INFN CCR
Istituto Nazionale di Fisica Nucleare
Commissione Calcolo e Reti

Corso di formazione per neoassunti nelle attività di Computing

4–7 Mar 2024
LNF

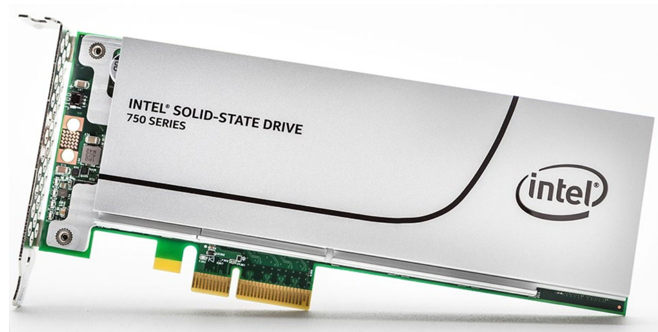
Dal laptop al supercalcolo

Alessandro Costantini, Daniele Cesini – INFN-CNAF
alessandro.costantini <at> cnaf.infn.it

Storage

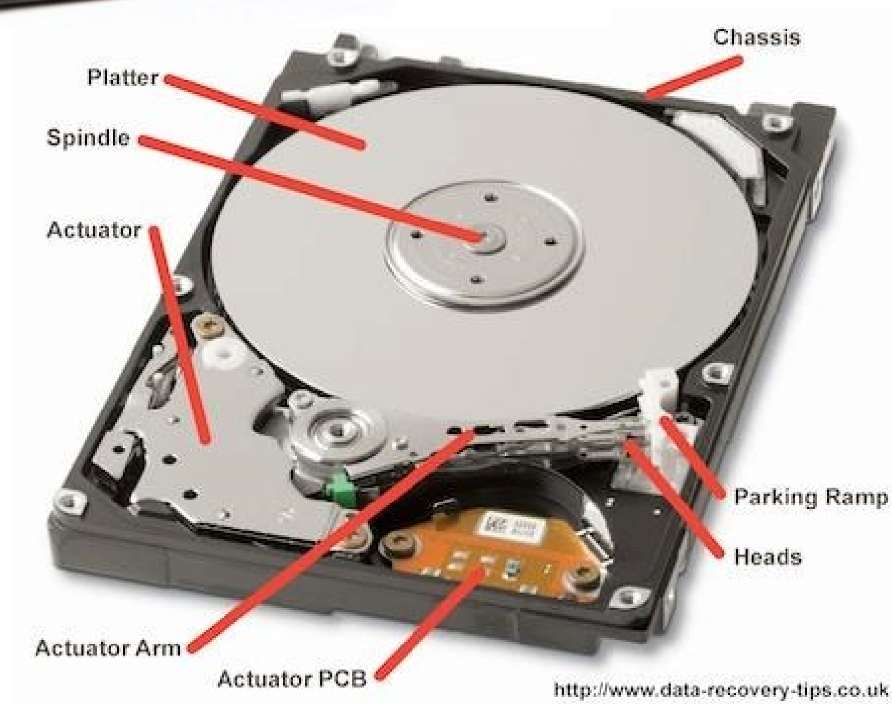
PC/Server storage

- Spinning disks
- Solid State Disks
- NVMe
- Tapes

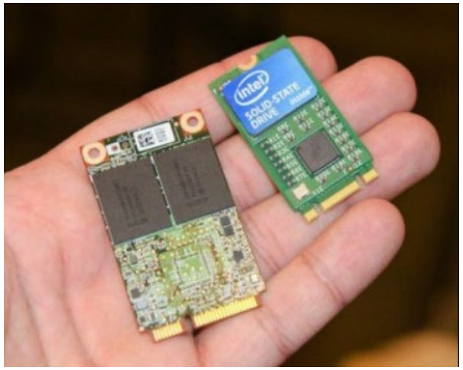


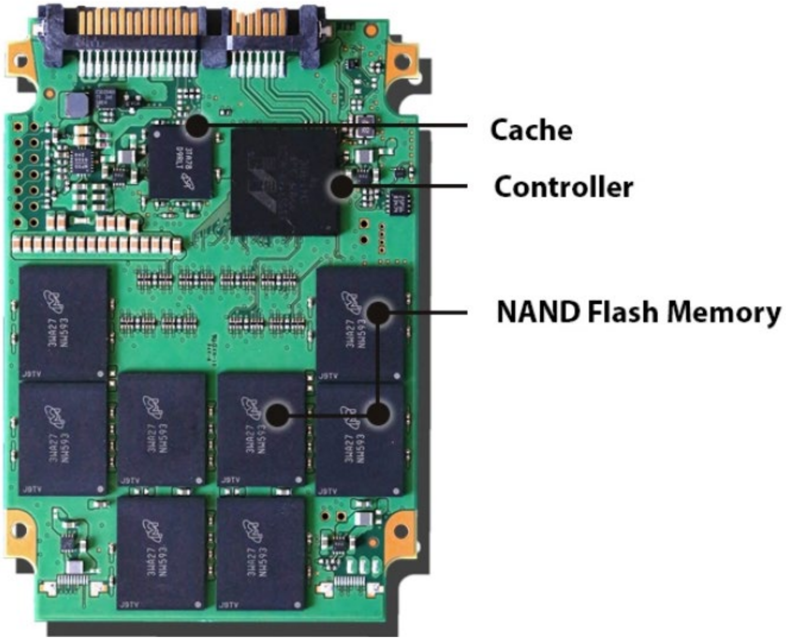
See also:
https://en.wikipedia.org/wiki/IOPS

+ # Inside an hard disk

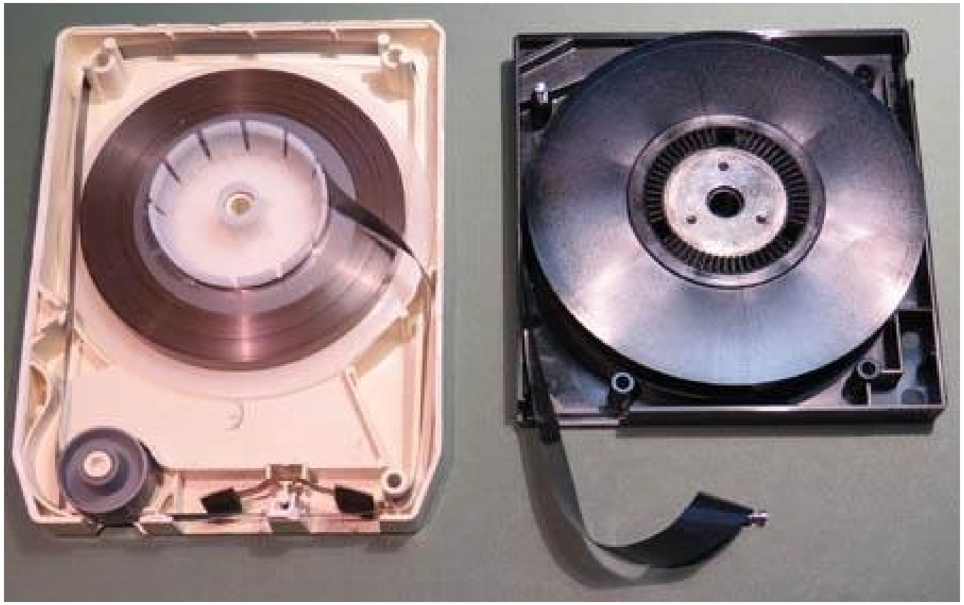


Inside an SSD





Inside a Tape Cartridge



Back to the '80s….

Commodore's datassette: a 90-minutes tape (45 minutes on each side) will hold on the order of 150 kilobytes on each side if no compression or fast loader is used.

Storage systems for a PC or server

Media	MB/s	IOPS	Capacity	Cost (Purchase)
HDD – Seagate Archive	100-150	100-200	8TB	$
SSD – Samsung EVO	400-500	100k-400k	500GB	$
NVMe Intel 400	2000 (read)	450k	400GB/1TB	$$
NVMe Violin 6000	4000	1M+	10TB	$$$
Tape T10000D	250MB/s	sequential	8.5TB	($$$)* Including driver and library

Different Quality of Services….different prices

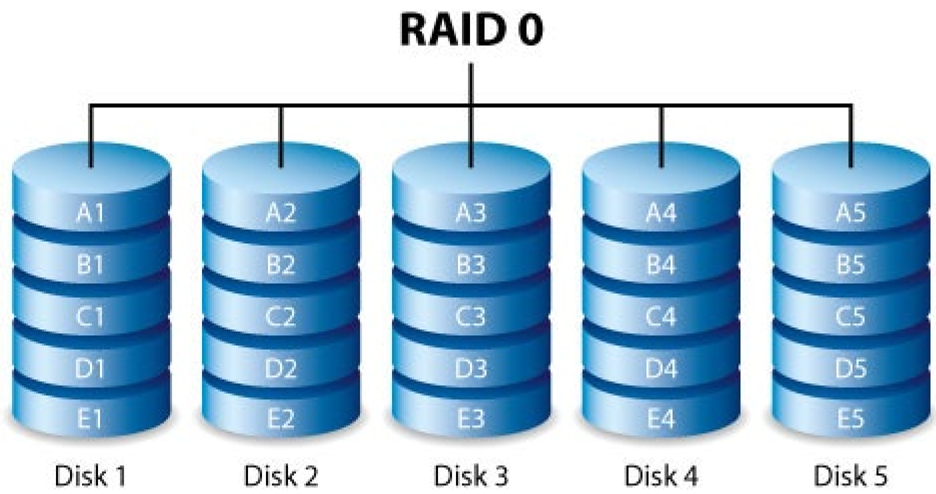
RAID systems

- **RAID**
  - "Redundant Array of Inexpensive Disks“ or "Redundant Array of Independent Disks”
  - is a data storage virtualization technology that combines multiple physical disk drive components into one or more logical units
- **Purposes**:
  - Data **redundancy**
  - Data access **performance** improvement
  - **Both**

You can see the RAID level of another type of QoS!!

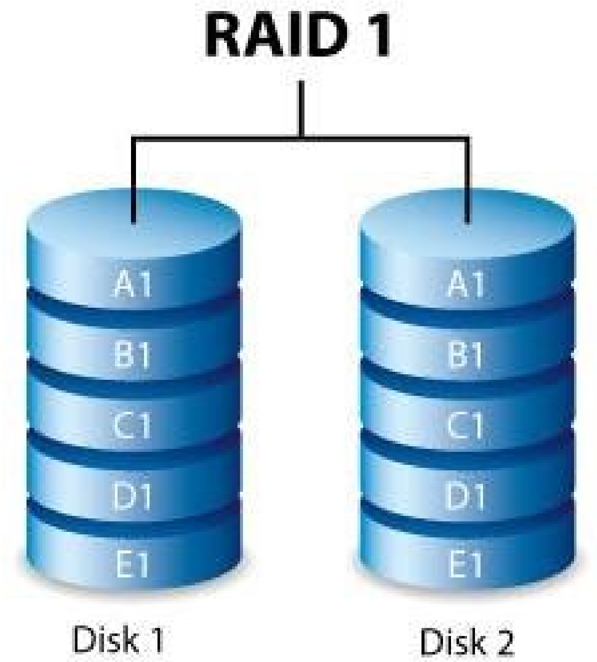
© https://www.lacie.com/it/it/manuals/lrm/raid/

RAID0: File striping



- Technique of segmenting logically sequential data, such as a file, so that consecutive segments are stored on different physical storage devices.
- Transfer rates up to *n* times higher than the individual drive rates
- Appears as a single disk with size equal n time the size of the smallest disk
- No redundancy

RAID1: File mirroring

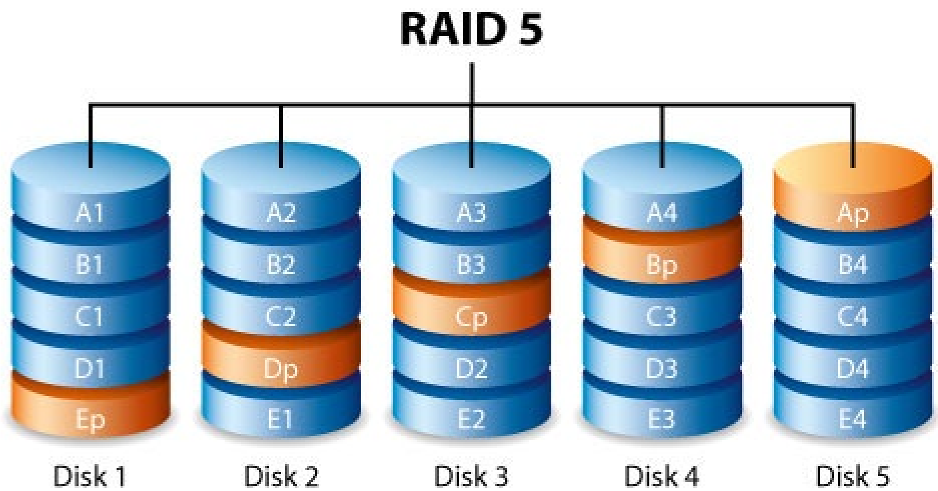


© https://www.lacie.com/it/it/manuals/lrm/raid/

- Exact copy
- Write performance remains at the level of a single disk
- Read performance can be increased to the one of the single disk depending on the type of I/O load
- Disk redundancy equal to the number of disks used **-1**

+

RAID5: Distributed parity



© https://www.lacie.com/it/it/manuals/lrm/raid/

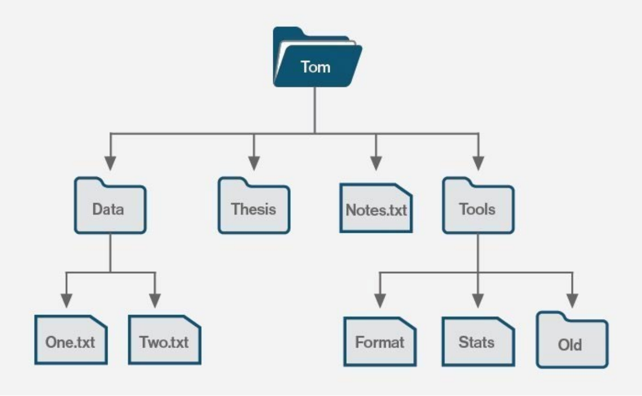
- Block-level striping with distributed parity
- Write performance is increased since all RAID members participate in the serving of write requests
- RAID 5 requires at least three disks
- In case of a failure of one disks it can be reconstructed

RAID6 has two parity disks
RAID5 with spare disks is also possible

https://en.wikipedia.org/wiki/Standard_RAID_levels

File systems

- A file system or filesystem (often abbreviated to fs) controls how data is stored and retrieved
  - **Space management**
    - allocate space in a granular manner, usually multiple physical units on the device. The file system is responsible for organizing files and directories, and keeping track of which areas of the media belong to which file and which are not being used
  - Filenames
    - A **filename** (or **file name**) is used to identify a storage location in the file system
  - Directories
    - File systems typically have **directories** (also called **folders**) which allow the user to group files into separate collections
    - This may be implemented by associating the file name with an index in a table of contents or an inode in a Unix-like file system



https://en.wikipedia.org/wiki/File_system

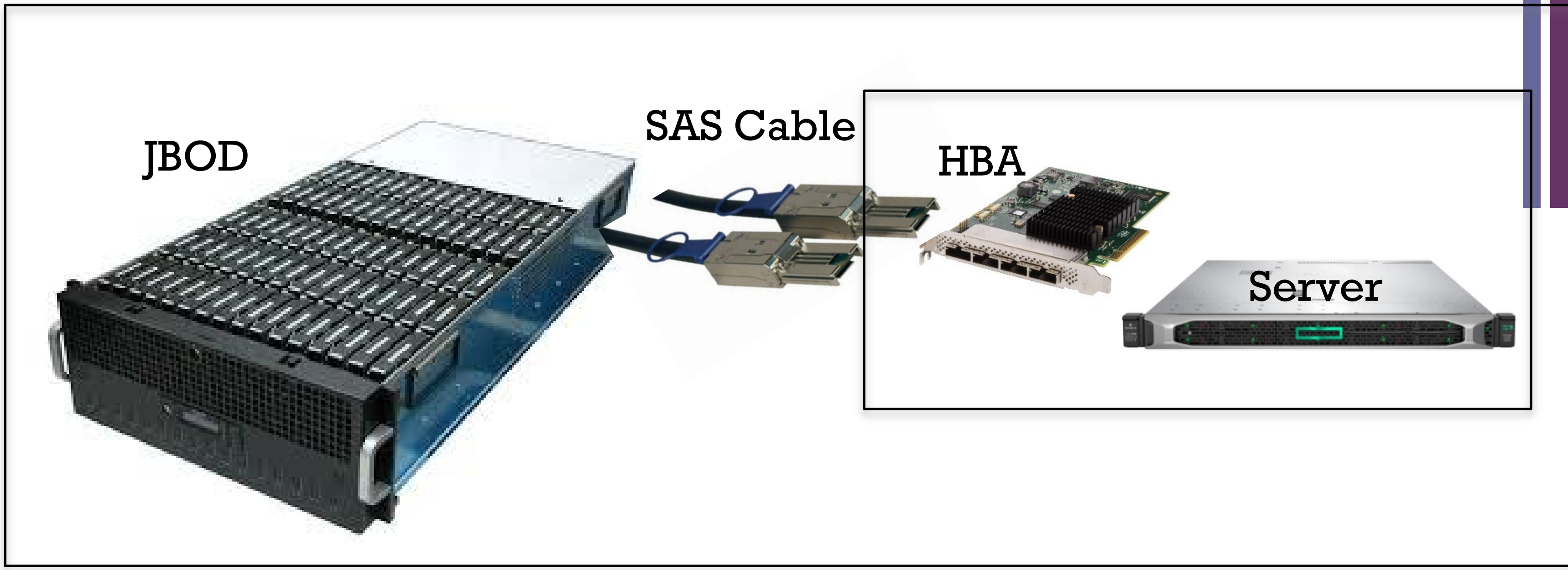
File systems /2

- **Fs Metadata**
  - **bookkeeping information** typically associated with each file within a file system.
    - The **length of the data** contained in a file may be stored as the number of blocks allocated for the file or as a byte count.
    - The **time that the file was last modified** may be stored as the file's **timestamp**.
    - **Creation time**
    - the time it was **last accessed**
    - the time the file's metadata was **changed**
    - the time the file was **last backed up**.
    - Other information can include:
      - the file's **device type** (e.g. block, character, socket, subdirectory, etc.)
      - its **owner** user ID and group ID
      - its **access permissions**
      - other file attributes (e.g. whether the file is read-only, executable, etc.).

POSIX Filesystems

- The **Portable Operating System Interface** (POSIX) is a family of standards specified by the IEEE Computer Society **for maintaining compatibility between operating systems**
- POSIX defines the application programming interface (API), along with command line shells and utility interfaces
- The family of POSIX standards is formally designated as IEEE 1003 and the international standard name is ISO/IEC 9945.
  - **Define POSIX filesystem characteristics and functions**
    - ie. the usual 'fopen'
- Most typical Linux filesystems are POSIX capable, ie.:
  - ext3, ext4, xfs, zfs

Direct Attached Storage systems



Direct Attached Storage (DAS)

- Direct-attached storage (DAS) is **digital storage directly attached to the computer** accessing it
  - hard drives
  - solid-state drives
  - CD-DVD readers
  - USB external drives
  - SAS JBOD

- A DAS does not incorporate any network hardware and related operating environment to provide a facility to share storage resources independently. The storage presented by a DAS to a connected host can of course be shared by that host

- A typical DAS system is made of a data storage device (for example enclosures holding a number of hard disk drives) connected directly to a computer through a host bus adapter (HBA). Between those two points there is no network device (like hub, switch, or router), and this is the main characteristic of DAS

- The main protocols used for DAS connections are ATA, SATA, NVMe, SCSI, SAS, USB, Fibre Channel

Storage and networking

- Spinning disks
- Solid State Disks
- NVMe
- Tapes

- Ethernet
- Infininband/omnipath
- SAS
- Fiber

Networked Storage Systems

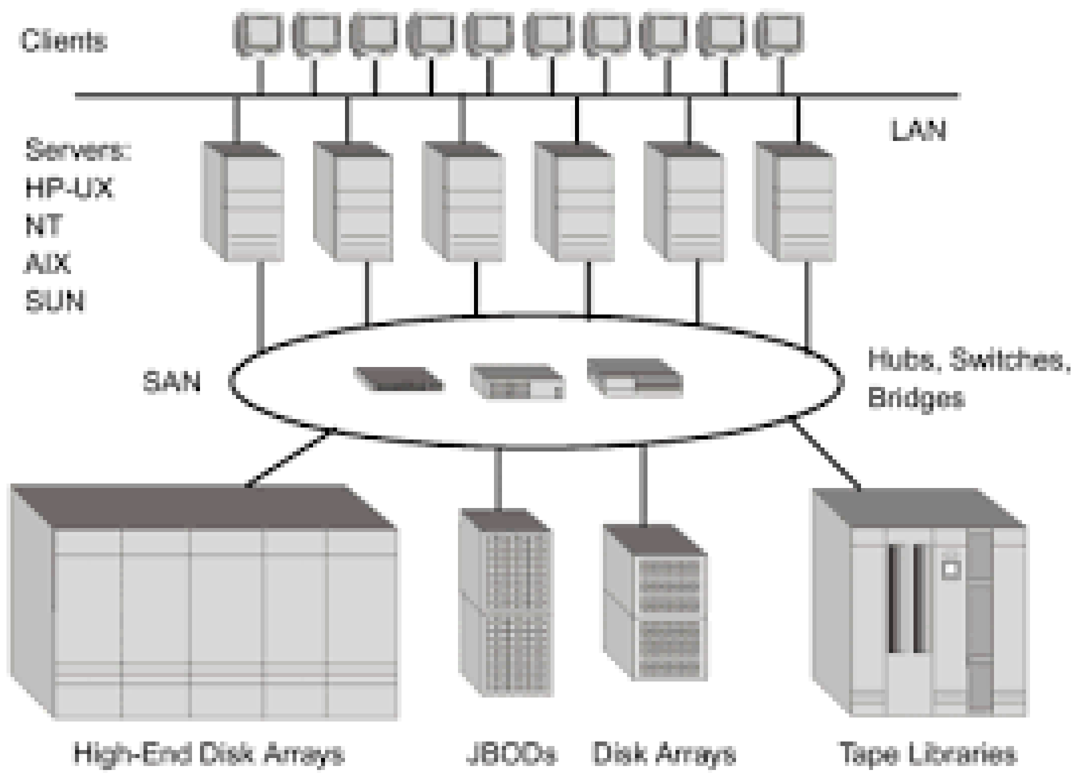
- **Network-attached storage** (NAS) is a **file-level** computer data storage server connected to a computer network providing data access to a heterogeneous group of clients
- NAS is often manufactured as a computer appliance – a purpose-built specialized computer.
- NAS systems are networked appliances which contain one or more storage drives, often arranged into logical, redundant storage containers or RAID
- Network-attached storage removes the responsibility of file serving from other servers on the network. They typically provide access to files using network file sharing protocols such as NFS, SMB/CIFS, or AFP.

NAS Appliances

Storage Area Network (SAN)

- **A dedicated network to store and access data**
- Storage devices (disk arrays or tape libraries) are accessible to servers as **block level** data storage
- The interconnection is made by distinct protocols, such as Fibre Channel, iSCSI or Infiniband
- Storage and network devices are dedicated to the SAN and can be heterogeneous
- On top of the SAN can be created a file system to access data at file level

Storage Area Network



SAN Components

- **HOST Layer**
  - Servers that allow access to the SAN and its storage devices
  - Through the host bus adapters (HBAs) can communicate with the storage devices in the SAN.
  - Often optical cables are used to connect the hosts and the storage system.
    - In this case a gigabit interface converter (GBIC) is used to convert light into digital signals (and viceversa)

- **Fabric Layer**
  - The SAN networking devices are called *fabric layers*
    - SAN switches
    - Routers
    - gateway devices
    - cables.
  - SAN network devices move data within the SAN, or between an *initiator*, such as an HBA port of a server, and a *target*, such as the port of a storage device.
  - SAN networks often have redundancy, so SAN switches are connected with redundant links

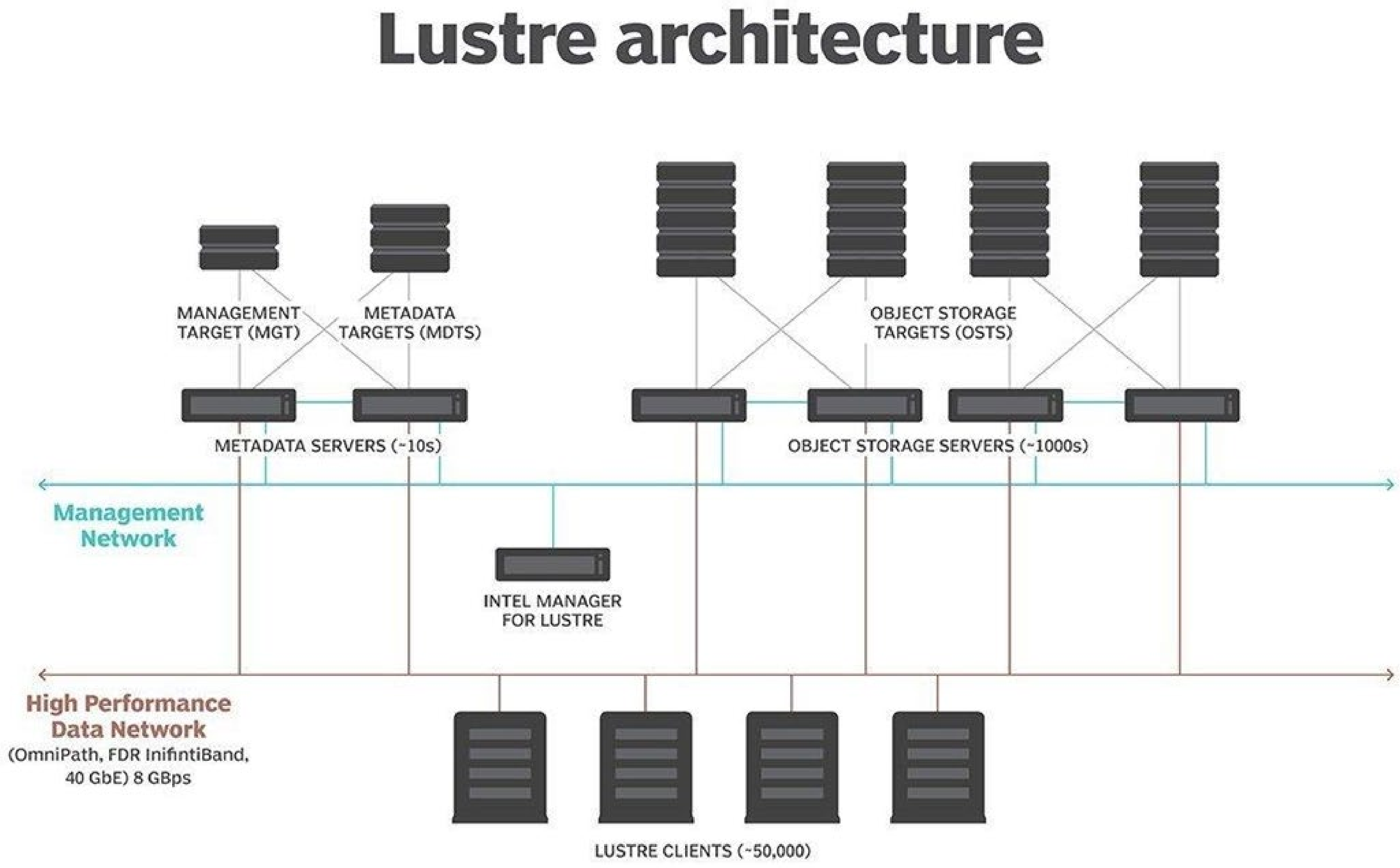
SAN Components

- **Storage Layer**
  - The various storage devices in a SAN.
    - hard disk – often organized in JBODs
      - Protected by RAID systems
    - magnetic tape devices - often organized in libraries
    - Every storage device, or even partition on that storage device, has a logical unit number (LUN) assigned to it.
      - A unique number within the SAN and every node in the SAN
        - The LUNs allow for the storage capacity of a SAN to be segmented and for the implementation of access controls
- **SAN Protocols**
  - the serialized Small Computer Systems Interface (SCSI) protocol, allows software applications to communicate, or encode data, for storage devices.
    - Built on top of the Fibre Channel-Switched Protocol
    - The internet Small Computer Systems Interface (iSCSI) over Ethernet and the Infiniband protocols may also be found implemented in SANs, but are often bridged into the fibre channel SAN

Parallel File Systems

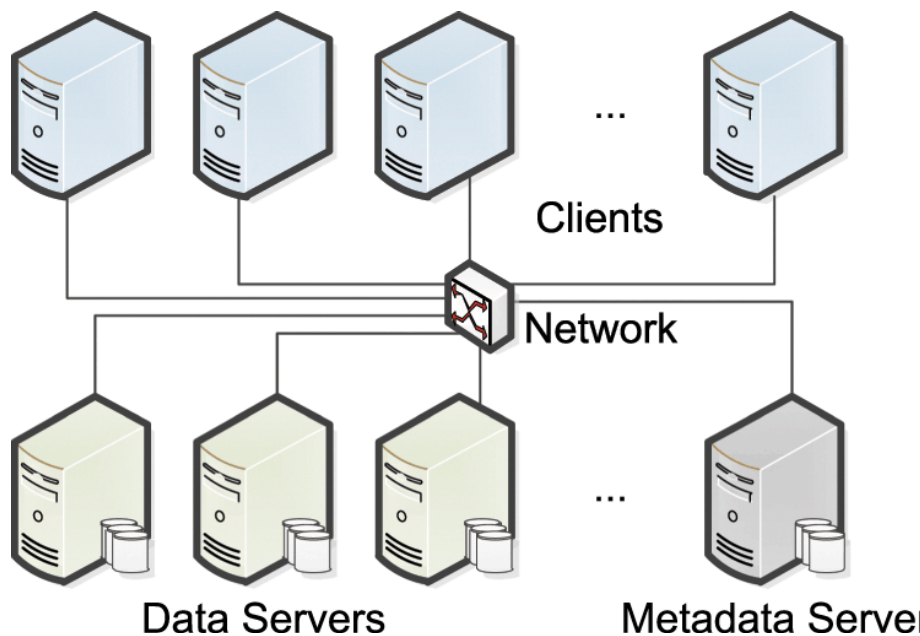
- A parallel file system **stores data across multiple networked servers**
- Facilitates high-performance access through simultaneous, coordinated input/output operations between clients and storage nodes
- Designed to support a **high multiplicity of clients**
- Breaks up a data set and distributes, or stripes, the blocks to multiple storage drives, which can be located in local and/or remote servers.
- Designed for performance and **highly concurrent access**
  - access to large files
  - massive quantities of data
  - simultaneous access from multiple compute servers
- Examples
  - IBM Spectrum Scale (GPFS)
  - LUSTRE

Parallel File System



Distributed File system

- *Distributed file systems* in general **do not share block level access** to the same storage but **use a network protocol to access files on multiple servers**
  - **Create a unique global namespace of distributed files**
- Distributed file systems allow files to be accessed using the same interfaces and semantics as local files
- Programs running on one computer use local interfaces and semantics to create, manage and access files located on other networked computers
- Examples
  - BeeGFS
  - GlusterFS
  - Ceph
  - GFS
  - HDFS
  - OrangeFS
  - IPFS



Distributed File system

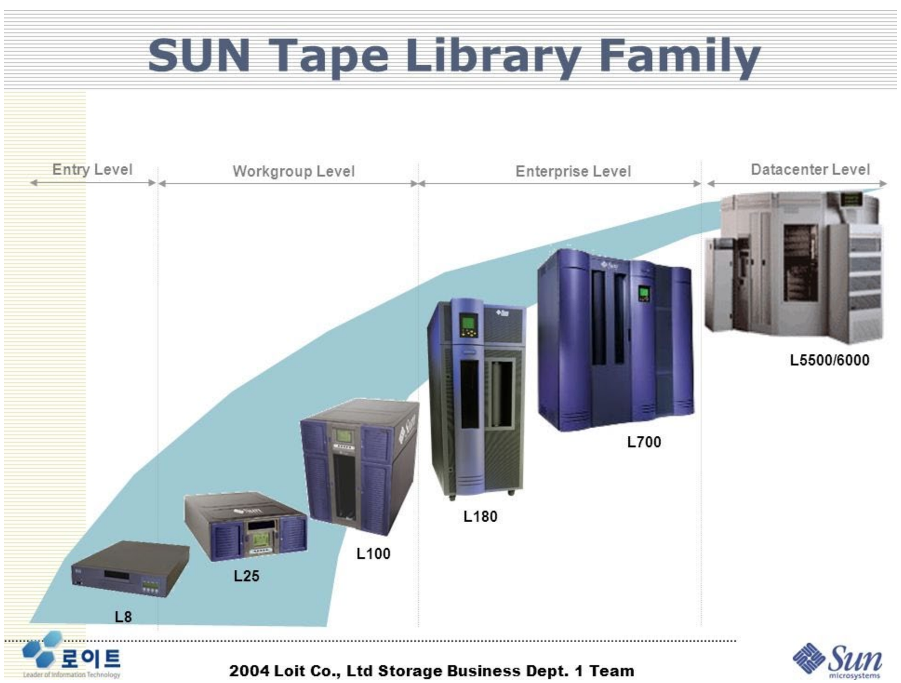
- *Distributed file systems* in general **do not share block level access** to the same storage but **use a network protocol to access files on multiple servers**
  - **Create a unique global namespace of distributed files**
- Distributed file systems allow files to be accessed using the same interfaces and semantics as local files
- Programs running on one computer use local interfaces and semantics to create, manage and access files located on other networked computers
- Examples
  - BeeGFS
  - GlusterFS
  - Ceph
  - GFS
  - HDFS
  - OrangeFS
  - IPFS

NB – there are various definition and various interpretation of the difference between parallel and distributed file system

Some of the quoted systems in the examples can be both!

Tape Area Network (TAN)

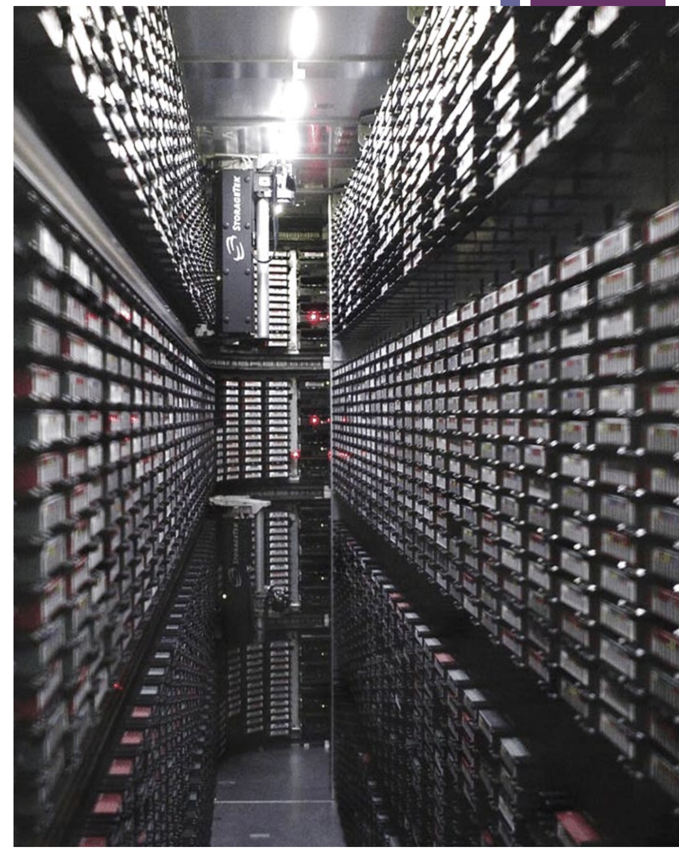
- Is the part of the SAN dedicated to the interconnection among servers, libraries and tape drives
- Tape drives can be installed in a central array and attached to the SAN, making them accessible to every server on the network



+

CNAF mass storage system

- 1 Oracle StorageTek SL8500 tape library
  - 10000 slots, 85PB capacity with present technology
- 17 T10000D tape drives
  - 250 MB/s throughput
- Disk buffer to perform writing and reading operations with tapes
- 80 PB of data
  - Especially scientific RAW data
  - Backup of CNAF service configurations, logs, repositories, etc.



QoS, Migration and Recall

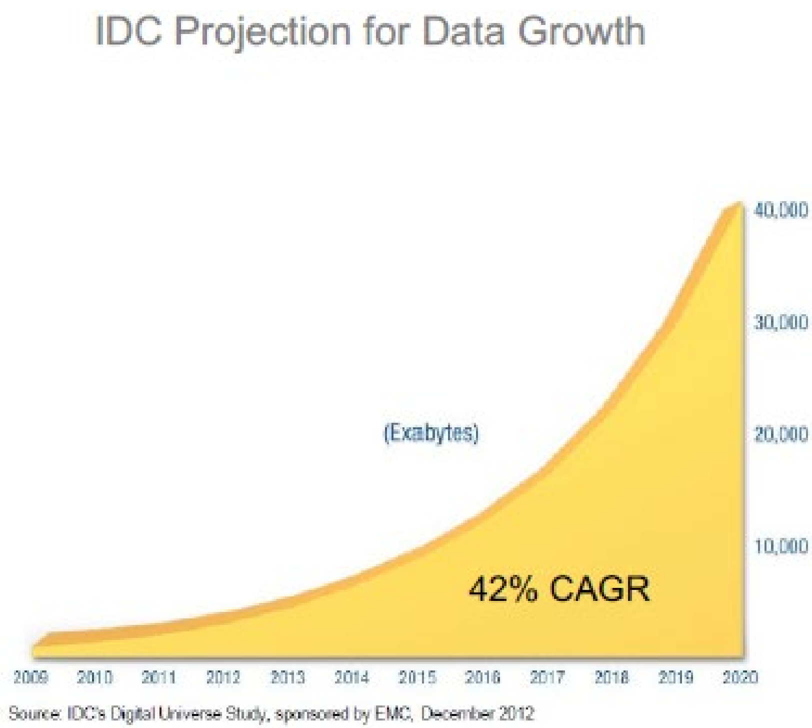
- **Migration** moving a file from disk to tape
- **Recall** moving a file from tape to disk
- **QoS** Ensures that a particular application or workload always gets a certain performance level
  - Typically expressed as IOPS
  - An increasing number of storage systems now claim to offer some form of QoS

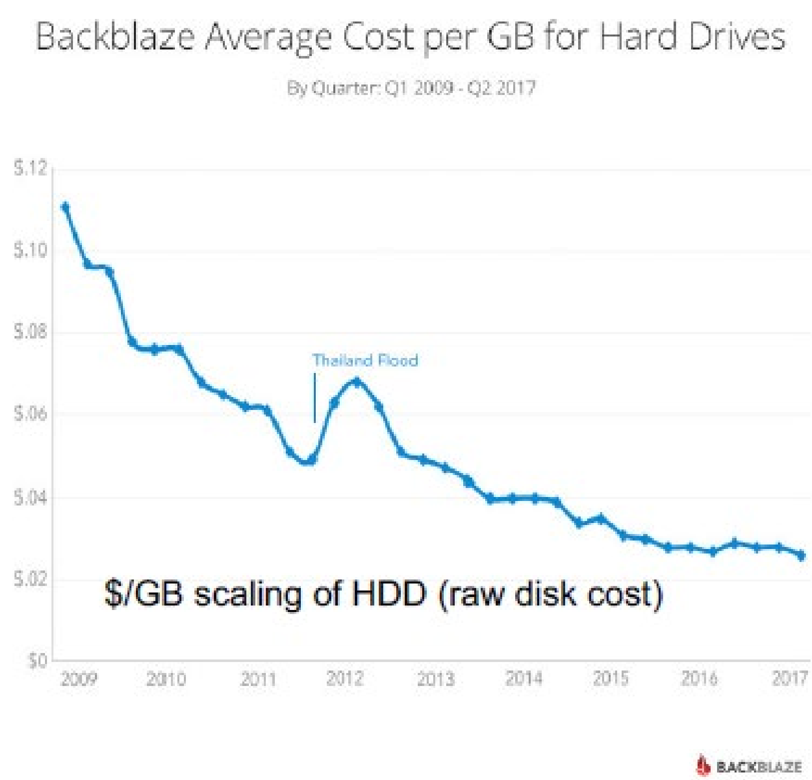
IBM Spectrum Protect

- Formerly TSM (Tivoli Storage Manager)
- A proprietary software designed by IBM, one of the leaders in data protection solutions
- Especially used for backup-archive purposes
- Offers a Hierarchical storage management (HSM) extension to manage migrations from disk to tape and recalls from tape to disk of data hosted on Spectrum Scale file systems.

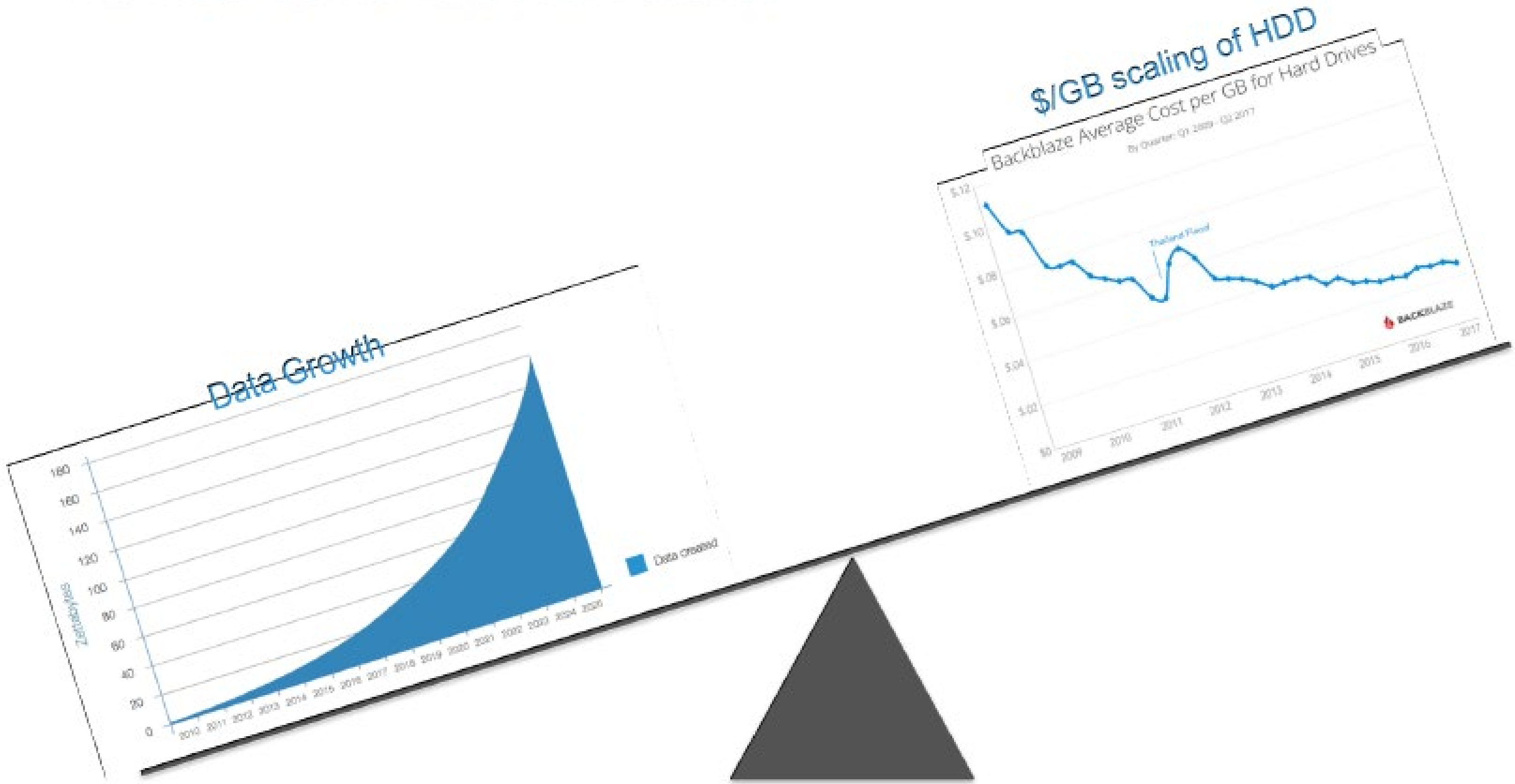
+ Tape's reinassance

- Data continues to grow exponentially while HDD scaling has stagnated
  - Driving demand for cost effective storage





The data center is out of balance



- But 80% Worldwide of files created are inactive
    - No access in at least 3 months

Tape advantage for long term storage

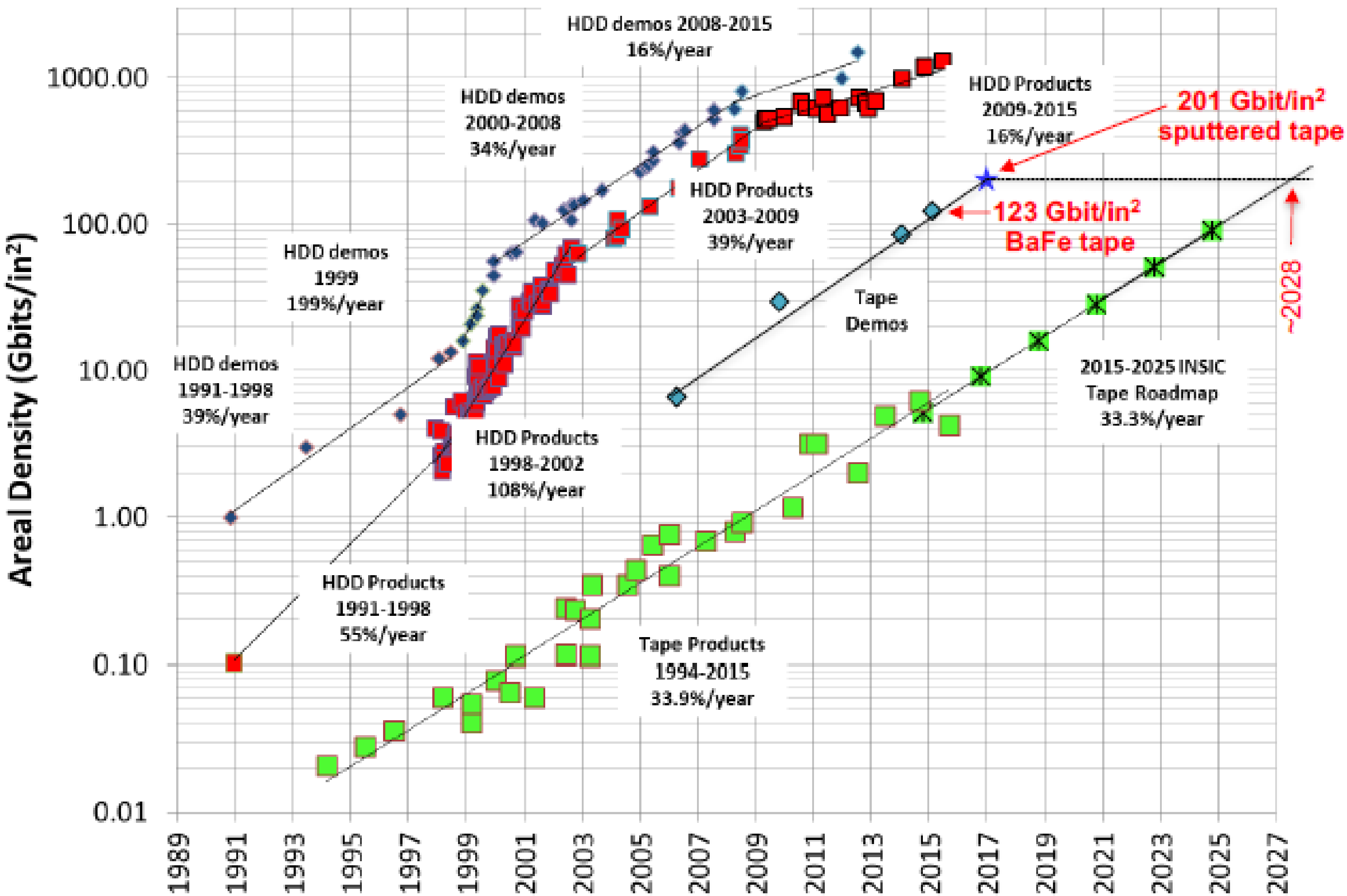
- Very **energy efficiency**
  - No power needed once data is recorded
- **Very secure**
  - Data is inaccessible while cartridge is not mounted
  - Portable
- **Very long expected data lifetime** (30+ years)
- **~~Very~~ reliable**
  - Read while write verification
  - Typically no data loss in case of drive failure
- **Main advantage is cost**
  - Recent study by Clipper Group on 9 years TCO of 1 PB that grows to 52 PB (52% CAGR): **6.7x TCO advantage of LTO Tape over Disk**

+
Magnetic tape evolution

Product Year	IBM 726 1952	LTO7 2015	TS1155 2017	Demo 2015 BaFe Tape	Demo 2017 Sputtered Tape
Capacity	2.3 MB	6 TB	15 TB	220 TB	330 TBytes
Areal Density	1400 bit/in^2	4.3 Gbit/in^2	9.6 Gbit/in^2	123 Gbit/in^2	201 Gbit/in^2
Linear Density	100 bit/in	485 kbit/in	510 kbit/in	680 kbit/in	818 kbit/in
Track Density	14 tracks/in	8.86 ktracks/in	18.9 ktracks/in	181 ktracks/in	246 ktracks/in

Areal Density >6.8M X

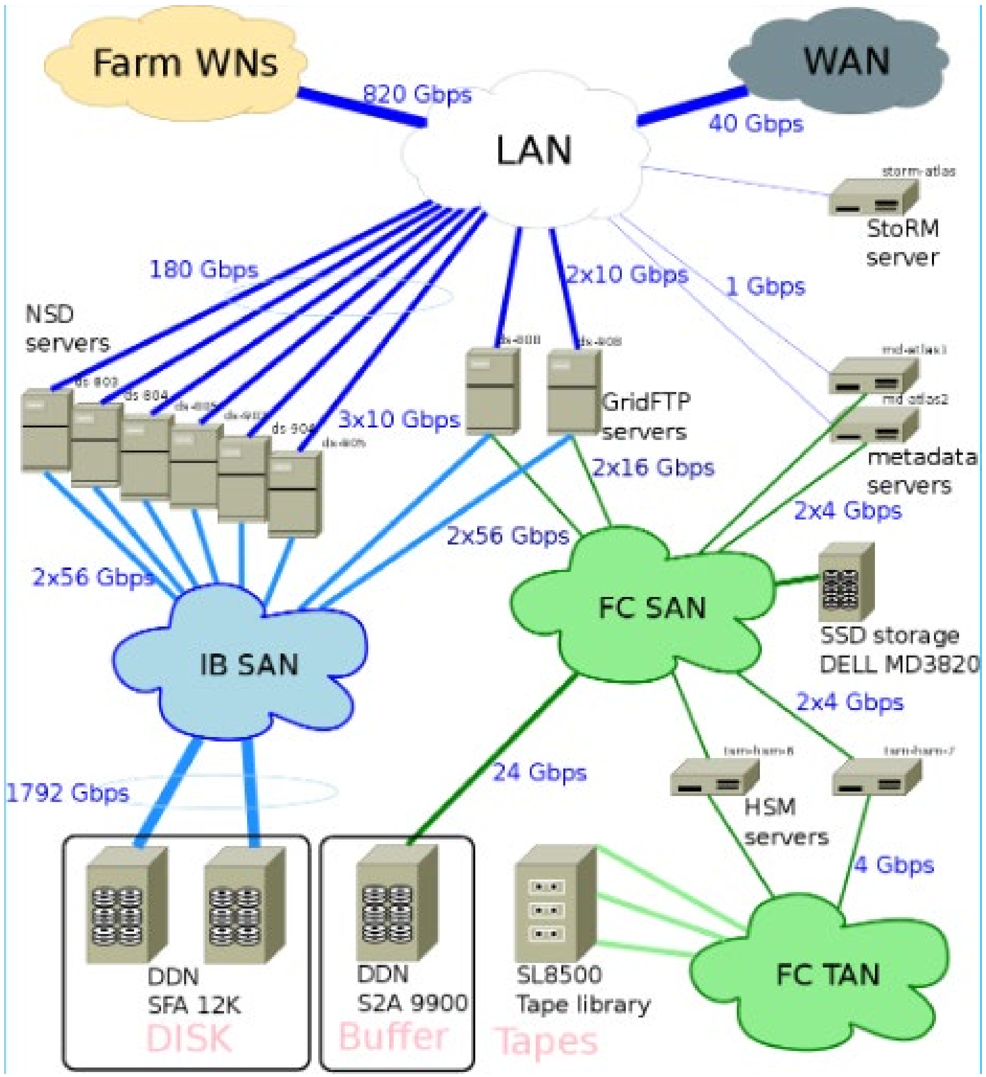
Areal density scaling



- 2015: IBM-FujiFilm demonstration of 123 Gb/in2 on BaFe tape
- 2017: IBM-Sony demonstration of 201 Gb/in2 on Sputtered Tape

+

SAN Example at CNAF



- 7 PB disk space
- 6 NSD servers (3x10 Gbps)
- 2 metadata servers (1Gbps)
- 2 GridFTP (XrootD) (2x10 Gbps)
- 2 HSM servers
- Metadata on SSD (mirrored)
- VM as Storm server
- Throughput required (5 MB/s/TB) = 21.5 GB/s
- Throughput available (6 NSD x 30 Gbps) = 22.5 GB/s

Storage Remote Access Services

- Service to access remotely the Storage system
- Implement operations like
  - List
  - Copy /transfer
  - Delete
  - Recall/migrate
  - Etc..
- Possibly based on standard or de-facto standard
- Implement various types of auth/authZ mechanisms
  - Username/password
  - Tokens
  - Digital certificates

Data transfer tools

- Asynchronous Data movement
  - Scp
  - Rsync
  - ftp
- Streaming Data
  - Flume
  - Kafka

Take away messages (Storage)

- Direct Attached Storage (DAS): devices connected to a computer/server
- Network Attached Storage (NAS): provides data access at file level to clients through the network
- Storage Area Network (SAN): a dedicated network infrastructure that provides access to storage devices at block level
- Parallel and distributed file systems stores data across multiple networked servers/devices providing fast access to multiple clients (possibly many thousands of clients)
  - Guarantee performance, high availability and redundancy of data
- A data center can provide storage resources with different Quality-of-Service
  - It is important to chose the right one for the user applications in order to maximize the performance and minimize the costs

References

- https://en.wikipedia.org/wiki/IOPS
- https://en.wikipedia.org/wiki/Template_talk:Bit_and_byte_prefixes
- https://en.wikipedia.org/wiki/Standard_RAID_levels
- https://www.cavium.com/Documents/TechnologyBriefs/Adapters/Tech_Brief_Introduction_to_Ethernet_Latency.pdf
- https://en.wikipedia.org/wiki/Storage_area_network
- https://www.ibm.com/support/knowledgecenter/en/SSETD4_9.1.3/lsf_admin/fairshare_about_lsf.html
- https://www.ibm.com/support/knowledgecenter/en/SSETD4_9.1.3/lsf_admin/backfill.html
- https://community.fs.com/blog/do-you-know-the-differences-between-hubs-switches-and-routers.html https://en.wikipedia.org/wiki/Computer_cooling
- https://en.wikipedia.org/wiki/Power_usage_effectivenes
- https://puppet.com/
- https://www.theforeman.org/
- https://grafana.com/

Block Vs Object storage

Summary

- **Introduction**
  - Scientific data needs and new challenges
- POSIX Standard
- Object Storage
- CEPH
- MinIO

I/O data

- I/O is commonly used by different applications to achieve goals like:
  - Storing output from simulations for later analysis
  - loading initial conditions or datasets for processing
  - checkpointing to files that save the state of an application in case of system failure
  - Implement 'out-of-core' techniques for algorithms that process more data than can fit in system memory

The I/O Challenge

- Data access is a huge challenge
- Data stored, moved and analyzed (also in real time)
- Findable Accesible Interoperable Reusable (FAIR principles)

- Using parallelism to obtain performance
  - O(n) n=TB
  - Critical to handle parallelism in all phases
  - Adding more compute nodes increases aggregate memory bandwidth and flops/s, but not I/O

Factors which affect I/O

- I/O is interaction with data Memory <> Disk
  - simply data migration
  - very expensive operation

- How is I/O performed
  - I/O Pattern
    - Number of processes and files
    - Characteristics of file access

- Where is I/O performed
  - Characteristics of the computational system
  - Characteristics of the Operating System and the File System

Object Storage vs. Block Storage

Point of Comparison	Object storage	Block storage
Data storage	Unique, identifiable, and distinct units called objects store data in a flat-file system.	Fixed-sized blocks store portions of the data in a hierarchical system and reassemble when needed.
Metadata	Unlimited, customizable contextual information.	Limited, basic information.
Cost	More cost-effective.	More expensive.
Scalability	Unlimited scalability.	Limited scalability.
Performance	Suitable for high volumes of unstructured data. Performs best with large files.	Best for transactional data and database storage. Performs best with files small in size.
Location	A centralized or geographically dispersed system that stores data on-premise, private, hybrid, or public cloud.	A centralized system that stores data on-premise or in private cloud. Latency may become an issue if the application and the storage are geographically far apart.

Summary #2

- Introduction
- **POSIX Standard**
  - Features
  - Architectures
  - Limits
- Object Storage
- CEPH
- MinIO

POSIX Standard

- POSIX (Portable Operating System Interface for Unix)
  - set of standards that define the Application Programming Interface (API) as well as some shell and utility interfaces
  - developed primarily for *nix operating systems
  - actually any operating system can utilize the standards
- The standards emerged from a project that began in 1985
- The family of POSIX standards is formally designated as IEEE 1003 and the international standard name is ISO/IEC 9945

POSIX IO Standard

- POSIX IO (not an official name) is the portion of the standard that defines the I/O interface for POSIX compliant applications.
- Functions
  - read(), write(), open(), close(), lseek(), fwrite(), fread(), …

- open
- read
- write
- close
- lseek
- llseek
- _llseek
- lseek64
- stat
- fstat
- stat64
- chmod
- fchmod
- access
- rename
- mkdir
- getdents
- fcntl
- unlink
- fseek
- rewind
- ftell
- fgetpos
- fsetpos
- fclose
- fsync
- creat
- readdir
- opendir
- fopendir
- rewinddir
- scandir
- seekdir
- telldir
- flock
- lockf
- lseekm
- lstat
- fstatat
- fopen
- fdopen
- freopen
- remove
- chown
- fchown
- fchmodat
- fchownat
- faccessat
- utime
- futimes
- lutimes
- futimesat
- link
- linkat
- unlinkat
- symlink
- symlinkat
- rmdir
- mkdirat
- getxattr
- lgetxattr
- fgetxattr
- xetxattr
- lsetxattr
- fsetxattr
- listxattr
- llistxattr
- flistxattr
- removexattr

POSIX-oriented operating systems

- Depending upon the degree of compliance with the standards, operating systems can be classified as
  - **POSIX-certified**
    - operating systems have been certified to conform to one or more of the various POSIX standards. This means that they passed the automated conformance tests
    - AIX, Solaris, macOS (since 10.5 Leopard)
  - **POSIX-compliant**
    - while not officially certified as POSIX compatible, comply in large part
    - Android (Available through Android NDK), *BSD, OpenSolaris
  - **POSIX for Microsoft Windows**
    - Cygwin, Microsoft POSIX subsystem Windows Subsystem for Linux
  - **POSIX for OS/2**
    - POSIX compliant environments for OS/2
  - **Compliant via compatibility feature**
    - not officially certified as POSIX compatible, but they conform in large part to the standards by implementing POSIX support via some compatibility feature
    - SymbianOS, Windows NT Kernel

POSIX IO: Metadata

- POSIX I/O prescribes a specific set of metadata that all files must possess (POSIX I/O calls)
  - **user** and **group** which owns the file
  - the **permission** that user and group has to read and modify the file
  - **attributes** such as the time the file was created and last modified
- Calls manipulate the metadata that POSIX dictates all files must possess
  - such as chmod() and stat() or other shell commands that provide command-line interfaces for these calls

POSIX IO: Metadata

- POSIX style of metadata certainly works but…
- Prescriptive
  - all files must possess metadata such as the user and group which owns the file, the permission that user and group has to read and modify the file, and attributes such as the time the file was created and last modified
    - Eg. "ls –l" in a directory with a 1M of file
- Inflexible
  - the ownership and access permissions for files are often identical within directories containing scientific data (for example, file-per-process checkpoints), but POSIX file systems must track each of these files independently.
- Not descriptive enough for many data sets
  - often resulting in README files effectively storing the richer metadata that POSIX does not provide.

POSIX IO: stateful

- POSIX I/O is stateful
  - rely on a state in time to perform an action: e.g to change the output given the determined inputs and state.
    - "**state**" is simply the condition or quality of an entity at an instant in time
  - A typical application might
    - open() a file
    - read() the data from it
    - seek() to a new position
    - write() some data
    - close() the file

POSIX IO: file descriptor

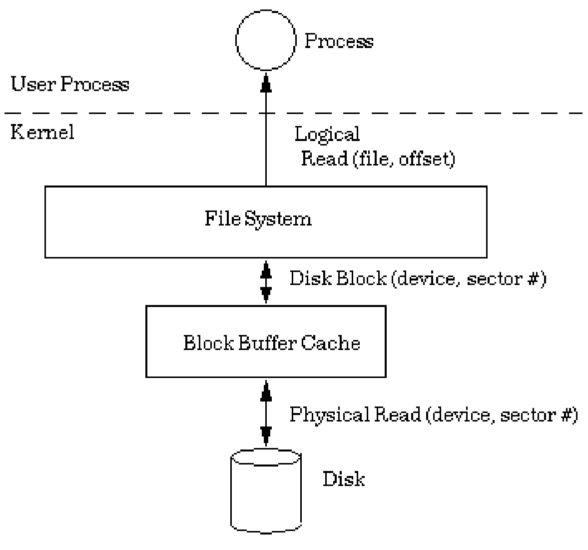
- POSIX I/O is stateful
  - reading and writing data is governed by some persistent state that is maintained by the operating system in the form of **file descriptors**.
  - File descriptors are central to this process
    - applications cannot read or write a file without first open()ing it to get a file descriptor
    - the position where the next read or write will place its data is generated by where the last read, write, or seek call ended

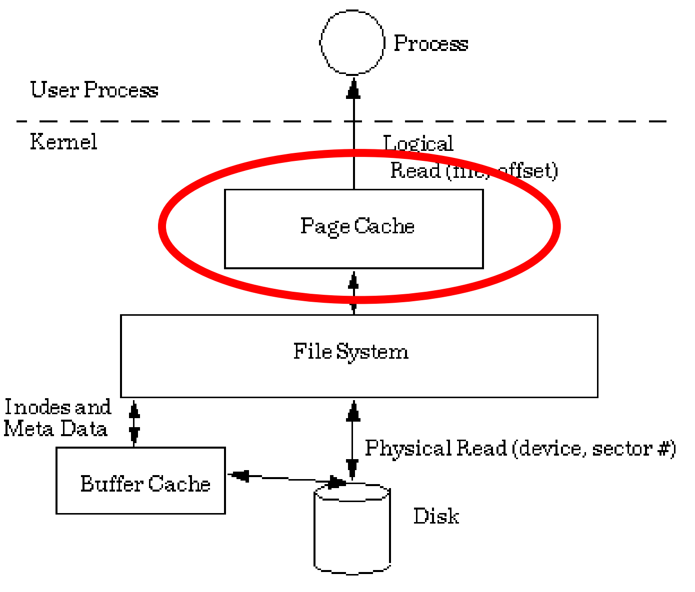
- **Writes must be strongly consistent:**
  - a write() is required to block application execution until the system can guarantee that any other read() call will see the data that was just written.

A "dirty" solution to stateful

- Instead of blocking the application until the data is physically stored on a slow (but nonvolatile) storage device, write()s are allowed to return control back to the application only after the data is written to a memory page.
- The operating system tracks "dirty pages" which contain data that hasn't been flushed to disk and writes those dirty pages from memory, back into their intended files asynchronously: **Page caching**.
- The POSIX consistency guarantee is still satisfied because the OS tracks cached pages and will serve read() calls directly out of memory if a page is already there.

A “dirty” solution to stateful





Images from http://sunsite.uakom.sk

Summary #3

- Introduction
  - Scientific data needs and I/ Challenges
- POSIX Standard
- **Object Storage**
  - Features
  - Architectures
  - Applications
- CEPH
- MinIO

What is object storage?

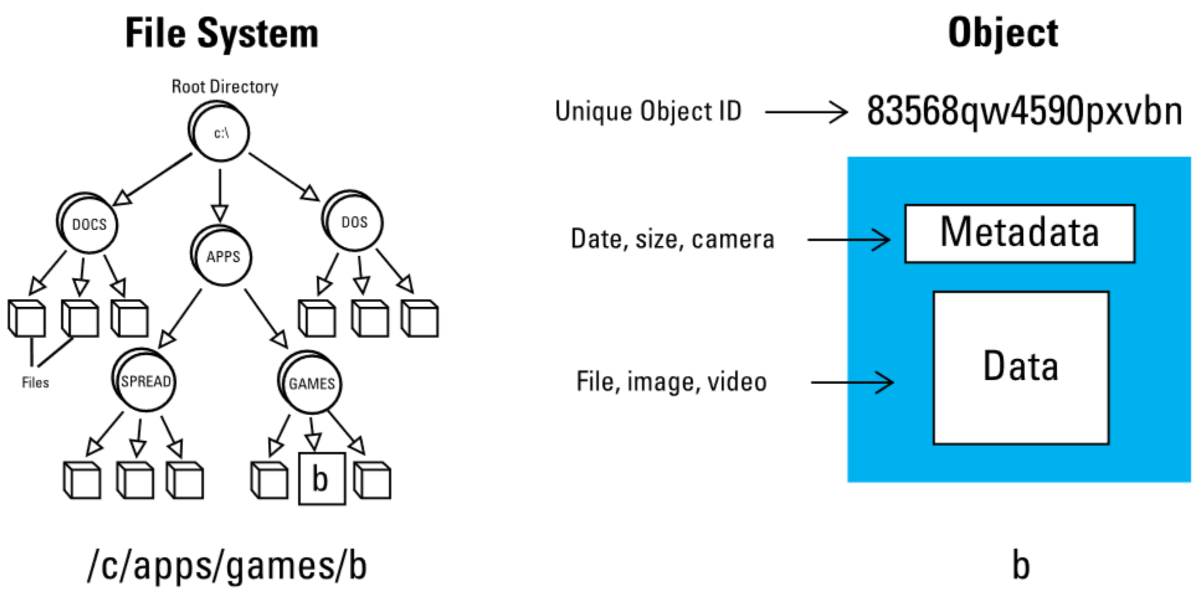
- **Object storage** (also known as object-based storage) is a computer data storage architecture that **manages data as objects**, as opposed to other storage architectures like file systems which manages data as a file hierarchy (Posix), and block storage (RBD) which manages data as blocks within sectors and tracks.

- Each object typically includes the data itself, a variable amount of metadata, and a globally unique identifier.

What is Object Storage used for?

- **Storage for Unstructured Data** such as music, media files, or text documents. Any type of data that doesn't have a distinct structure to it, has metadata (ex. a song's artist, album title, etc.), and likely won't be manipulated often is a great fit for Object Storage. Some popular products that use object storage in this category that you might recognize are Netflix and Spotify who use is to store their media files.
- **Backup and Recovery** of critical business applications and workloads. With the rise of digital products, mobile devices and the internet, consumers and enterprises expect applications to always be on and functioning. Due to the highly resilient nature and low cost of Object Storage, many businesses use it to backup their data and workloads to ensure business continuity and to prevent data loss in the event of a disaster.
- **Archived data for long term retention**. Sometimes customers specifically in the financial services industry and healthcare industry have requirements to keep data under retention or records for a certain time period (x number of years). Since this data will persist and not be manipulated frequently, object storage is a perfect cost-effective solution for this use case.
- **Cloud Native Applications** for a persistent data store. As businesses look to modernize their approach to application development in an effort to minimize the time to bring their solutions to market, they need a data store that will scale and not cause costs to sky rocket. Object Storage is a great solution for this as applications can connect directly to the object store and will allow for data to scale simply effectively as the business grows with its number of users and locations.
- **Data Lake for Analytics**. With the acceleration of the number of devices generating data (Smartphones, smart devices, IOT sensors etc.), there will be lots of data circulating around that can be processed for intelligent insights. Current storage solutions such as NAS and others are just not effective enough to support this vast growth of data being produced through these various sources. Object Storage can be a great solution for storing all types of data (structured, semi-structured, and unstructured data) that will give businesses a place to dump data before processing and analyzing in order to enable critical insights.

FS vs Object


File System
Root Directory
c:\
DOCS
APPS
DOS
Files
SPREAD
GAMES
b
/c/apps/games/b
Object
Unique Object ID
83568qw4590pxvbn
Date, size, camera
Metadata
File, image, video
Data
b

Object Metadata

- The flat organizational structure also enables object storage to provide a much richer metadata component for the object.
  - **Non-editable metadata**
    - Some metadata cannot be edited directly. This metadata is set at the time of object creation or rewrite
  - **Editable metadata**
    - **Fixed-key metadata**: Metadata whose keys are set, but for which you can specify a value.
    - **Custom metadata**: Metadata that you add by specifying both a key and a value associated with the key

Fixed-key metadata

- **Access control metadata**
  - Object Storage uses Identity and Access Management (IAM) and Access Control Lists (ACLs) to control access to objects.
- **Content Type**
  - Also known as MIME type, allows browsers to render the object properly
- **Content Disposition**
  - Content-Disposition allows to control presentation style of the content, for example determining whether an attachment should be automatically displayed or whether some form of action from the user should be required to open it

Fixed-key metadata

- **Content Encoding**
  - The Content-Encoding metadata can be used to indicate that an object is compressed
- **Content Language**
  - The Content-Language metadata indicates the lanuage(s) that the object is intended for
- **Cache Control**
  - Can control whether and for how long browser and Internet caches are allowed to cache your objects

Custom Metadata

- **Custom Metadata**
  - Custom metadata is metadata that can be added and removed.
  - Custom metadata are specified in the form of *key:value*
    - *Limited (10MB; 100 istances)*

Key Features of Object Storage

- One of the big draws of object oriented storage is its simplicity.

- There are only a few commands for object based file systems:
  - PUT (basically a "write" — PUT the object into the storage)
  - GET (basically a "read" — GET the object from the storage)
  - DELETE (delete the object)
  - HEAD (returns an object's metadata but not the data itself)

Write-once…

- Data is write-once
  - Editing an object means creating a completely new copy of it with the necessary changes
  - It is up to the user of the object store to keep track of which object IDs correspond to more meaningful information like a file name.
  - there is no need for a node to obtain a lock an object before reading its contents
  - There is no risk of another node writing to that object while its is being read
- The only reference to an object is its unique object ID
  - a simple hash of the object ID can be used to determine where an object will physically reside (which disk of which storage node)

…Read many

- Objects are comprised of
  - an object ID
  - Data
- Any metadata for an object (such as a logical file name, creation time, owner, access permissions) can be managed separately
- Objects' immutability restricts them to write-once, read-many workloads
  - Object storage cannot be used for scratch space or hot storage
  - applications are limited to data archival.

Applications

- Unique ID provides **greater scalability**, enabling an object storage system **to support faster access** to a much higher quantity of objects or files
- Object-storage systems allow retention of massive amounts of unstructured data.
- Object storage is used for purposes such as storing photos on Facebook, songs on Spotify, or files in online collaboration services, such as Dropbox.

Architecture and Structure

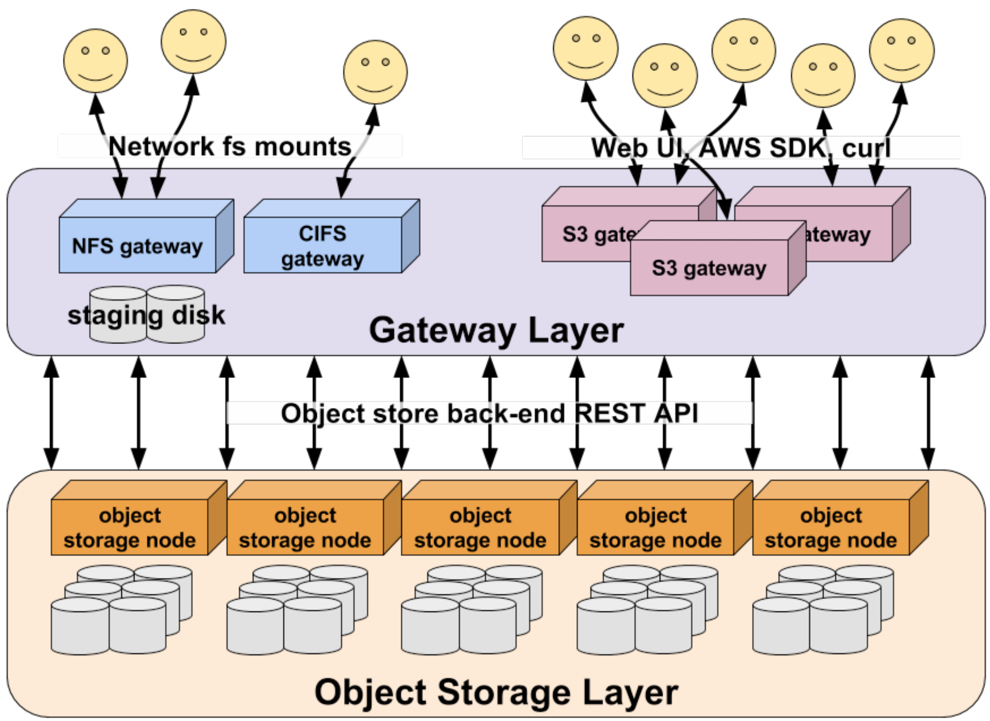
- Object storage systems are software defined scale-out architectures that can leverage commodity servers and storage.
  - IT planners can add additional storage nodes as their capacity demands grow.
  - Ideal for a use case where a lot of data needs to be stored for a long period.

Architecture and Structure

- Object storage systems are software defined scale-out architectures that can leverage commodity servers and storage.
  - IT planners can add additional storage nodes as their capacity demands grow.
  - Ideal for a use case where a lot of data needs to be stored for a long period.

IT DEPENDS!

Object Storage Cluster



Object-based file systems

- Some distributed file systems use an object-based architecture
  - metadata is stored in metadata servers
  - data is stored in object storage servers.

- Abstraction of the distinct servers to present a full file system to users and applications.
  - IBM Spectrum Scale (also known as GPFS),
  - Dell EMC Elastic Cloud Storage
  - Ceph
  - Lustre

Cloud storage

- The vast majority of cloud storage available in the market leverages an object-storage architecture. Some notable examples are
  - Amazon Web Services S3, which debuted in March 2006,
  - Google Coud Storage released on May 2010
  - Rackspace Files (whose code was donated in 2010 to Openstack project and released as OpenStack Swift)
  - MinIO Multi-Cloud Object Storage

Side-by-side comparison

	OBJECT STORAGE	BLOCK STORAGE
PERFORMANCE	Performs best for big content and high stream throughput	Strong performance with database and transactional data
GEOGRAPHY	Data can be stored across multiple regions	The greater the distance between storage and application, the higher the latency
SCALABILITY	Can scale infinitely to petabytes and beyond	Addressing requirements limit scalability
ANALYTICS	Customizable metadata allows data to be easily organized and retrieved	No metadata

Use case

Object storage

- Storage of **unstructured data** like music, image, and video files.
- Storage for backup files, database dumps, and log files.
- Large data sets. Whether you're storing pharmaceutical or financial data, or multimedia files such as photos and videos, storage can be used as your **big data** object store.
- Archive files in place of local tape drives.

Block storage

- **Ideal for databases**, since a DB requires consistent I/O performance and low-latency connectivity.
- Use block storage for **RAID Volumes**, where you combine multiple disks organized through stripping or mirroring.
- Any application which requires **service side processing**, like Java, PHP, and .Net will require block storage.
- Running **mission-critical** applications

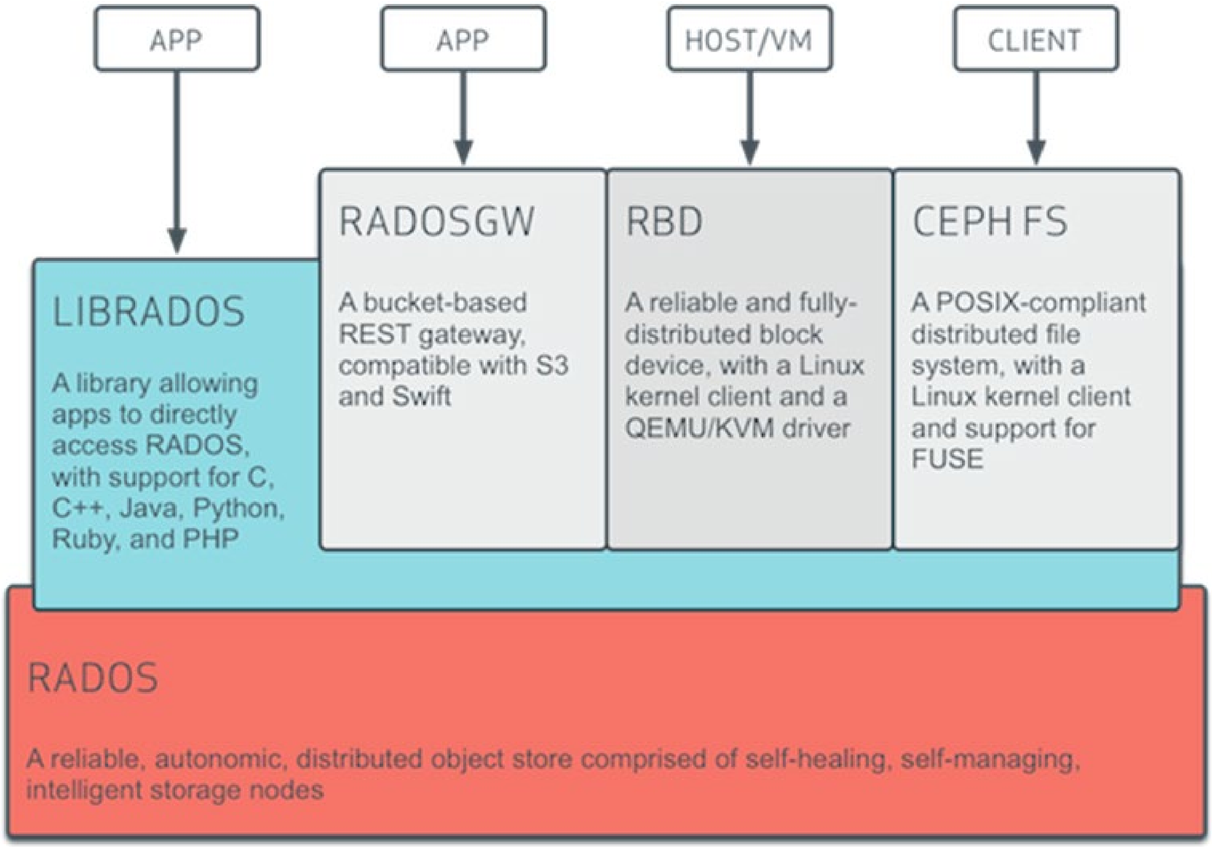
Summary #4

- Introduction
  - Scientific data needs and I/ Challenges
- POSIX Standard
- Object Storage
- **CEPH**
- MinIO

CEPH Features

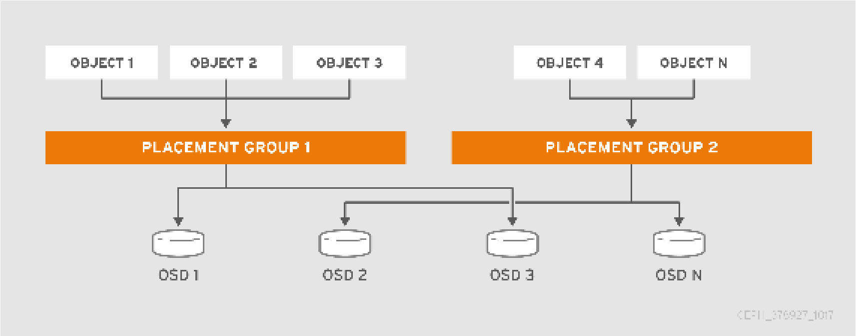
- In CEPH everything is an object
- No database for object position on the cluster
- There is a “rule” to place where store data on the cluster:
  - Each node of the cluster can calculate the object position

CEPH Architecture

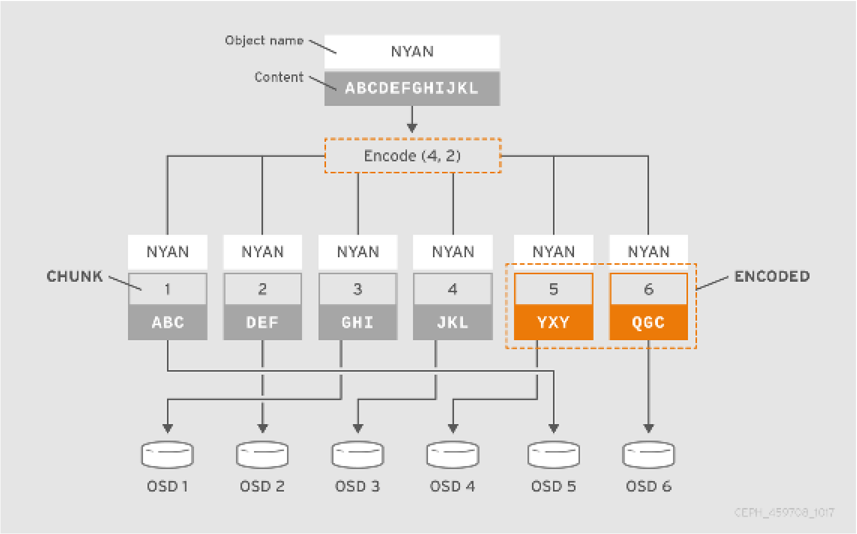


CEPH Replication strategy

Replicated



Erasure coding

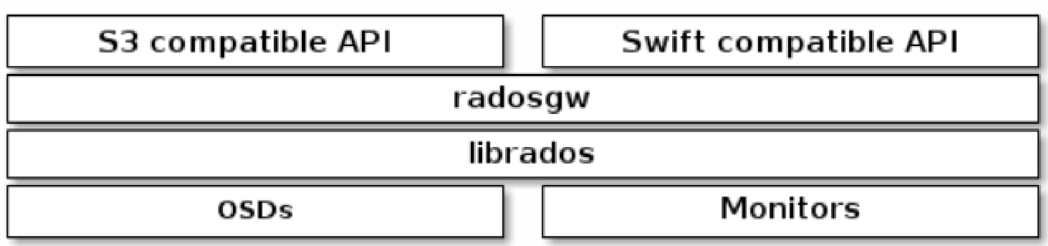


CEPH Object Gateway

- Ceph Object Gateway is an object storage interface built on top of librados to provide applications with a RESTful gateway to Ceph Storage Clusters. Ceph Object Storage supports two interfaces:

1. **S3-compatible:** Provides object storage functionality with an interface that is compatible with a large subset of the Amazon S3 RESTful API.

2. **Swift-compatible:** Provides object storage functionality with an interface that is compatible with a large subset of the OpenStack Swift API.



API support

Ceph Object Gateway S3 API

Ceph supports a RESTful API that is compatible
with the basic data access model of the Amazon S3 API.

Feature	Status	Remarks
List Buckets	Supported	
Delete Bucket	Supported	
Create Bucket	Supported	Different set of canned ACLs
Bucket Lifecycle	Supported	
Policy (Buckets, Objects)	Supported	ACLs & bucket policies are supported
Bucket Website	Supported	
Bucket ACLs (Get, Put)	Supported	Different set of canned ACLs
Bucket Location	Supported	
Bucket Notification	Supported	See S3 Notification Compatibility
Bucket Object Versions	Supported	
Get Bucket Info (HEAD)	Supported	
Bucket Request Payment	Supported	
Put Object	Supported	
Delete Object	Supported	
Get Object	Supported	
Object ACLs (Get, Put)	Supported	
Get Object Info (HEAD)	Supported	
POST Object	Supported	
Copy Object	Supported	
Multipart Uploads	Supported	
Object Tagging	Supported	See Object Related Operations for Policy verbs
Bucket Tagging	Supported	
Storage Class	Supported	See Storage Classes

Ceph Object Gateway Sfift API

Ceph supports a RESTful API that is compatible
with the basic data access model of the Swift API.

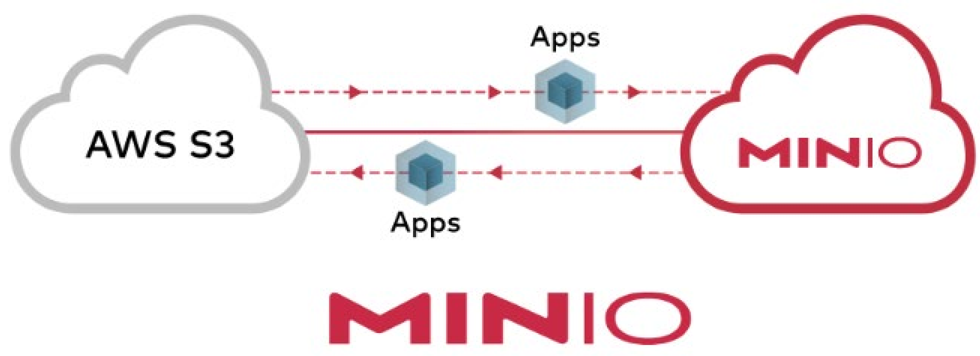
Feature	Status	Remarks
Authentication	Supported	
Get Account Metadata	Supported	
Swift ACLs	Supported	Supports a subset of Swift ACLs
List Containers	Supported	
Delete Container	Supported	
Create Container	Supported	
Get Container Metadata	Supported	
Update Container Metadata	Supported	
Delete Container Metadata	Supported	
List Objects	Supported	
Static Website	Supported	
Create Object	Supported	
Create Large Object	Supported	
Delete Object	Supported	
Get Object	Supported	
Copy Object	Supported	
Get Object Metadata	Supported	
Update Object Metadata	Supported	
Expiring Objects	Supported	
Temporary URLs	Partial Support	No support for container-level keys
Object Versioning	Partial Support	No support for `X-History-Location`
CORS	Not Supported	

Summary #5

- Introduction
  - Scientific data needs and I/ Challenges
- POSIX Standard
- Object Storage
- CEPH
- **MinIO**

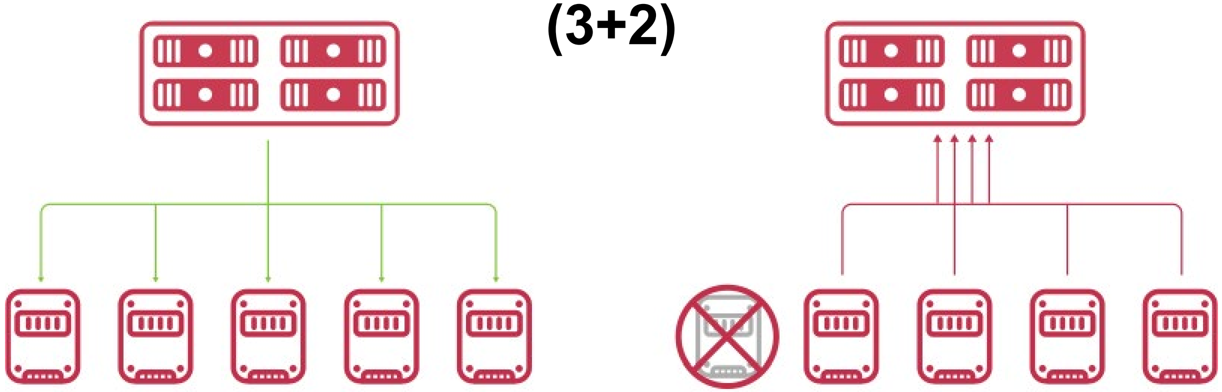
MinIO Features

- MinIO is adistributed object storage server written in Go, designed for Private Cloud infrastructure
- Providing S3 storage functionality. Suited for storing unstructure
backups, a



MinIO replica strategy

ERASURE CODING
(3+2)



- Green = **The original data comes into the system, is encoded into data and parity blocks and written to 5 drives.**
- Red = **When 1 drive fails, data on any other 3 drives can be decoded into the original data.**

MinIO User interface