Cloud Storage

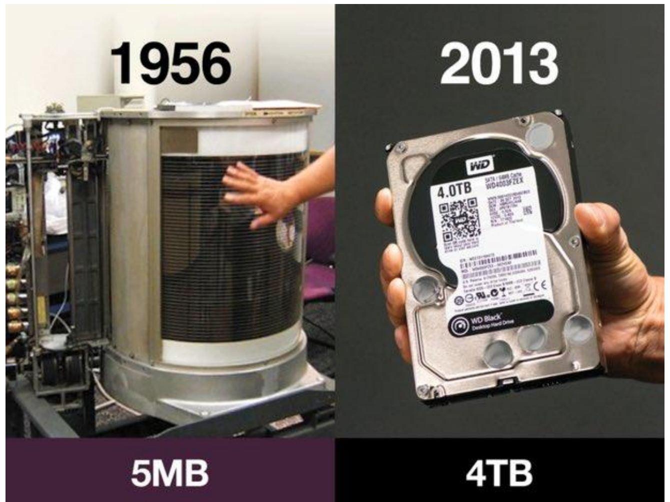
Cloud Storage and Distributed File Systems

Reference:

- ***[cam] Chapter 13***
- ***Material provided by instructor (learning ceph book)***

Storage Technology Evolution

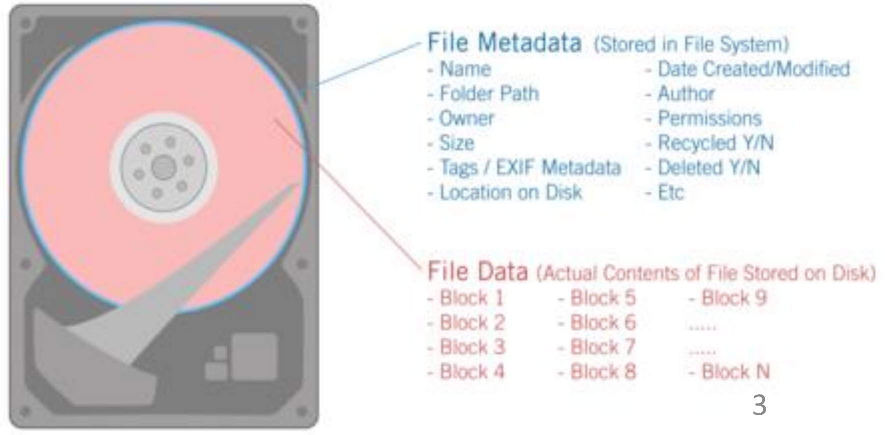
- In the last decade storage technologies have evolved at an accelerated pace: the volume of data stored and the access speed have increased tremendously
- Hard Disk technologies, in particular, has evolved providing significant more capacity and performance
- Their capacity evolved from Megabytes to Terabytes with a significant reduction in costs
- The transition to the Solid State Drive (SSD) technology allowed to remove moving parts from Hard Drives thus ensuring a significant reduction in data access time, from hundreds milliseconds to tens milliseconds



Parameter	1956	2016
Capacity	3,75MB	10TB
Average Access Time	~600ms	2,5-10 ms
Average Life Span	~2000 Hours	~22500 Hours
Price	9200$/MB	0,032$/MB
Physical Volume	1,9m^3	34cm^3

Block storage and File System

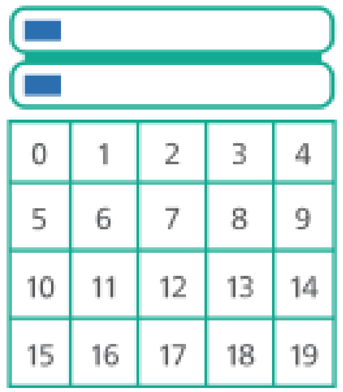
- The simplest type of storage that can be implemented with a hard drive is ***block storage***, the possibility of store a continuous blocks of raw unstructured data
- A block, often referred as physical record, is a sequence of bytes or bits, blocks can directly stored on the physical sector of the hard drive
- On top of a set of blocks, a ***file system*** is usually created: the file system organizes the data into files and manages the metadata required to store files on different blocks
- A file system is usually adopted by OS to store user's and systems' files

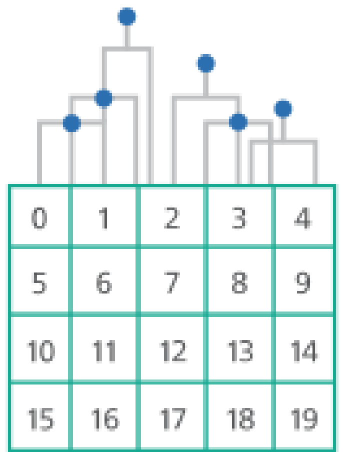


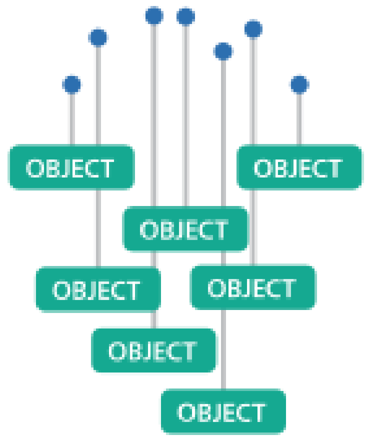
Object storage

- Recently a new storage type has been created, the ***object storage***
- Objects are bundled data (for instance a file) with the corresponding metadata
- Each object has associated a unique ID, that is calculated based on the file content and meta-data
- Applications can access the object via its ID, the set of metadata is not defined a-priori and can be extended
- Each object is immutable, a change produces another version that is stored as a new object (an incremental change system can be introduce to minimize data replication in the system)

Comparison



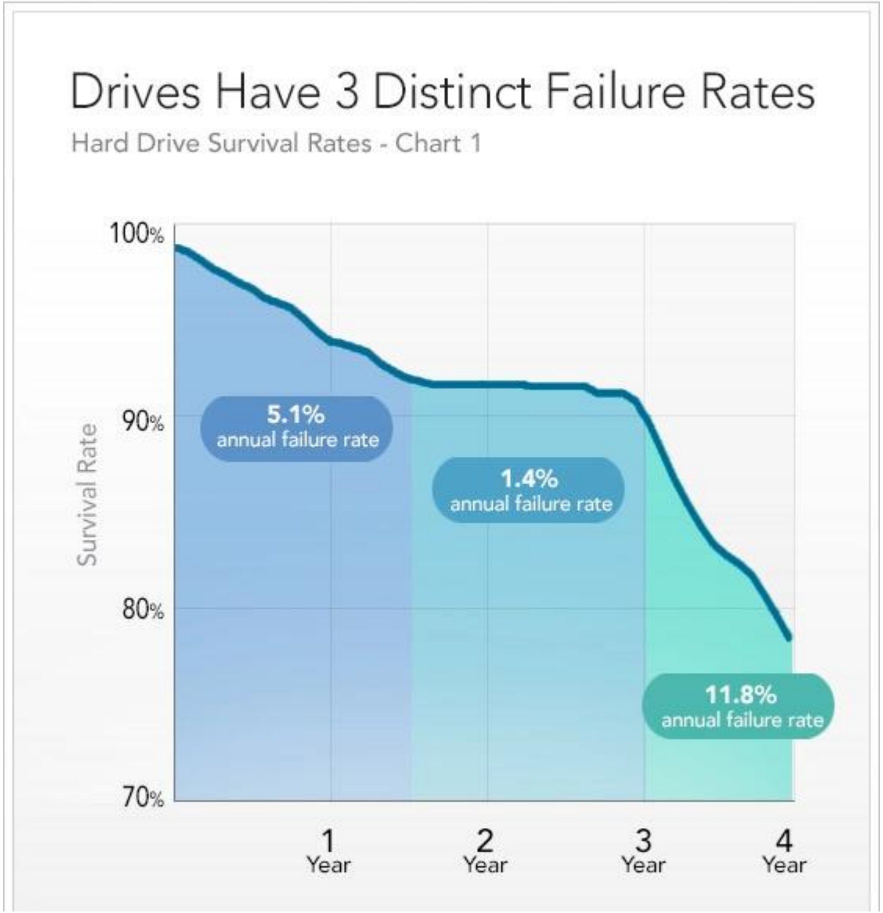




Block	File	Object
Client Via OS	Client Via OS	Client is App
Fixed Sys Attributes	Fixed Sys Attributes	Custom Metadata
Transactional Data	Shared Changing File	Shared Semi-Static File
Performance	Access, Single Site	Scalable, Multi-Site

Single Server Storage Model

- The storage of the traditional computing model is composed of one or more hard drives connected to **one server**
- Hard Drives can fail, their lifespan depends on the load, however, on average they have an expectancy of approximately 5 years
- In order to handle **failure on a single server** the RAID technology is employed, to ensure replica and fault tolerance
- The maximum storage is limited to the physical number of bays (hard drives slots) of the server
- In order to overcome this **physical limitation** distributed file systems are employed

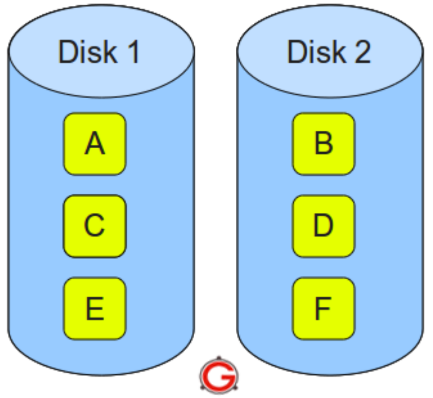


RAID

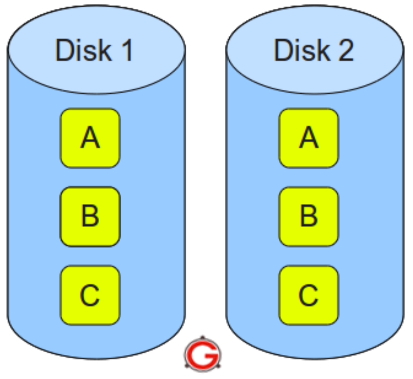
- RAID (Redundant Array of Independent Disks) is a technology that combines multiple physical hard drives into one or more logical hard drive (virtual)
- Different RAID schema are possible depending on the purpose: data redundancy, performance improvement or both
- RAID functionalities can be implemented in hardware (through a RAID controller to which the hard drives are connected) or via software
  - In the former, the RAID controller takes care of managing the hard drivers, consequently RAID functionalities are hidden from the OS, which access only to the logical hard drive
  - In the latter, RAID functionalities are implemented by the OS which manages access to the physical hard drives and the creation of the logical unit

RAID Schema

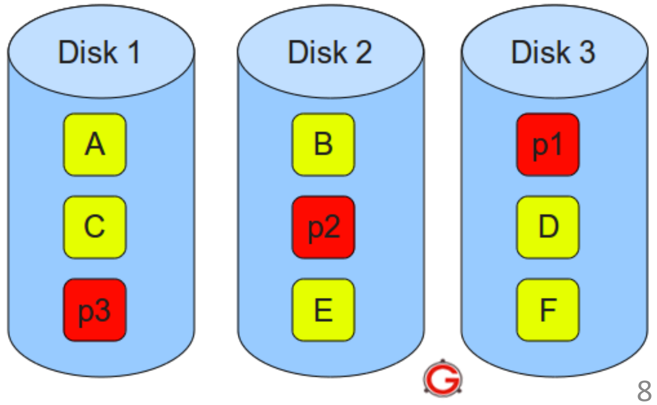
- According to the RAID schema adopted, the blocks of the logical hard drive are stored in the physical drives
- Three schema are popular:
    - **RAID 0**, adopted for performance, where blocks are striped across the physical hard drive for performance
    - **RAID 1**, adopted for maximum redundancy, where blocks are mirrored across the physical hard drives for redundancy
    - **RAID 5**, tradeoff between performance and redundancy, where blocks are striped across hard drives, however, a parity block (e.g. calculated via XOR) is added to a set of block to ensure fault tolerance, i.e. to recover the data if a failure occurs



RAID 0 – Blocks Striped. No Mirror. No Parity.



RAID 1 – Blocks Mirrored. No Stripe. No parity.



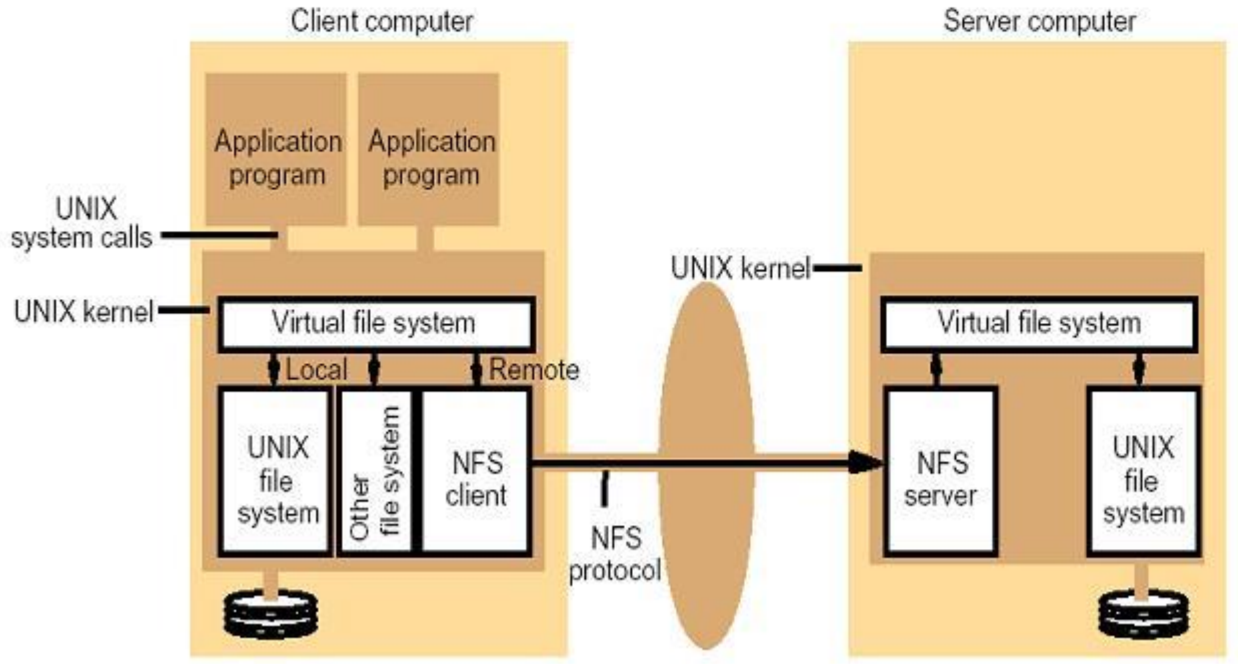
RAID 5 – Blocks Striped. Distributed Parity.

Distributed File System

- In order to overcome the limitations of a single server storage, Distributed File Systems (DFS) were defined
- A DFS allows to improve server’s storage capabilities by distributing the file system over different servers
- Data transfer and synchronization is achieved exploiting a local LAN
- A DFS can be used to create a large file system that puts together all the storage of different servers or it can be used to enlarge the storage of a single server by extending its capacity through the storage another server that usually have large storage capacity
- Different distributed file systems have been defined, one of the first examples (still widely adopted today) is NFS or Network File System

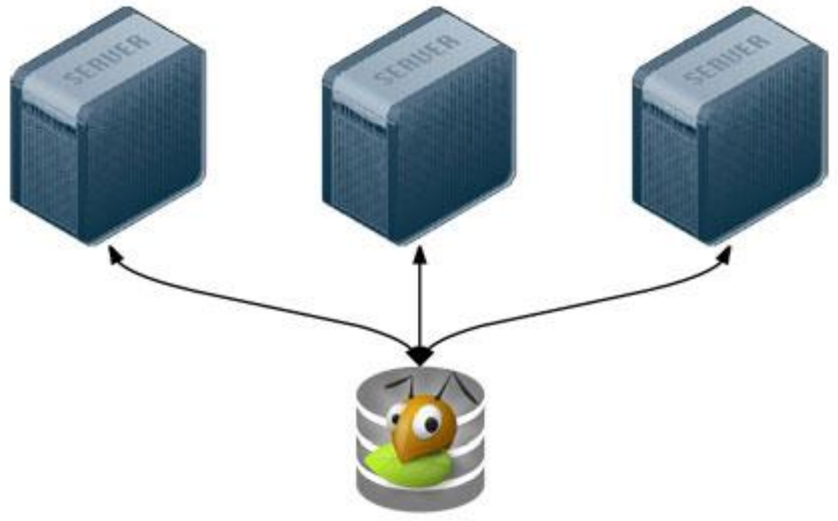
NFS

- NFS adopts a client-server approach: a central server offers access to clients to its local file system
- NFS defines a communication protocol between client and server, the messages are synchronous messages in order to ensure consistency and simplify the implementation
- The remote file system is accessed by applications running in each client in the same manner of local files are accessed
- All NFS functionalities are implemented inside the Linux kernel, NFS functionalities are hidden from applications



GlusterFS

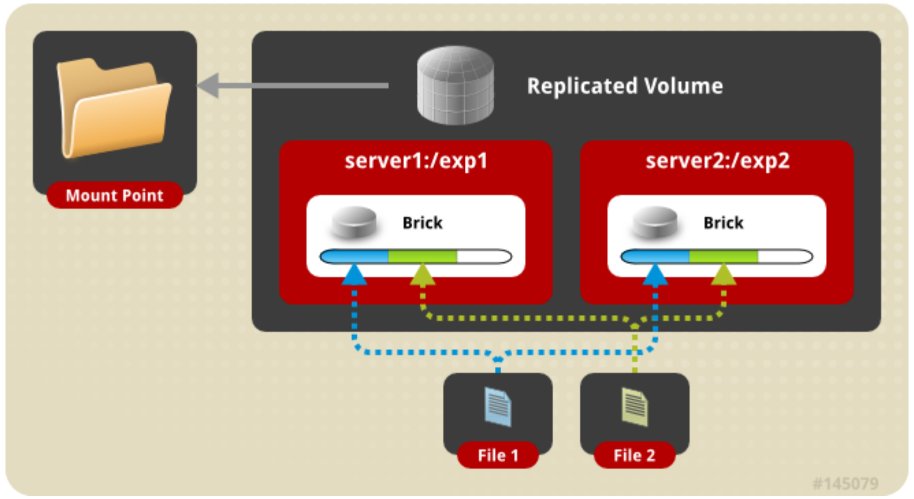
- Although NFS is a shared storage its **centralized architecture** refrains its usage in practical deployments
- **Distributed** alternatives are usually adopted to increase resiliency to failure and guarantee scalability exploiting storage locally available
- *GlusterFS* is an example of DFS that creates a distributed file system piecing together the storage capabilities available on all nodes
- GlusterFS can be used locally in the same way is configured NFS
- There is no distinction between clients and server, *all the nodes participate offering some of the local storage*

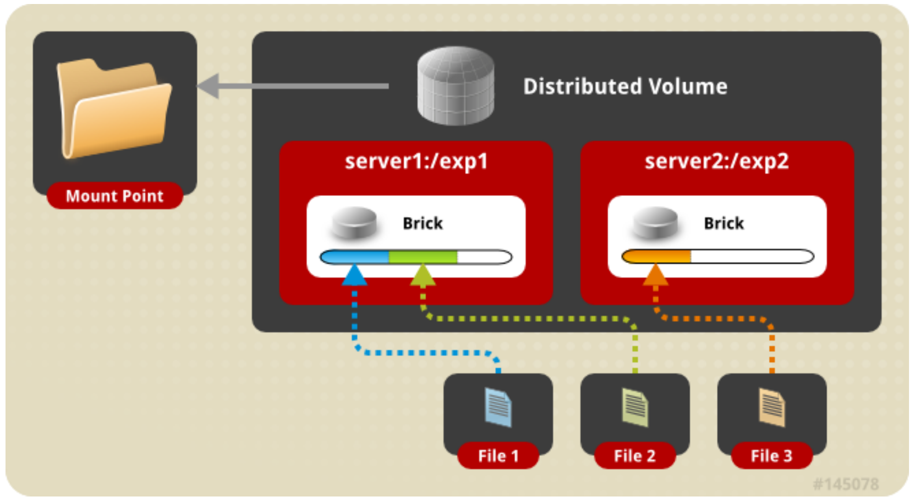


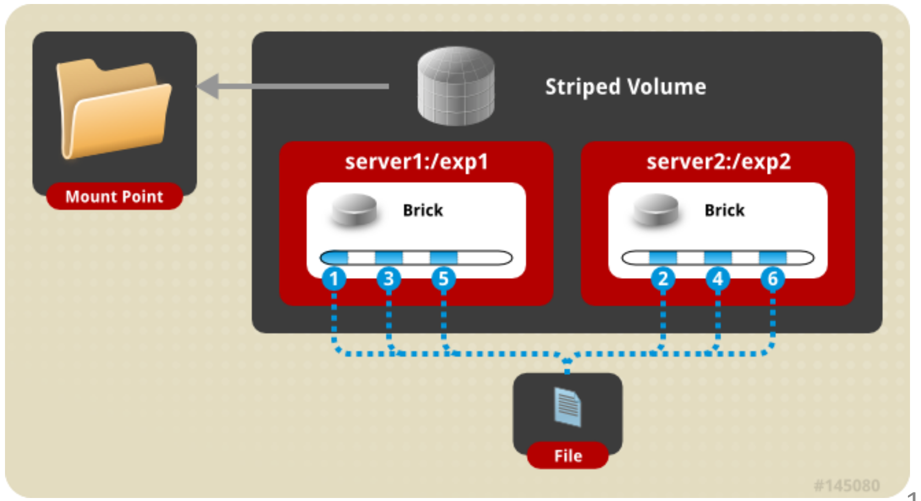
GlusterFS (Distributed, replicated volume)

GlusterFS – Basic Modes

- GlusterFS is *highly configurable*, with different levels of redundancy and replica
- Basic configuration includes: **replicated** volumes, **distributed** volumes and **striped** volumes

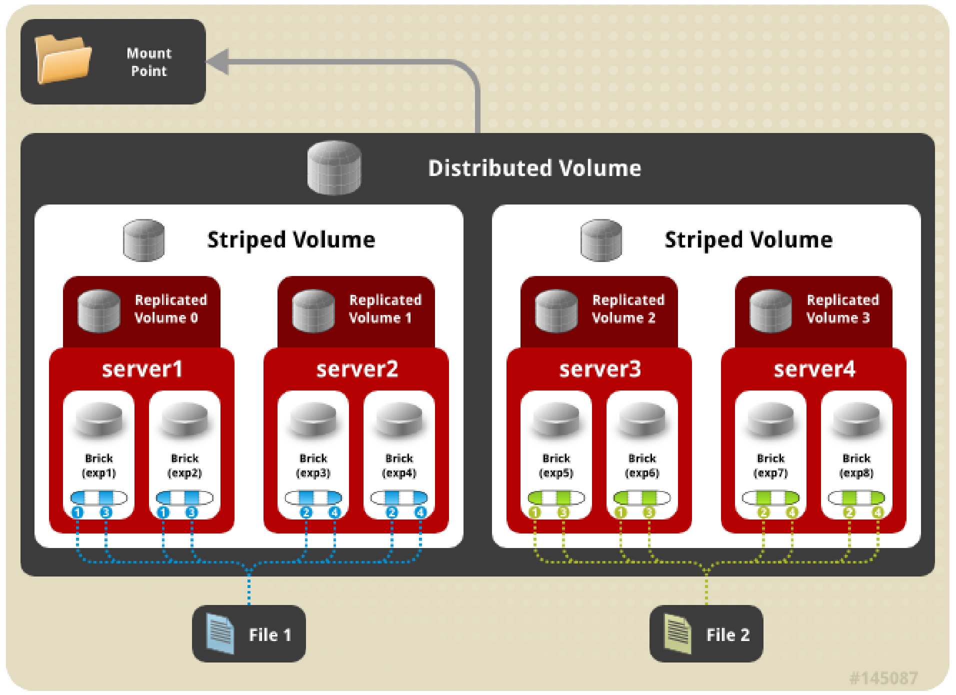


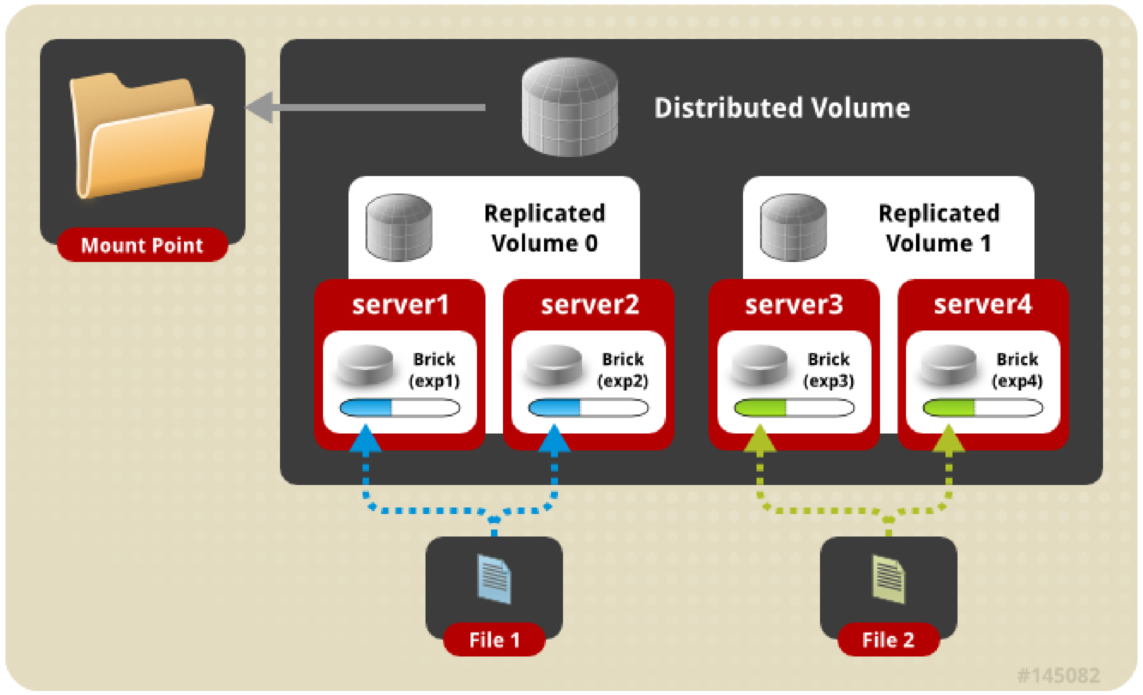




GlusterFS – Advanced Modes

- In order to meet different requirements different combination of basic modes are allowed: **striped replicated** and **distributed replicated**



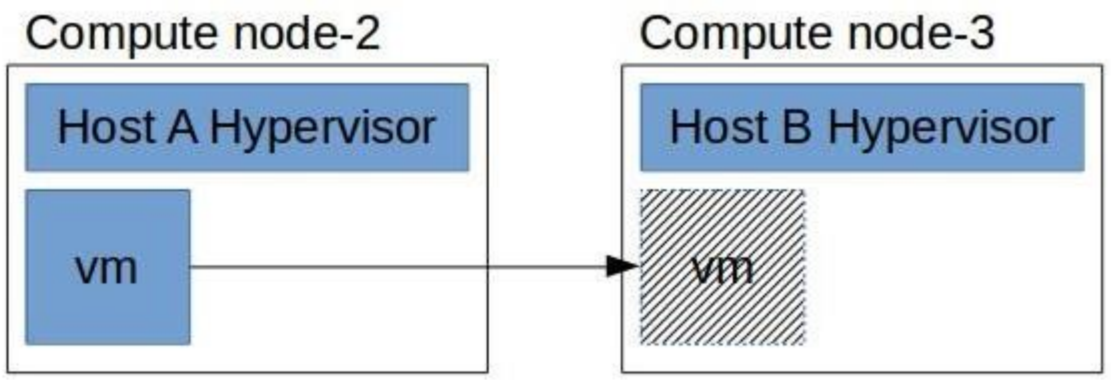


Cloud Storage

- Storage requirements of cloud computing platforms is different from the traditional computing model
- Its main requirement is that the storage scales, i.e. it can scale with the amount of data that the platform has to store, for instance:
  - It scales with the number of VMs that the platform has to handle
  - It scales with the number and the size of the volumes created
  - It scales with the amount of data the data generated by different service, e.g. the object storage Swift
- Exploiting local hard drives in cloud computing platforms is feasible, the hypervisor creates virtual drives or volumes for the VMs on the local hard drive of the physical machine on which it is installed
- Such solution, however, is not scalable as it still has the limitations of the traditional computing model

VM Live Migration

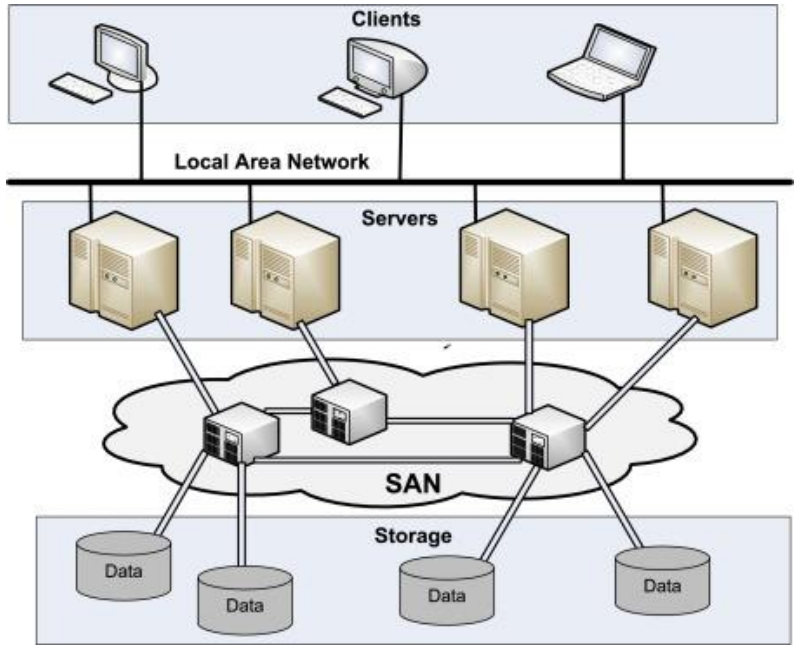
- In addition to this, using the local storage does not allows to implement VM live migration
- **Live Migration** allows a VM to be moved dynamically from one compute node to another, while still the VM is up and running
- Live Migration a cloud platform to move a VM minimizing the down-time and without turning the VM off



Live migration requires a shared storage, otherwise, it needs the VM to be shut off and the virtual hard drive to be moved from the local storage of a compute node to another with significant downtime

Storage Area Network

- In order to overcome this limitation a Storage Area Network (SAN) can be introduced
- A SAN is a high-speed network of specific high-capacity ***storage devices***
- Those ad hoc storage devices (they are not regular servers) are connected to the servers that compose the cloud platform to provide block-level storage
- The storage can be exploited by the platform to run its services, e.g. create VMs virtual hard drive, create volumes, etc.
- While SAN storage devices can include tape libraries, disk-based devices are more common, they use high speed communication technologies like Fiber Channel
- Ad hoc protocols are introduced for the communication between servers and storage devices, one of them is the iSCSI protocol, an IP-based storage networking standard for linking data storage facilities, which provides block-level access to storage devices by carrying SCSI commands over a TCP/IP network



Software Storage Solutions

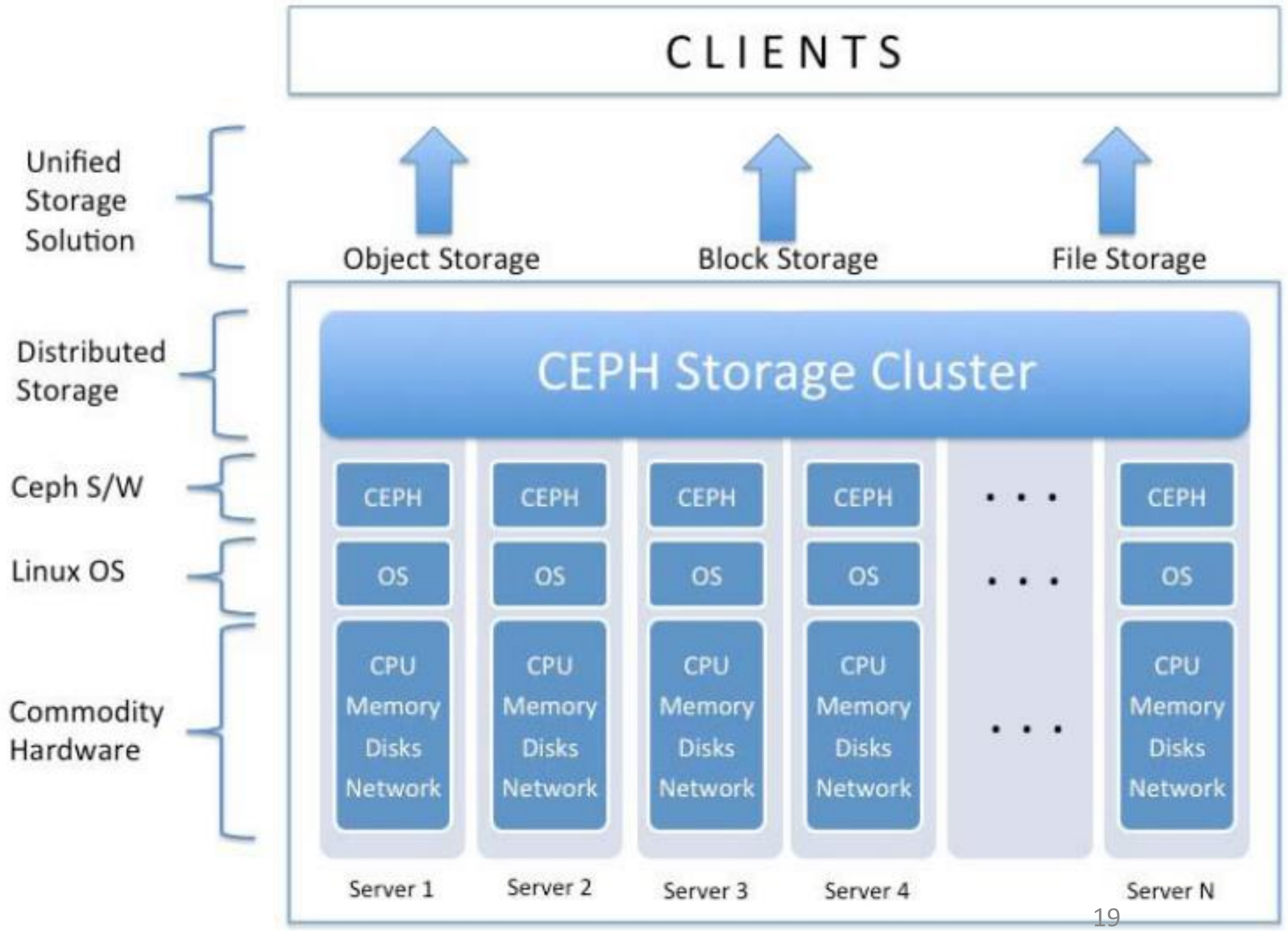
- SANs require ad-hoc hardware solutions that are **complex and expensive**
- For these reasons not all cloud deployments can justify them
- In order to overcome the need of expensive storage solutions, recently distributed storage systems have gained popularity as inexpensive alternatives to SANs
- These solutions, inspired by the literature on DFS, have as main goal to provide a unified storage solution to offer scalable, high-performing, without single point of failure that is based on **general-purpose commodity hardware**
- Among those solutions, one of the most popular at the moment is ***Ceph***

Ceph

- Ceph provides an enterprise-grade, robust, and highly reliable storage system on top of commodity hardware
- At the core of Ceph there are the following features:
  - Every component must be scalable
  - There can be no single point of failure
  - The solution must be software-based, open source, and adaptable
  - Ceph software should run on readily available commodity hardware
  - Everything must self-manageable wherever possible
- The result is that ceph handles by itself data replication and failure recovery, consequently there is no need anymore to employ technologies like RAID for reliability and failure handling

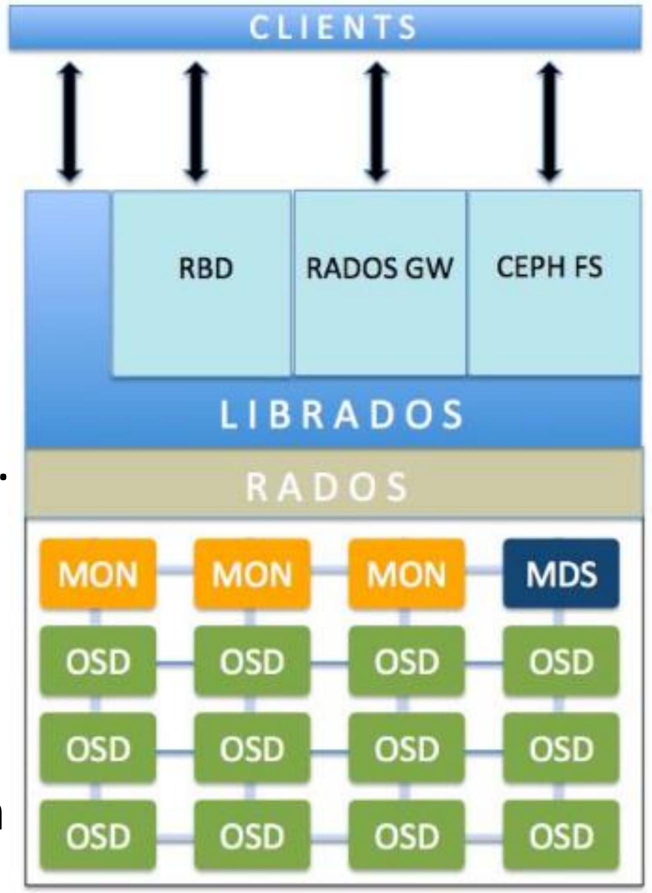
Ceph Services

- It can be installed on regular servers to create a distributed storage system (a **Ceph cluster**) in which the capacity of all the hard drives of all the servers is made available
- Ceph offers different storage types: object storage, block storage and file system storage



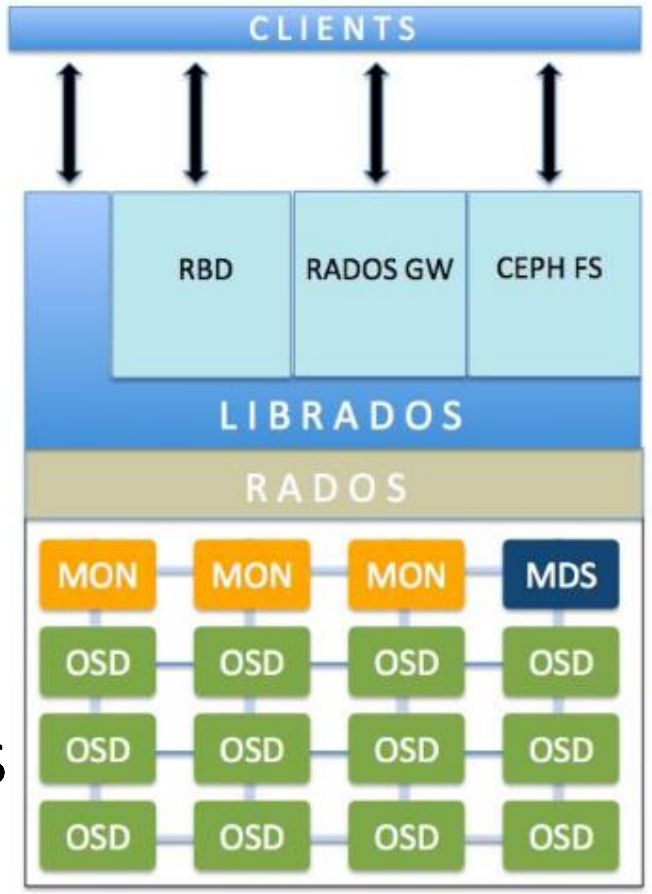
Ceph Architecture

- A Ceph storage cluster is made up of several different software services
- Each of these service takes care of unique Ceph functionalities and it is separated from the others
- Clients (i.e. applications or cloud services) interact with one of such components depending on the required storage service type
- At the **core** of Ceph we have **RADOS (Reliable Autonomic Distributed Object Store)**
- Everything in Ceph is stored in the form of objects, and the RADOS object store is responsible for storing these objects, irrespective of their data type. Other storage types are created on top of this basic object storage
- The RADOS layer makes sure that data always remains in a consistent state and is reliable. For data consistency, it performs data replication, failure detection, and recovery, as well as data migration and rebalancing across cluster nodes
- Applications that want to access to the object storage service interacts with RADOS via LIBRADOS, a library that offers a native interface to RADOS available for different programming languages



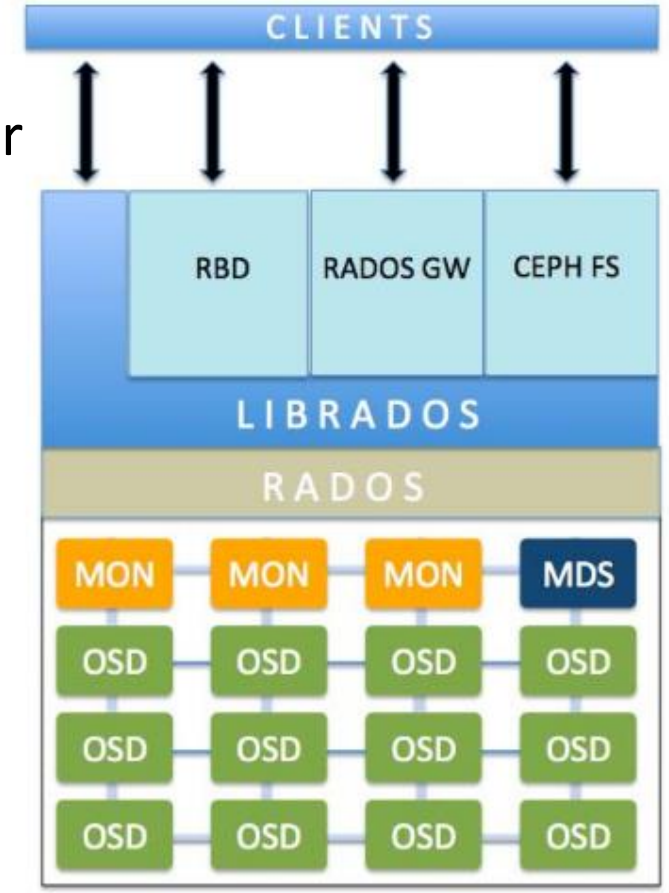
Ceph Architecture

- In order to offer clients storage services other than the object storage, additional services are created on top of LIBRADOS:
  - ***Ceph Block Device, formerly known as RADOS block device (RBD)***, provides block storage, which can be mapped, formatted, and mounted just like any other disk to the server. A Ceph block device is equipped with enterprise storage features such as thin provisioning and snapshots. It can be exploited to create volumes or virtual hard drives for VMs
  - ***Ceph File System (CephFS)*** offers a POSIX-compliant, distributed filesystem of any size. CephFS relies on Ceph MDS, Ceph Metadata Server (MDS), which keeps track of file hierarchy and stores metadata only for CephFS. A Ceph block device and RADOS gateway do not require metadata, hence they do not need a Ceph MDS daemon.



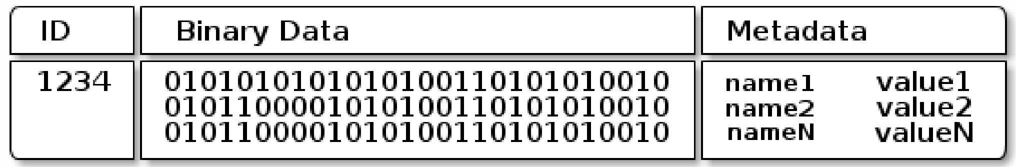
Ceph Architecture

- Core of Ceph is the RADOS layer, which provides all Ceph's features, including distributed object store
- Functionalities like high availability, reliability, no single point of failure, self-healing are provided by implementing in the RADOS layer the CRASH algorithm
- The RADOS layer functionalities are implemented in a distributed manner by the services OSD and MON
- **<u>OSD</u>**, Ceph Object Storage Device, is the most important building blocks in Ceph, as it takes care of storing the actual data on the physical hard disk drives of each node in the cluster handling single I/O operations on the disks. Usually, for each physical hard drive a different OSD instance is created.
- **<u>MON</u>**, Ceph monitor, is responsible for monitoring the health of the entire cluster. These monitor services maintain the cluster state and store critical cluster information (both status and configuration). A set of MON instances is deployed in order to ensure fault tolerance

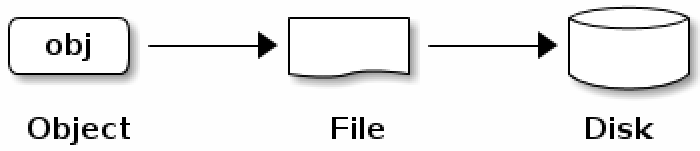


Ceph Object

- A Ceph object comprises data and metadata components that are bundled together and provided with a globally unique identifier
- The unique identifier makes sure that there is no other object with the same object ID in the entire storage cluster, and thus guarantees object uniqueness
- Unlike file-based storage, where files are limited by size, objects can be of enormous size along with variable-sized metadata
- Objects are stored in Object-based Storage Device (OSDs) in a replicated fashion, which provide high availability. When the Ceph storage cluster receives data-write requests from clients, it stores the data as objects.
- The OSD daemon then writes the data to a file in the OSD filesystem

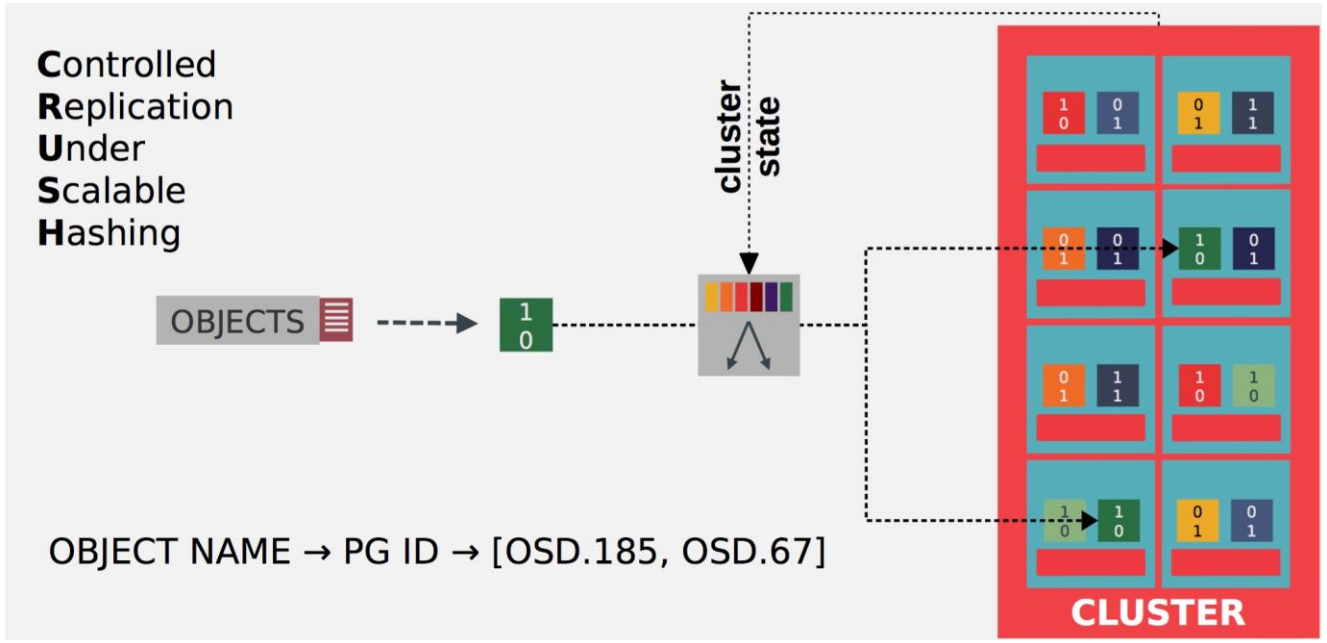


ID	Binary Data	Metadata
1234	010101010101010100110101010010 010110000101010011010101010010 010110000101010011010101010010	name1 value1 name2 value2 nameN valueN



CRUSH Algorithm

- Ceph Clients and Ceph OSDs both use the CRUSH algorithm to efficiently compute information about object location
- Through this algorithm both clients and OSDs can compute the location of an object (fro write or read) independently instead of having to depend on a central lookup table
- CRUSH provides a better data management mechanism compared to older approaches, and enables massive scale by cleanly distributing the work to all the clients and OSD daemons in the cluster
- CRUSH uses intelligent data replication to ensure resiliency, which is better suited to hyper-scale storage



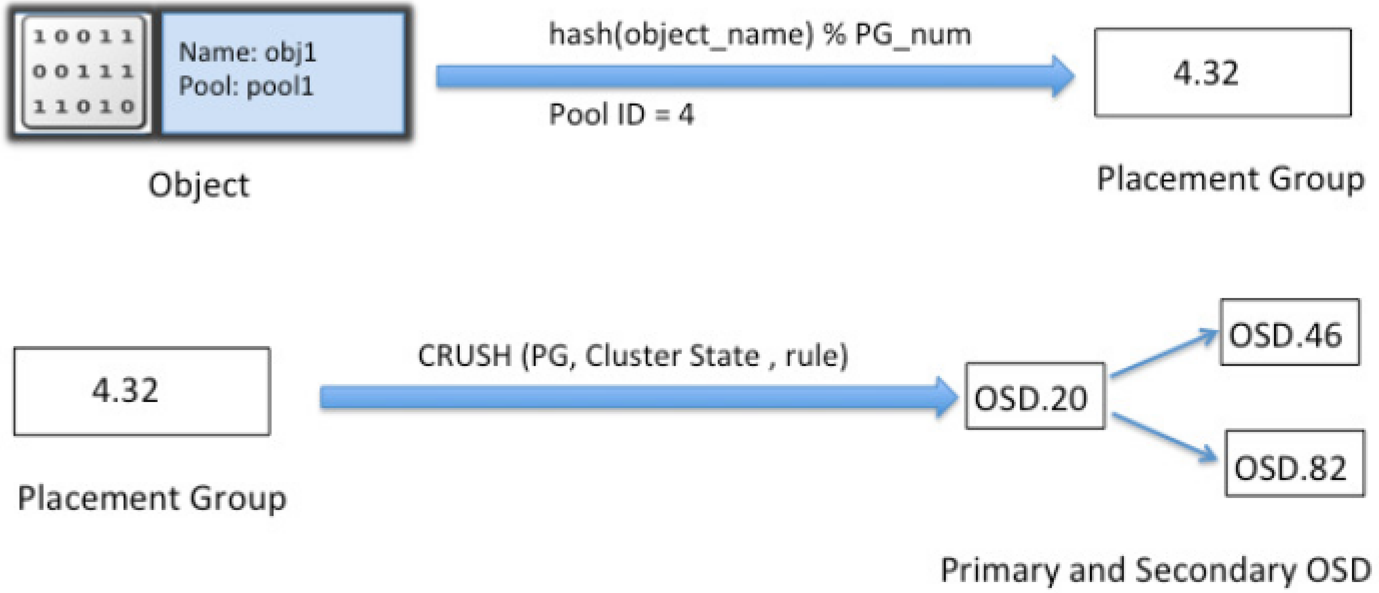
The algorithm computes the location of an object based on the current cluster status

Cluster Map

- The status of the cluster is represented through a Cluster Map
- The Map can be retrieved from any MON instance includes the following information:
  - The Monitor Map contains indicates the current epoch, when the map was created, and the last time it changed
  - The OSD Map contains the list of pools, replica sizes the list of OSDs and their status (e.g., up, in)
  - The PG Map contains the details on each placement group such as the PG ID, the Up Set, the Acting Set, the state of the PG (e.g., active + clean)
  - The Cluster Map contains a list of storage devices, the failure domain hierarchy (e.g., device, host, rack, row, room, etc.), and rules for traversing the hierarchy when storing data
- A placement group determines the actual OSD (and consequently the physical hard drive) on which an object is saved

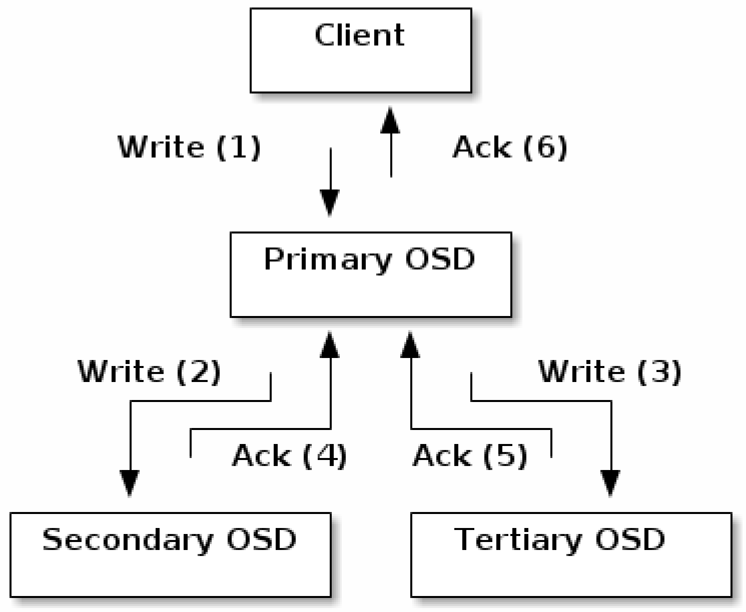
Data Placement

- Before Ceph Clients can read or write data, they must contact a Ceph Monitor to obtain the most recent copy of the cluster map
- Then, the client first derives the placement group by applying a hash function to the object ID
- In order to compute the actual OSDs to save the objects the CRUSH algorithm is executed to translate the placement group on the list of OSDs based on the cluster state
- The result is a list of ordered OSDs, a primary OSD and a set of secondary OSDs that store the object (primary OSD) and its replicas (secondary OSDs)



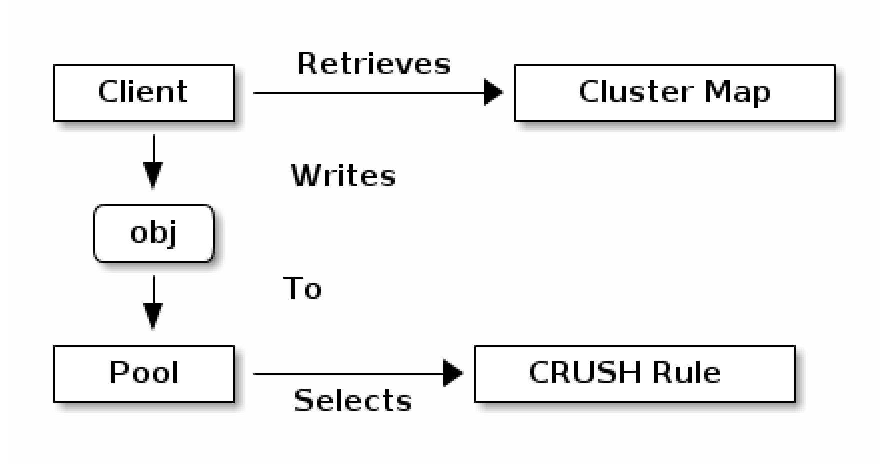
Data Replication

- The client writes the object to the identified placement group in the primary OSD
- Then, the primary OSD with its own copy of the CRUSH map identifies the secondary and tertiary OSDs for replication purposes
- After that it replicates the object to the appropriate placement groups in the secondary and tertiary OSDs (as many OSDs as additional replicas)
- Eventually it responds to the client once it has confirmed the object is stored successfully
- With this mechanism, Ceph OSDs relieve Ceph clients from that duty, while ensuring high data availability and data safety



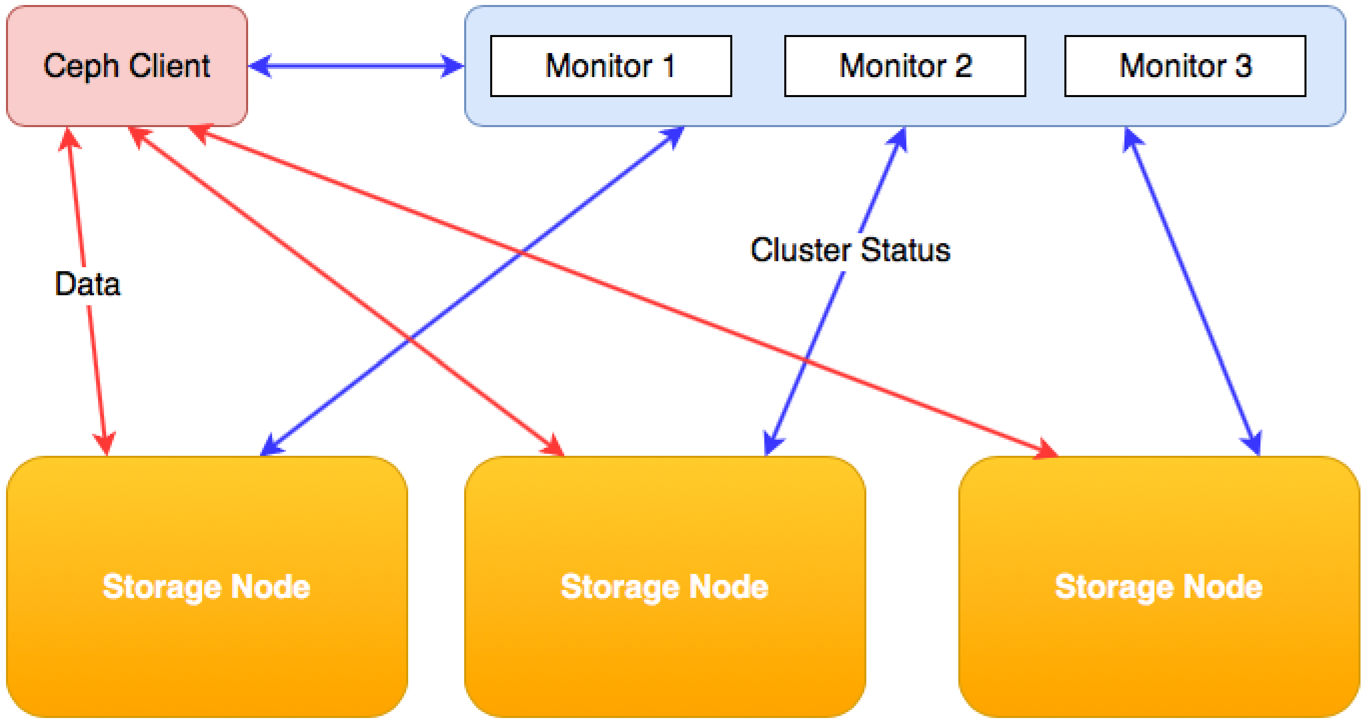
Pools

- The Ceph storage system supports the notion of ‘Pools’, which are logical partitions for storing objects
- Ceph Clients retrieve a Cluster Map from a Ceph Monitor, and write objects to pools
- Different pools can be instantiated to accommodate data of different applications, which also have different requirements for instance in term of replication



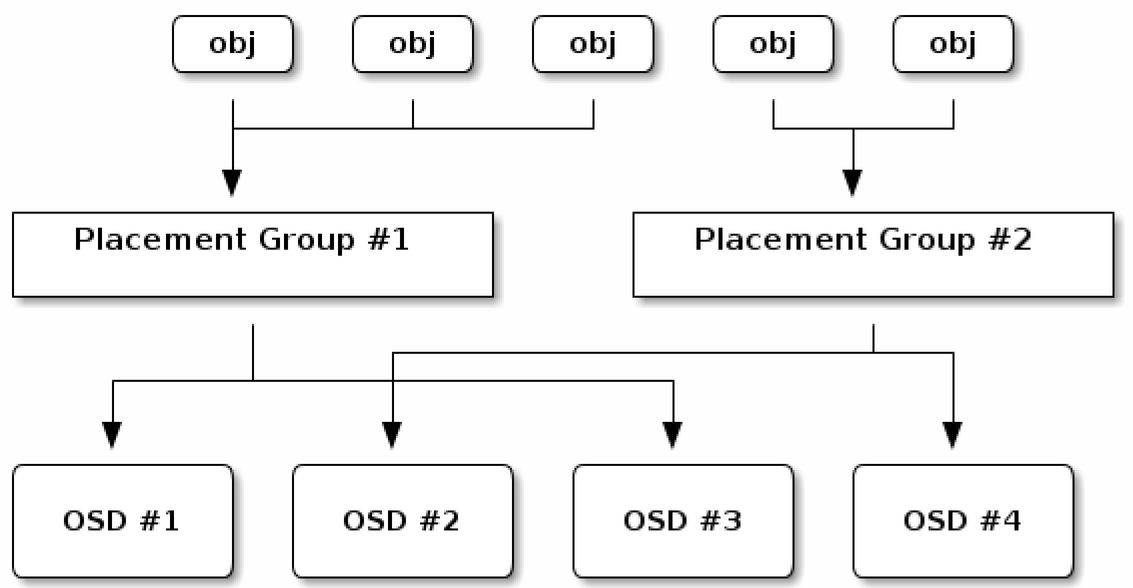
MON Redundancy

- A Ceph Storage Cluster can operate with a single monitor; however, this introduces a single point of failure (i.e., if the monitor goes down, Ceph Clients cannot read or write data)
- For added reliability and fault tolerance, Ceph supports a cluster of monitors. In a cluster of monitors, latency and other faults can cause one or more monitors to fall behind the current state of the cluster.
- Ceph must have agreement among various monitor instances regarding the state of the cluster. Ceph always uses a majority of monitors (e.g., 1, 2:3, 3:5, 4:6, etc.) and the Paxos algorithm to establish a consensus among the monitors about the current state of the cluster.



Placement groups computation

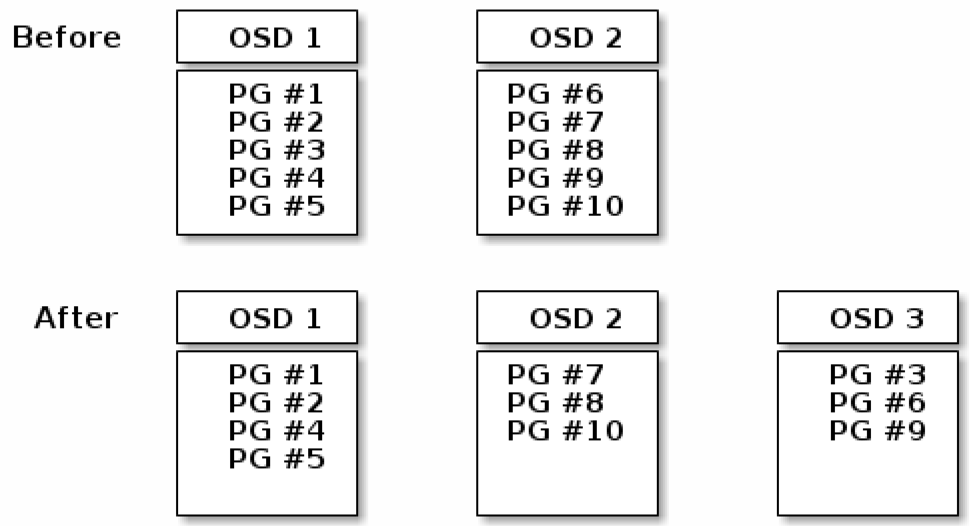
- Each pool has a number of placement groups. CRUSH maps PGs to OSDs dynamically.
- When a Ceph Client stores objects, the hash will map each object to a placement group
- The CRUSH algorithm maps each placement group to one or more Ceph OSDs
- This mapping is computed based on the current failure zones of the cluster, so data can be considered safe and available even if some components fail
- In addition to this, groups are mapped in order to ensure proper data balancing across OSDs, proportionally to the physical disk size of each one



The definition of failure zones does not only take into account different hard drives and different servers but it can be modified in order to include other factors like the network infrastructure or the architecture of the power supply system

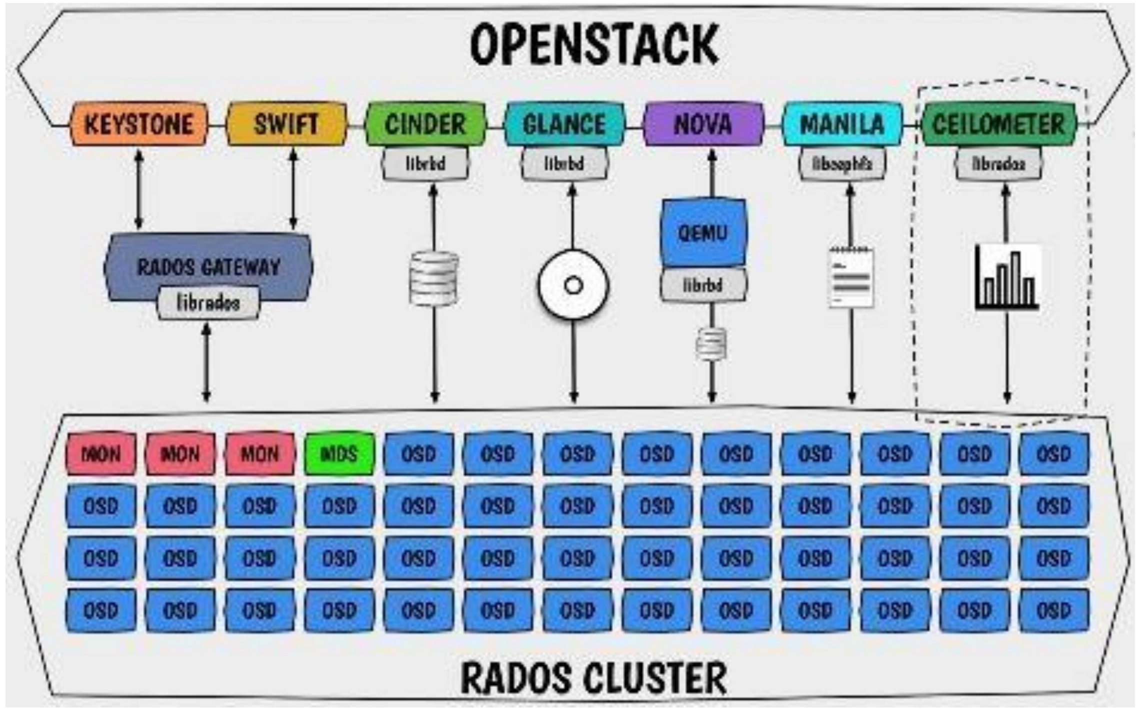
Recovering and Rebalancing

- Mapping objects to placement groups creates a layer of indirection between the Ceph OSD Daemon and the Ceph Client
- This layer of indirection between the Ceph OSDs and the Ceph Client allows the Ceph Storage Cluster to grow (or shrink) and rebalance where it stores objects dynamically, for instance due to failures
- For instance, when you add a Ceph OSD Daemon to a Ceph Storage Cluster, the cluster map gets updated with the new OSD and consequently all the mapping of the placement groups
- This changes the object placement, thus resulting in rebalancing, i.e. movement of data across different OSDs to ensure proper balance across OSDs



Caph and OpenStack

- Many OpenStack services can exploit Ceph as cloud storage technology
- For instance
  - Cinder can exploit Ceph to store VM volumes, in this case Ceph block storage is exploited
  - Swift can exploit Ceph to store the its objects by using the default object storage service
  - Ceilometer can use Ceph to store telemetry data
  - Glance can use Ceph to store OS templates for Instance creation
  - Nova can use Ceph to store VM virtual hard drives



Storage as a Service

- Among the services offered by cloud provider one of them is the Storage as a Service (STaaS)
- This service offers access to cloud storage capabilities over the Internet
- In other words they offer access to their cloud storage infrastructure
- Often STaaS services adopts the Object storage model, they exposes a REST interface to store/read objects
- This can be used for backup or long term storage of large data
- One example is the Swift OpenStack service that can be used to expose a Ceph cloud storage for instance
- AWS exposes a large set of cloud storage services, from S3 storage (similar so Swift) or AWS Glacier (for cheap long term storage)