Object Storage: Adoption, Practice and Deployment

An Outlook Report from Storage Strategies NOW

June 24, 2013
By Deni Connor and James E. Bagley
Client Relations: Phylis Bockelman

Storage Strategies NOW
8815 Mountain Path Circle
Austin, Texas 78759
(512) 345-3850

SSG-NOW.COM
Executive Summary ........................................................................................................................................................................5
History ............................................................................................................................................................................................ 6
What is object storage? .................................................................................................................................................................. 6
Process Data as Objects ............................................................................................................................................................. 6
Access Objects via Internet Protocol and RESTful API ........................................................................................................... 6
Retrieve and update Objects based on Metadata ...................................................................................................................... 7
Retrieve and update Metadata without Retrieving Data from an Object ................................................................................ 7
Command Interface .................................................................................................................................................................... 7
Permissions and Functional Policies ......................................................................................................................................... 7
Capabilities Queries and Provisioning .......................................................................................................................................... 7
Multi-Tenant Usage and Billing Controls ........................................................................................................................................ 7
Runs on commodity x86 servers or purpose-built systems ..................................................................................................... 7
Object Storage Fundamentals ....................................................................................................................................................... 8
NAS and block interfaces .......................................................................................................................................................... 8
Alternative to RAID and traditional Replication ..................................................................................................................... 8
Load Balancing .......................................................................................................................................................................... 8
Data Protection .......................................................................................................................................................................... 8
Tagging, Search and Indexing ................................................................................................................................................... 8
Redundancy and Availability .................................................................................................................................................... 9
Data Distribution ....................................................................................................................................................................... 9
Unstructured Data ..................................................................................................................................................................... 9
Scalability ................................................................................................................................................................................... 9
Implementations ........................................................................................................................................................................... 10
Software Implementations ....................................................................................................................................................... 10
Commodity hardware ............................................................................................................................................................... 10
Purpose-built systems .............................................................................................................................................................. 10
Platform as a Service .............................................................................................................................................................. 10
Vendor/Product Profile ........................................................................................................................................................... 11
Vendor Name: Cloudian ........................................................................................................................................................... 11
Use case profile ....................................................................................................................................................................... 12
Sponsor

Cloudian
Dear Reader,

With the explosive growth of unstructured data, there needs to be a type of storage system that can store, analyze, manage and archive that data in an economical and effective fashion. Some organizations have turned to scale-out NAS systems to accomplish this goal, but they have been limited by the scalability and cost of the system.

With the advent of object storage systems, also comes confusion. How is object storage different than scale-out NAS? How does it differ from content-addressable storage? Why do so many large scale cloud storage systems, such as Amazon S3 and Facebook, rely on object storage?

SSG-NOW takes a look at technologies and products when such confusion exists in a specific market. This report will solve that problem and answer your questions about object storage. If as a customer, you have further questions about object storage, how it is implemented and deployed and where it is best used, please feel free to contact us. We will enjoy talking to you.

Deni Connor
Principal Analyst
dconnor@ssg-now.com

James E. Bagley
Senior Analyst
jbagley@ssg-now.com
Object storage refers to a class of storage systems that include protocols that are designed to operate over an Internet Protocol (IP) interface and provide efficiencies of scale that differ from typical file-based protocols. While the use cases for object storage have been predominantly based around cloud storage applications, they are useful for many other applications. For example, object storage can be effectively used for archival applications, large-scale data analytics and high-performance computing, surveillance and financial services – anywhere there is a large amount of data that needs to be stored and managed. Another area of object storage interest is in the data protection and disaster recovery software space, where systems are designed to create the metadata on an object store specific to the ability to recover data.

A primary characteristic of object storage is the ability to use metadata, that is, data that describes the attributes of the information stored in an object, and can include application-specific and future use-specific matter. This is generally a deeper set of descriptors than exist in the typical file systems such as CIFS and NFS. Even with portable operating system interface (POSIX) extensions, these file systems do not automatically keep an application-specific set of metadata, nor do they have a way for an application to control the placement of metadata and separate the metadata on a priority basis for future access.

While the metadata handling is a characteristic of object storage that provides an advantage for many operations, it does require the modification of applications to use the different storage protocol as opposed to simply operating with the file protocols or lower level block storage protocols. An example is a typical big-data analytic that asks ‘what has changed in the last four hours?’ Without specific change-related metadata stored in rapidly accessible areas, that query could take many days to answer, obviating the intent of the query.

An advantage of the IP protocol interface and use of RESTful APIs goes beyond the clear benefit of being able to access storage across the widest area possible. It also provides for placement policies based on the metadata provided by the application. This can result in the placement of data where it will be accessed the most, or archiving information without any other activities. Large web-scale applications can access data without driving through file-based trees or indices, but rather based on the content of the object within various storage operations, which are typically a simple set of commands like GET, PUT and DELETE. This facilitates the important nature of cloud storage (public, private and hybrid), which allows elasticity without intervention by administrative activities that need to decide specifically where to place data and what to change when deletion commands are processed.

While we have specifically focused on object storage as a class, it is important to note that many products by many vendors deliver network attached storage (NAS) functionality in various file formats through the use of an underlying object storage system. These systems can claim that they are object stores, but unless the underlying protocols are available at an IP level, it can be argued that these systems fall out of the definition used here.

A simple way to view object storage is that it is necessary to operate public cloud storage, but the use cases extend into many other types of storage applications. Organizations that require object storage have choices between software implementations, use of commodity or purpose-built hardware and software, or Platform-as-a-Service (PaaS), which is similar to cloud storage services.
History

Object-based storage can look back to 2001 when Peter Braam of Carnegie Mellon University conceptualized the Lustre File System. He called the implementation Lustre and founded a company called Cluster File Systems, who would commercialize the code base. Cluster File Systems was acquired by Sun Microsystems in 2007, and the technology was marketed as a solution of high-performance computing. When Oracle bought Sun in 2010, technology advancement of Lustre stalled until a company called Whamcloud was founded with the goal of extending its use and supporting it. Whamcloud, in turn, was acquired by Intel in 2012 and the intellectual property of Lustre was sold by Oracle to Xyratex.

Another popular implementation of object-based storage was EMC's Centera, first introduced in 2001 when EMC acquired FilePool. Centera became popular with the passage of financial controls with the Securities and Exchange Commission. The SEC’s ruling, 17a-3 and 4, required that brokers and dealers retain records of financial transactions on non-rewritable, non-erasable media, such as WORM. EMC led the battle in getting Centera ‘compliant’ with SEC regulations.

What is object storage?

In general, object storage has the following attributes:

1. Processes data as objects, which contain both data and an extensible set of data descriptors called metadata;
2. Accesses objects via an Internet Protocol interface and RESTful APIs;
3. Retrieves objects based on metadata;
4. Retrieves an object’s metadata without retrieving the data content;
5. Contains a command line interface (CLI) that typically includes object read, write, delete, data read, data write, data delete and metadata update functions;
6. Provides permissions and functional policies;
7. Allows capabilities queries and provisioning;
8. Provides multi-tenant security, usage and billing functions; and,
9. Runs on commodity x86 servers or purpose-built systems.

While object storage processes a command and interface protocol, underlying details of the storage are not necessary for the using application to understand. Having said this, many object stores have the ability to designate service levels and access priorities, and may include the ability to designate objects for different speeds of devices, such as capacity-optimized storage versus speed-optimized storage, such as solid state drives.

Process Data as Objects

Objects include both data in the form a bytes, and descriptors of the data (metadata). Depending on the protocol and capabilities of the object store, objects may be grouped into units, sometimes called ‘chunks’ or ‘blobs.’ An important characteristic of objects is that the descriptor set is defined by the using application and is extensible and can apply to a given group of objects or just individual objects. Objects are typically identified by the user or department within an organization along with other parameters that can include permissions and metadata-specific items.

Access Objects via Internet Protocol and RESTful API

An Internet Protocol interface allows objects to be accessed across the worldwide web using HTTP and HTTPS commands. This means that access can be extended to a class of client device that includes servers, personal computers, tablets, smart phones and other Web-based appliances, such as security cameras and industrial control devices.
Retrieve and update Objects based on Metadata

A major advantage of an object store is the ability to find objects based on attributes designated by the application that created the object. This greatly facilitates ‘Big Data’ analytics because only the objects that fit a certain criteria need to be delivered by the object store. For example, if an object describes an electronic transaction such as a retail order, metadata could include the type of product(s) ordered. This can be accomplished without the database administration normally involved in creating a row and column type access that needs to be rigidly defined in advance. Similarly, if the object contains archival data, the metadata could include subjects to facilitate future searches, such as geo data, subject contents and compliance related information.

Retrieve and update Metadata without Retrieving Data from an Object

Often the metadata will contain all of the information necessary to perform an application function and can be retrieved without using the bandwidth necessary to retrieve all of the data in the object. This benefits both the speed of applications and reduces resources used for application data processes.

Command Interface

The command interface to an object store is commonly referred to as a representational state transfer (REST or RESTful) interface. Depending on the intended use case, the object store command interface would minimally include object read, write, delete, data read, data write, data delete and metadata update functions. Beyond the basic input-output functions, the command interface could include a variety of additional commands as further described in the following information.

Permissions and Functional Policies

Like standard files, most object stores, especially those accessed from beyond a network’s firewall, will need to have a series of access permissions as well as functional policies that include who can create or delete objects. The permissions could include commands to access centrally controlled permissions such as Windows Active Directory or the Lightweight Directory Access Protocol (LDAP).

Capabilities Queries and Provisioning

Most command line interfaces and GUIs include the ability to query specific information from the object store, such as capacity utilization, capacity available and even what commands are available. These commands are closely related to provisioning commands where users and organizations can be allocated capacities, service levels, bandwidth throttles and other functionality access.

Multi-Tenant Usage and Billing Controls

Even if an object store is used by a single organization, its access by various departments may be controlled and monitored by an administration interface. Since most object stores can function as stand-alone cloud storage service devices, the ability to securely segregate tenants and bill or chargeback for usage, including capacities used, storage speed and input-output bandwidth used should be available.

Runs on commodity x86 servers or purpose-built systems

Object storage systems are characterized by their ability to run on industry-standard x86 platforms or purpose built appliances. In these cases, software is installed on these ‘nodes’ and provides the functionality of the object storage system. For this reason, one of the advantages of object-based storage is its cost-effectiveness to deploy.
Object Storage Fundamentals

NAS and block interfaces

While technically not a requirement of object storage and specifically out of the scope of an object store, most vendors offer alternate access to the object store beyond the REST interface. These interfaces typically include standard file protocols like CIFS and NFS. These interfaces can also be extended to block interfaces including iSCSI, Fibre Channel and even InfiniBand. It is important to note that these interfaces do not include important object storage features like an extensible set of metadata. This means that there is an administrative burden associated with defining and providing any metadata beyond that managed by a given file system. Since block systems, by definition, do not have application-specific metadata, additional application work needs to be performed to find and develop the metadata within objects. Nonetheless, many object stores are ‘front-ended’ by devices that communicate in file and block protocols and turn around and communicate with an object store in its REST protocol. These devices and interfaces are most prevalent for data protection and disaster recovery applications, which create the metadata necessary to recover data from the object store. Other use cases where a front-end connected to a public cloud for primary storage usage are growing, however, it should be noted that unless additional work is performed, the object store features for big data analytics are not automatic.

Alternative to RAID and traditional Replication

Object stores implement data protection at a higher level than RAID and replication. While each object store may have strategies that include RAID, objects are usually replicated individually and depending on system parameters, may exist on multiple physical storage systems and even in multiple geographic locations. In any event, an object store has data protection and healing at the object level, which is important for truly elastic storage, immutability in archives and the ability to rapidly access objects from anywhere in the world and anywhere in the object store.

Load Balancing

The REST interface has an inherent capability for transparent load balancing. Since details such as block or file tree namespace are not specified, the object store has the ability to keep its own set of load balancing and bandwidth control parameters based on service level designation rather than physical device. This makes provisioning of storage automatic as the application creates new objects within boundaries established by permissions.

Data Protection

Object stores need to maintain an internal data protection scheme that, like load balancing, is transparent to the users and applications accessing them. The richest command protocols include policy development that designates levels of protection that can span the number of copies of a given object group to maintain and where. At lower levels, RAID schemes and other storage-system-wide protection strategies can be deployed in a manner that can be automated and include policies for storage tiers that range from solid state devices, through high-speed and high-capacity drives, to tape. Metadata separation also adds another dimension for data corruption detection.

Tagging, Search and Indexing

The existence of metadata, which can be tracked in a separate thread than the actual data, provides high speed accessibility for tagging, search and indexing without needing to track and scan actual data. This provides very rapid access to parameters for exotic applications, such geo-location and facial recognition, as well as rapid access for more typical applications like email and database archives.
Redundancy and Availability

Objects have a particular set of characteristics that enhance high availability with a minimal amount of redundancy. Objects can carry their own error detection and correction information, and the comparison of areas of content can detect corruption easier than the typical RAID methods which use a parity check (extra bits created mathematically) on various data components (blocks and characters). Consider a typical RAID implementation that ‘stripes’ data across multiple physical devices and makes multiple copies of the data. If the original content was corrupted, the RAID system is simply protecting corrupted data. Objects can be automatically replicated across systems and locations in order to enhance availability and reliability of data.

Data Distribution

Object scaling is a feature of object stores that can intelligently distribute objects across multiple locations in order to maximize security by only storing partial information at a given location, minimize access latency by placing objects physically closer to the area of most frequent access and by policy based on regulatory parameters. Objects carry the information necessary to do this automatically without the development of complex namespace designations.

Unstructured Data

The growth of unstructured data is a major driving factor for object storage. Metadata that identifies geo-location, date and time of a given photograph provides significant information beyond the unstructured, compressed bits of a JPEG. It also adds a dimension of knowledge for revision control to typical documents such as DOCX, PDF and PPTX. An object grouping can include recognizable parameters which allow a mixture of file types to be easily reconciled and delivered.

Scalability

A primary reason for cloud storage to be implemented as object stores is the ability to transparently scale objects into the existing infrastructure at a lower cost than rigid namespace and block storage constraints. This is because an object store creates an abstraction layer between the physical equipment consisting of servers and storage arrays versus the objects and using applications. Automation can replace administration in these use cases.
Implementations

For the purpose of this report, we will examine implementations in the dimensions of software, commodity hardware, purpose-built systems and Platform as a Service (PaaS).

Software Implementations

Object store software vendor examples include companies, such as Cloudian.

Software implementations can be installed as virtual machines or on dedicated physical machines. The advantage of a software implementation is that using organizations can deploy without a major capital expenditure by using existing equipment or commodity hardware. Trial and limited editions are often available allowing the test of the object store for a given use case before additional expenditure on shared or dedicated hardware occurs. Organizations should choose the vendor based on a match of features to the intended use case.

Commodity hardware

Implementations of software such as Cloudian on commodity-based servers are increasingly common. The implementation of object storage is often eased by the use of readily available or existing servers that can be introduced into the network and provisioned as object storage nodes. Customers that choose this implementation are able to configure each node for performance and capacity and easily add nodes economically as the system grows.

Purpose-built systems

A purpose-built system has the advantage of a single vendor supplying both hardware and software. Generally speaking, purpose-built systems provide increased benefits in support since the vendor has control over the system parameters, which are critical to the application. The primary disadvantage is that these systems represent a fairly large capital expenditure and the user is responsible for the ongoing operating expense. These costs can potentially be offset in terms of total cost of ownership when factors like scale and density are considered, as well as price-performance. Users entering into an arrangement should carefully evaluate their needs against the capacity and scalability of the system being considered. Other parameters include replacement cost and time to replace in case of a disaster, as well as the vendor’s overall ability to provide support and the proverbial ‘one throat to choke.’

Platform as a Service

While arguably a fancy name for a cloud storage service, a number of vendors fall into this category including Amazon S3.

As the old saying goes, if it floats or flies, rent it, don’t buy it. The same could be said for organizations new to object storage or ones that will benefit from the elastic nature of cloud object storage. PaaS providers are aggressively looking to expand their user base and market share and are often willing to provide flexible contracts at the outset. The obvious disadvantage, particularly for IT organizations that are comfortable with managing and controlling the cost of their own infrastructure, is the incremental commitment to a particular PaaS provider as more feature are implemented and heavier reliance on a particular supplier’s command interface results.
Vendor/Product Profile

Vendor Name: Cloudian
Product Name: Cloudian Object Storage Software
Link to website: http://www.cloudian.com/overview.html
Link to data sheet: http://www.cloudian.com/cloud-storage-products/cloudian-cloud-storage-platform.html

Product Description: Cloudian helps organizations address their big data storage challenges by providing software to build private or public clouds using commodity servers. The Cloudian solution requires minimum IT involvement, dramatically reduces the costs of cloud deployment and management—and, as a result, is helping make “Storage-As-A-Utility” a reality.

Cloudian serves two primary markets: enterprises and service providers. For enterprises, Cloudian offers software that helps them build private or hybrid storage clouds at a significantly lower total cost of ownership compared to traditional storage solutions. For service providers, Cloudian offer software that helps them offer new services (such as Storage-as-a-Service and Backup-as-a-Service).

Cloudian software delivers a full S3 API compliant, multi-tenant, multi-data center, multi-region cloud storage solution that enables customers to cost-effectively deploy an extremely scalable and reliable object storage service within public and private clouds. The platform leverages a fully distributed, peer-to-peer architecture, with no single point of failure. The solution easily scales from one node up to thousands of nodes across multiple data centers and regions, supporting petabytes of data. Cloudian also provides a comprehensive user interface for both end user applications as well as administrative functions, including billing, monitoring and provisioning.

The Cloudian architecture consists of x86-based servers/storage nodes that are able to scale horizontally. The architecture supports automatic replication in which distributed nodes and the objects they store are distributed across available nodes. Cloudian enables the concept of dynamic consistency levels, which automate failover to other nodes transparently and non-disruptively. These consistency levels can be ranked in order of priority of failover -- the software automatically attempts to meet the highest consistency level possible. If it can't fail over to this level, it will fail back to the next priority specified in the policy.

The product also provides policy-based encryption at rest and retention policies for lifecycle management. When an object reaches the expiration date set for it, the Cloudian software automatically deletes the object and an audit log of deletions is updated.

Cloudian enables various use cases by partnering and integrating with Citrix Cloud Platform, OpenStack, Apache CloudStack and with a variety of cloud gateway vendors such as Cloudberry, Computer Associates, CommVault, Ctera, IBM and TwinStrata. The product is compatible with a variety of applications and tools due to its unmatched S3 compatibility.
Use case profile

@nifty

Type of Organization: Japan’s largest Internet Service Provider. Nifty is a subsidiary of Fujitsu. The company has 2,000 corporate customers.

Location: Shinigawa, Tokyo, Japan

Environment: Cloudian Multi-Tenant S3 Cloud Storage

Challenges:
- 2,000 enterprise users of the Nifty Cloud
- Unlimited storage capacity
- VMware-compatible
- Compatibility with Amazon S3 API
- Use of commodity servers to enable low price per GB
- Scale-out peer-to-peer architecture with no single point of failure
- NOSQL database to support large-scale distributed processing
- No downtime

Solution:
- Reviewed over 40 vendors
- Implemented Cloudian Cloud Storage software solution
- Use of S3-compatible cloud gateways, such as Microsoft's StorSimple and TwinStrata
- Use of commoditized storage nodes
- In full scale production in Q3 2011, continues to deploy additional capacity

Benefit of Cloudian solution:
- Requires no change to existing application
- Supports distribution of storage clusters across data centers and dispersed geographic locations
- S3 simplifies hybrid cloud deployments, integrations and migrations
- Unlimited scalability
- High-availability