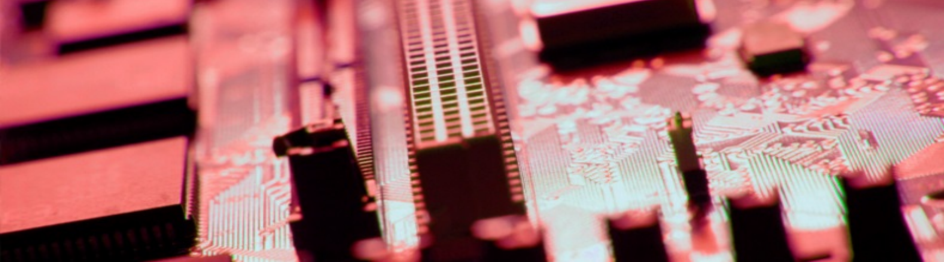


LCI workshop
on high performance clustered computing

Linux Clusters Institute: Block vs Object Storage

Georgia Tech, August 15th – 18th 2017

J.D. Maloney | Storage Engineer
National Center for Supercomputing Applications (NCSA)
malone12@illinois.edu

What Are We Trying To Solve

Data Growth

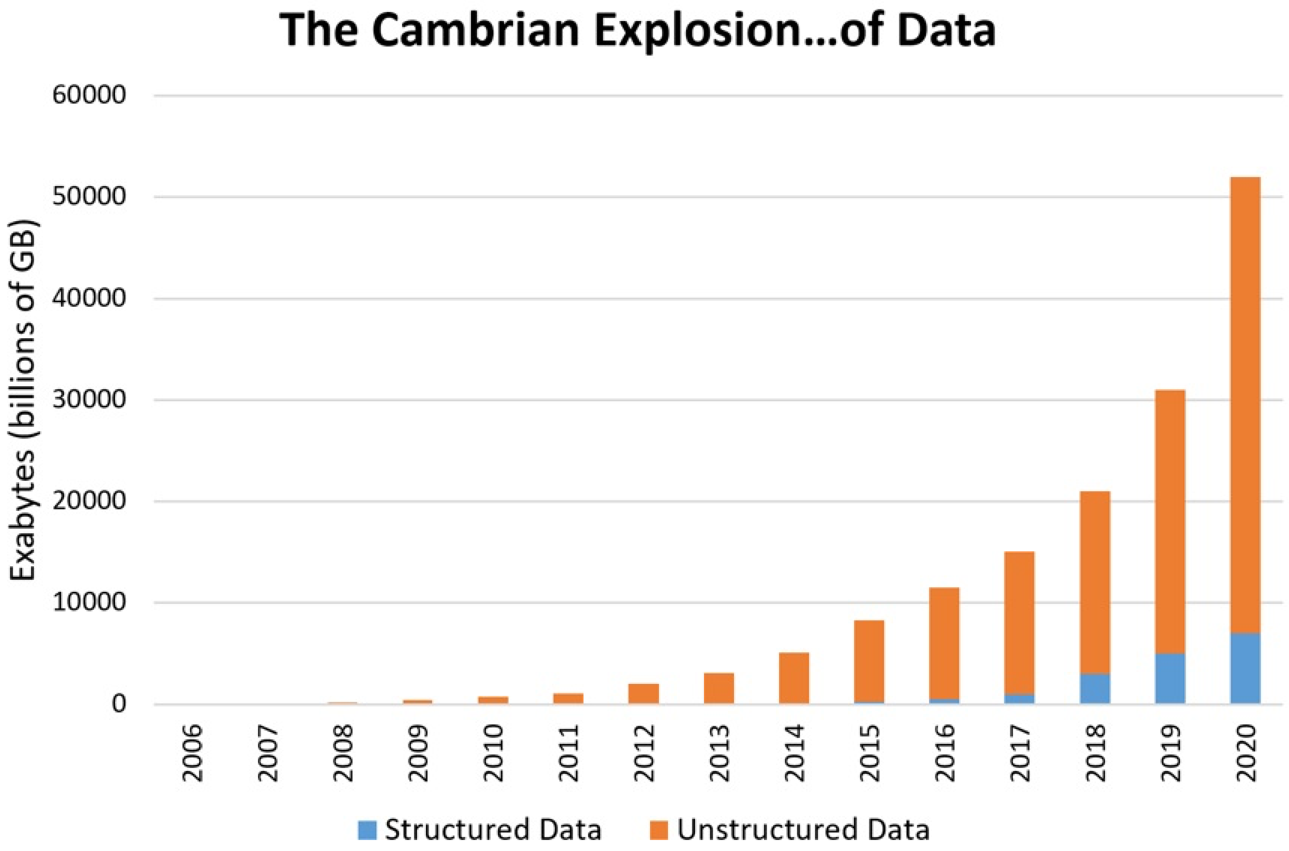


Image Credit: http://www.eetimes.com

Structured Data

- Vast majority of data does not fall in this category
- Data that lives in a format that can be parsed or queried
    - Relational Databases
    - Sensor metric data
    - Tables and spreadsheets
- Machines can understand it, and analyze it
- More understandable to people as data can be coalesced into meaning that helps make decisions
    - This data easily turned into charts or tables
    - What is put in research findings to represent the output of the work

Unstructured Data

- Vast majority of data does fall in this category
- Data that can’t be queried easily to get out relevant on needed information
- Big examples
  - Images
  - Video
  - Text Files
  - Sound files
- Can come out of things with image sensors, microphones, or other data capture instruments
- Emails, social media posts, human generated text

Research Data Growth

- How does this resonate with the data that we deal with in HPC?
  - Unstructured data sounds a lot like the data scientists gather to process on our machines
- Almost all areas of science dive into these types of datasets when doing their work
  - Studying human interaction/trends via social media
  - Imaging the galaxy or things on the earth
  - It’s a by-product of us capturing the world around us

Research Data Growth

- How have handled this data in the past?
  - Throwing data into directories of course
  - Organize by date/sensor/location etc.
  - File trees are easy for people to see and navigate to find things
- What's going on now?
  - Datasets are getting bigger (more quantity, more size)
  - Resolution increases have led to rapidly growing file sizes
  - Lower cost of data gathering tools leads to vast increase in data
- Why did the user just try and create directory with 25 million files?
  - How this can manifest itself on in computing workloads

Object Storage

What It Is

- What we’d call files now -- now called objects, dropped into a bucket
- Just like you pour grains of sand into a bucket, that’s how objects are treated in object storage systems
- A great way to store unstructured data at large scales
- Much more flexible answer than POSIX file based solutions that are commonly deployed today

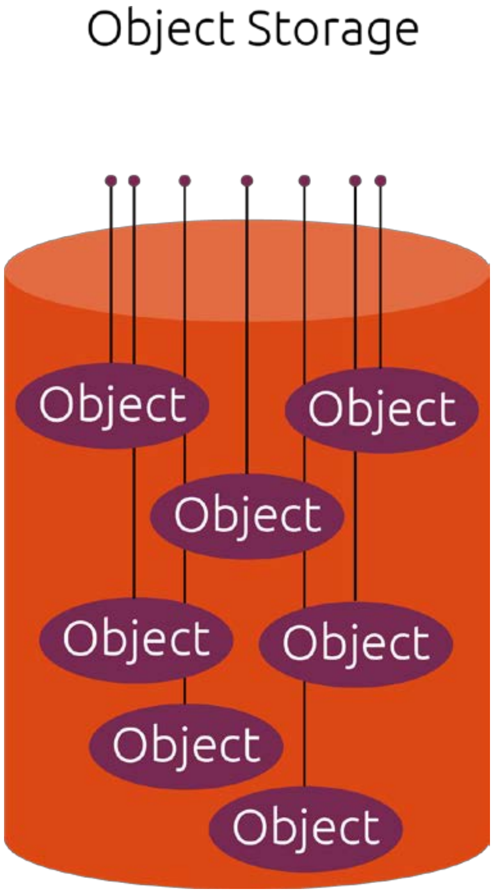
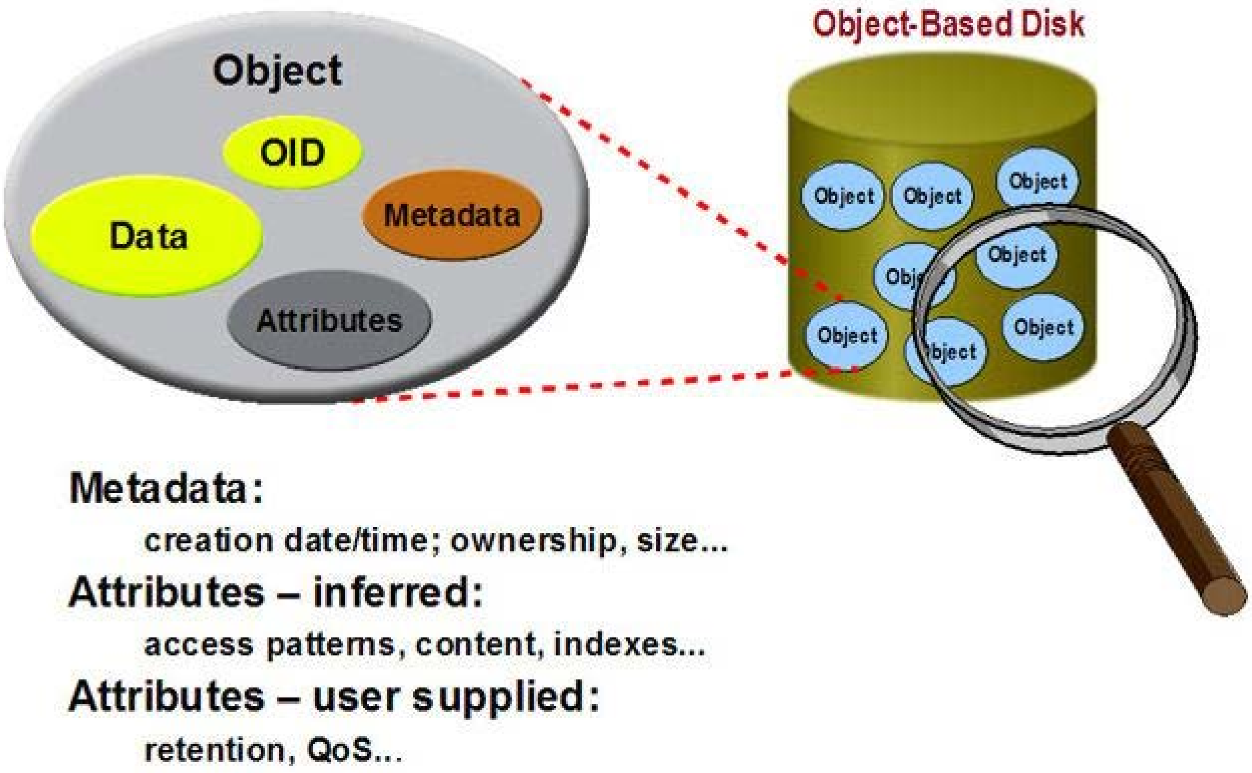


Image Credit: http://ubuntu.com

How It Works

- Each object is given a unique ID that can be used to retrieve the object from the system when it is needed by the user or application
- Data is stored in one flat level, no concept of hierarchy between different files
- Objects have metadata and attributes about them stored along with the object
  - Metadata defined by the system
  - Attributes are user supplied
- Access is done via GETs and PUTs

How It Works



Metadata:

creation date/time; ownership, size...

Attributes – inferred:

access patterns, content, indexes...

Attributes – user supplied:

retention, QoS...

Image Credit: http://storagegaga.com

How It Works

- Object storage systems can contain many different buckets for different user's data as well as different data sets
- Buckets can have quotas applied to them like normal file systems can to ensure fair use of system
- Attributes can support things like tagging that allows easier sorting and aggregating of objects

What It Solves

- The super deep directory problem
- Managing large amounts of unstructured data easily without the burden of POSIX
  - Directory locking is no longer an issue for highly concurrent file access to data in the same dataset
  - The CRUSH map in Ceph and the rings in Swift reduce metadata bottleneck seen on traditional file systems
- Ease of portability between different systems
  - Absolute file paths are no longer important
  - Object ID’s hold the key to everything and don’t change once the object is put into the system

What It Solves

- The scaling limitations we're starting to see with POSIX and the constraints it has
- Many in the industry see object based storage as key to making a successful jump to Exascale Computing
  - Large vendors are developing file systems that are object based, designed to be ready for the next wave of large machines
  - Intel has DAOS (Distributed Application Object Storage)
  - Seagate in their A200 appliance

The Challenges

Not a perfect system, there’s a reason we all still have traditional file systems

- Hard for humans to visualize and understand
  - No one wants to try and fish through a bucket with millions of objects by hand, practically impossible
  - Object IDs have no bearing on the content of the file, unlike a file name in a directory
  - Change for many years of historic use, needs to overcome the precedent that has been around
- New tools need to be developed to view and manage data
  - Lots of tools rely and file paths to retrieve data, easy to rewrite but takes the effort
  - View tools to understand the different attribute and metadata tags

The Challenges

Application Support

- Biggest Challenge facing object use
- High adoption in web-like areas, and great for bulk archival of data but traditional HPC workloads are a long ways off from native use
- Adopting legacy code to handle new file access is not always and easy task and requires programmer time and money to fund it
- Characteristics of object storage don't provide the same performance that traditional file systems do

The Challenges

Application Support (cont)

- Parallel access to the same object is not a concept which is how some big HPC applications
  - Would require fundamental overhauls to how code is written
- Many applications in all areas, but including HPC rely on a stack of dependencies which all need to able to handle the new backend

```
ion = dataSo
getConnection
connection.
tSQL = "SELECT
statement.exec
Set.next()) {
```

Difference Between Block & Object

To Put It Simply

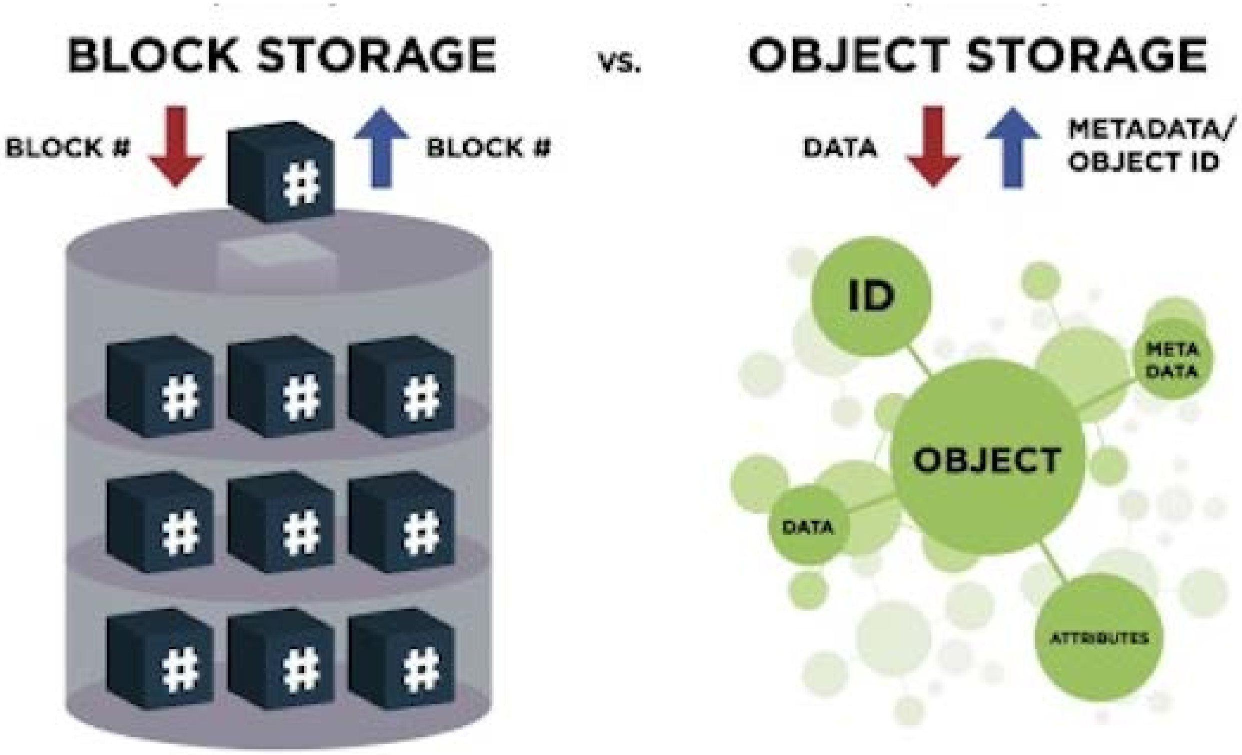


Image Credit: http://druva.com

Like But Not Same

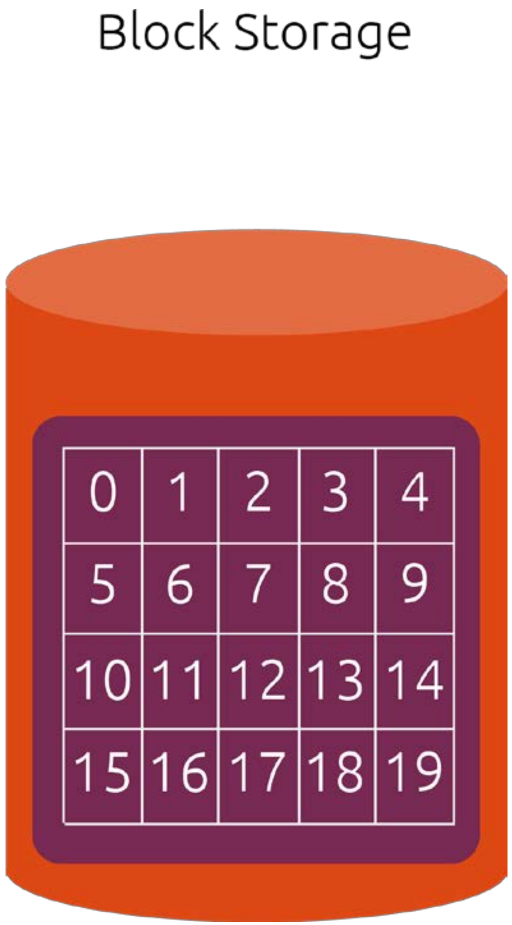
- Neither Block or Object is the same as traditional file storage, but they can sometimes get confused with each other
- Both are heavily used in cloud based deployments
  - Usually object storage holds user data, files users use on the system
  - Block storage holds virtual machines
  - Openstack for example has Swift for Object and Cinder for Block
- Neither are the same as traditional file storage

≠

Block Storage

What It Is

- A block is a very raw unit of storage, a file can span many different blocks, or just take up one
- Blocks don’t store any metadata, just data
- Blocks each have an address that is used to reference and retrieve them for use
- All blocks are the same size as each other, they aren’t dependent on what they store



Block Storage

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19

Image Credit: http://ubuntu.com

How It Works

- Chunks of data are served to hosts for their use
- The block presented to the host can be adjusted in size to appear as a volume of whatever the desired size is
- Host machine (physical or virtual) that receives the block storage then formats the block storage to whatever it desires (commonly EXT4, XFS, ZFS, etc.)
- Fabric connecting systems can be something like Fiber Channel, but often now done over Ethernet to machines
  - For example how Openstack uses block storage for volumes

How It Works

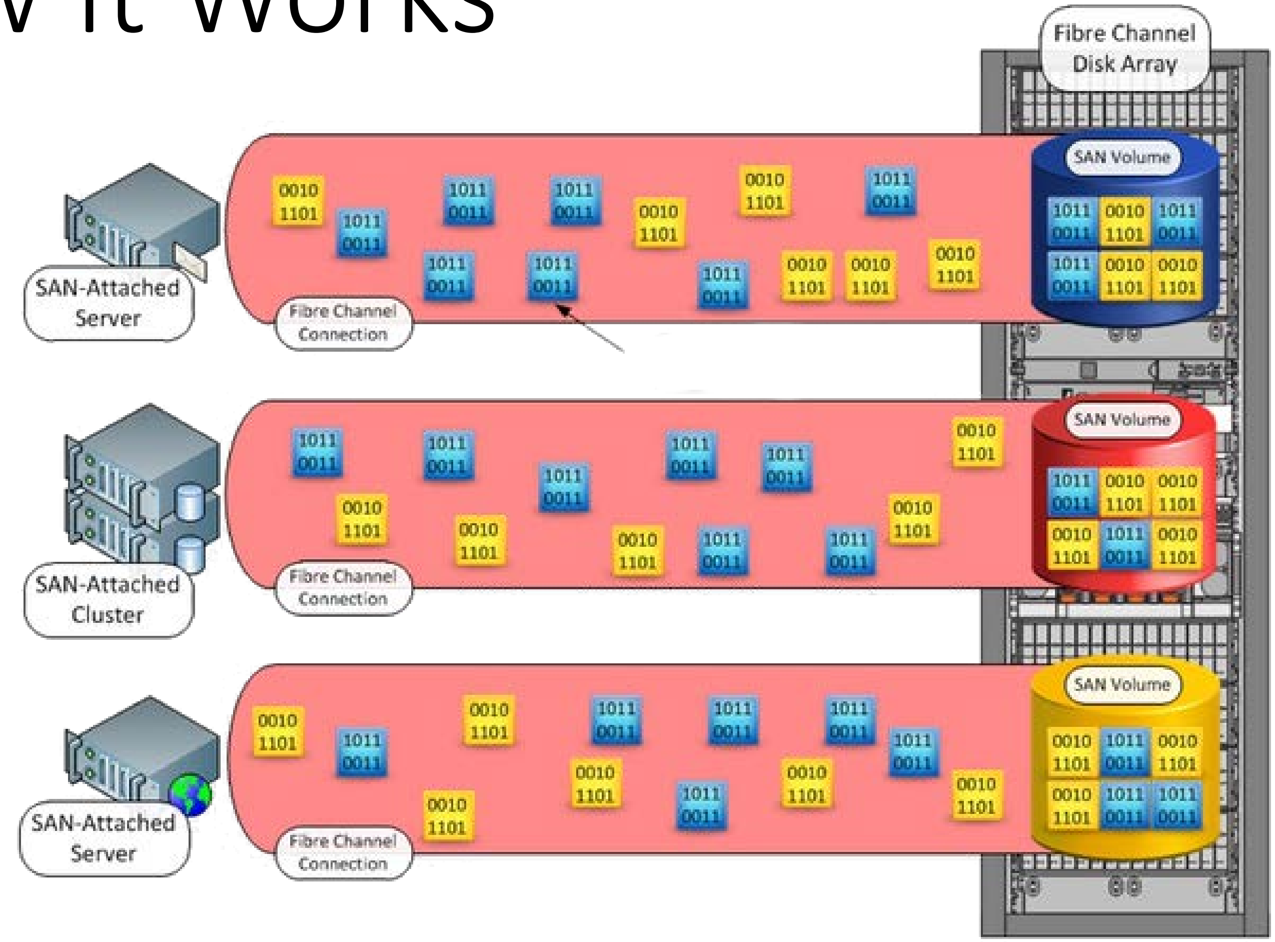


Image Credit: http://arstechnica.com

What It Solves

- Allows a large centrally managed storage system to serve capacity to many hosts
- Reduces the burden and overhead of management of disks, instead of having a bunch of isolated storage pools local to the hosts
- Enables very efficient use of storage capacity in the organization
  - Don’t have to decide the amount of storage needed in machine on purchase
  - Not limited to what the machine can physically hold
  - Disk can be added and provisioned when needed

What It Solves

- Easy snapshot ability versus traditional file storage
  - Only the part of the file that changes needs to be updated with the changes
- Performance upgrades either disk or network can boost performance for all systems that are leveraging the service
- Disk that looks local is used by a lot of applications and something people are used to

The Challenges

- Ties availability of disk to many machines (physical or virtual) to one central system
  - Outage of the main system causes loss of capacity for many machines
  - All eggs in the same basket principle
- Fabric contention for the SAN and/or the block storage servers can be an issue where as local disk is bound to the system it serves
- Network performance and latency can become a bottleneck even with expensive disk is used
- By itself, doesn't manage data, just provides the capacity to store the data

Acknowledgements

- Thanks to Sam Linston from University of Utah for help with slide development content
- Members of the SET group at NCSA for slide review
- Members of the steering committee for slide review

Questions