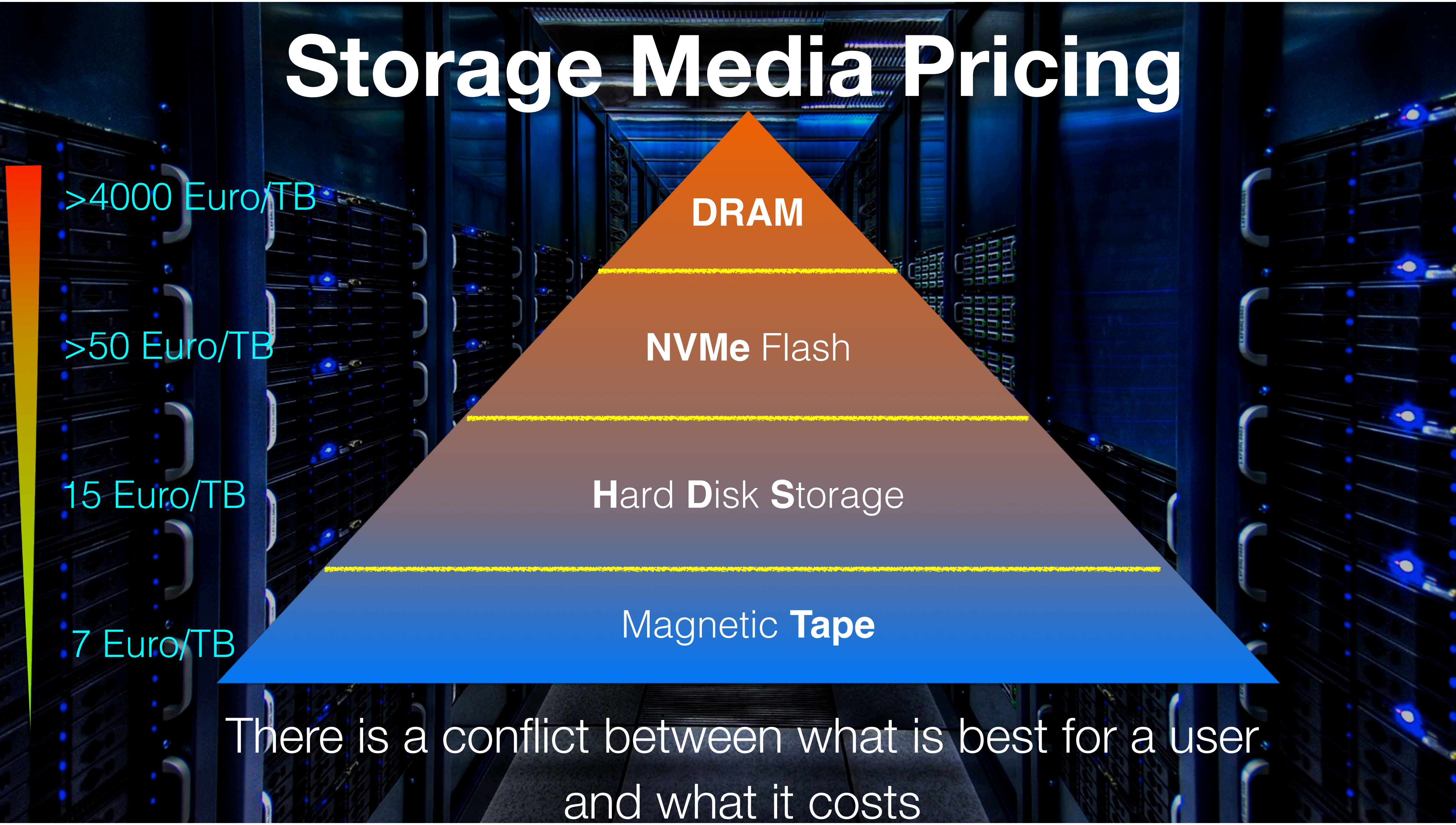
2023 Server Configuration

96 x 18 TB HDDs
per server - **1.7 PB**

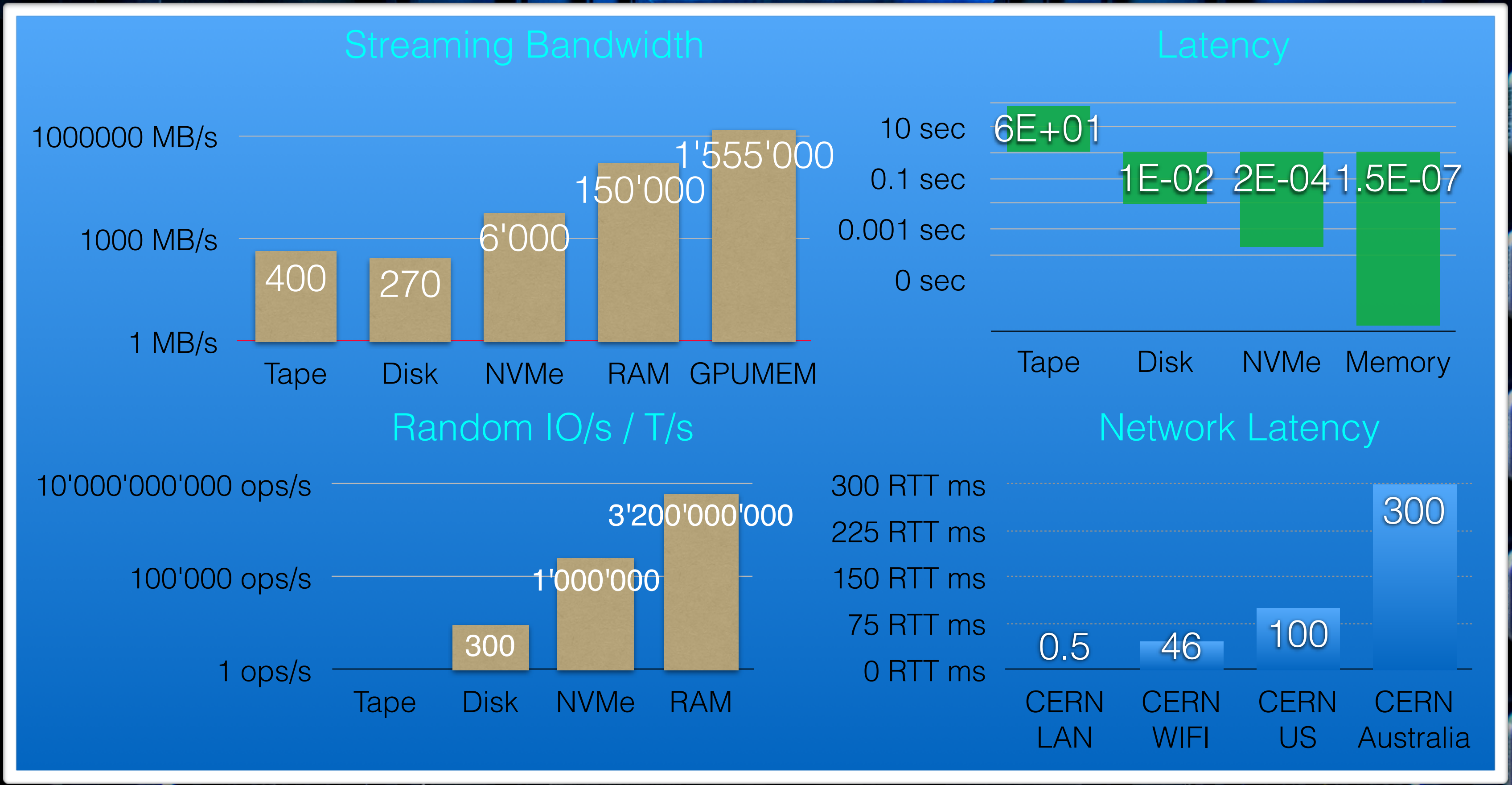
2 x 16**-**core CPU
256 GB Memory
100 GE Ethernet

XFS Filesystems
on JBODS

CERN Tape Robot

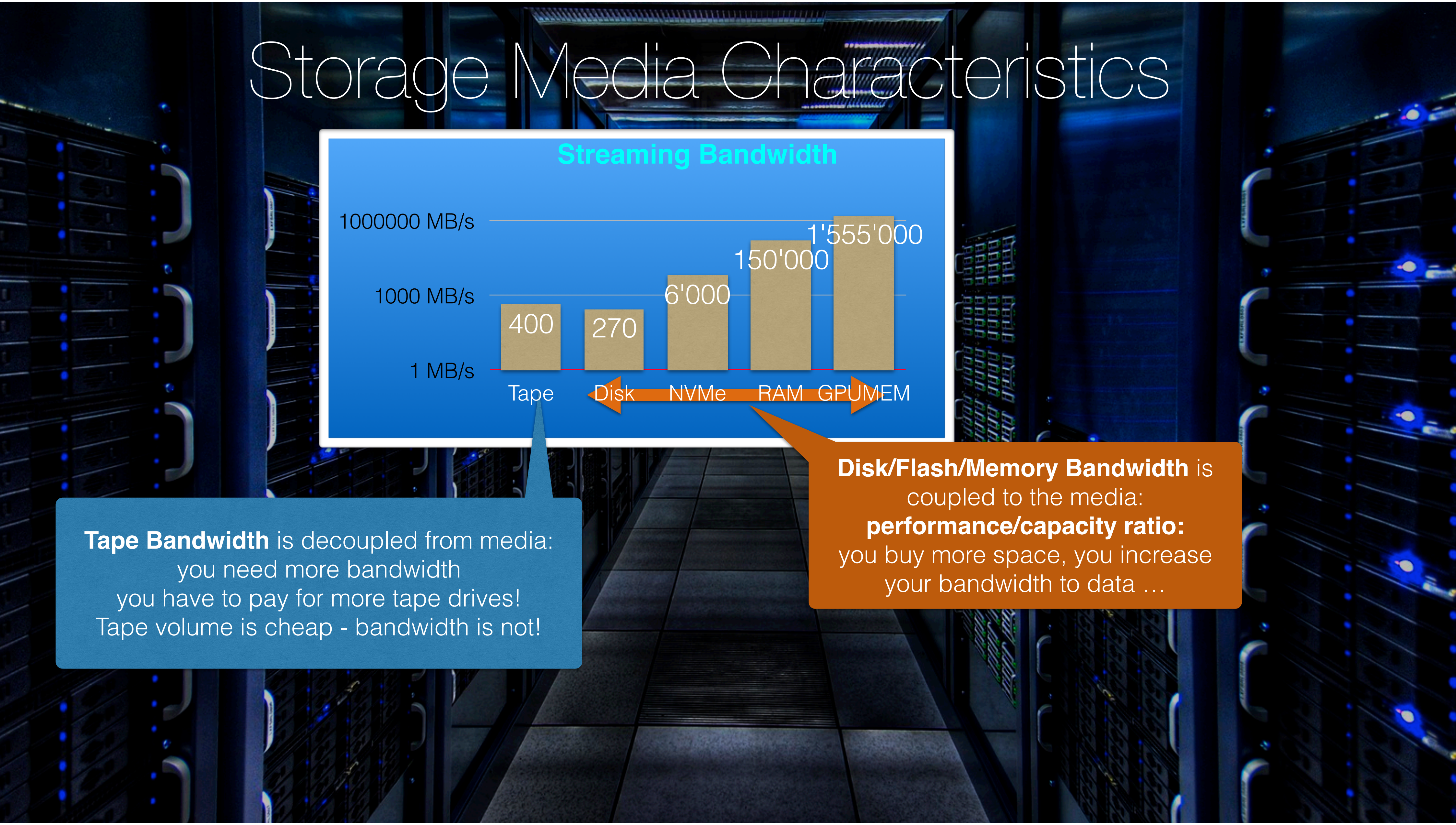

Storage Media Pricing
>4000 Euro/TB
>50 Euro/TB
15 Euro/TB
7 Euro/TB
DRAM
NVMe Flash
Hard Disk Storage
Magnetic Tape
There is a conflict between what is best for a user and what it costs

Storage Media Characteristics



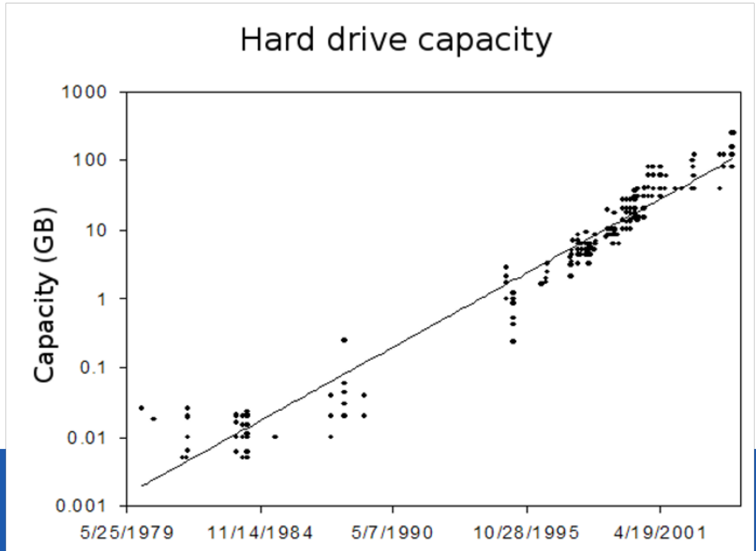
Disclaimer:

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


Storage Media Characteristics
Streaming Bandwidth
1000000 MB/s
1000 MB/s
1 MB/s
400
270
6'000
150'000
1'555'000
Tape
Disk
NVMe
RAM
GPUMEM
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 …

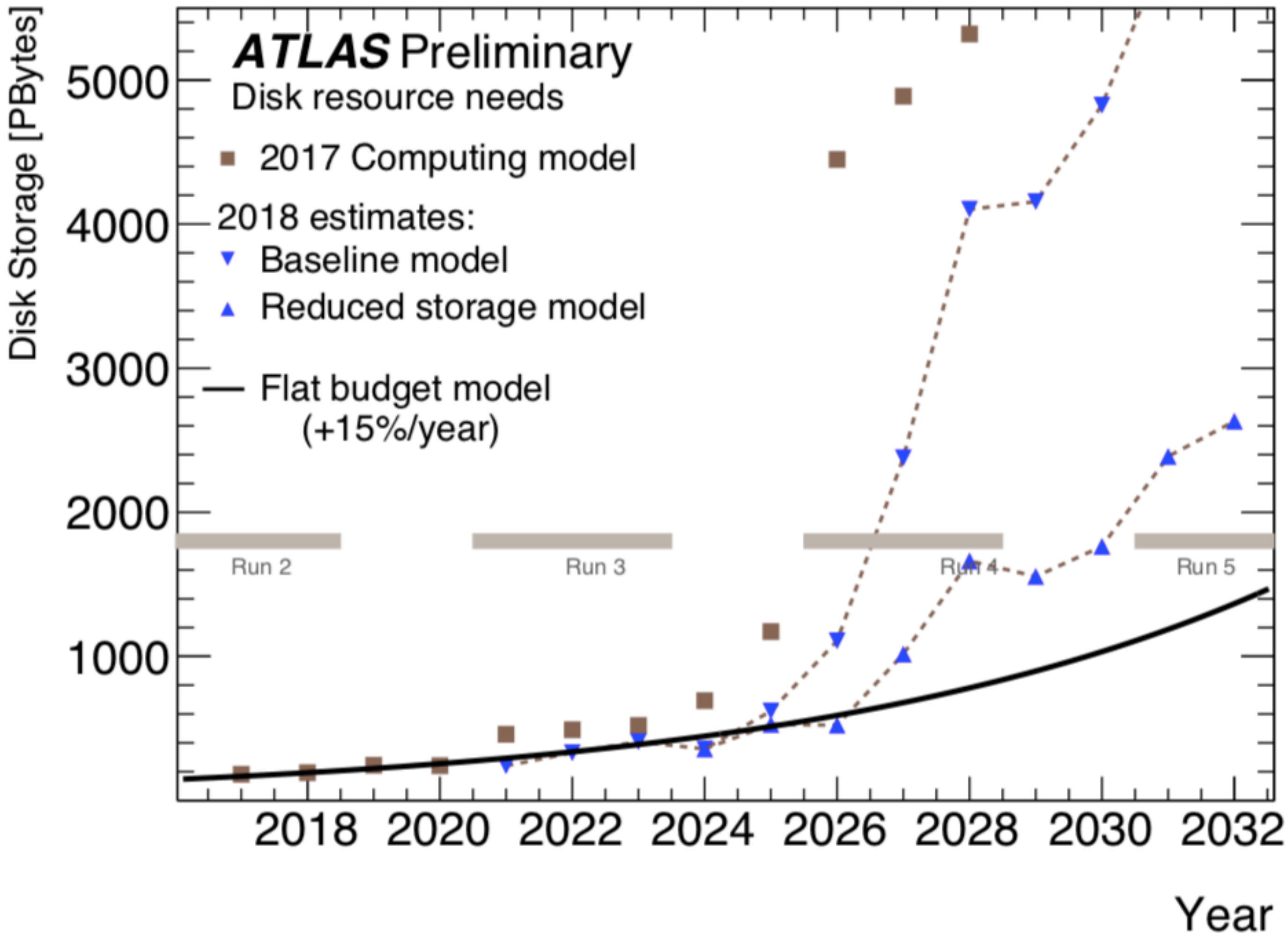
Storage: the cost problem

- **Many sciences have flat/limited budgets - data capacity increases by technology evolution**
- **'Moores law' for disk storage works today like:**
  we expect ~20% more space for the same money the year after … people use 'thin provisioning' for savings



Long term problem:
a given technology has physical limitations, sometimes technology has to be changed to continue the growth …

Example: Storage in LHC Run-4



Disk storage requirements can not be met with a flat budget after 2026!

What experiments look at:

- CMS can save 20% with **lossless** LZMA **compression** in raw data
- ATLAS R&D to save 20% with **lossy compression** in derived data formats

Do HDD and Tapes disappear?

- Today **everybody** has almost only **SSDs in laptops, desktops, tablets, phones**
- **T**he **enterprise HDD** market was growing - 1.5 ZByte of HDD space sold in 2021, but declined by 10% in 2022, expected to grow again in 2023
- **H**yper**S**cale **D**ata **C**enter (Alphabet, Meta, Amazon++) store most data on HDDs and tapes - after all it is about minimising costs!
  Same for World Wide LHC Computing GRID - no SSDs - 0.75 EB HDD 1.2 EB tape
- Technological progress: 27 PB in an one Open Compute Tape Rack are coming ….


Data Formats & Access Patterns

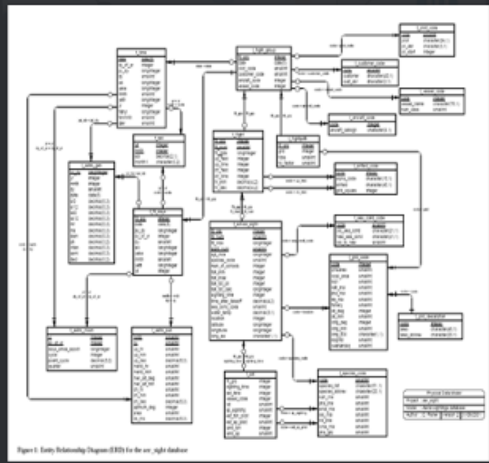
Contents of Data…

- most of data stored is **unstructured data** (photo, audio, video)

data for humans

best match
Scale-Out Storage
see Exercise 1 EC/RAIN

- small fraction of **structured data** (DBs, derived data, meta data)

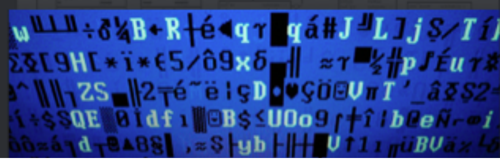


machine data

mostly stored in
Scale-Up Storage
see Exercise 1 RAID

- **Raw Data**

(LHC)



best match
Scale-Out Storage
Tiered Storage

Data Interchange Formats

- The data format you use has significant impact on performance and when stored on storage space required - example formats
  - XML
  - JSON
  - PROTOBUF, FLATBUFFER
  - Parquet, ARVO, ROOT, custom …

Data Interchange Formats

XML

- **XML**
  - Extensible mark-up language
  - Complex
  - Hard to read
  - Slow to parse
  - Not human friendly

```xml
<?xml version="1.0" encoding="UTF-8"?>
<WindowElement xmlns="http://windows.lbl.gov"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://windows.lbl.gov/BSDF-v1.4.xsd">
<Optical>
<Layer>
  <Material>
      <Name>Perfect Diffuser</Name>
      <Manufacturer>ACME Surfaces</Manufacturer>
      <Thickness unit="Meter">0.150</Thickness>
      <Width unit="Meter">1.000</Width>
      <Height unit="Meter">1.000</Height>
  </Material>
  <DataDefinition>
      <IncidentDataStructure>TensorTree3</IncidentDataStructure>
  </DataDefinition>
  <WavelengthData>
      <LayerNumber>System</LayerNumber>
      <Wavelength unit="Integral">Visible</Wavelength>
      <SourceSpectrum>CIE Illuminant D65 1nm.ssp</SourceSpectrum>
      <DetectorSpectrum>ASTM E308 1931 Y.dsp</DetectorSpectrum>
      <WavelengthDataBlock>
          <WavelengthDataDirection>Reflection Back</WavelengthDataDirection>
          <AngleBasis>LBNL/Shirley-Chiu</AngleBasis>
          <ScatteringDataType>BRDF</ScatteringDataType>
          <ScatteringData>{ 0.318309886 }</ScatteringData>
      </WavelengthDataBlock>
  </WavelengthData>
</Layer>
</Optical>
</WindowElement>
```

Data Interchange Formats

JSON

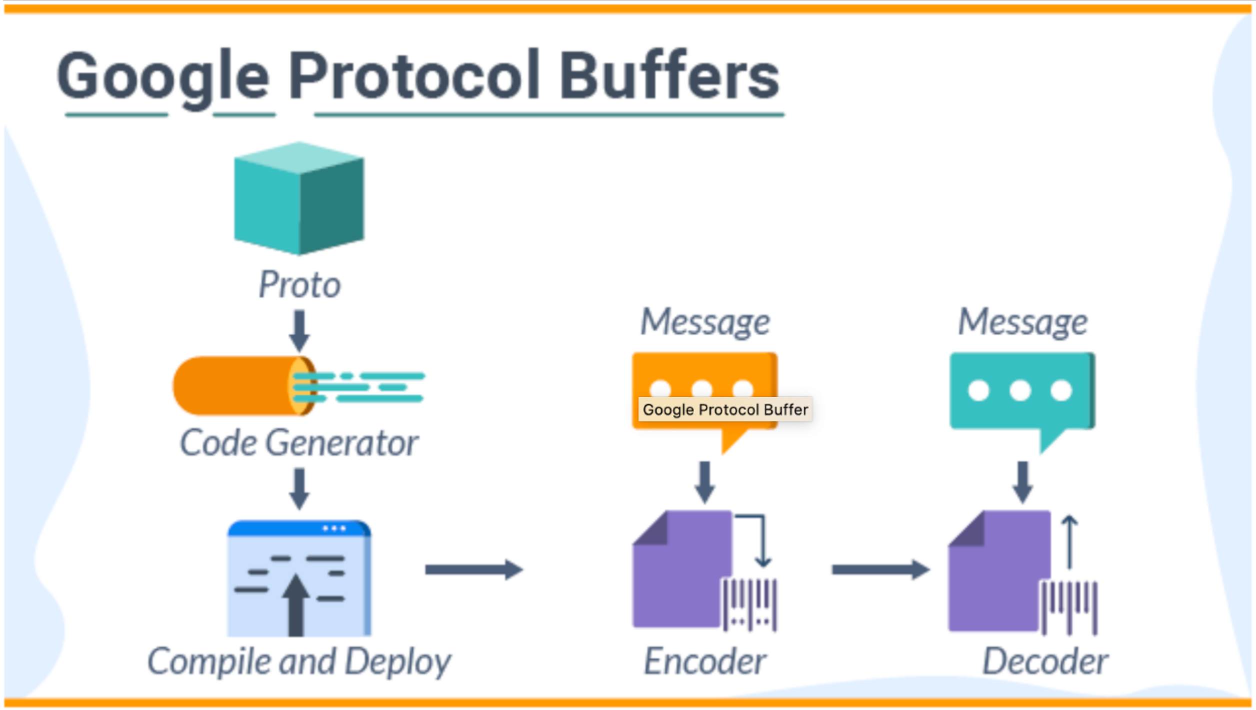
JSON

- Language independent data format derived from JAVA script
- UTF-8 character encoding
- Types
  - Number
  - String
  - Boolean
  - Array
  - Object (K-V list)
- Fast to parse, human readable, and dense
  (much better then XML)


```json
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

Data Interchange Formats

PROTO`BUF



- Protocol buffers is **language-neutral,** platform-neutral extensible mechanism for serializing structured data over the network
- Data structure is **defined in a schema** .proto file, which is compiled into an API for a given language
- Supported for almost **all platforms** and **language** bindings
- Supports **simple schema evolution** (extension of schma)

Data Interchange Formats

PROTOBUF

```
syntax = "proto2";

package tutorial;

message Person {
  optional string name = 1;
  optional int32 id = 2;
  optional string email = 3;

  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }

  message PhoneNumber {
    optional string number = 1;
    optional PhoneType type = 2 [def
  }

  repeated PhoneNumber phones = 4;
}

message AddressBook {
  repeated Person people = 1;
}
```

Compiled PYTHON API

```
class Person(message.Message):
  __metaclass__ = reflection.GeneratedProtocolMessageType

  class PhoneNumber(message.Message):
    __metaclass__ = reflection.GeneratedProtocolMessageType
    DESCRIPTOR = _PERSON_PHONENUMBER
  DESCRIPTOR = _PERSON

class AddressBook(message.Message):
  __metaclass__ = reflection.GeneratedProtocolMessageType
  DESCRIPTOR = _ADDRESSBOOK
```

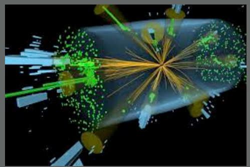
Compiled C++ API

```
// name
inline bool has_name() c
inline void clear_name(
inline const ::std::str
inline void set_name(co
inline void set_name(co
inline ::std::string* m

// id
inline bool has_id() const;
inline void clear_id();
inline int32_t id() const;
inline void set_id(int32_t value);

// email
inline bool has_email() const;
inline void clear_email();
inline const ::std::string& email() const;
inline void set_email(const ::std::string& value);
inline void set_email(const char* value);
inline ::std::string* mutable_email();

// phones
inline int phones_size() const;
inline void clear_phones();
inline const ::google::protobuf::RepeatedPtrField< ::tutorial::Person_PhoneNumber >& phones() const;
inline ::google::protobuf::RepeatedPtrField< ::tutorial::Person_PhoneNumber >* mutable_phones();
inline const ::tutorial::Person_PhoneNumber& phones(int index) const;
inline ::tutorial::Person_PhoneNumber* mutable_phones(int index);
inline ::tutorial::Person_PhoneNumber* add_phones();
```

Data Storage Formats

High Energy Physics - no XML, no JSON!

unstructured raw data - each physics event is stored in a compound block - events are assembled during data taking from many detector systems



1 2 3 4 5 6 7 8

File

N

compressed raw data with individual events

structured data - data is stored optimised for volume and access patterns

1..8

z1	y1	x1		…..	a1
z2	y2	x2			a2
z3	y3	x3			a3
z4	y4	x4			a4
z5	y5	x5			a5
z6	y6	x6			a6
z7	y7	x7			a7
z8	y8	x8			a8

File

C(Z) C(Y) C(X) ….. C(a)

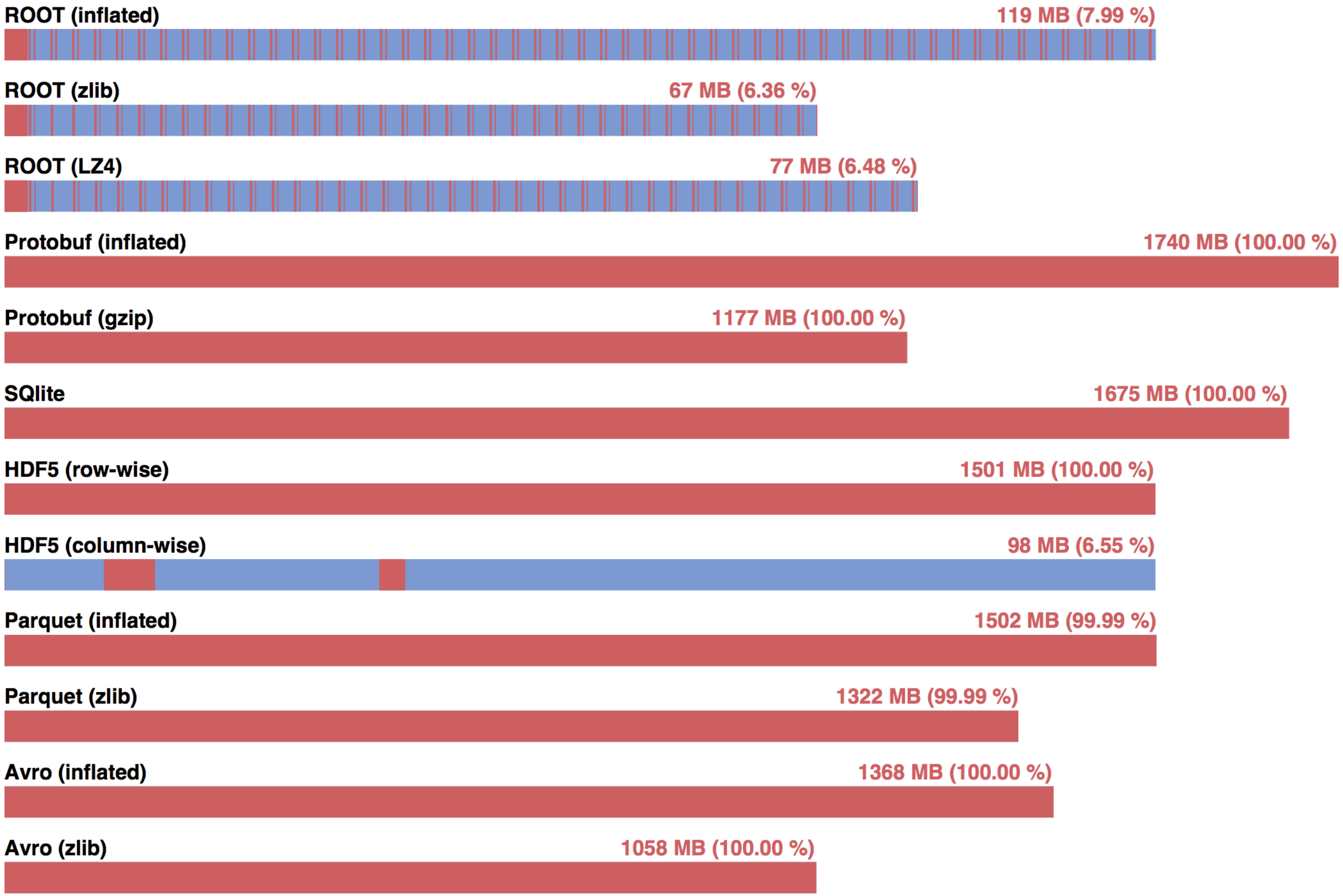
File

ROOT Data Analysis Framework

Data Formats & Storage Access Patterns

in selective analysis use cases

read pattern (read) in a selective physics analysis workflow

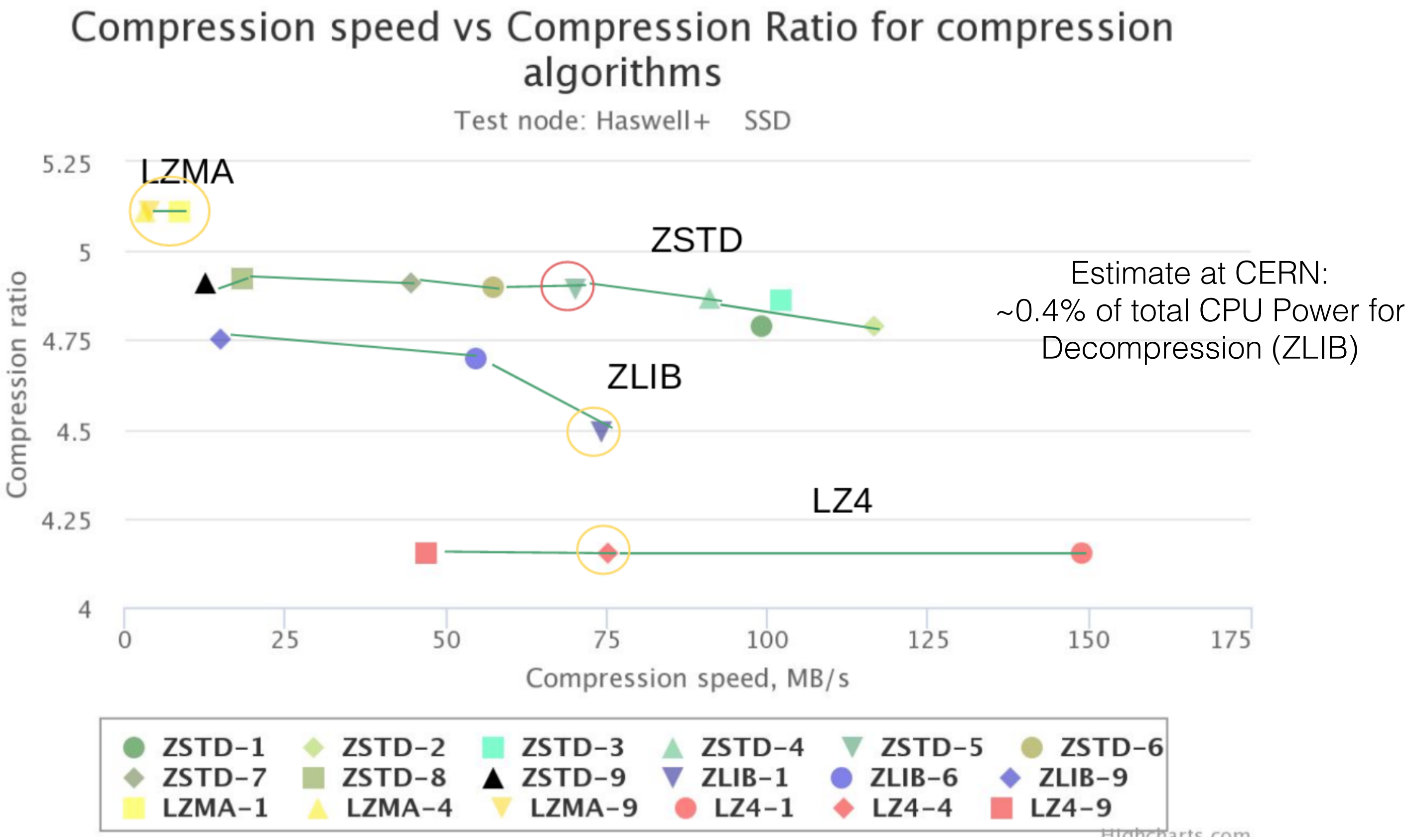


- ***ROOT format optimized*** for small size and minimum IO payload

- predictable read patterns allows to use latency compensation techniques (see the following)

jobs@CERN like 100.000 people watching all a different movie with 1 MB/s streaming average

Data Formats & Compression Algorithms



- often **compression** done on **application side**
- **LZMA** cheapest for storage, most expensive for CPU
- best **algorithm** has to be selected **per use case** (de-/compression speed)
- compression inside storage systems rarely a benefit for physics data

Data Transformations to improve Compression Efficiency

Depending on Data Type, people use delta, zigzag, byte **transpositions** etc.

1. **reshuffle** data in memory
2. **compress** reshuffled data

byte 1	byte 2	byte 3	byte 4	Integer
0	1	2	3	value 1
4	5	6	7	value 2
8	9	10	11	value 3
12	13	14	15	value 4

value 1	value 2	value 3	value 4	
0	4	8	12	bytes 1
1	5	9	13	bytes 2
2	6	10	14	bytes 3
3	7	11	15	bytes 4

example of byte transposition for an int32

File Size Reduction of a CMS Nano file applying transformations

CODEC (l=9)	Filter	FileSize	FileSize no Filter	Size Overhead	No Filt
zstd	T+Z(0,4,8)	2486704370	2942503102	+9.9%	+30.1%
lz4	T+Z(0,4,8)	3017303943	3794744333	+33.4%	+67.8%

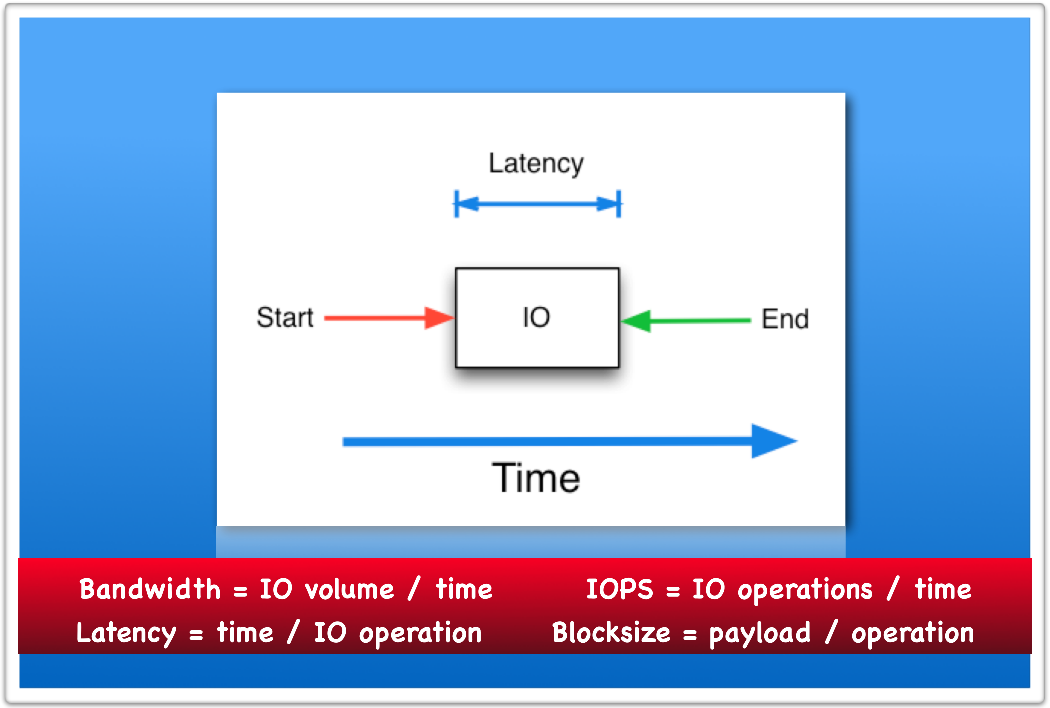
ROOT **RNTuple**

- next generation ROOT data format benefits from these

Same Data requires **~15% less space**

Optimizations used in IO systems

IO Language



- Bandwidth
- IOPS
- Latency
- Blocksize

The standard model of storage systems

$$\mathcal{L}_{SM} = \underbrace{\frac{1}{4}W_{\mu\nu}\cdot W^{\mu\nu} - \frac{1}{4}B_{\mu\nu}B^{\mu\nu} - \frac{1}{4}G^{a}_{\mu\nu}G^{\mu\nu}_{a}}_{\text{kinetic energies and self-interactions of the gauge bosons}}$$
$$+ \underbrace{\bar{L}\gamma^{\mu}\left(i\partial_{\mu} - \frac{1}{2}g\tau\cdot W_{\mu} - \frac{1}{2}g'YB_{\mu}\right)L + \bar{R}\gamma^{\mu}\left(i\partial_{\mu} - \frac{1}{2}g'YB_{\mu}\right)R}_{\text{kinetic energies and electroweak interactions of fermions}}$$
$$+ \underbrace{\frac{1}{2}\left|\left(i\partial_{\mu} - \frac{1}{2}g\tau\cdot W_{\mu} - \frac{1}{2}g'YB_{\mu}\right)\phi\right|^{2} - V(\phi)}_{W^{\pm}, Z, \gamma \text{ and Higgs masses and couplings}}$$
$$+ \underbrace{g''\left(\bar{q}\gamma^{\mu}T_{a}q\right)G^{a}_{\mu}}_{\text{interactions between quarks and gluons}} + \underbrace{\left(G_{1}\bar{L}\phi R + G_{2}\bar{L}\phi_{c}R + h.c.\right)}_{\text{fermion masses and couplings to Higgs}}$$

Storage → **Meta Data** + **Data**

implicit
explicit

IO Type Categories

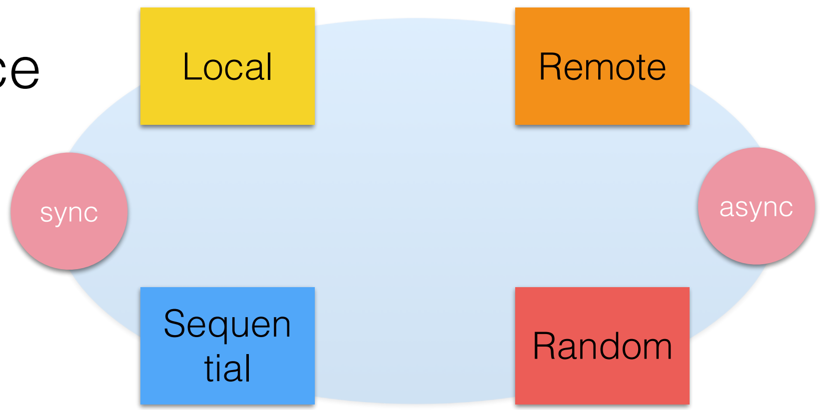
local storage device

end-user analysis

forward reading

video streaming

bulk data analysis



remote storage device

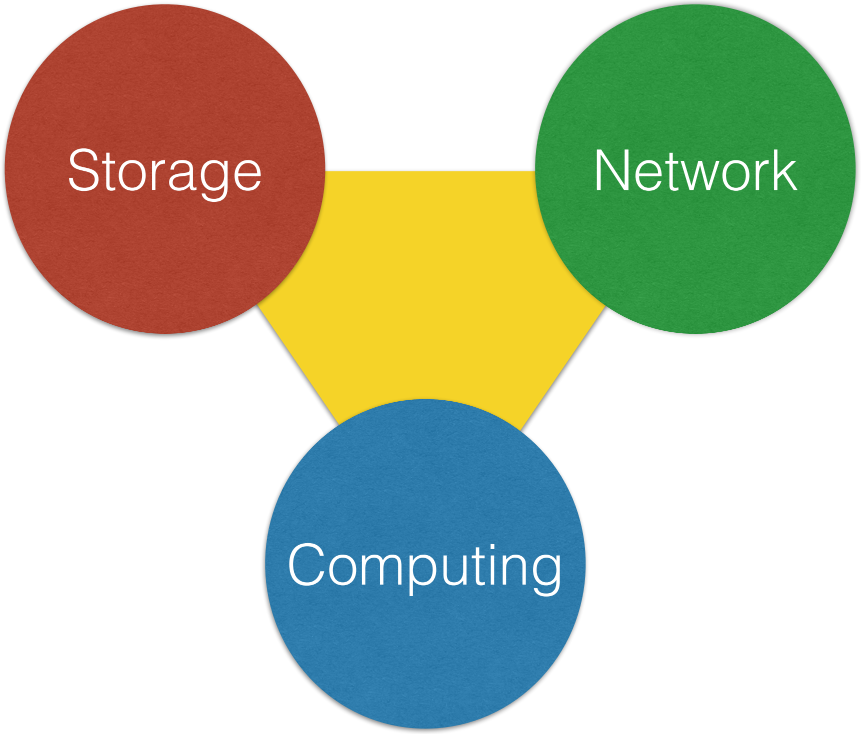
performance baseline
often given by network

big data analysis

seek + read (sparse)

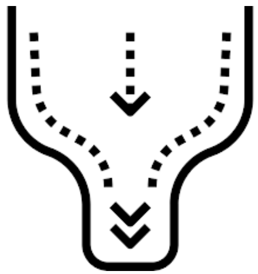
selective data analysis

IO foundation



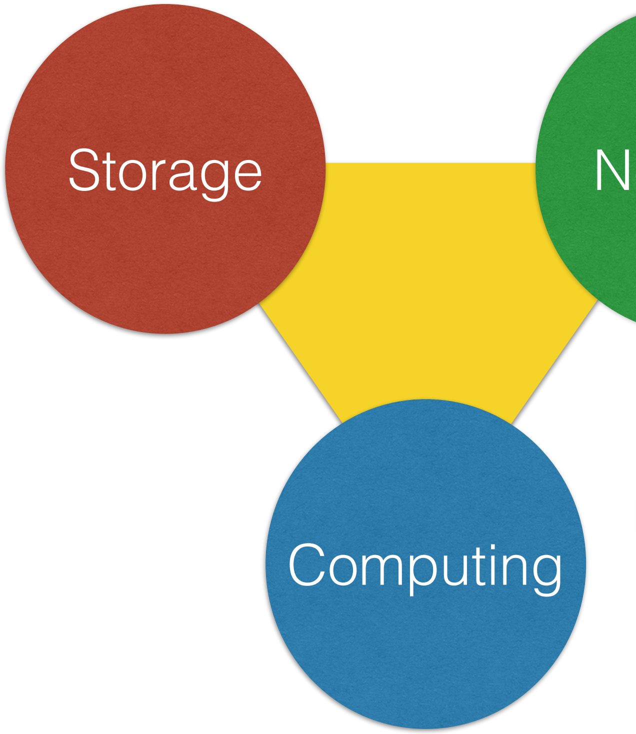
Building blocks for an IO system …

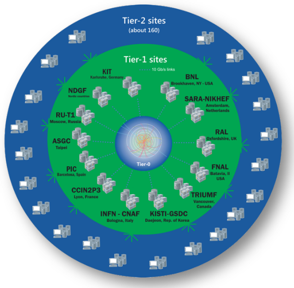
= Bottlenecks for an IO system …



Storage System Flavours

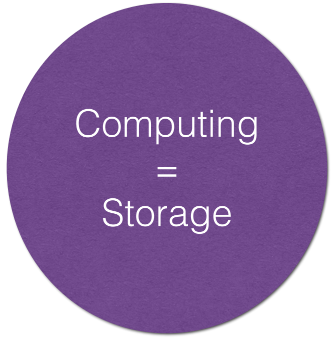
Distributed Storage Systems





The GRID!

Hyper-converged Storage Systems

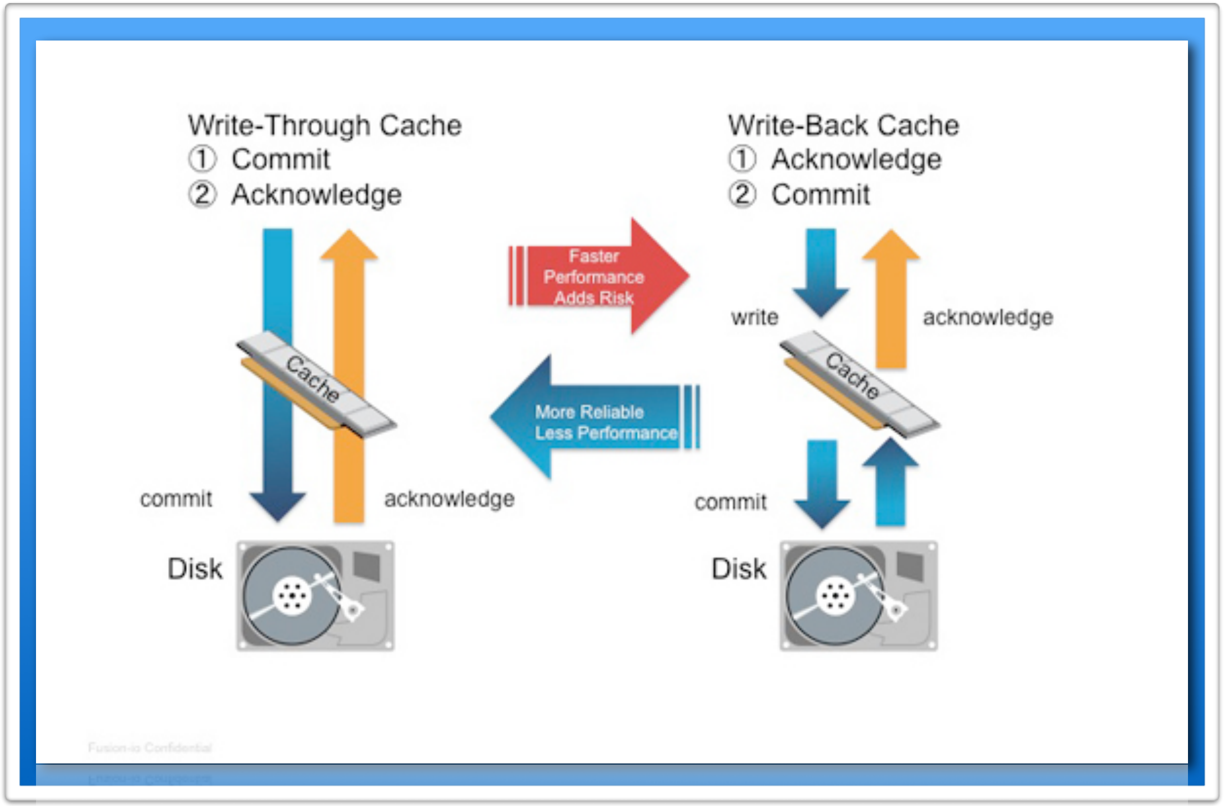


Hadoop!

Caching Strategies

Which strategy is best for low latency?

What is the latency difference when reading?



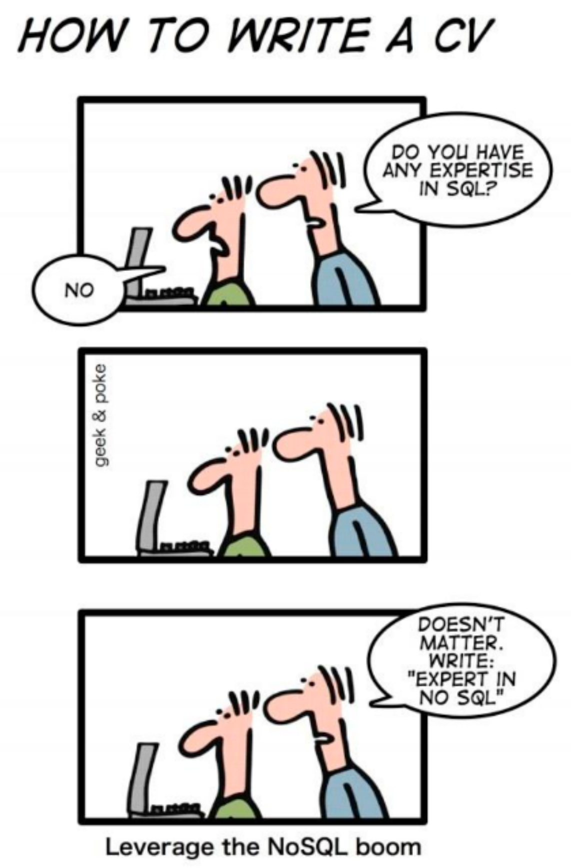
What is the danger when using a write-back cache?

Data Stores & APIs

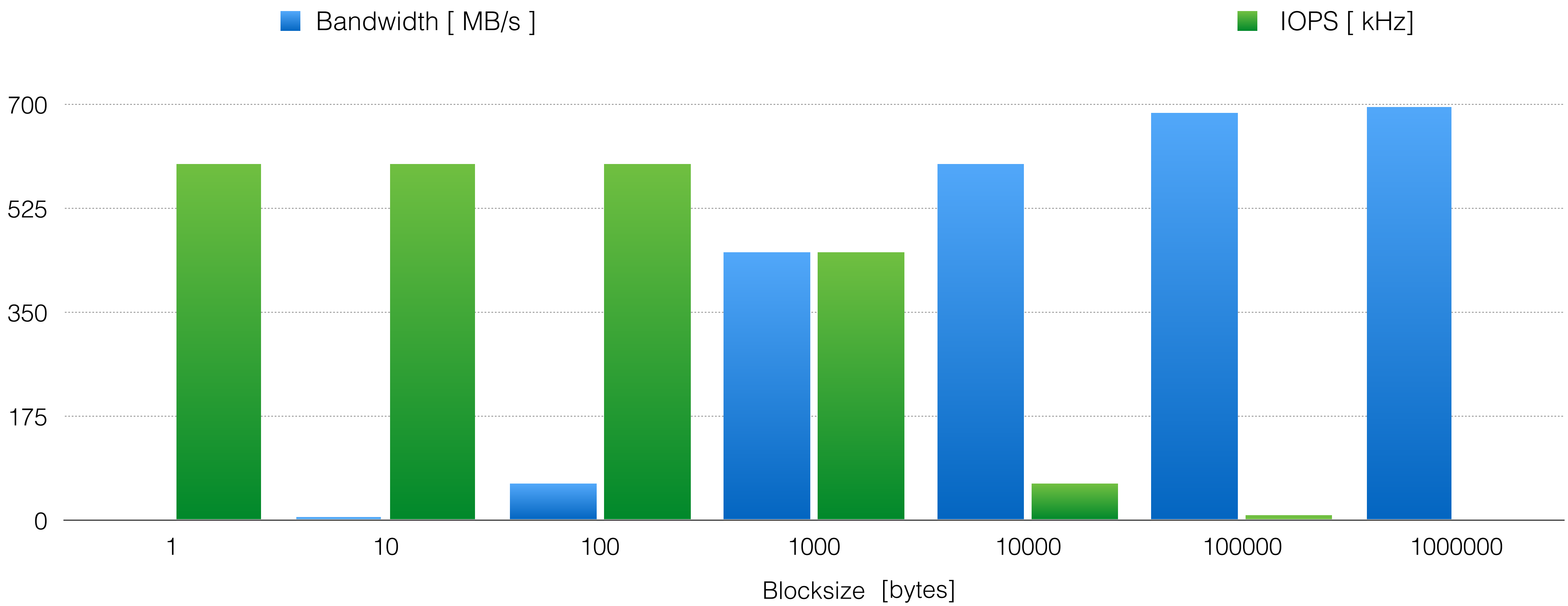
- File Systems : *hierarchical namespace (tree structure)*
  POSIX `open, read, write, close file`

- 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



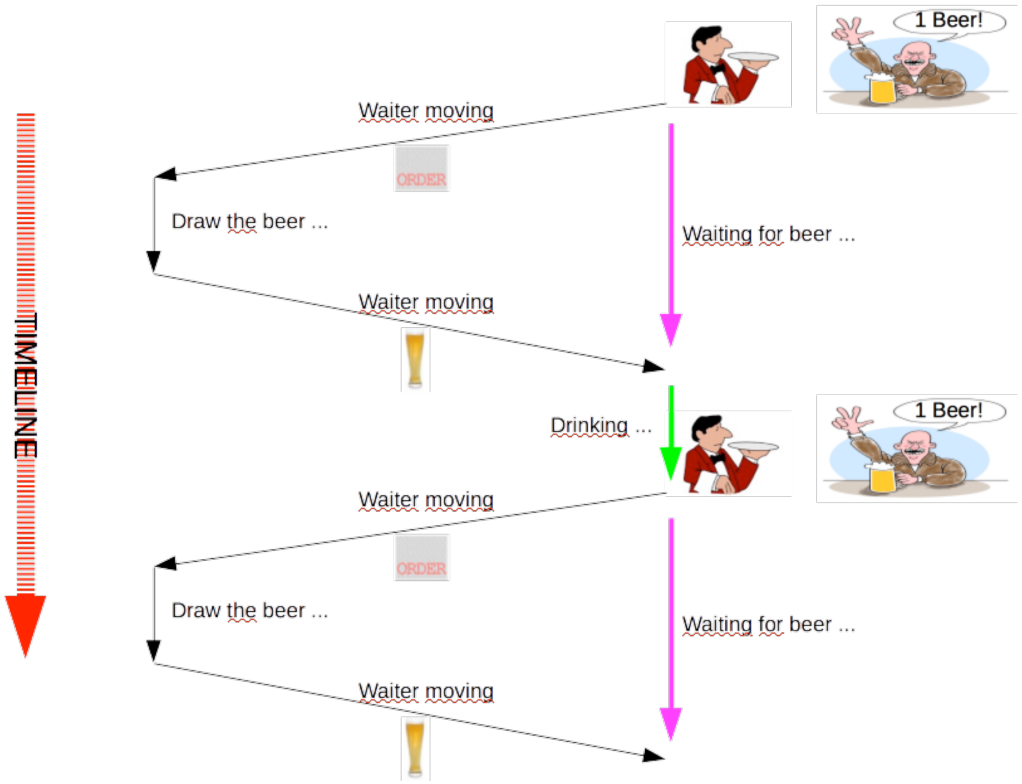
Which bandwidth bottleneck can you see?
Which IOPS bottleneck can you see?

Impact of **Latency**

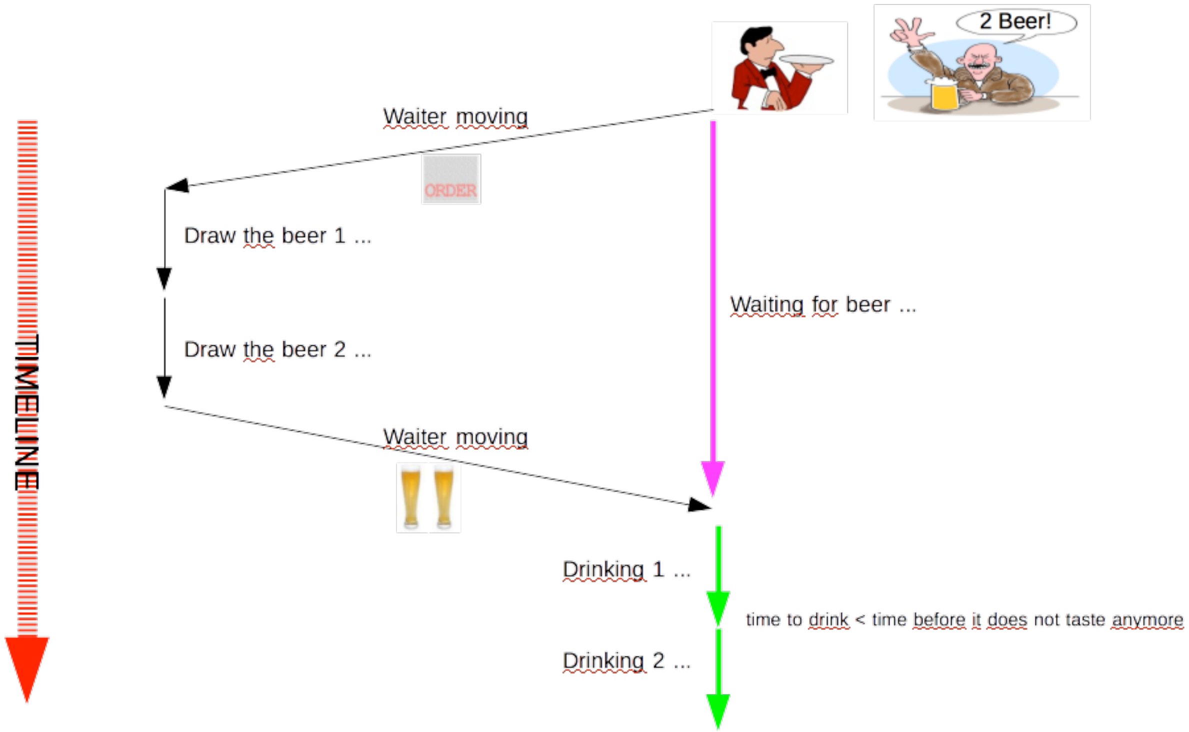
in analysis use cases

- **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

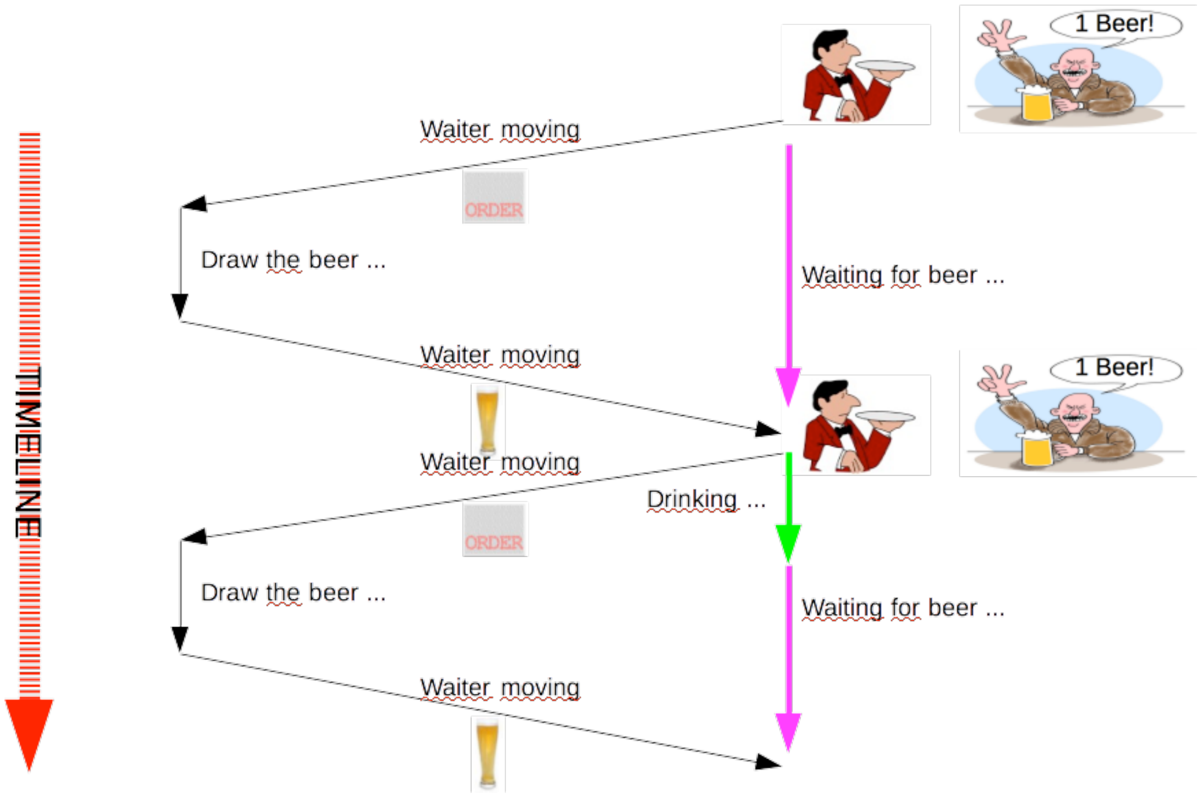
Analogy: ordering beer with a high-latency waiter - uncompensated



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

1 Redundancy Technology

- **RAID** technology

2nd hour today

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

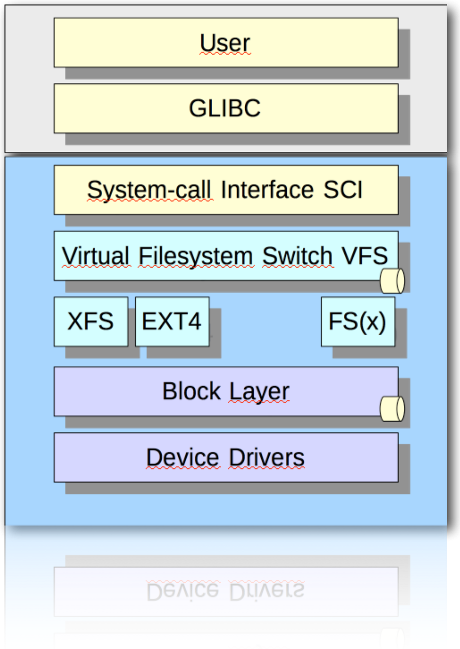
2 Cloud Storage Technology

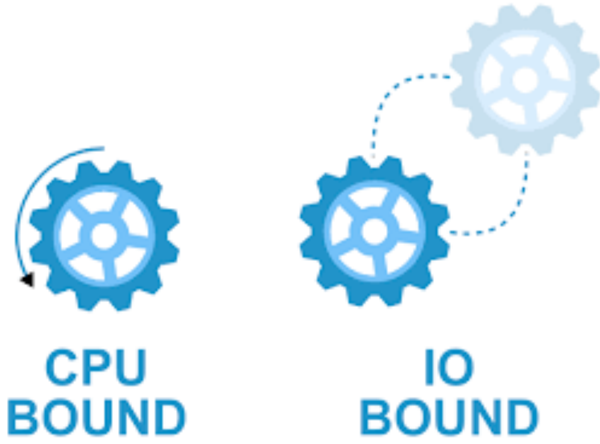
3rd + 4th hour Monday

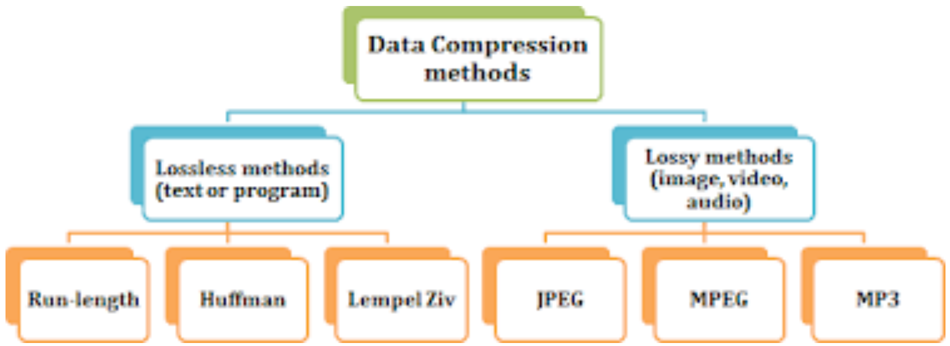
- **Scalability, Replication, Namespace, Placement toy MonteCarlo**

Exercises at https://cern.ch/setcp

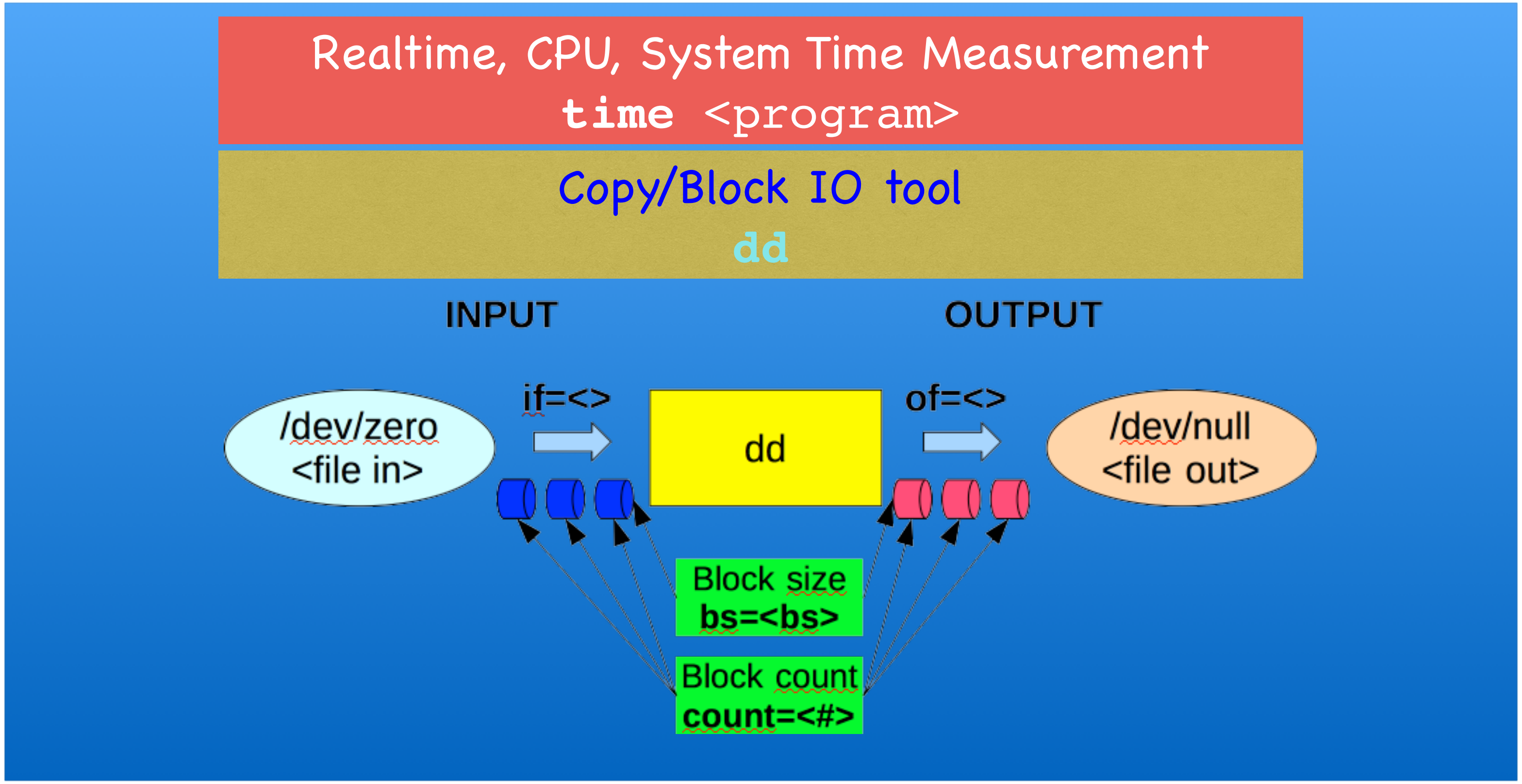
Tutorial - Exercise 0







Useful Linux Command



Useful Linux Commands

- Trace system calls of a program

  `strace <program> [program args]`

  `-c` count sys calls

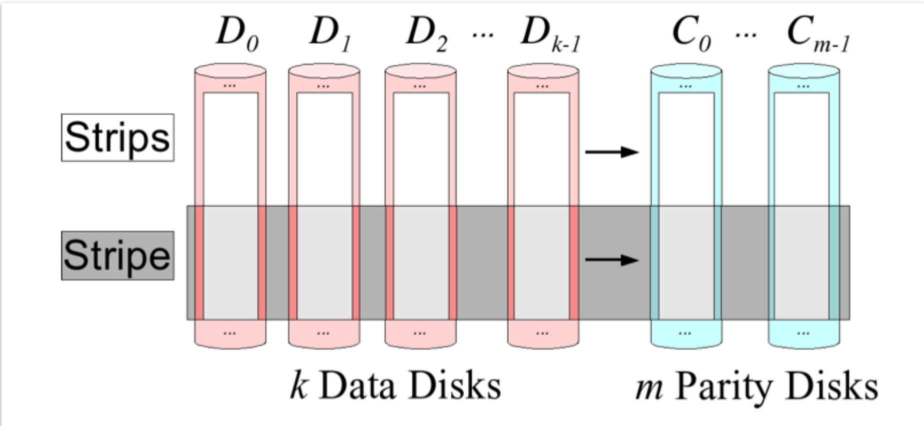
  `-ttt` show high time resolution

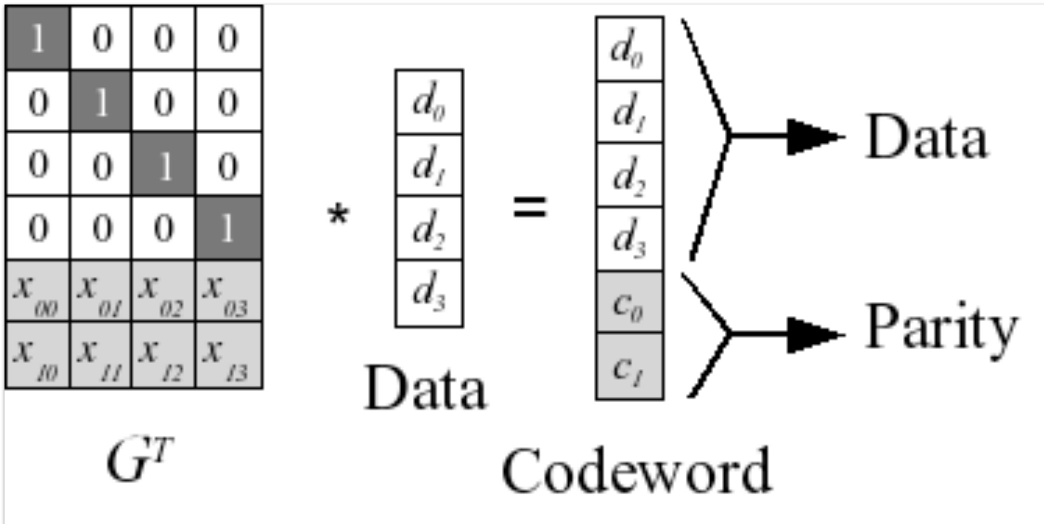
  → `man strace` !

Tutorial - Exercise 1

How to store ...

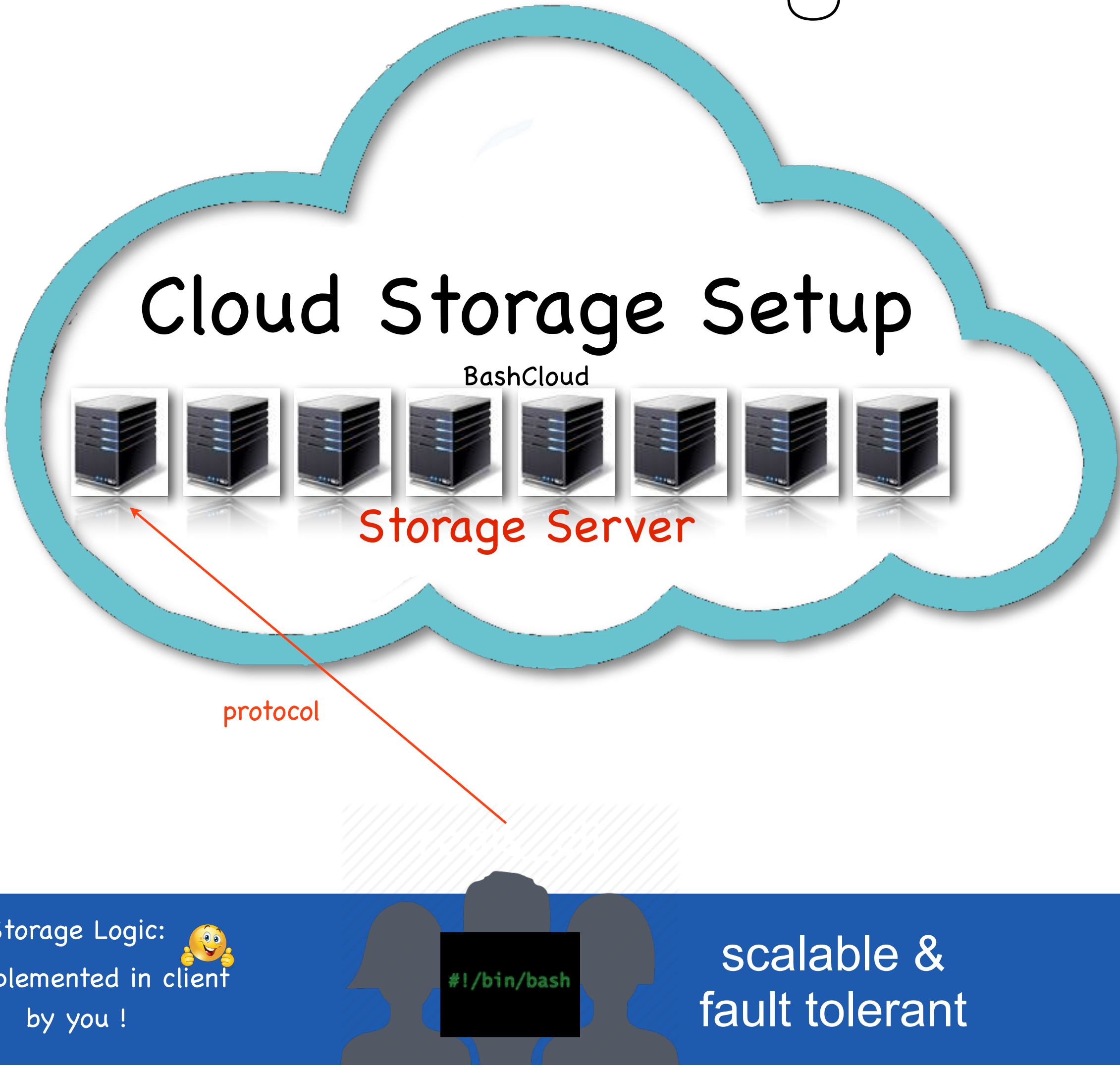
Redundancy RAID/RAIN





Tutorial - Exercise 2

Cloud Storage

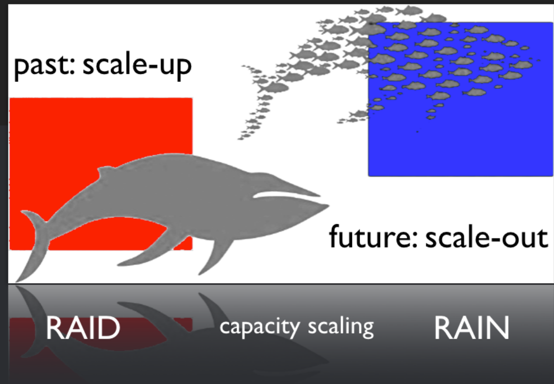


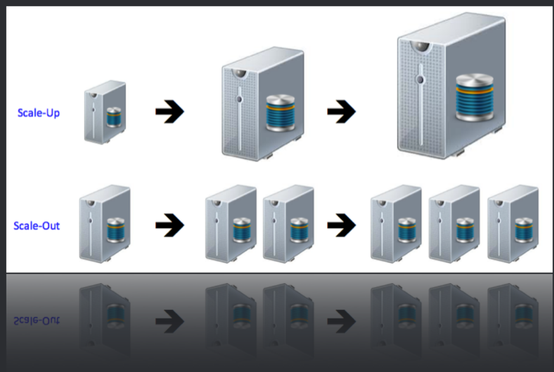
Storage Logic:
→ implemented in client by you !

scalable &
fault tolerant

Cloud Storage= Scale-Out Storage

‘Big Data - The Solution’
From Scale-Up to Scale-Out Storage

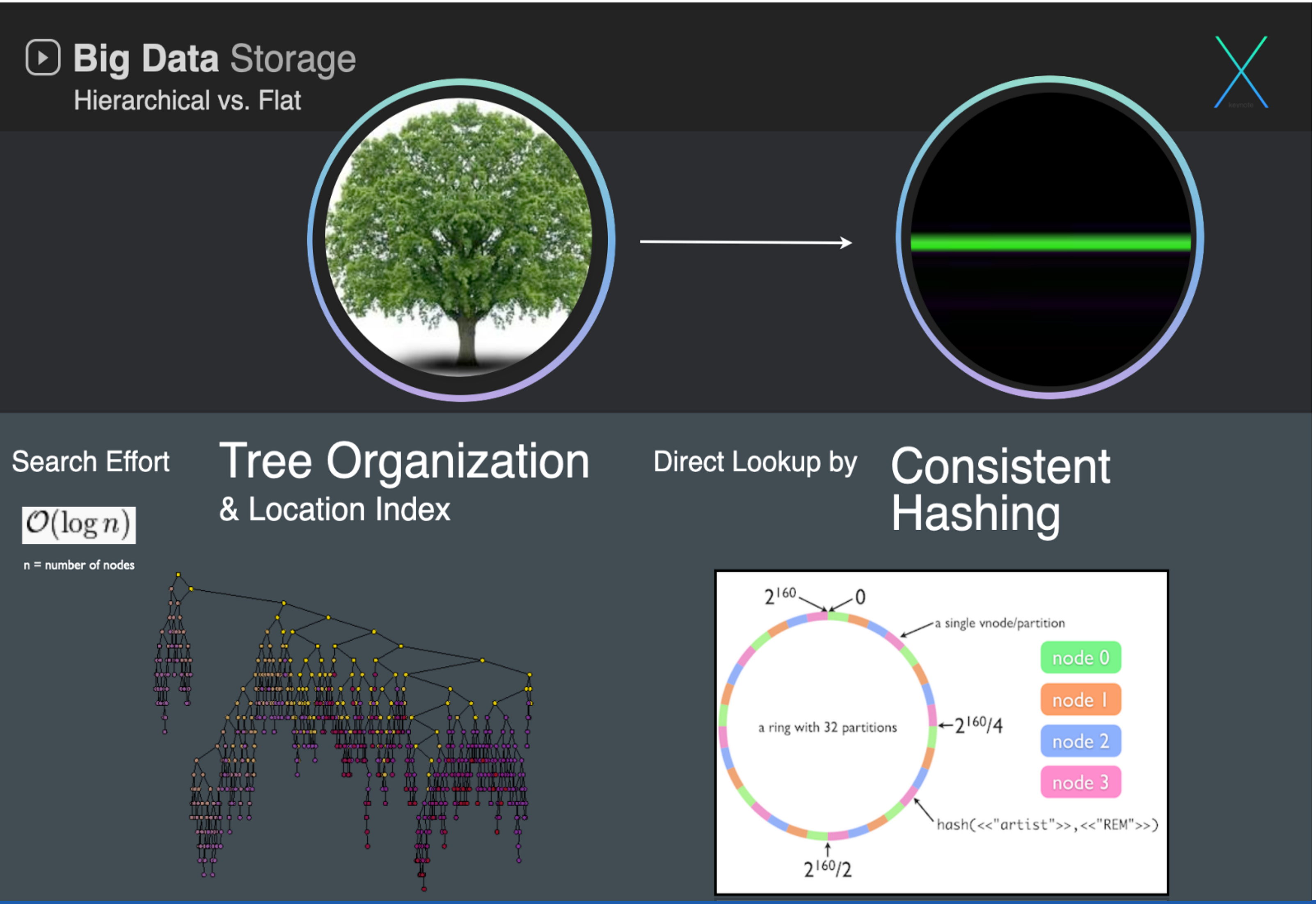




- structured data with central view & strong consistency
- predictable access
- SPOF
- slow growth rate

- 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?

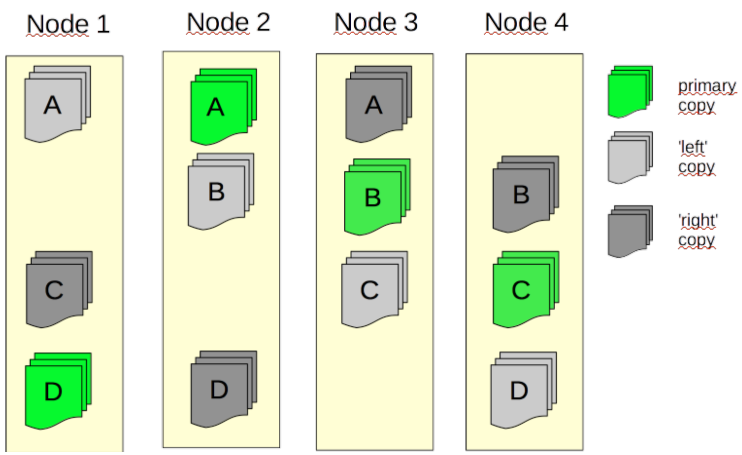
Filesytem:
tree search

Object Storage:
consistent hashing

Cloud / Object Storage

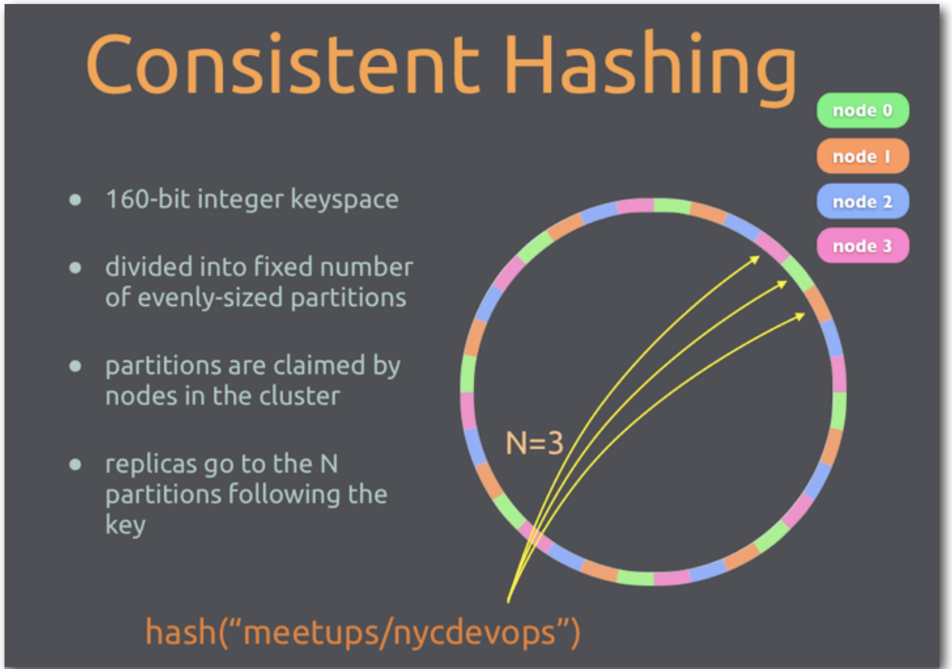
- basic principles for the exercise
  - sharding: files are placed and located using a distributed hash table (DHT)
    - the DHT can be changed to change the storage configuration
    - files are located computing the SHA1 checksum of their filename in hex representation
  - files get replicated to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

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

https://cern.ch/setcp

THANK YOU