Università degli Studi di Padova

J. Pazzini
PADOVA UNIVERSITY, INFN

2 - DATA STORAGE

Management and Analysis of Physics Datasets - Module B

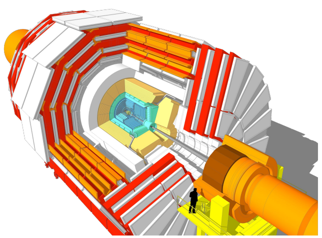
Physics of Data

A.A. 2023/2024

DATA STORAGE NEEDS

Different necessities for various data-storage solutions:

Raw Data



+ (MC) Simulations

DAQ

Write continuously at high-rate

Processing

Write every few hours / days

Reprocessing

Write every ~year

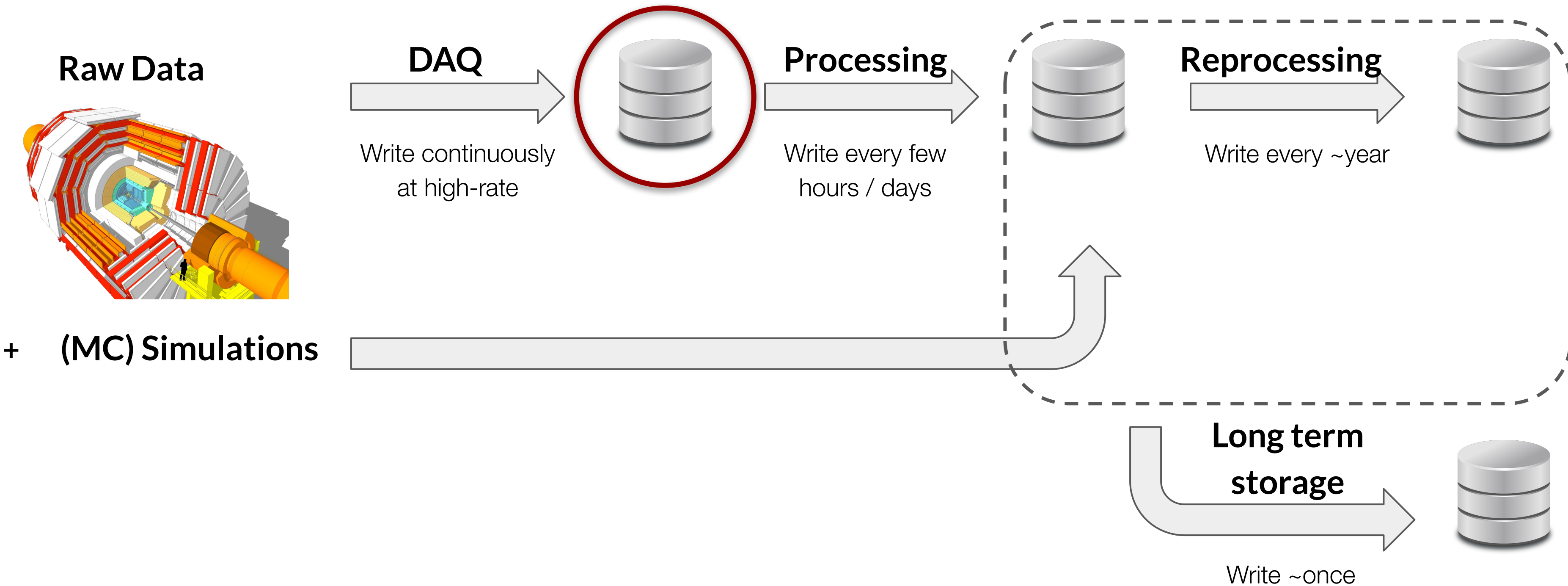
Long term storage

Write ~once

DATA STORAGE NEEDS

Different necessities for various data-storage solutions:

Large throughput + huge number of operations + small amount of data stored

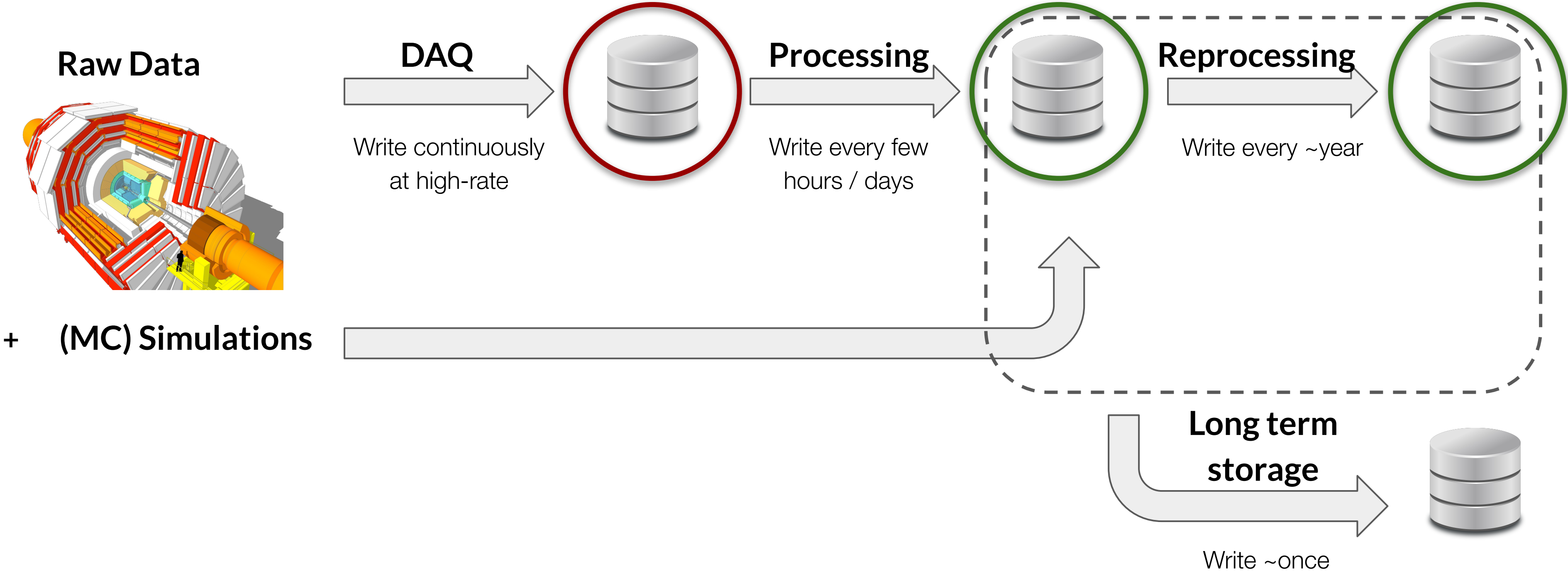


DATA STORAGE NEEDS

Different necessities for various data-storage solutions:

Large throughput + huge number of operations + small amount of data stored

Average number of operations + large amount of data stored



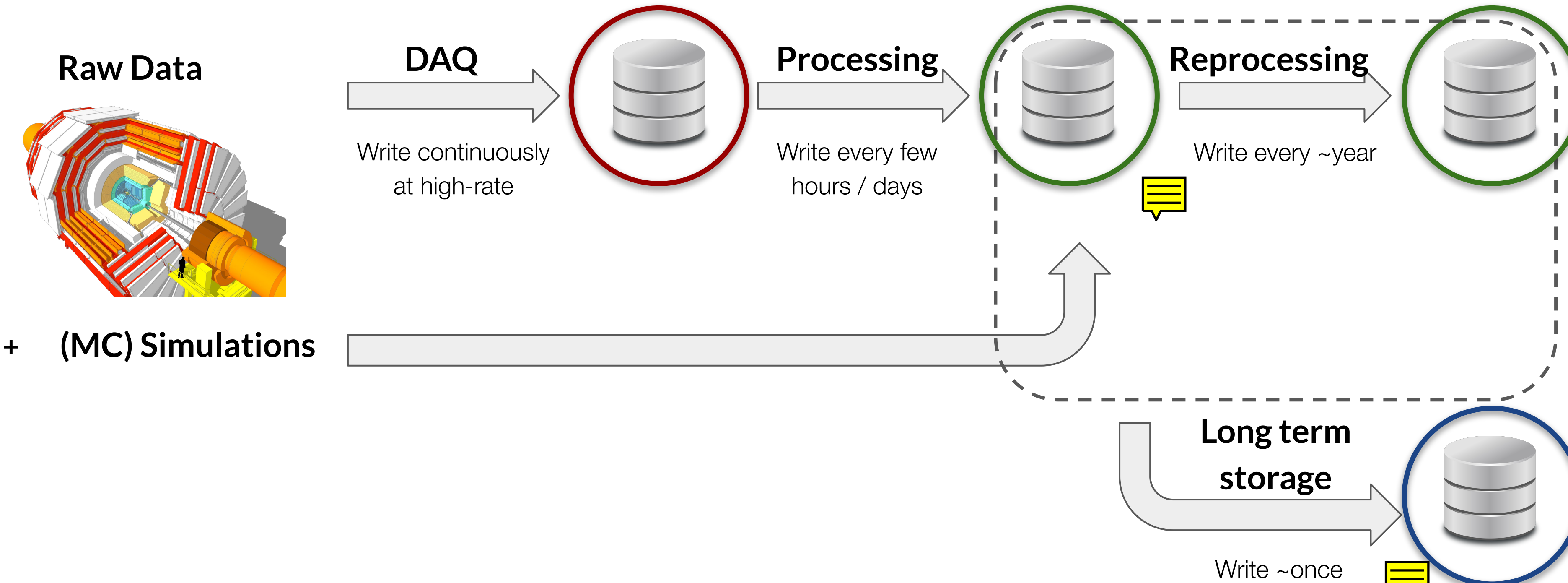
DATA STORAGE NEEDS

Different necessities for various data-storage solutions:

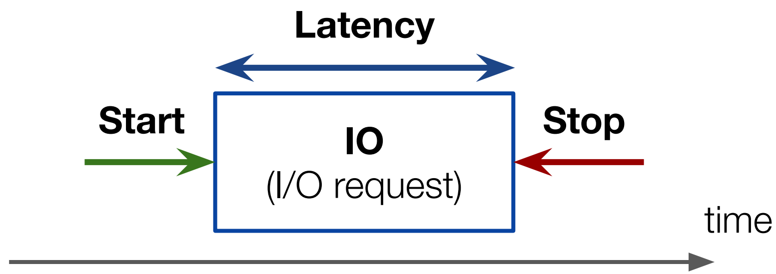
Large throughput + huge number of operations + small amount of data stored

Average number of operations + large amount of data stored

Write once + read once every few years + very large amount of data stored



STORAGE CONCEPTS AND DEFINITIONS



IO: a single request for an Input/Output operation (e.g. read/write/update/…)

Latency: time taken to complete a single IO request

[IO] Rate: number of IO that can be performed by a storage in a second of a given workload

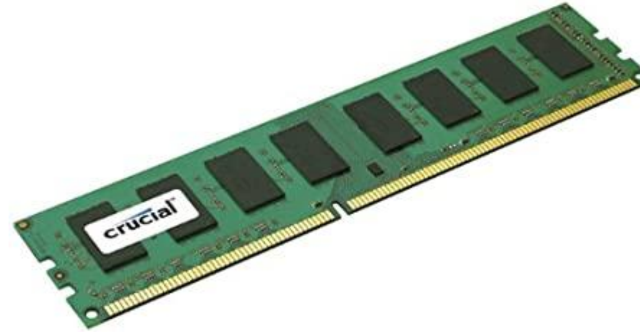
IO operations per second → ***IOPS***

Block size: average size (in bytes) of a IO requests of a given workload

Throughput: data transfer rate to and from the storage media in megabytes per second

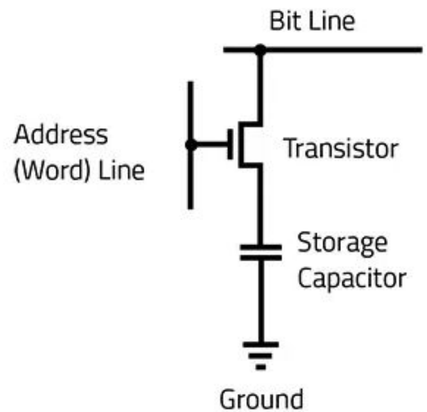
⇒ Throughput = [IO block size] x [IOPS]

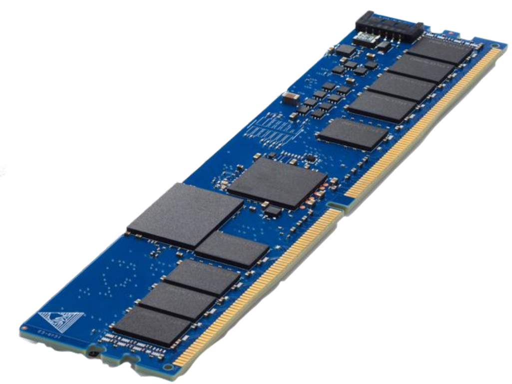
DRAM and Non-Volatile Memory



DRAM

- Semiconductor memory technology (NAND cells storing 1 bit of data)
- Data is not persisted, only temporary storag cells (capacitors and transistors)
- Extremely low latency (0.1 μs)





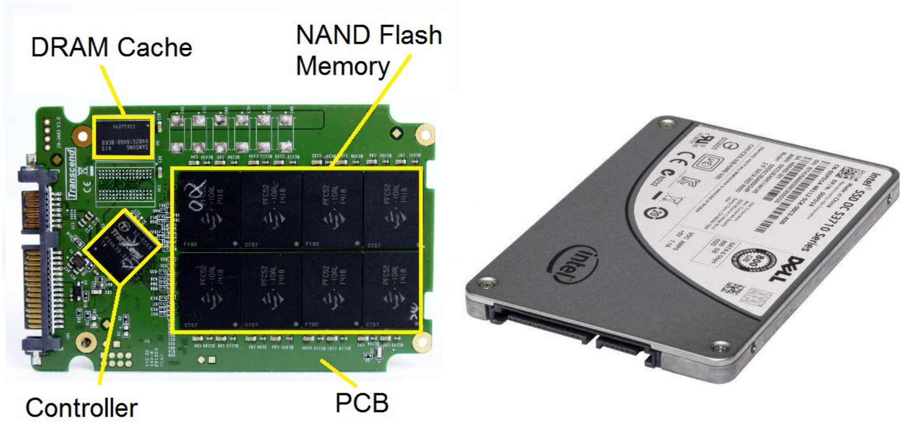
Non-volatile memory (NVM)

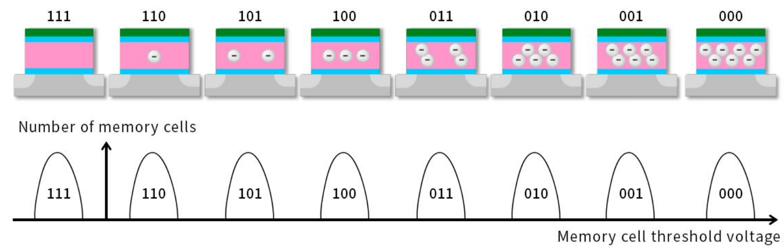
- A number of different technologies
- Hold data even if device is turned off
- Higher storage capacity than DRAM
- Very low latency (1 μs)

Extremely specialized solution for storage
Very rarely used in “standard” applications

Solid State Drives (SSDs)

- Mostly based on NAND flash chipset for data storage and a controller for caching, error handling, etc
- Commonly Multi-Level charge trap instead of a single-value NAND cell
- No mechanical components (as in DRAMs and NVMs)
- Optimized performance implementations available (such as NVM Express SSDs)
- Relatively short latency (10-100 μs)
- Fast READ but slower WRITE IO
- Limited number of WRITE per block over a SSD lifetime





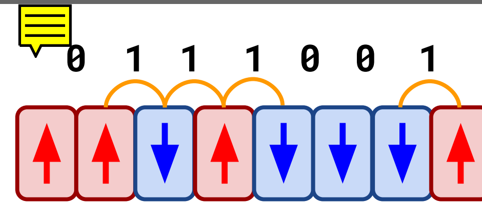
Have become quite widespread as their price deceased over time

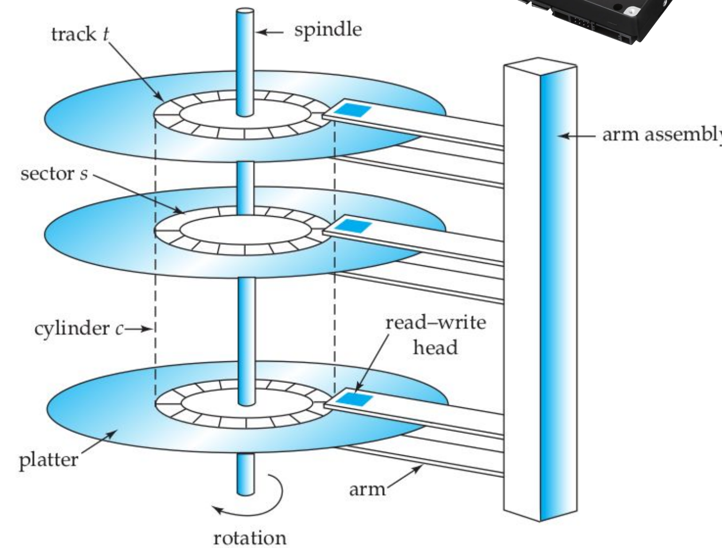
Useful for large-storage low-latency applications

Magnetic (Hard) Disks Drives (HDDs)

- A number of (double-sided) spinning platters covered in magnetic media
- Every bit of information is actually a flip in the magnetic field across two domains
- Data is stored in contiguous sectors (typically ~0.5 to 4 kBytes) of the platter tracks
- IO requests are managed by a moving arm assembly placing its heads on the appropriate track to READ/WRITE data from sectors

- A number of latency sources (10 **m**s):
  1. Time to move the head to the proper track (seek time)
  2. Time for the disk to rotate and align to the appropriate sector (rotation delay)





Still very widely used due to large capacity and lower cost wrt SSDs

Mostly for mid-high latency applications only

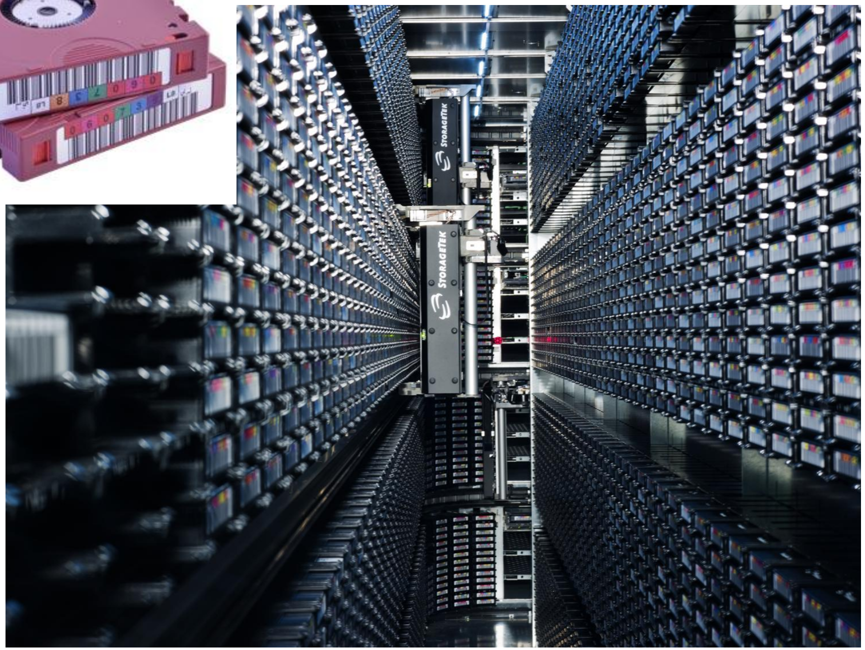
STORAGE MEDIA

Magnetic Tape (yes, really…)

- Magnetic tape rolled up in cartridges
- Usually managed by robotic arms to move/swap cartridges
- Data (e.g. entire files) are stored sequentially (contiguous) over the tape

- From a IO request to the actual operation to be complete a number of steps occurs:
    1. Robotic arms fetch the cartridge
    2. Unroll the tape to the point where the data
    3. Read sequentially all memory locations

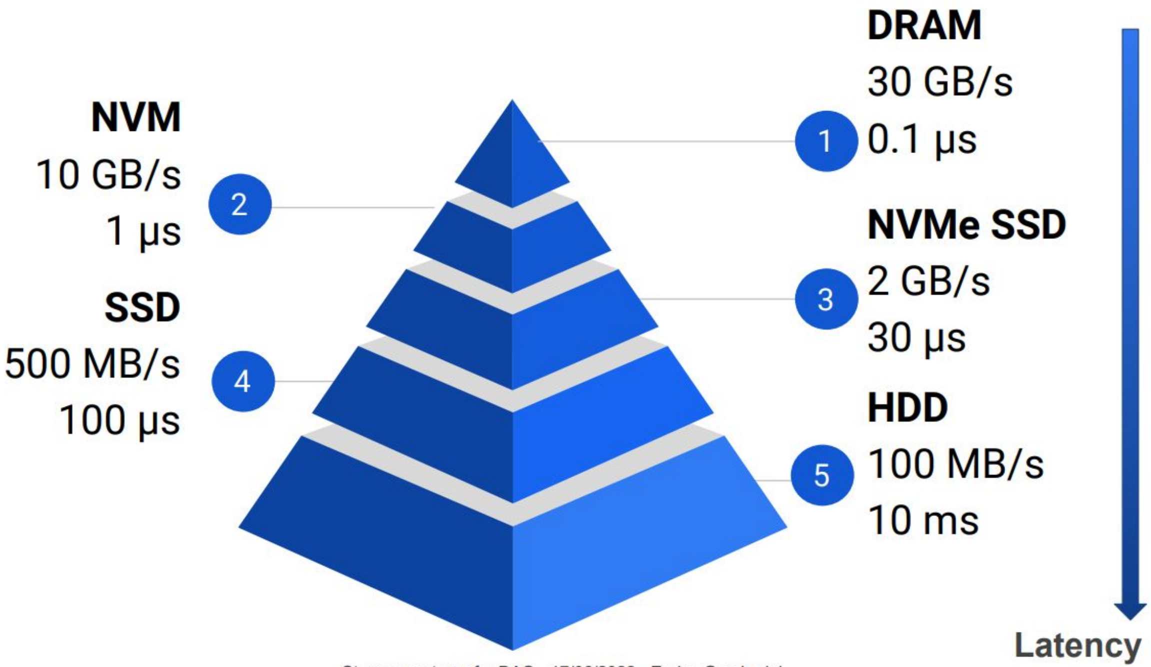
- While the actual READ IO is fast (!) the overall latency if of the order of 100 **s** (or more)



Used for write-once operations and long term safekeeping

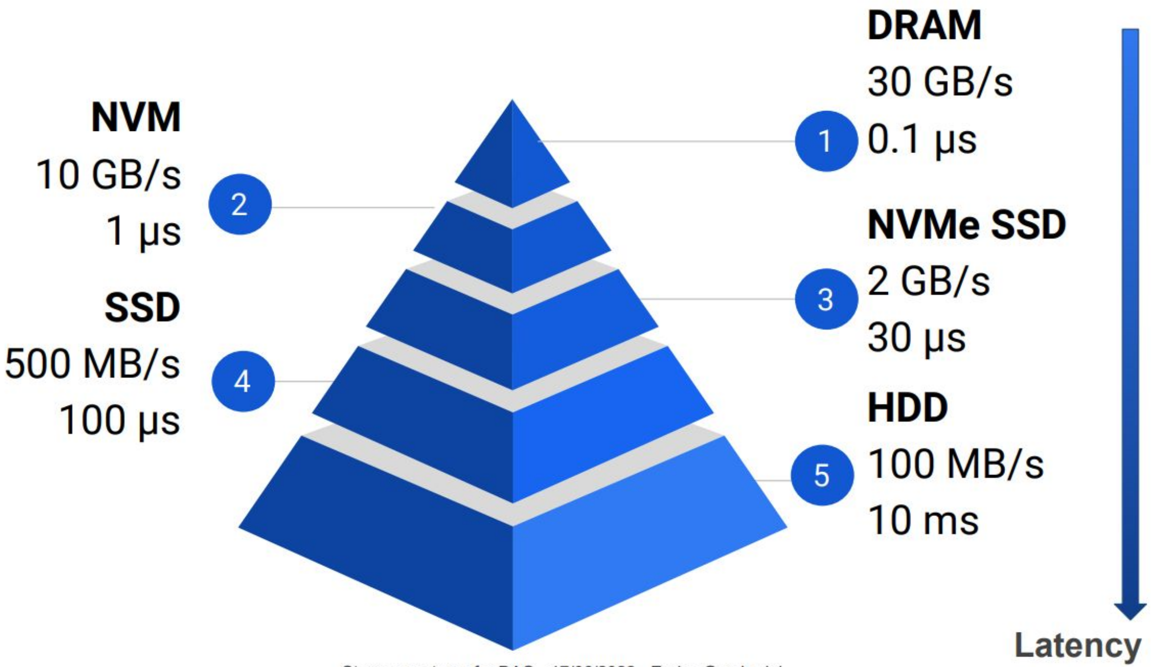
~0.8 EB (1 EB = 1 million TB) of data stored in tape @ CERN as of today

STRIKING A BALANCE



Storage systems for DAQ - 17/06/2022 - Enrico Gamberini

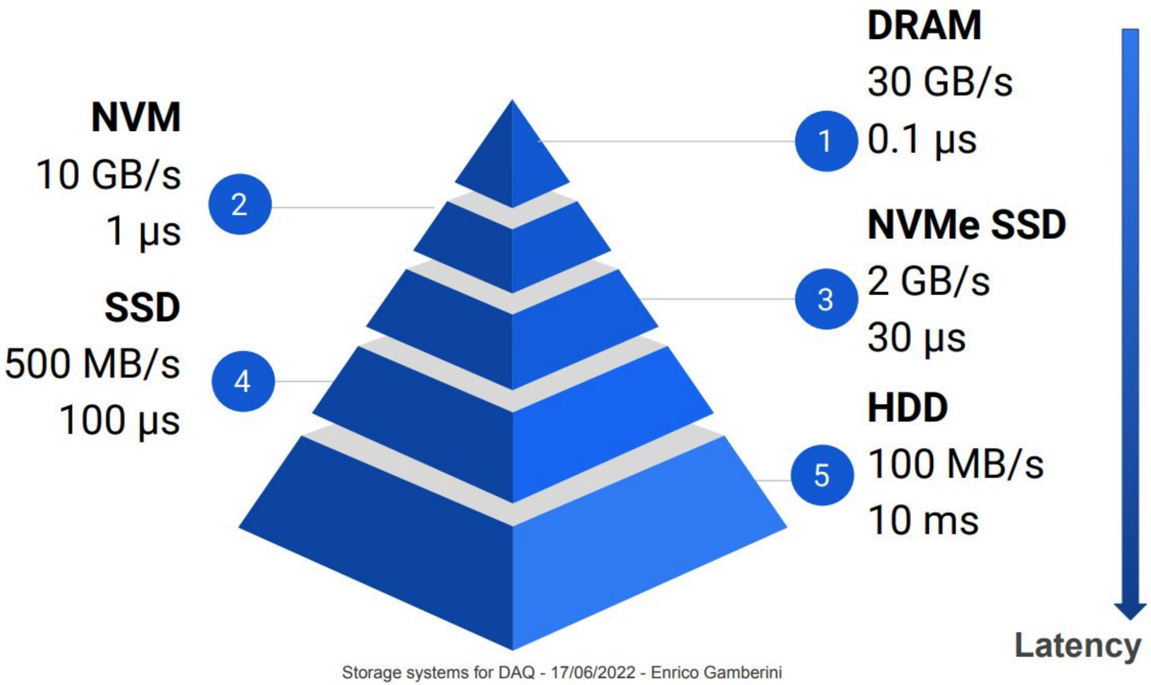
STRIKING A BALANCE



Storage systems for DAQ - 17/06/2022 - Enrico Gamberini

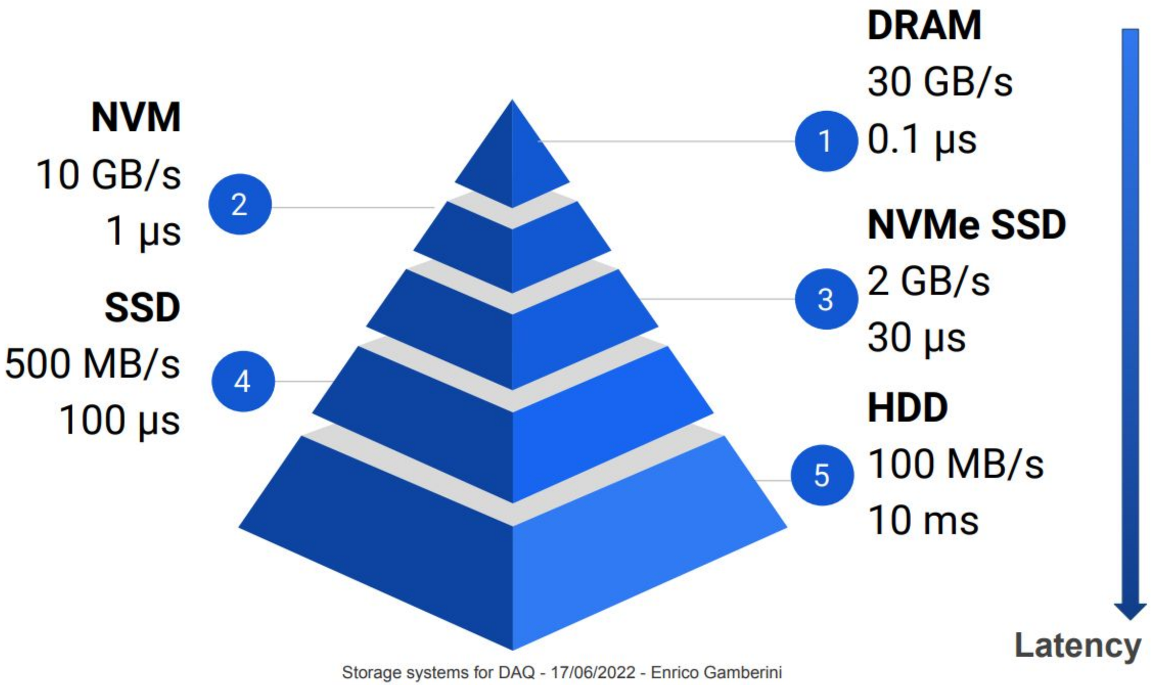
	Capacity per unit	Latency	$/TB	Speed
RAM	16 GB	10 ns	7000 $	$10\,\mathrm{GB\,s^{-1}}$
SSD	500 GB	10 µs	200 $	$1\,\mathrm{GB\,s^{-1}}$
HD	6 TB	3 ms	25 $	$150\,\mathrm{MB\,s^{-1}}$
Tape	20 TB	100 s	20 $	$500\,\mathrm{MB\,s^{-1}}$

STRIKING A BALANCE

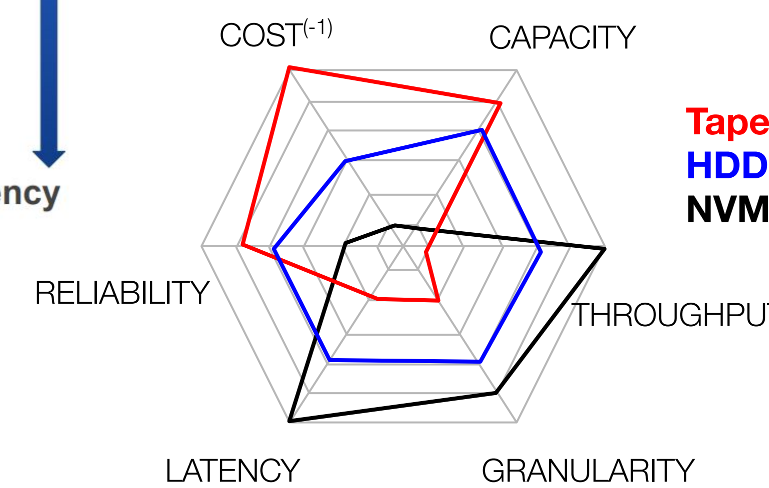


	Capacity per unit	Latency	$/TB	Speed	reliability
RAM	16 GB	10 ns	7000 $	$10\,\mathrm{GB\,s^{-1}}$	volatile
SSD	500 GB	10 µs	200 $	$1\,\mathrm{GB\,s^{-1}}$	poor
HD	6 TB	3 ms	25 $	$150\,\mathrm{MB\,s^{-1}}$	average
Tape	20 TB	100 s	20 $	$500\,\mathrm{MB\,s^{-1}}$	good

STRIKING A BALANCE



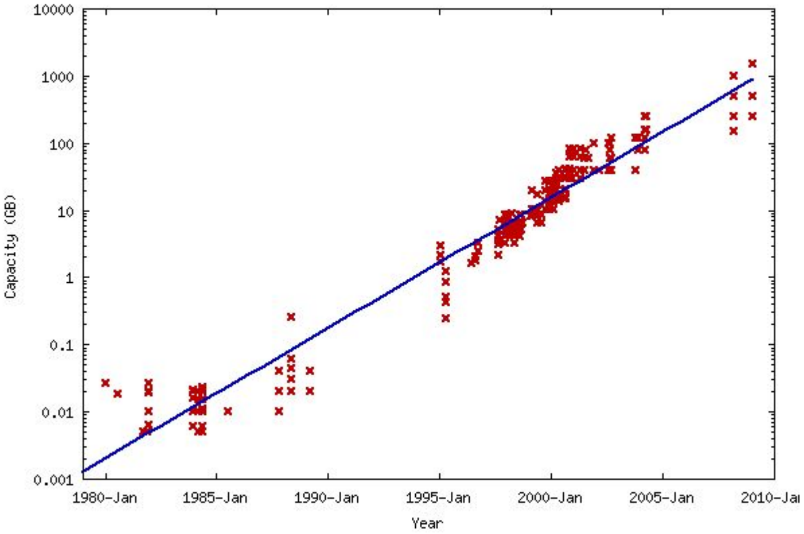
	Capacity per unit	Latency	\$/TB	Speed	reliability
RAM	16 GB	10 ns	7000 \$	$10\,\mathrm{GB\,s^{-1}}$	volatile
SSD	500 GB	10 µs	200 \$	$1\,\mathrm{GB\,s^{-1}}$	poor
HD	6 TB	3 ms	25 \$	$150\,\mathrm{MB\,s^{-1}}$	average
Tape	20 TB	100 s	20 \$	$500\,\mathrm{MB\,s^{-1}}$	good

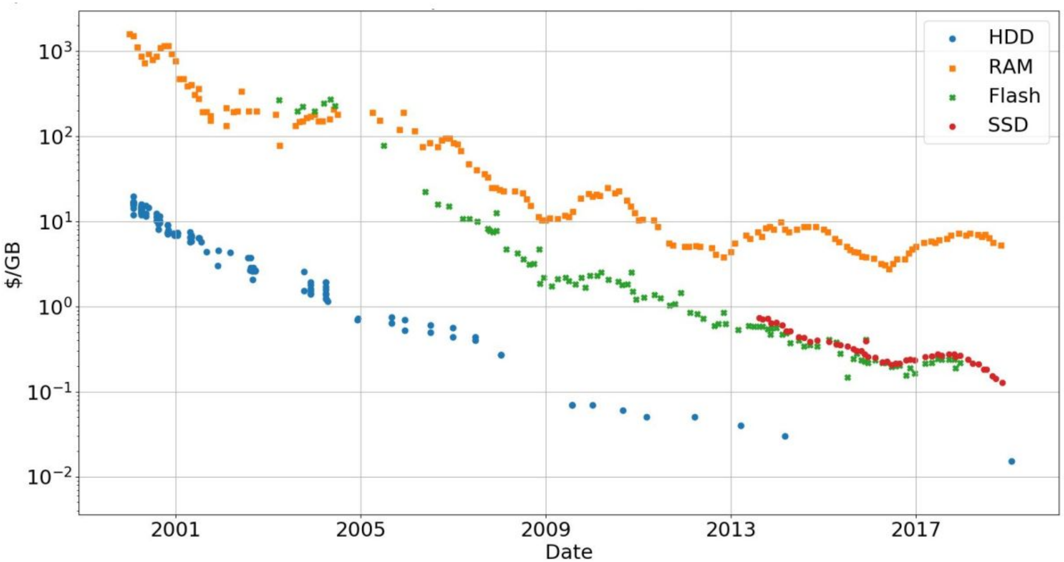


KRYDER'S LAW

Mark Kryder in 2005:
"The storage density (amount of data stored per unit of disk/chip area) doubles every 13 months"

This would imply a doubling of the storage capacity at a fixed cost ~ every year (***not really happening anymore***)





Storage systems for DAQ - 18/01/2020 - Adam Abed Abud

Nonetheless, the balance across all these figure of merits evolves with time following the technological advancements

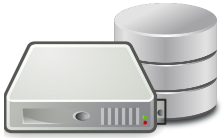
SCALABILITY

Most (if not all datasets) are expected to grow over time

- Ongoing experiments
- Industrial data
- Users' produced data
- …

How to handle an increase in the size of the dataset maintaining or even optimizing the performance?

Let's assume that we start from *here:* a system with a given storage solution
→ How do we scale this solution for a larger/increasing datasets?



SCALABILITY

Vertical (Up) → increase the capacity/performance of the available resources by buying better (more performant / higher capacity) HW

SCALABILITY

Vertical (Up) → increase the capacity/performance of the available resources by buying better (more performant / higher capacity) HW

Horizontal (Out) → chain together more resources buying additional *commodity HW* and creating a ***distributed system***

SCALABILITY

Vertical (Up) → *pros:*

- No real changes in how users access the resources
- Fast access time to the HW resources

cons:

- The larger the storage solution, the more expensive it gets
- There's an upper limit to the solution one could get at a given time (e.g. a max SSD storage capacity)
- Stop operations →upgrade →resume

SCALABILITY

Vertical (Up) → *pros:*

- No real changes in how users access the resources
- Fast access time to the HW resources

cons:

- The larger the storage solution, the more expensive it gets
- There's an upper limit to the solution one could get at a given time (e.g. a max SSD storage capacity)
- Stop operations →upgrade →resume

Horizontal (Out) → *pros:*

- Can grow to *very* large systems for cheap(er than scale-up)
- Easy to scale as you go (wire up or remove HW resources if/when needed)

cons:

- Multiple HW resources "connected" together → network bandwidth is a very important (possibly limiting) factor
- More complex solutions to access all (shared/distributed) HW resources → different from local systems

WHAT ABOUT *YOUR* STORAGE?

We can use a variety of tools to assess the characteristics of a storage solution.

Most commonly in `Linux` we can use `dd` and `fio`

`dd` is a unix/linux function that allows you to make copies of files at a block level, accessing the raw storage performances.

$$\texttt{\$ dd}\quad \overbrace{\texttt{if=[...]}}^{\text{Input file}}\quad \overbrace{\texttt{of=[...]}}^{\text{Output file}}\quad \overbrace{\texttt{bs=[...]}}^{\text{Block size (in Bytes)}}\quad \overbrace{\texttt{count=[...]}}^{\text{N blocks to copy}}$$

We can use it to measure the read/write throughput of your disk

WHAT ABOUT *YOUR* STORAGE?

Write throughput

Read from a "device file" full of zeros `/dev/zero` and write it as a file onto the disk in blocks of 10 MB

```
pazzini@pazzini-x1:~$ dd if=/dev/zero of=delete_this_file bs=10M count=100
100+0 records in
100+0 records out
1048576000 bytes (1,0 GB, 1000 MiB) copied, 0,631432 s, 1,7 GB/s
```

Read throughput

Read from a physical file and "fake" its writing into another device drive file that discards all data written to it `/dev/null`

```
pazzini@pazzini-x1:~$ dd if=delete_this_file of=/dev/null bs=10M count=100
100+0 records in
100+0 records out
1048576000 bytes (1,0 GB, 1000 MiB) copied, 0,178132 s, 5,9 GB/s
```

WHAT ABOUT *YOUR* STORAGE?

Write throughput

Read from a "device file" full of zeros `/dev/zero` and w e disk in blocks of 10 MB

```
pazzini@pazzini-x1:~$ dd if=/dev/zer count=100
100+0 records in
100+0 records out
1048576000 bytes (1,0 GB , 1,7 GB/s
```

Read throughput

Read from into another device drive file that discards all data wri

```
pazzini@pazz elete_this_file of=/dev/null bs=10M count=100
100+0 records
100+0 records
1048576000 byte ,0 GB, 1000 MiB) copied, 0,178132 s, 5,9 GB/s
```

BASED ON THROUGHPUT, GUESS THE TYPE OF STORAGE MEDIA