DDN STORAGE

architectingIT

Nine Critical Features for Object Stores

All you need to know to review object storage technology and help focus your decisions on choosing the right technology. An independently authored guide, written by Chris M Evans of Architecting IT and sponsored as a PDF by Data Direct Networks.

INTRODUCTION

Object storage is a relatively new market segment that has continued to grow steadily and is starting to find more reasons for adoption. For the uninitiated, object stores are used to hold large volumes of unstructured data, where each "object" is essentially a file with no specific format (also called a binary file). Object stores can hold any type of data, from small objects that can be human-readable files to media (audio and video) or other industry-specific formats (oil & gas, medical imaging and so on).

The benefit of using an object store over traditional storage is multi-fold. Block-based systems (e.g. Fibre Channel and iSCSI) don't scale out well and have no real understanding of the data being stored. They are "dumb" block devices that serve content with low latency and high granularity. File systems place some structure onto data, putting file objects into hierarchies (folders/directories) and attaching metadata to those objects. However, the metadata is typically only based on the information needed to store the file (time created, time updated, access rules).

Object stores go one step further and remove the folder hierarchy. Objects are stored with extensible metadata that is typically highly searchable. In terms of scale, object stores can grow to multiple (if not hundreds) of petabytes in size, usually with no restriction on data geography. The use of object stores is gaining adoption in the enterprise as the platform offers benefits over traditional forms of storage. Block-based storage arrays don't scale well and have issues with data protection (e.g. RAID) with large numbers of HDDs and SSDs. File-based systems are restricted by the scalability of the file system itself, either in terms of object count, concurrent or parallel access and recovery time to check the consistency of the file system structure. Object stores represent a simpler, more scalable solution and one that is easily accessed over standard web-based protocols.

Possibly the biggest challenge for IT organisations looking to adopt object stores is choosing how to use the platform and how to evaluate vendor product offerings. Object stores use web-based protocols and so require a degree of coding to use. This is changing, as we will discuss later.

From a features perspective, there are many aspects to object stores that make one platform more appropriate than another. In the remainder of this post, we identify, categorise and describe what IT organisations should be looking for in an object storage platform.

FEATURE 1 – SCALABILITY – BIG AND SMALL

As already discussed, object stores are designed to scale much further than traditional data stores like scale-out NAS. Vendor offerings run into the multi-petabyte capabilities, with the option to store billions of objects. However, achieving high scalability is much more than simply measuring object counts and data volume. Considerations include:

- **Object size.** How well can the object store deal with small and large objects? How are small objects handled?
- **Capacity limits.** Are there any real limits on capacity? Does capacity growth require adding more hardware or software nodes? Can I simply expand storage?
- **Tiering & Caching.** How does the object store manage tiering of data? As capacities grow, naturally a large volume of data will be inactive and present an opportunity to archive to cheaper media. At this point, tiering becomes a critical capability. Flash media can also be used to accelerate performance when used as either a caching or tiering layer.
- **Metadata Management.** As the object store grows, how well is metadata managed? Does the size of the object store affect the performance of search?
- **Object Access.** As the object store grows, does the access time of any individual object increase (hopefully not at all)?

The last point is particularly important for building out object stores that will deliver access to many object store/retrieve requests in parallel, such as systems serving as the backend of a CDN network. Increasing the number of objects in an object store shouldn't significantly increase the retrieval time, or more importantly the "time to first byte", which is the time taken to start to stream an object back to the requestor from the point of receiving the request.

Of course, we shouldn't forget that object stores may need to start small and not be required to have an initial footprint in the hundreds of terabytes or petabyte range. The capability to have a small entry-level capability helps reduce the barriers to entry for object storage adoption, with the added requirement to be able to scale linearly from small to large with minimal operational impact.

FEATURE 2 - DATA PROTECTION

The idea of data protection covers many aspects in an object store. Compared to traditional "primary" storage, object stores are likely to be used for the long-term retention of data, so data durability becomes an important factor. We can think of durability as the need to ensure that no logical corruption occurs on the data being stored, due to a range of errors including hardware read failures and data corruption.

Modern hard drives are hugely reliable compared to the devices in use quarter of a century ago. Despite this, drives do suffer from read errors and other transient problems. Object stores should execute a range of on-disk management functions, including data scrubbing, CRC checking and rebuilds of corrupted or inconsistent data. These background tasks represent processes to keep data healthy where long term retention is critical.

The second area to consider is that of hardware failure protection. Most modern storage arrays implement RAID (Redundant Array of Inexpensive Disks) as a method to recover data from loss due to hardware failure. RAID has scalability issues as data volumes start to rise. Storage vendors have implemented dual and even triple parity to protect against multiple drive failures with large HDD capacities, however, elongated drive rebuild times make RAID impractical for the bulk of data in an object store.

The alternative is to protect data using erasure coding schemes. Erasure coding describes a process of dividing and transforming data into a number of redundant pieces, a minimum count of which is needed to recover the original information. An encoding scheme might, for example, translate data into 12 pieces, with any 8 required to rebuild the original data. The 12 pieces can be distributed across multiple drives, servers/nodes or even geographically to provide high resiliency. In a 12/8 scheme, distributing the data across three locations means the loss of any one location could be tolerated.

Object stores should provide erasure coding with variable protection values, based on the customer needs. As erasure coding has a significant processing overhead, RAID can also be used to protect smaller objects and improve access performance. Where data is distributed geographically, the impact of rebuild over the network becomes important. Therefore, the specific implementation of the erasure coding system (and the need to retrieve data across the WAN) will directly impact on recovery time and customer SLAs (Service Level Objectives). This issue can also occur when local LAN latency is high – any distributed network-based recovery will always be impacted by network performance. Fast recovery is important as unprotected data needs to be re-protected quickly, to avoid potential data loss.

FEATURE 3 - SEARCHING, INDEXING AND METADATA

The ability to search and retrieve data in an object store is one of the most critical requirements. Compared to structured data like databases and file systems, object stores keep data in a flat hierarchy, with only a small amount of logical or physical separation, such as buckets or pools. This means that every object stored needs to have plenty of information to make data retrieval easy.

Object stores typically store data using one of two methods; either the end user sets the name of the object (which could look like a standard file name) or the object is stored and accessed using a system-generated object ID (OID). Object IDs are typically long strings of characters and numbers, randomly generated by the object store itself.

Where OIDs are used, metadata is critical. The object store user may also maintain a separate database of object IDs and their uses. Metadata provides information on the object itself (system metadata) such as object size, access permissions, the user creating the object and so on. User metadata extends the information stored with each object and is application specific information used to deliver search and indexing capability.

The performance of metadata searches should be independent to the amount of data stored in the object store itself. This is a critical requirement in managing scalability.

FEATURE 4 - PERFORMANCE

In our discussion of requirements so far, performance is a theme in the implementation of scalability, data protection and search. When object stores were first developed, the idea of performance wasn't a key consideration, as many object stores were simply used as long-term archives or backup repositories. Increasingly, object platforms are being used for much more active data, either as active archives or the repository for media and other streamed content.

The result is the need for object storage platforms to provide high throughput, scale performance linearly and to handle a high level of concurrent requests. The need for concurrency is especially important where object platforms are used as the backing store for CDNs (content delivery networks) or other Software as a Service (Saas) solutions. Concurrency means the ability to both stream many objects at the same time and to be able to handle a high number of individual requests per second. In terms of metrics, typical measurements are based on IOPS (I/Os per Second) and throughput (MB/s or GB/s).

FEATURE 5 - SECURITY

As with any data store, security is a key feature. In object stores, security features cover a number of aspects.

With the volume of data likely to be retained in an object store, *multi-tenancy* becomes very important. Business users (either separate departments in an organisation or separate organisations) want to know that their data is isolated from access by others. This means having separate security credentials and offering encryption keys per customer or object within a customer.

Object stores typically provide access to data through authentication keys that are supplied on an HTTP call to the object store itself. These keys are credentials rather than typical user/password combinations as the data could be passing over the public internet. The wider task of managing credentials is part of identity management features that can also provide integration into standard platforms such as LDAP and Microsoft Active Directory.

Access to individual objects or buckets will be assigned through access control lists, that determine either individual or group-level access to data. Many object stores will allow the access controls to be set and managed through the same web-based REST interface used to store and retrieve data.

In addition to managing identity, security has to be provided through data encryption, both in flight and at rest. Typically, in-flight protection is achieved at the protocol level, using TLS (e.g. HTTPS). At rest, data should be encrypted to protect from direct access, either at the physical server level or drive/device level. The specific point or implementation of encryption can depend on how the end users want to manage encryption keys. Data could be encrypted before or while being added to the object store.

FEATURE 6 - COMPLIANCE & AUDITING

Compliance is another aspect to data security that focuses on meeting regulatory requirements on the retention of data in specific controlled industries such as healthcare and finance. Typically, compliant systems need to be able to provide immutability of data, offer object versioning (so changes can be tracked), implement object locking or WORM (write once read many), again for immutable data. Most object stores don't update data in place, compared to block and file-based systems. This provides a degree of control that works well with compliance requirements.

Auditing complements compliance, providing a trail that shows how data was stored in an object store system. The audit trail can also provide additional information such as the migration of data between tiers, checksum validation on content (to ensure no tampering) and all accesses to individual data objects or buckets.

FEATURE 7 - DEPLOYMENT MODELS

Object storage has been at the forefront of the move towards software-defined storage or SDS. The nature of large scale-out deployments has meant object stores work well with the cost model of commodity hardware and vendor-supplied software. As a result, we see many object storage implementations based on software only.

The use of commodity hardware, of course, doesn't suit all requirements. Many potential customers may be unwilling or unable to manage the process of sourcing and building a bespoke object storage solution, preferring instead to take a combined hardware and software solution from the vendor. In this case, vendors need to offer appliances to suit the needs of the customer, potentially in partnership with server and storage vendors already in the customer's data centre. Why? Because support models, in-house skills and deployment blueprints will already be based on the preferred hardware vendor of choice. For ultimate flexibility, vendors are likely to offer three choices:

- **Software only** – either as a VSA (virtual storage appliance) or deployed natively onto hardware.
- **Appliance** – a dedicated hardware appliance, built as a white-box or in conjunction with one of the main hardware providers.
- **Cloud** – deployed as an instance in the public cloud.

For each option, customers should expect full interoperability and consistent management interfaces.

FEATURE 8 - PROTOCOL SUPPORT & STANDARDS

Initial object stores were based on the HTTP(S) protocol, using REST-based API calls to store and retrieve data. The use of HTTP is flexible in that data can be accessed from anywhere on the network (either local or wide-area), however, applications have to be coded to use object stores, compared to accessing data stored in scale-out file systems. As a result, vendors have moved to add NFS and SMB support to their products, allowing data to be stored and retrieved through standard file-based protocols. To fully support scale-out capabilities, support should include parallel file systems.

Extending protocol support means existing applications can be easily ported or amended to use object stores for their data. Also worth considering is the difference in architecture that is provided by using an object store that emulates a file store, compared to a scale-out file store. The underlying data isn't stored using a structure based on inodes and directories, so the concept of an FSCK (file system scan) after a system crash doesn't apply. This has big implications for scalability and performance of object stores supporting file systems compared to traditional file systems. (continued….)

Protocol support also needs to extend to adopting de-facto or industry standards. For object stores, this means working with S3 and Swift, two of the "standards" that have gained widespread popularity. Amazon's early entry into the object market with the S3 (Simple Storage Service) platform released in 2006 has made the S3 API a standard that many vendors have chosen to follow because it is well established, mature and comprehensive. Swift has developed from the object storage component of the OpenStack project.

FEATURE 9 - TOTAL COST OF OWNERSHIP

No summary of object storage would be complete without a discussion on pricing and TCO. The most obvious licencing model is one based on capacity – add more usable or raw capacity to the platform and pay more for the licence in practical increments. Vendors also have the option to charge per node, which means end users need to make sure the hardware they deploy provides the most capacity possible.

There is also the option to charge by feature, although some vendors will see the opportunity to create a comprehensive charging structure, inclusive of all feature options. This is certainly more competitive from an end user perspective, where hidden additional costs can be a problem.

Calculating TCO raises one interesting question on the efficiency of object storage platforms. Scale-out node designs employ compute, system memory and disk or flash storage to deliver a certain amount of user capacity. When building on white-box hardware, the efficiency of software has a direct correlation on the cost of building a solution. As yet, there are no practical benchmarks to compare the efficiency of object stores and this remains one area that needs some development by the industry.

CONCLUSIONS

We have highlighted nine critical features of object stores. Vendors will implement these features in ways that complement their product architectures. When deciding on what platform is right to employ in your business, some of these critical features will be rate more highly than others. The list is presented here in no particular order, however, as a prospective customer, the aim should be to work through this list and determine those critical features that deserve more investigation.

THE AUTHOR

Chris M Evans has worked in the technology industry since 1987, starting as a systems programmer on the IBM mainframe platform, while retaining an interest in storage. After working abroad, he co-founded an Internet-based music distribution company during the .com era, returning to consultancy in the new millennium. In 2009 he co-founded Langton Blue Ltd (www.langtonblue.com), a boutique consultancy firm focused on delivering business benefit through efficient technology deployments. Chris now writes a popular blog at http://blog.architecting.it, attends many conferences and invitation-only events and can be found providing regular industry contributions through Twitter (@chrismevans) and other social media outlets.

Email: mailroom@architecting.it
Twitter: @architectingit