Block Storage vs File storage

This is a key topic to understand the fundamental differences between Block and File storage.

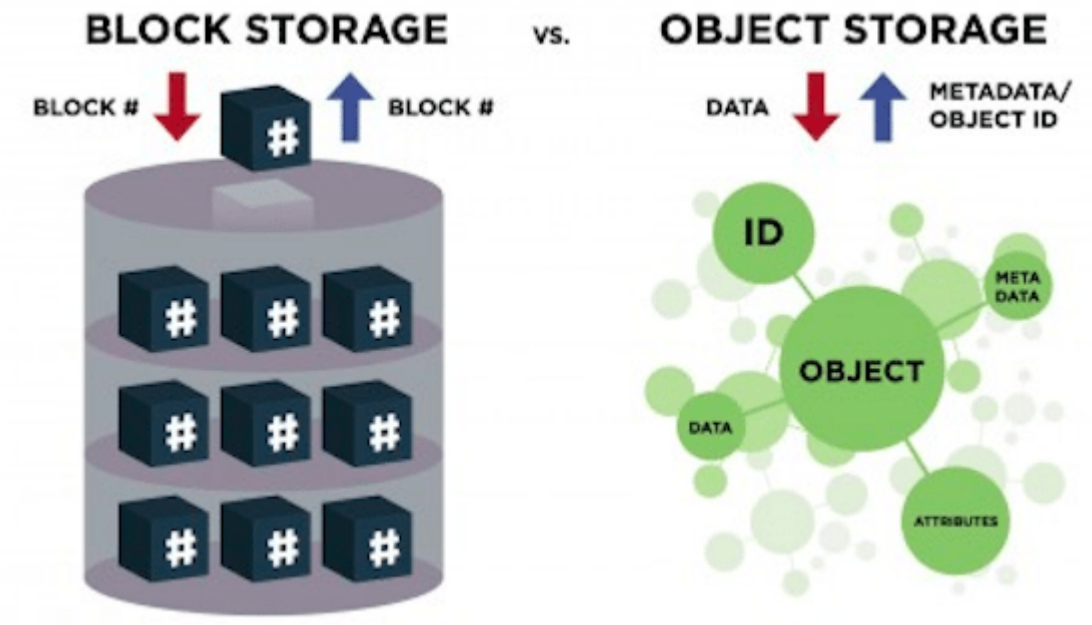
Block Storage vs File storage:

It is important to understand the differences between Block and file storage in general and understand where one would use each.

What is object storage? Object storage (also referred to as object-based storage) is a general term that refers to the way in which we organize and work with units of storage, called objects. Every object contains three things:

1. **The data itself**. The data can be anything you want to store, from a family photo to a 400,000-page manual for assembling an aircraft.
2. **An expandable amount of metadata**. The metadata is defined by whoever creates the object storage; it contains contextual information about what the data is, what it should be used for, its confidentiality, or anything else that is relevant to the way in which the data is used.
3. **A globally unique identifier**. The identifier is an address given to the object in order for the object to be found over a distributed system. This way, it's possible to find the data without having to know the physical location of the data (which could exist within different parts of a data center or different parts of the world).

How block storage and object storage differ :-



With block storage, files are split into evenly sized blocks of data, each with its own address but with no additional information (metadata) to provide more context for what that block of data is. You're likely to encounter block storage in the majority of enterprise workloads; it has a wide variety of uses.

Object storage, by contrast, doesn't split files up into raw blocks of data. Instead, entire clumps of data are stored in, yes, an object that contains the data, metadata, and the unique identifier. There is no limit on the type or amount of metadata, which makes object storage powerful and customizable. Metadata can include anything from the security classification of the file within the object to the importance of the application associated with the information. Anyone who's stored a picture on Facebook or a song on Spotify has used object storage even if they don't know it. In the enterprise data center, object storage is used for these same types of storage needs, where the data needs to be highly available and highly durable.

However, object storage generally **does not** provide you with the ability to incrementally edit one part of a file (as block storage does). Objects have to be manipulated as a whole unit, requiring the entire object to be accessed, updated, then re-written in their entirety. That can have performance implications.

Another key difference is that block storage can be directly accessed by the operating system as a mounted drive volume, while object storage cannot do so without significant degradation to performance. The tradeoff here is that

so without significant degradation to performance. The tradeoff here is that,

unlike object storage, the storage management overhead of block storage (such as remapping volumes) is relatively nonexistent.

What problems does object storage solve?

Object storage is ideal for solving the increasing problems of data growth. As more and more data is generated, storage systems have to grow at the same pace. What happens when you try to expand a block-based storage system beyond a hundred terabytes or beyond multiple petabytes? You may run into durability issues, hard limitations with the storage infrastructure that you currently have, or your management overhead may go through the roof.

Solving the provisioning management issues presented by the expansion of storage at this scale is where object storage shines. Items such as static Web content, data backup, and archives are fantastic use cases. Object-based storage architectures can be scaled out and managed simply by adding additional nodes. The flat namespace organization of the data, in combination with its expandable metadata functionality, facilitate this ease of use.

Another advantage to object storage its responsiveness to the need for resiliency while mitigating costs. Objects remain protected by storing multiple copies of data over a distributed system; if one or more nodes fail, the data can still be made available, in most cases, without the application or the end user ever being impacted. (Downtime? What downtime?) In most cases, at least three copies of every file are stored. This addresses common issues including drive failures, bit-rot, server and failures, and power outages. This distributed storage design for high availability allows less-expensive commodity hardware to be used because the data protection is built into the object architecture.

What about the tradeoffs?

Object storage has the potential to provide IT departments a great deal of value. It can save money in infrastructure costs by allowing the organization to use less-expensive hardware, it can reduce management time through ease of scalability, as well as provide tremendous flexibility for certain types of storage needs.

But, as exciting as it sounds, object storage is not the answer to all your storage problems. Sometimes, block storage is a far better fit. There are use cases where object storage performs beautifully, scales out seamlessly, and solves all sorts of management headaches, but in other situations, it outright fails to meet the needs of your application.

You have to decide which type of architectural approach is appropriate for your needs, as you balance the requirements for a scalable storage solution that provides resilience and performance. The primary issues are eventual consistency or strong consistency. Object storage systems are eventually consistent while block storage systems are strongly consistent.

Eventual consistency can provide virtually unlimited scalability. It ensures high availability of data that needs to be durably stored but is relatively static and will not change much, if at all. This is why storing photos, video, and other unstructured data is an ideal use case for object storage systems; it does not need to be constantly altered. The downside to eventual consistency is that there is no guarantee that a read request returns the most recent version of the data.

Strong consistency is needed for real-time systems such as transactional databases that are constantly being written to, but provide limited scalability and reduced availability as a result of hardware failures. Scalability becomes even more difficult within a geographically distributed system. Strong consistency is a requirement, however, whenever a read request must return the most updated version of the data.

Workloads for object versus block storage

Object storage works very well for unstructured data sets where data is generally read but not written to. Static Web content, data backups and archival images, and multimedia (videos, pictures, or music) files are best stored as objects. Databases in an object storage environment ideally have data sets that are unstructured, where the use cases suggests the data will not require a large number of writes or incremental updates.

Geographically distributed back-end storage is another great use case for object storage. The object storages applications present as network storage and support extendable metadata for efficient distribution and parallel access to objects. That makes it ideal for moving your back-end storage clusters

across multiple data centers.

It is not recommended you use object storage for transactional data, especially because of the eventual consistency model outlined previously. In addition, it's very important to recognize that object storage was not created as a replacement for NAS file access and sharing; it does not support the locking and sharing mechanisms needed to maintain a single accurately updated version of a file.

Because block-level storage devices are accessible as volumes and accessed directly by the operating system, they can perform well for a variety of use cases. Good examples for block storage use cases are structured database storage, random read/write loads and virtual machine file system (VMFS) volumes. However, since block storage has essentially no additional storage-side metadata that can be associated with a given block other than the address of that block, performance degrades in geographically distributed systems. The further the block storage gets from the application, the more the performance suffers due to latency issues.

Object storage in practice Despite what some people suggest, object storage is not an emerging technology. Data stored as objects have already approached the exabyte scale (1000 petabytes) representing trillions of objects. Companies like Amazon (with S3) provide object storage via its public cloud platform at massive scale, while object storage can be implemented in the company data center using technology like OpenStack's Swift or EMC's Atmos.

When you begin to think about what types of items you should move into object storage, start with the low-hanging fruit. Take a look, for example, at low I/O workloads such as network share, which may be on a NAS device. In this instance, you are limited to the size of the unit. Without an easily expandable option, you are forced to overprovision in order to leave room for future expansion for the users, resulting in underutilization. By moving this workload to an object store, you're not limited to the amount of space each unit holds. Nodes can be added easily within the object storage paradigm, allowing full use of the disks you purchased.

Regardless of the path you choose, it is important to familiarize yourself with the advantages and limitations of the architecture in order to get the most

value for your company.