Quantum.

The Quantum ActiveScale Object Storage Platform

Architecture, Features, and Differentiators

CONTENTS

Introduction

More than two-hundred and thirty years after Benjamin Franklin wrote "In this world, nothing is certain except death and taxes," it's abundantly clear that "data growth" belongs on any modern list of certainties. Data is fuel that drives insight and innovation, but it can become unmanageable and unusable if not handled effectively.

Understanding the lifecycle of data is the key to making it benefit, not burden, the organization. The typical data lifecycle involves three stages – creation or acquisition, active use, and preservation. Each of these stages has unique requirements for storage.

During the first stage, data comes into being, thanks to a camera, a sensor, an IoT device, a transaction, or a calculation. Storage performance is key so that streaming information is not lost.

The second stage involves active use of the data by humans and/or machines, such as researchers or AI algorithms. Again, storage performance is key, and data must frequently be moved and copied to support different steps in its processing. These steps often span local data centers and the cloud.

After active use, the focus moves to preservation. Historically, retained data required little additional processing, but increasingly, and across many industries, data must be kept accessible, with the expectation that it will be needed again in the future, to be re-processed, re-analyzed or monetized in new ways. It is this stage that many organizations struggle with the most. How to keep massive, ever-growing quantities of valuable data protected and available over the long term, without breaking the bank?

Classic NAS and object storage architectures break down at scale. RAID and replication don't provide sufficient protection or storage efficiency and managing multiple storage devices or separate tiers of storage takes too much administration time.
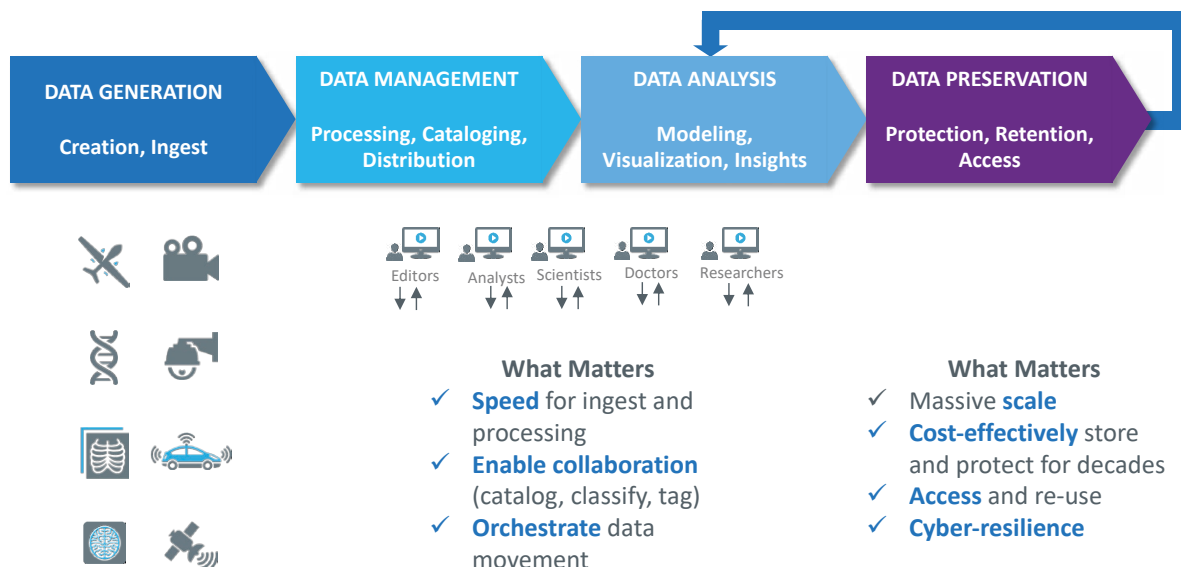

Figure 1 - The Lifecycle of Data

These are the challenges that Quantum ActiveScale was born to solve. With over 150 US and international patents, ActiveScale is the industry's first and only object storage platform architected for both active and cold data. It's simple to manage and always available, with unlimited scalability. Seamless, fully integrated support for both active and cold storage classes yield the lowest total cost of ownership, and data is always safe thanks to advanced data durability and security technology.

This paper provides a detailed review of the ActiveScale architecture, specifically calling out features and abilities that drive its unique combination of simplicity, scalability, performance, availability, data durability, security, and low TCO.

The ActiveScale Software Architecture

OVERVIEW

ActiveScale is typically deployed as an appliance on Quantum hardware but is also available as software for installation on qualified 3rd party hardware and as a fully-managed service with a subscription model, Quantum Object Storage Services.

Figure 2 is a high-level view of the components of the ActiveScale architecture, which are discussed in detail in the sections that follow. It is useful to think of the capabilities and technologies within ActiveScale as being organized into several categories, or layers.

At the top is the Data Access layer, consisting of the customer-facing interfaces for data storage and retrieval. Next is a Data Services layer, comprising a host of optional functions and features that may be applied to data stored in the system.

Underpinning these layers is the most important one, the ActiveScale Architecture layer. The lower-level functions and technology that appear here are fundamental to the capabilities and value of ActiveScale as a durable, secure, cost-effective repository for data. Most of the patents and unique intellectual property that differentiate ActiveScale from all other object stores are found here.

Management and monitoring tools and interfaces span across all layers, tying everything together into a simple to manage system regardless of scale, and security is woven throughout the software stack.
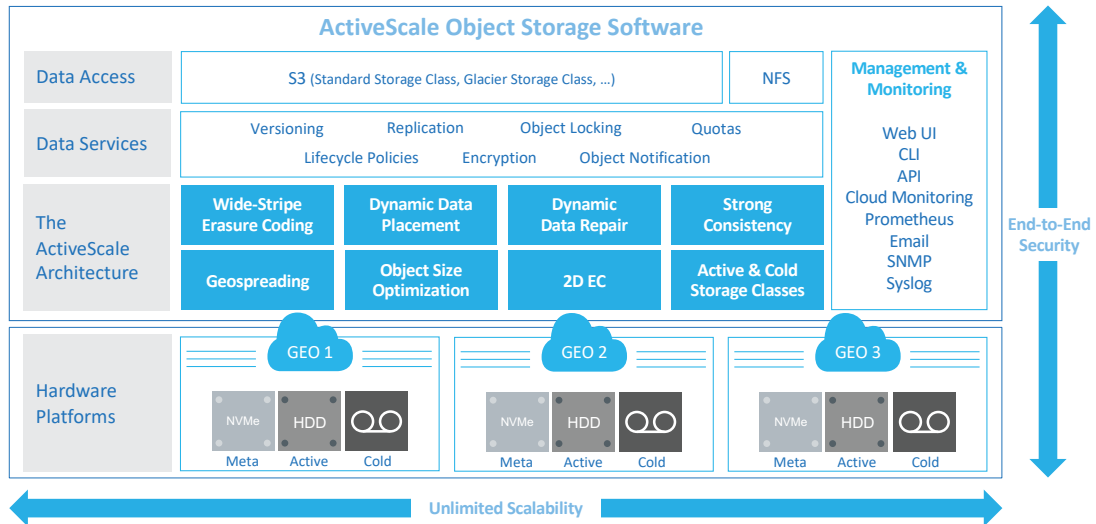
Figure 2 – ActiveScale High-Level Architecture

TWO-LAYER ARCHITECTURE WITH COLD STORAGE CLASS

It's probably no surprise that ActiveScale, like most other object storage systems, stores user data on hard disk drives (HDDs). HDDs have reasonable cost per TB and perform very well when aggregated together into arrays and accessed with many parallel streams. They are good for storing active data, sometimes called hot or warm data. This is data that's in use for current projects or expected to be immediately available for reference.

In addition to user data objects, all object stores, including ActiveScale, generate and store metadata. Metadata includes attributes and policies attached to objects or buckets, internal data regarding erasure code spreads, custom object metadata added by users or applications, and many other categories of information that are critical to the operation of the system. Nothing happens in an object store without metadata being manipulated. Metadata operation performance is far more critical to overall system performance than the performance of the bulk storage – any object store is only as fast as its metadata. To ensure the lowest latency and highest performance, ActiveScale stores metadata on blazing fast NVMe Flash storage.
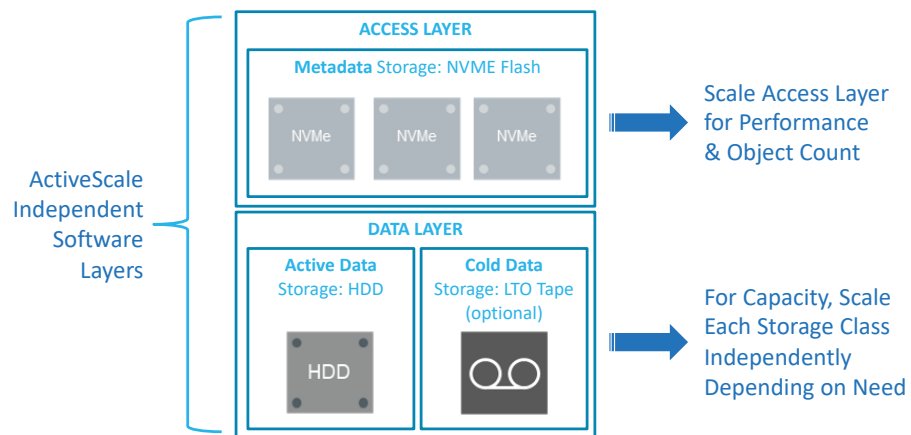

Figure 3 - ActiveScale Two-Layer Architecture with Cold Storage Option

Not all data is active. Some needs to be kept for possible reference or re-use in the future but doesn't have to be immediately accessible. The more data an organization has, and the longer the required retention time, the greater likelihood that most of the data will be cold. Research by analyst firm IDC estimates that 60% of all data is cold.
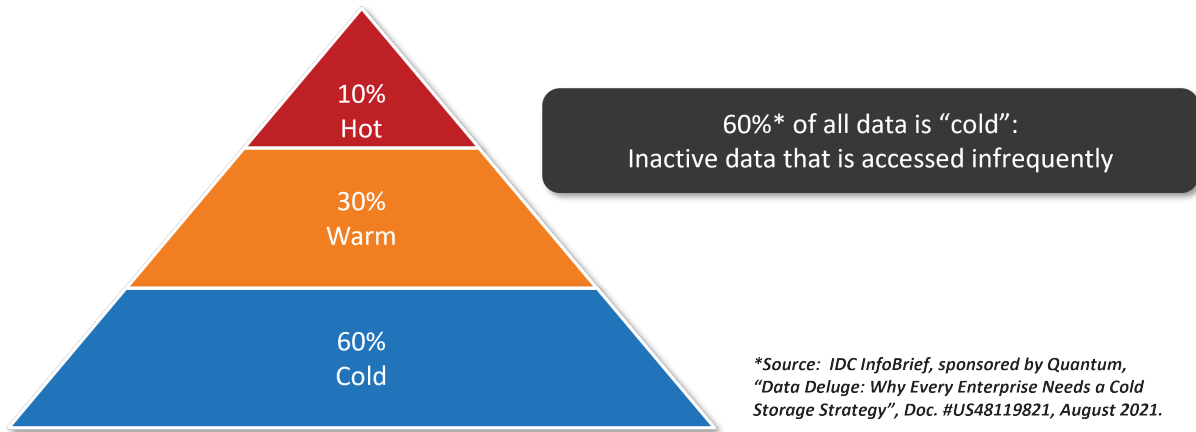

Figure 4 - Cold Data

Most object stores treat all data the same, forcing organizations to pay the same cost per TB to store hot data and cold data. This can be very expensive when storing petabytes to exabytes of data, and it's not necessary. ActiveScale is the first and only object store to offer an integrated tape-based cold storage class, accessed using standard S3 (Simple Storage Service) Glacier class API calls. ActiveScale Cold Storage is discussed in detail later in this document.

OBJECT STORAGE ENCODING AND DATA DURABILITY

ActiveScale uses Reed-Solomon erasure coding to encode all data stored in the system – including cold storage – taking advantage of the hardware acceleration capabilities built into modern microprocessors. The way the erasure coding is implemented results in extremely high data durability, up to 19 nines, or 99.99999999999999999%. Data durability refers to the ability of a system to withstand failures without losing data, and it can be measured in terms of the probability of losing objects over time. A durability rating of 19 nines corresponds to the probability of losing one object out of every ten billion objects, every one billion years. Data stored in ActiveScale is extremely safe.

At a high level, incoming objects are encoded into a number of data shards, and a number of parity shards. All shards contain periodic embedded CRCs (cyclical redundancy checks) that enable ActiveScale to verify the integrity of an individual shard at any time. If a shard is damaged or lost, the full object can still be recovered, and the damaged shards reconstructed using the remaining data and parity shards. When a GET operation is submitted for an object, ActiveScale reads the corresponding shards, validates and strips out the CRCs, and reassembles the shards into the object, which is then returned to the requestor.

Storage Policies

Storage policies control the configuration of several erasure coding parameters, most importantly spread width and safety. The spread width represents the number of disk drives the encoded data shards and parity shards are spread across for a given stored object. Safety determines how many simultaneous drive losses can be tolerated without data loss. For example, data stored using a storage policy of 18/5 is encoded into 18 shards. Any five shards of those 18 can be lost and the data will remain intact and readable. ActiveScale uses wide spreads – up to 20/4 – which provide greater protection from failures and higher storage efficiency than more common narrow spreads (like 4/2) used by others.

Locality

ActiveScale is a location-aware system. Based on configuration information entered during installation and upgrades, and automatic detection of resources by the software, ActiveScale knows the location of each storage resource in the system. Storage is arranged in a simple 4-level hierarchy: Disk → Node → Rack → Datacenter. Using this location information, data is spread across storage resources to maximize protection from storage failures at any level of the hierarchy, from single disk to full site failures.

Disk Drive   **Storage Node**   **System Rack**   **Data Center Location**
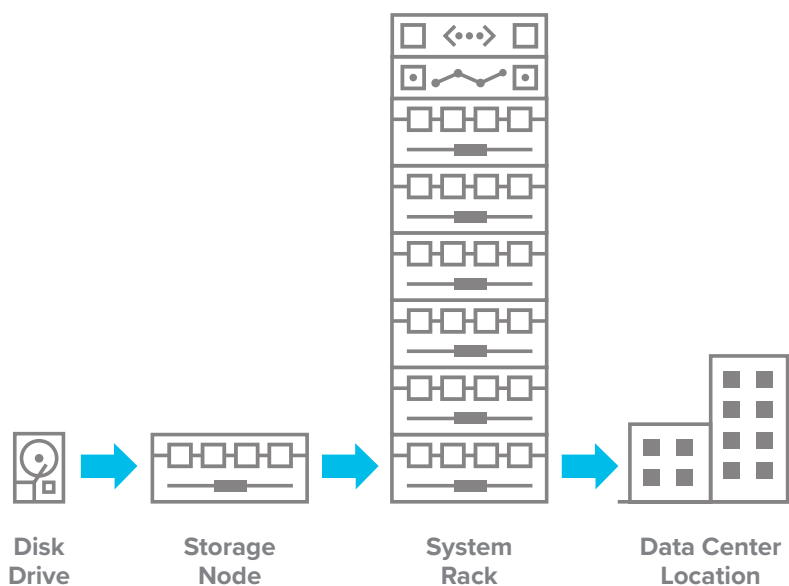
Figure 5 - Storage Hierarchy

System Resilience Example

To better picture how Storage Policies and Locality interact, consider the following example. An object is stored with an 18/5 policy (spread width 18 and safety 5) into an ActiveScale system occupying a single data center rack. The rack contains 6 nodes, each containing dozens of individual HDDs.

ActiveScale will encode the object into 18 shards and place each one on an independent HDD. The 18 HDDs are chosen in a balanced way across the Disk/Node/Rack/Data Center hierarchy. In

this case there is only a single rack in a single data center, so all disks are selected from this rack. There are six nodes in the rack, so 3 drives are chosen per node to store the encoded shards. The leftmost part of Figure 6 illustrates this distribution, with blue boxes representing drives containing the encoded data shards.

Now consider what happens when failures occur. Because the object was stored with an 18/5 policy, any 13 parts of the 18 may be used to reconstruct the object (18-5 = 13). If a single drive fails in one of the nodes, the safety level is temporarily reduced by one to four, until the affected objects are repaired onto other drives in the system.
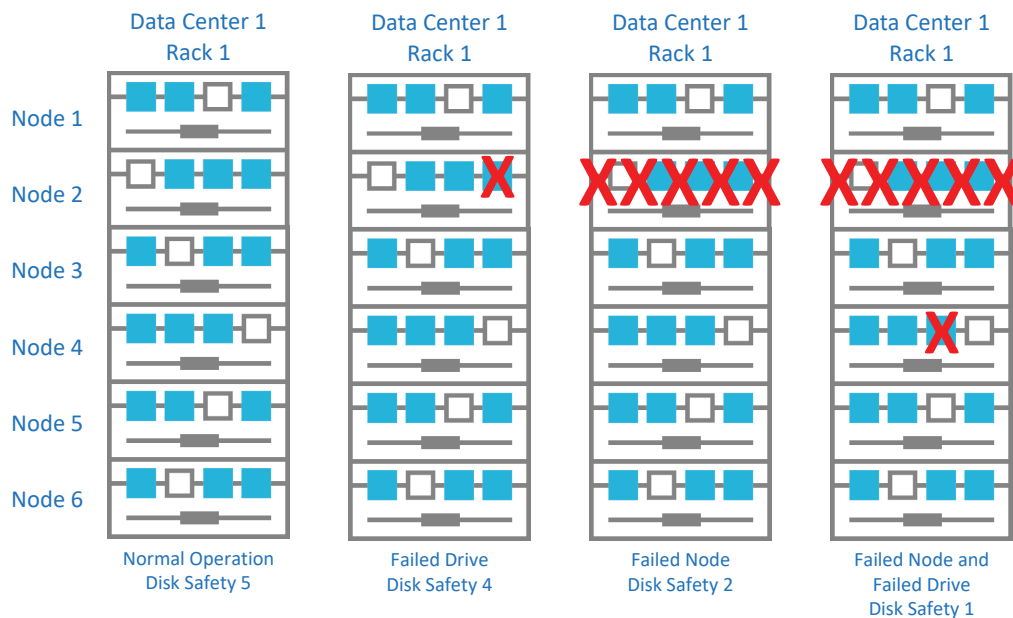

Figure 6 - Data Resilience Example

If an entire node fails, the safety level for our example object is temporarily reduced by three (from five to two), since three drives containing parts of the object are unavailable. Repair of the object begins immediately but is not instantaneous. This is OK – the remaining safety factor of two (equivalent to the protection provided by RAID 6) means that the object is safe even if two additional drives containing parts of the object fail.

This is a simple single-site example, but the same principles apply for larger single and 3GEO systems. Erasure coded shards are always spread in a way that maximizes resilience to disk, node, rack, and full site failures. This ability to absorb numerous, simultaneous failures without data loss is what provides ActiveScale's extremely high level of data durability.

Object Size Optimization for Performance and Scalability

Objects can be very small, very large, or any size in between. Very small and very large objects present problems for all object storage systems. They make it difficult to maintain high performance and efficiently allocate system resources like RAM and CPU cores. ActiveScale implements several

innovative techniques to optimize operations based on the size of the incoming objects, including segmentation of large objects and aggregation of small objects. This enables high performance regardless of object size while not unreasonably burdening the system.

Large Object Segmentation

While S3 multipart upload can help improve the performance of large object uploads, sometimes it's not enough. A multipart object can have at most 10,000 parts. For very large objects, each individual part can still be too large for an object storage system to handle efficiently. In ActiveScale, incoming large objects are divided into superblocks, which are up to 64MiB in size. For example, a 30MB object will occupy a single superblock, and a 70MB object will occupy two superblocks. Each superblock is further divided into data shards that the EC codec uses to calculate parity shards. Data shards and parity shards, with their associated CRCs, are written to disk based on the storage policy.

This patented (US8738582B2) system of dividing large objects into a series of superblocks makes them easier for the system to handle and enables parallel processing, which speeds up both `PUT` and `GET` operations.

Small Object Aggregation

ActiveScale defines small objects as objects less than 7MiB in size. Beginning with ActiveScale software v5.7, a new, more durable method of dealing with small objects has been introduced. This new technique is covered by patents US10120576B2 and US10719497B2 and is known as small object aggregation.

Small objects present a special problem for all object storage systems, which must deal with multiple segments on multiple disks to store and retrieve each erasure encoded object. For large objects this is an advantage because it enables parallel processing. But for small objects it's inefficient, requiring many more IOPS compared to a simple non-erasure-encoded write.

When a small object is received by ActiveScale, instead of being handled individually, it is first concatenated with other small objects into a larger container, up to 32MiB in size. This entire container object is then encoded into data shards and parity shards at once, just like the large object segments discussed above. Because the same encoding technique is used, small objects are stored with the same durability as large objects, which is not always true for other small object approaches.
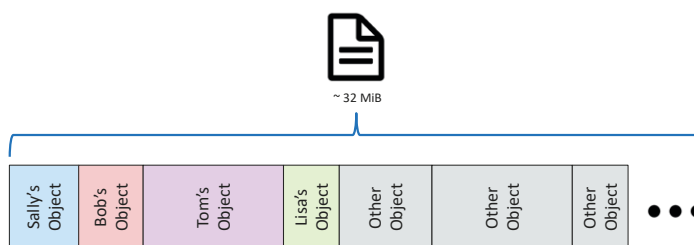

Figure 7 - Small Object Aggregation

Because Reed-Solomon is a *systematic* erasure code, the original input data is contained, unmodified, in the data shards. With small object containers, this may mean that multiple small objects are contained in a single data shard. ActiveScale keeps track of which data shard contains each small object, and where in the shard that object is located. When a `GET` operation is requested for a small object, only the bytes necessary are read from within the data shard. This is how ActiveScale increases efficiency when reading and writing small objects compared to other approaches.

Metadata Protection

The importance of metadata to an object storage system was mentioned above and can't be overstated. Metadata must be protected and always available. ActiveScale stores multiple copies of the object and system metadata in a highly available distributed key-value store (patent US9965539B2). The overall key space is sharded across all nodes in the system. Nodes are organized in groups of three, each group handling a portion of the metadata. As additional nodes are added to the system, the metadata is redistributed in the background, transparently. This scheme enables full read/write operations to continue even in the event one node fails. In larger systems, it's possible to shard the metadata across groups of five nodes, a configuration that can survive the simultaneous loss of two nodes and still maintain full read/write functionality.

In addition to these copies on NVMe flash, metadata is also stored with the encoded data on disk according to the storage policy. This provides equal durability for the metadata as the data and provides a copy that can be used to recover the metadata in the unlikely event of a disaster that wipes out all the copies that reside on NVMe flash.

STRONG CONSISTENCY

In the beginning, Amazon's S3 API specification indicated data was stored using an "eventual consistency" model. This meant that if you executed a `PUT` to replace an existing object, then immediately executed a `GET` against the object, the `GET` might return the old value or the new value of the object, depending on timing. This model favored low latency over accuracy, prioritizing reading and writing data quickly over always returning the latest version of data. Bucket metadata (e.g. `LIST` output) and object data were treated the same way. Third-party object storage systems using the S3 API generally mirrored this approach.

The alternative to eventual consistency is strong consistency. With strong consistency, `GET` requests always return the latest version of an object or metadata. There is no need to worry about a request returning stale information. Strong consistency is always preferred to eventual consistency, provided high-performance can be maintained. This becomes clear when organizations attempt to migrate applications from on-premises servers to the cloud. Applications designed to write to on-prem block or file storage assume strong consistency. Simply "lifting and shifting" such an application onto eventually consistent storage will not work. The application must be modified to take the differing consistency model into account, or third-party shim tools must be used to hide the change from the application, increasing complexity.

In December 2020, Amazon changed the consistency model for S3 to strong consistency. The change was welcomed because it made it easier to migrate applications to AWS. But with that change, object storage systems using eventual consistency could no longer claim to be fully S3 compatible. Some vendors have since updated their consistency models, but not all.

ActiveScale has always featured strong consistency for both data and metadata. This change in AWS behavior required no adjustments to ActiveScale, and validated Quantum's approach.

GEOSPREADING

Most object storage systems include a method to protect stored data across multiple sites. Frequently this consists of two or three-site replication, where objects written to one site are replicated wholesale to separate object storage systems housed at one or more additional sites. This approach is simple to understand but has major deficiencies. It forces customers to purchase multiple separate systems that must be managed individually, and it introduces a huge amount of overhead – 3x the amount of storage is needed for three sites vs. a single site.

ActiveScale can use replication (discussed later in this document), but also includes a superior technology known as Geospreading, or 3GEO. Rather than connecting multiple independent systems with replication, the ActiveScale 3GEO approach distributes the components of a single system across three sites.
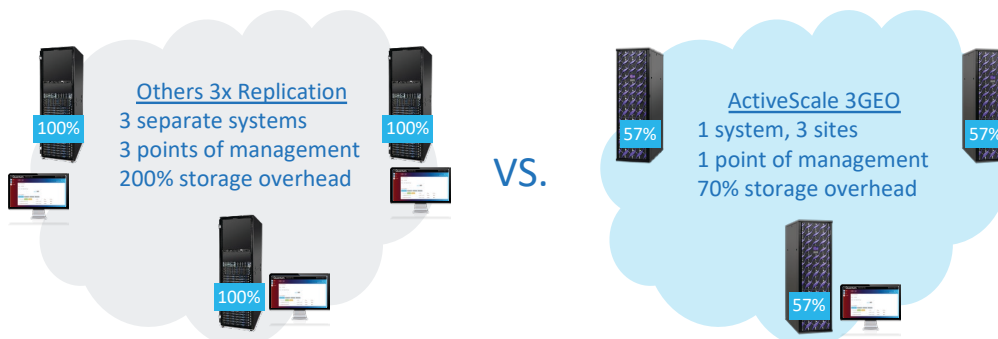

Figure 8 - ActiveScale 3GEO vs. 3-Site Replication

Spreading a single system across three sites has major benefits. By spreading a single set of erasure encoded data and parity shards across sites, the overhead of writing two additional full copies of the data (as with replication) is avoided. High data durability and availability are assured, and only a single system needs to be purchased and maintained. Because of ActiveScale's strong consistency, objects written at any site are immediately available at all sites.

Metadata in a 3GEO system is protected by placing a copy at each site. This ensures that a 3GEO system will not only withstand the loss of a single site without losing any data, but that the system will remain functional and usable from the remaining two sites when the third is down.

Three-site replication has one advantage over the 3GEO approach in that it can protect against the complete failure of two of the three sites, but this is a very rare requirement.

DYNAMIC DATA PLACEMENT

Dynamic Data Placement, or DDP is another patented technology that sets ActiveScale apart from other object storage systems. Many object storage systems use some form of Chord protocol in a ring-based architecture as their basis. Chord provides a relatively simple way to implement a distributed hash table, but as systems grow it becomes very limiting. ActiveScale DDP is a fundamentally different and much better approach.

The Problem with Ring

In ring-based object stores, some number of nodes collectively manage an address space and corresponding data storage. Incoming objects are spread across the nodes using a hash algorithm. This works well under ideal conditions, but as every storage administrator knows, conditions are never ideal for very long. When more nodes are needed to scale up, or when the system must work around unavailable or failed hardware, ring-based systems must execute a process known as a rebalance. The rebalance process is the Achilles heel of ring architectures, limiting their practical scale.
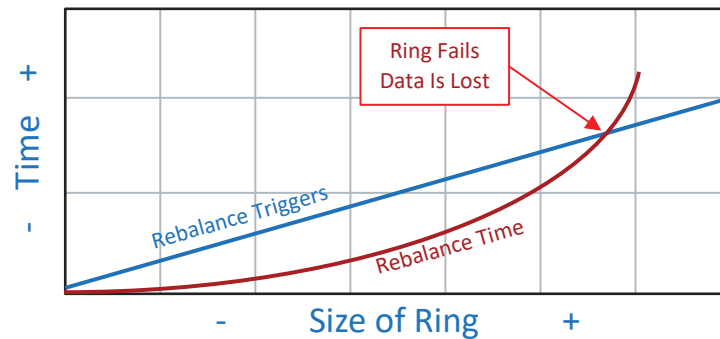

Figure 9 - Ring Architecture Scale Limitation

Rebalancing a ring involves re-assigning the address space across the nodes, to include new nodes or remove failed nodes. As a result of the address range changes, some data is left hosted on nodes that don't have a matching address range. This data must be physically moved onto the nodes where it belongs, a process that can take weeks or even months for a large system. Until this move is complete, exception processing is used to find data that isn't where the algorithm expects it to be. In the case of node failures, some of the data has a lower level of durability until the rebalance process completes.

The larger a ring is, in terms of disks, nodes, storage capacity, and number of objects, the longer it takes to execute a rebalance after every expansion or failure. More data must be moved, which takes longer, therefore more exception processing is needed, which further saps system resources, impacting performance. In the case of failure recovery, there is also the direct decrease in performance from the loss of the failed nodes. This increase in rebalance

time is not linear – doubling the size of a system will more than double the time it takes to complete a rebalance.

Consider also that the more components a system has, the more frequent failures will be. Failures trigger rebalance events, so as a ring-based system grows in scale, the frequency of rebalance triggers increases. When the pace of failures exceeds the rate at which the system can complete rebalancing tasks, the result is data loss. Ring architectures simply don't scale.

The DDP Difference

ActiveScale's patented (US8433849B2, US10078552B2, US10379951B2) data placement approach was designed to support unlimited scalability. Ring-style rebalancing is never required. Instead of being based on fixed hash ranges, DDP places data based on attributes of the underlying storage devices. Because these attributes vary over time, data placement is constantly being balanced and optimized – it's dynamic.

Within ActiveScale, each storage device (HDD) is treated as an independent entity known as a blockstore. Each blockstore has a set of attributes that include its location (Datacenter, Rack, Node, etc.), free capacity, online/offline status, and more. DDP uses this metadata to determine the most ideal location to write data and keeps track of data locations using a key/value store.

More specifically, for each write, DDP uses the storage policy to determine the data center, rack, and node to be used. Then attributes such as online status, free capacity, and blockstore availability are used to determine the best blockstore on which to write the data.
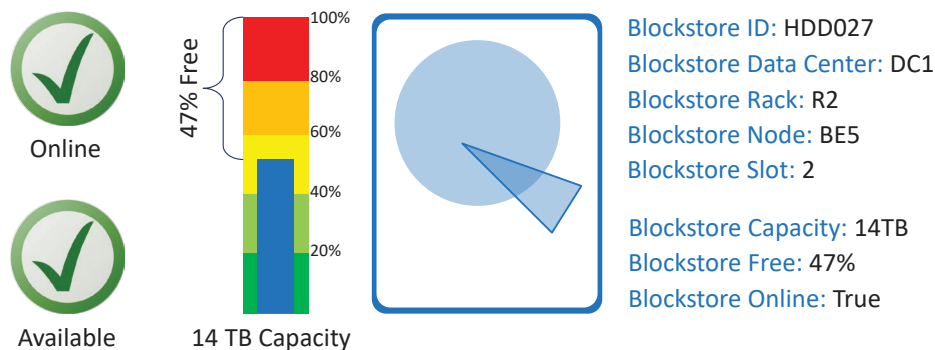
Online

Available   14 TB Capacity

47% Free

100%
80%
60%
40%
20%

Blockstore ID: HDD027
Blockstore Data Center: DC1
Blockstore Rack: R2
Blockstore Node: BE5
Blockstore Slot: 2

Blockstore Capacity: 14TB
Blockstore Free: 47%
Blockstore Online: True

Figure 10 - Blockstore Attributes

Data Placement Example

Recall that incoming objects to ActiveScale are processed through the erasure coding algorithm, resulting in shards of data that must be written to disk. DDP uses the storage policy and its knowledge of system locality to narrow down where to write each shard in terms of which data center, which rack, and which node to optimize resource usage and maximize data availability in case of failure.

Next, DDP must choose a specific disk on which to write the shard. It does this by evaluating all the disks in the node. Various evaluation criteria are examined, and disks are eliminated from consideration that don't meet the criteria. Figure 11 shows a simplified example that illustrates how DDP decides where to write a shard.

The simplest criterion is whether the disk is online or offline. Disks may be offline for a variety of reasons, both temporary and permanent, but writes are not allowed when a disk is offline. In this example, HDD04 is offline, so it's not a candidate for writing the shard.

Free storage capacity is also examined. Here, disks with less than 50% of their capacity free are removed from the selection set. Using a capacity criterion this way ensures that objects are distributed evenly across the system. The 50% threshold is an example – on a live system this may be different and will change over time. HDD01 is eliminated from consideration in our example because it is more than half full.

Blockstore availability is also a factor. A blockstore may be online, but responding too slowly, not able to be read or written, or otherwise unavailable. Unavailable blockstores - such as HDD5 in this example - are not candidates for writing the shard.

The result of this process is that for this hypothetical node, only disks HDD02 and HDD03 remain as candidates for the write after evaluating all criteria. ActiveScale will write the part to one of these two disks essentially at random.

BlockStore ID	Online?	Free %	Available?	Candidate for Write?
HDD01	✓		✓	No: Less Than 50% Capacity Available
HDD02	✓		✓	Yes
HDD03	✓		✓	Yes
HDD04	✗		✓	No: Drive Not Online
HDD05	✓		✗	No: Drive Not Available

Figure 11 - Dynamic Data Placement Example

DDP and System Events

Because DDP treats each disk in the system as an independent blockstore, it can adjust in a resilient way to system events, both positive (system expansions) and negative (disk failures). When capacity is added to a system, the new disks are empty. Based on a probabilistic weighting function these initially empty disks are selected for writes moderately more frequently

than the existing, occupied disks. This continues until system capacity usage is balanced across all blockstores. Because this balancing occurs over time, I/O hotspots are not created on the new storage.

Similarly, when a disk or node fails, affected object shards are regenerated using the surviving data and parity and are simply written to any eligible disk based on the criteria listed above. New object writes are directed to surviving disks and nodes per the storage policy and selection criteria. There is never a need for a massive, disruptive rebalancing event like with ring-based systems. DDP allows ActiveScale to continually adjust to conditions and remain in balance.

DYNAMIC DATA REPAIR, INTEGRITY AND DURABILITY

DDP does a fine job of distributing data and recovering from hard failures, but there is a threat much more insidious than failed disks. Known as silent data corruption, data degradation, or by the more picturesque moniker "bit rot," this threat affects all storage media in some way. In simple terms, bit rot is the accumulation of errors due to physical and environmental factors. Small numbers of errors are easily corrected by onboard ECC using stored parity, but when enough errors accumulate, they cannot be corrected, and a "1" that was written may be read as a "0." The result may be a distorted image, a corrupted database, or even a completely unreadable file. Traditionally, these types of data damage are only discovered at the worst possible time – when access to the data is needed.

ActiveScale is designed to store data safely, and indefinitely. A patented (US8386840B2, US9588862B2, US10379953B2) function known as Dynamic Data Repair (DDR) runs continuously in the background to identify and correct small errors before they become large enough to irrevocably damage stored data. Regardless of the reason for the error – bit rot, a simple write error, or even malicious tampering – DDR will detect and repair the problem.

"Smart" Object Shards

In ActiveScale, objects have metadata associated with them, but so do individual object shards. This makes the object shards "smart." The metadata includes things like the unique identifier and location of the shard within the blockstore, the parent object the shard belongs to, the bucket the object resides in, and a sequence number representing which fraction of the parent object corresponds to the shard. Perhaps most importantly, a hash of the shard is calculated and the result stored when the shard is written.

Intelligent Data Monitoring

The hash values stored for each object shard are key to the first stage of DDR, intelligent data monitoring. In the background, a small army of out-of-band monitoring processes continually crawls through all the object shards stored on an ActiveScale system. Their job is to examine the object shards for consistency. The number of processes used scales up as the system grows. They all run in parallel, but at a low priority so they don't impair regular I/O.

First, the metadata stored with each object shard is compared to the values stored in the ActiveScale databases. If there is a mismatch, a repair action for the object shard is queued. If

the metadata matches, the shard content itself is checked for consistency by reading the shard, running it through the hash algorithm, and comparing the result with the stored value. If the hashes match, the shard is consistent, and the monitoring process moves on to examine the next object shard. If the hashes don't match, this indicates the shard is corrupted, and a repair task for the object shard is queued. Important to note is that this check is done entirely local to the node where the shard is stored. No data movement over the network is required for bit-rot verification.
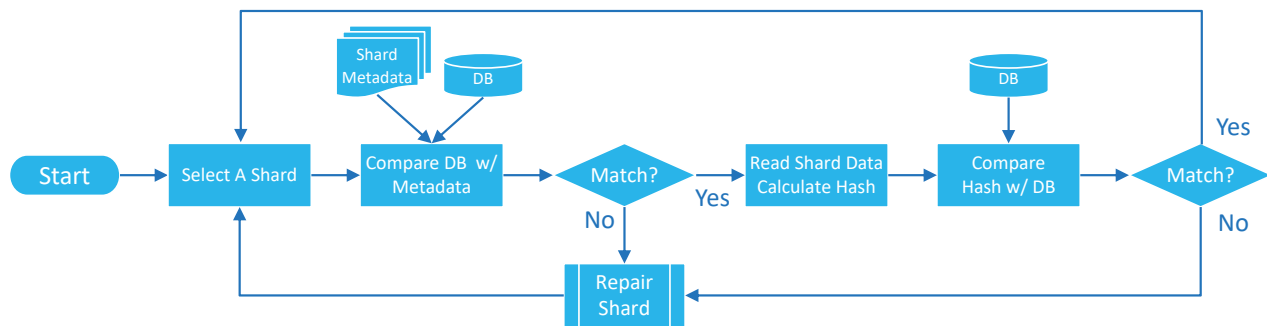

Figure 12 - Intelligent Data Monitoring Process

Proactive Data Repair

Due to the redundant nature of the erasure encoding used in ActiveScale, detecting a few damaged or corrupted object shards isn't a cause for panic. An object can suffer the loss of multiple shards and still be readable. Although it's not an emergency, damaged objects need to be repaired as soon as possible to restore their expected durability.

Remember that in the metadata stored for each object shard includes the name of the parent object, for example `Sally's Spreadsheet.xls`, along with the sequence number indicating which fraction of the parent object the shard represents.

For example, assume that `Sally's Spreadsheet.xls` was stored into ActiveScale and encoded into 15 shards, with sequence numbers 1-15. Further assume that shard five has been flagged as inconsistent and must be repaired. The repair process will first reconstruct `Sally's Spreadsheet.xls` from the remaining object shards. `Sally's Spreadsheet.xls` will then be re-encoded into 15 shards, as it was originally.

Now the process gets efficient. Other object storage systems must re-write entire repaired objects back to disk. Because ActiveScale knows which shard of the object was damaged, it discards the unneeded shards (1-4 and 6-15 in our example) and writes only the repaired shard (five) back to any eligible disk in the system, completing the repair. This fine-grained approach minimizes the system resources needed for repairs and is only possible due to the "smart" nature of the object shards.

Figure 13 - Shard Repair Process

ACTIVESCALE COLD STORAGE

ActiveScale software version 6.1 introduced a revolutionary cold storage class. ActiveScale is now the only object storage system with an integrated class of storage designed for cold data, accessed using industry standard S3 Glacier Class APIs. ActiveScale Cold Storage (ASCS) was born from Quantum's experience designing cold storage architectures for web scale companies and public cloud hyperscalers. It provides extreme data durability, availability, security, scalability, and ease-of-use – just like the public cloud – but resides in the data center of your choice. For organizations that need to store petabytes to exabytes of unstructured data, ASCS can do the job at up to 80% lower cost compared to traditional all-disk object storage systems.

Several patented Quantum innovations (US11216196, US20210294514), combined synergistically, make this capability possible, including two-dimensional erasure coding (2D EC) and redundant array of independent libraries (RAIL) technology.

Tape is Tough

Although the general idea of storing objects on tape media is not new, the approaches historically used have been naïve and flawed. Using a simple object to tape gateway gets objects on tape and preserves retrieve performance but writing an object to only one tape isn't durable. Writing multiple copies of an object to two or three tapes improves durability, but costs 100% - 200% in storage overhead, seriously diluting the economic benefits of using tape. Erasure coding an object and spreading it across multiple tapes seems smart – it solves durability with only moderate overhead – but having to load five, six, or more tapes to retrieve a single object doesn't scale, requiring high numbers of expensive tape drives and once again negating the attractive economics of tape. Finally, these approaches can't always maintain object availability when an entire site, library or gateway box goes down. ASCS solves all these problems and more.

	Object on 1 Tape	Object Replicated on 3 Tapes	Object EC Across Multiple Tapes	ActiveScale 2D EC
Durability	Low	Medium	Medium	High
Overhead	Low	High	Medium	Low
Cost	Low	High	Medium	Low
Read Efficiency	High	High	Low	High

Figure 14 - Legacy Approaches vs. ActiveScale 2D EC

Two-Dimensional Erasure Coding

2D EC is the technique of erasure coding data down the length of a single tape, and simultaneously performing erasure coding across a group of tapes as shown in Figure 15. By spreading object data and parity down the length of a tape, most object retrievals can be accomplished with a single tape mount, even if the tape has areas of damage. Across tape EC protects the data against major

problems such as the complete loss or failure of a tape. To increase efficiency, objects are not written to tape individually. They are grouped into larger BLOBs and written a few hundred GB at a time. This enables large streaming writes, minimizes the work the tape robot needs to do loading and unloading tapes, and minimizes tape wear. 2D EC provides high data durability – up to 19 nines – with overhead as low as 15% and high store and retrieve efficiency.

For more details on 2D EC and how it's unique, refer to the video on Quantum.com titled "Bringing Tape Economics to Object Storage: A Comparison of Solution Architectures."

Figure 15 - 2D EC and RAIL Architecture

Redundant Array of Independent Libraries

2D EC is a powerful technique for protecting data, but its value is maximized when it is combined with another Quantum innovation, the Redundant Array of Independent Libraries, or RAIL architecture. Typically, organizations needing lots of tape storage purchased very large, monolithic tape libraries, and this architecture is still popular. But large libraries have several downsides. First is the challenge of finding a place to put them. Second is the high entry cost. Third is availability. These expensive libraries have lots of redundancy and high availability features, but no single tape library is as highly available as multiple totally separate units. Fourth is performance. The modern "active archive" can require more performance than even a pair of robots can provide.

RAIL is the concept of using a cluster of relatively small, scalable tape libraries such as the Quantum i6 or i6H to store data, instead of a single large monolithic library. The exact quantity and configuration of libraries (number of slots and drives per library, for example) may be tailored to achieve the performance, density, and cost profile required. Expandability is achieved by "scaling up" the individual libraries in the cluster, "scaling out" by adding additional libraries to the cluster, or both.

ASCS leverages RAIL to store the 2D erasure coded data. For normal retrievals, only a single tape mount on a single tape library is required as discussed above. But if major problems occur, such as the failure of a library or even destruction of an entire site, all data remains safe and accessible via the remaining libraries in the RAIL cluster. This is only possible due to the combined use of 2D EC and RAIL.

ActiveScale Data Access Methods

Data may be stored to and retrieved from ActiveScale using several different access methods, including two different S3 storage classes (Standard and Glacier) and NFS.

S3 API COMPATIBILITY

ActiveScale's support of the AWS S3 API set is extensive. Though ActiveScale's responses will usually be identical to those of AWS, there are some cases where ActiveScale's response will be slightly different. Depending on how an application is coded and which functions are used, this may or may not matter. Full documentation of ActiveScale S3 API support, including notes on differences versus AWS, is included in the ActiveScale OS S3 API Reference Guide. Organizations considering ActiveScale who are concerned about application compatibility should work with a Quantum Solution Architect to gain access to a physical or virtual ActiveScale system for testing.

S3 STORAGE CLASSES AND RESTORE TIERS

S3 Storage Classes are partially supported by ActiveScale. As mentioned previously, objects `PUT` using S3 Standard class are stored on disk, and objects `PUT` using S3 Glacier class are stored to tape when the ActiveScale Cold Storage option has been implemented. But AWS has many minor storage classes, such as Infrequent Access (IA), Reduced Redundancy, and others. These other storage classes are treated as "synonyms" of S3 Standard class or S3 Glacier class, as shown in Table 1.

To maintain compatibility, ActiveScale will not reject a request that includes an unsupported storage class in the `x-amz-storage-class` header. Instead, it will use either the S3 Standard storage class or the S3 Glacier class, as appropriate, and store the requested storage class in the object metadata so that it is visible during a `GET Object` or `HEAD Object` operation. In all cases, objects are stored with the configured durability policy.

The result is that though the requested storage class may not be supported by ActiveScale in exactly the same fashion as AWS, objects will be stored with the same or better durability and access characteristics than requested. As an example, consider the S3 Reduced Redundancy storage class. ActiveScale will store objects destined for S3 Reduced Redundancy with the same durability as objects destined for S3 Standard class, not, in fact, with reduced durability.

ActiveScale prioritizes restore requests to the best of its ability based on the specified restore tier, with `Expedited` requests taking top priority, followed by `Standard` (the default), and finally `Bulk`.

Storage Class	Supported?
DEEP_ARCHIVE	Treated as synonym of GLACIER
GLACIER	Yes, w/ActiveScale Cold Storage only
GLACIER_IR	Treated as synonym of STANDARD
INTELLIGENT_TIERING	Treated as synonym of STANDARD
ONEZONE_IA	Treated as synonym of STANDARD
REDUCED_REDUNDANCY	Treated as synonym of STANDARD
STANDARD	Yes
STANDARD_IA	Treated as synonym of STANDARD

Table 1 - Storage Classes & Synonyms

S3 is the de facto standard language for communicating with cloud storage. ActiveScale's extensive API-level compatibility ensures that the vast majority of applications designed for use with AWS will work with ActiveScale without modification.

Object Workflows with ASCS

Regardless of whether the ActiveScale Cold Storage option is used, applications may always `PUT` and `GET` objects using S3 Standard class, and ActiveScale will store them on the internal disk tier.

With the introduction of ASCS comes the ability for applications to use S3 Glacier class API calls to direct ActiveScale to store objects on tape. Retrievals from ASCS also follow the same rules as native AWS Glacier. Upon determining an object is in cold storage using `HEAD`, the application issues a `RESTORE` and polls for updates. When the object has been copied back to disk, the application uses a `GET` to access it. The object copy remains on the disk tier for a configurable amount of time before being purged. Because ASCS uses standard API calls and workflows, any application that understands S3 Glacier class storage may use ASCS with no modifications.

In addition to directly storing objects to disk (S3 Standard class) or cold storage (S3 Glacier class), S3 lifecycle policies may optionally be used. Lifecycle policies are set at the bucket level within ActiveScale, enabling objects stored in S3 Standard class to be transitioned to S3 Glacier class in the background, with ActiveScale moving the objects from disk to tape appropriately. Applications use the `HEAD` command to determine whether objects reside in Standard or Glacier class.

S3 STANDARD
- PUT objects land on disk in S3 STANDARD class
- On retrieve (GET) objects accessed from disk

S3 GLACIER
- PUT objects land on disk in S3 GLACIER class
- Objects *moved* to tape by ActiveScale
- On retrieve, *copy* made back to disk (RESTORE), accessed from disk (GET)
- Copy remains on disk for configurable period before expiration

S3 STANDARD W/LIFECYCLE POLICY
- PUT objects land on disk in S3 STANDARD class
- Per policy, objects transition to S3 GLACIER class
- Objects *moved* to tape by ActiveScale
- On retrieve, app determines object class (HEAD)
- Retrieve proceeds based on S3 STANDARD or S3 GLACIER class

Figure 16 - Object Workflows with ASCS

NFS

ActiveScale also has a native NFSv3 POSIX compliant interface, which stores files internally as objects. Up to 65,535 top level directories may be created and exported, and each directory may have multiple exports with unique tags, for example to provide different permissions to different clients.

Because ActiveScale is intended to be used primarily as an object storage system, the NFS interface is designed for object-like workflows that have medium to large files and modest IOPS requirements. It is not designed to replace a high-performance clustered NAS system. Files stored via NFS may only be accessed via NFS, and objects stored with S3 may only be accessed via S3. Simultaneous multi-protocol access to files and objects is not currently supported. Instructions for configuring the NFS interface and a list of limitations are included in the ActiveScale Online Documentation under Advanced Topics.

Identity Model

ActiveScale has a three-tier hierarchical identity model, consisting of System Administrators, Accounts, and S3 Users, as illustrated in Figure 13. This model provides flexibility for shared use of a single ActiveScale system, including the delegation of some administrative responsibilities. Larger organizations can benefit from the efficiencies of maintaining one ActiveScale system for use by multiple parts of the organization and multiple applications, and service providers can offer object storage services with isolated administrative domains.

This model largely mirrors to the way AWS is structured, though there are minor differences as described in the documentation. ActiveScale System Administrators are analogous to AWS themselves. Accounts are similar to AWS accounts, and S3 Users are much like AWS IAM Users.
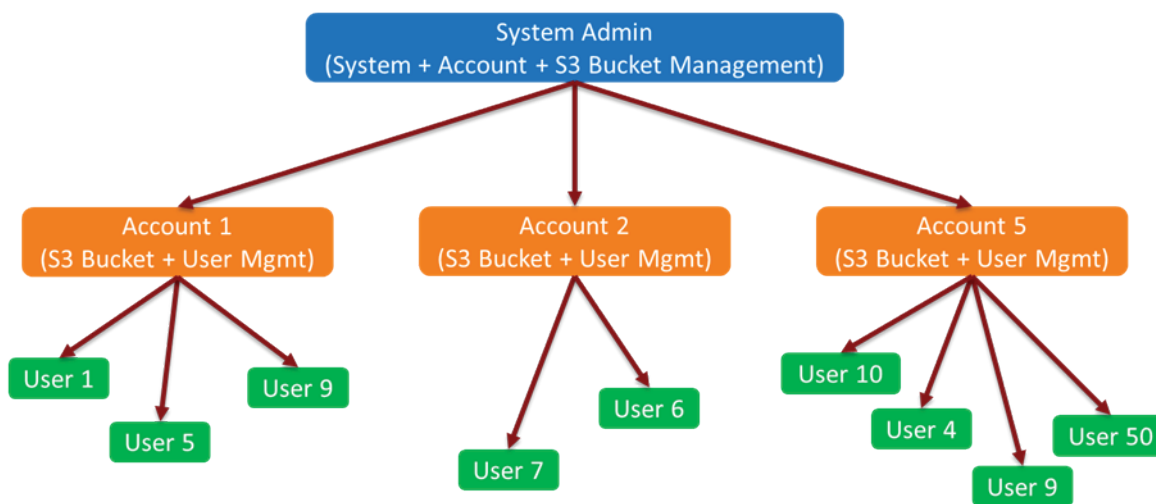

Figure 17 - ActiveScale Identity Model

System Administrators primarily use the ActiveScale System Manager GUI (ActiveScale SM) to manage the overall ActiveScale system. For scripting and automation, a CLI and REST API are also available. Management of S3 buckets and NFS volumes is done via the ActiveScale View GUI. An Active Directory group may be used when multiple individuals require System Admin access. The System Administrator does not have access to objects within buckets, but to some extent may manage Accounts on behalf of their owners.

Accounts manage S3 users, buckets, and objects, as well as NFS volumes, and can be billed for the capacity they use. Whereas the System Administrator is often someone in the corporate IT organization, Accounts are commonly assigned to application owners or DevOps teams. Accounts can create and delete buckets, control bucket permissions, and manage special bucket properties such as versioning, replication, object lock, and object lifecycle policies. They can have multiple linked users, which are not visible to other Accounts. Accounts exclusively perform their functions using the ActiveScale View GUI, and do not have access to ActiveScale SM, the management CLI or REST API.

S3 Users are created by, and belong to, Accounts. S3 Users PUT and GET objects to and from buckets and can set bucket permissions if given the appropriate access by the Account. Since S3 Users are the entities that initiate S3 operations, they each have one or more API keys. Although S3 Users may correspond to individuals in the organization, they are commonly associated with applications that need to use ActiveScale as an object repository.

	System Administrator	Account	S3 User
What is managed	• System • Accounts • S3 Users • S3 Buckets	• S3 Buckets • S3 Users • S3 Objects	S3 Objects
ActiveScale System Management GUI ActiveScale CLI Restful Management API	✓	✗	✗
ActiveScale View (S3 Bucket GUI)	✓	✓	✗

Table 2 - Identity Model Summary

Data Services

In addition to the basic ability to store and protect objects, ActiveScale software includes a range of data services. These services are included in the base software, requiring no additional installation. Their use is completely optional, offering advanced functionality for organizations or applications that require it. These services are configured at the bucket level unless otherwise noted.

Data Service	Functionality
Versioning	Keep multiple versions of an object in the same bucket
Lifecycle Policies	Expire objects and versions or transition them between storage classes
Encryption	System-wide or object-level encryption of objects & metadata at rest and in transit
Object Lock	Blocks object version deletion for a pre-defined period
Account Quotas	Limit system capacity used for object and file storage per account
Data Pipeline Service	Enables real-time notifications of S3 operations
Bucket Replication / Hybrid Cloud Support	Asynchronous bucket-level replication from ActiveScale to ActiveScale or AWS

Table 3 - ActiveScale Data Services

VERSIONING

Object versioning may be enabled on a bucket either at the time of creation, or later. Versioning is useful for maintaining a history of changes to an object and allows older versions to be accessed in case an object is accidentally overwritten or deleted. Versioning is also a prerequisite for certain other functions such as replication and object lock.

Once enabled, versioning may not be disabled, but it can be suspended. When objects are stored into version-enabled buckets, they are given a version identifier. Multiple objects may share the same key but will have unique version identifiers. Object deletions are handled differently. The object isn't deleted, instead a delete marker is created and set to the current version. Accessing a previous version of an object simply requires specifying the object version identifier in the GET, and similarly, an object version may be permanently deleted by executing a DELETE against a specific object version.
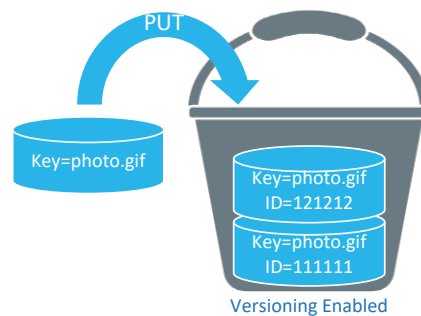

Figure 18 - Object Versioning

LIFECYCLE POLICIES

Object lifecycle management (OLM) policies enable managing storage cost by controlling the existence of objects and versions and the storage class on which they reside. Policies may automate actions such as these:

- Expire object versions older than 365 days

- Transition objects and versions older than 30 days to cold storage

- Expire objects older than 3650 days (10 years)

- Transition noncurrent object versions to cold storage

The list above contains just a few representative examples, many other options are possible. OLM policies may be applied to buckets programmatically via the PutBucketLifecycle operation, or by pasting the appropriate JSON-formatted text into the bucket's Properties page within the ActiveScale View (AS View) tool. To apply a policy to only some objects in a bucket, a object key prefix may be specified.
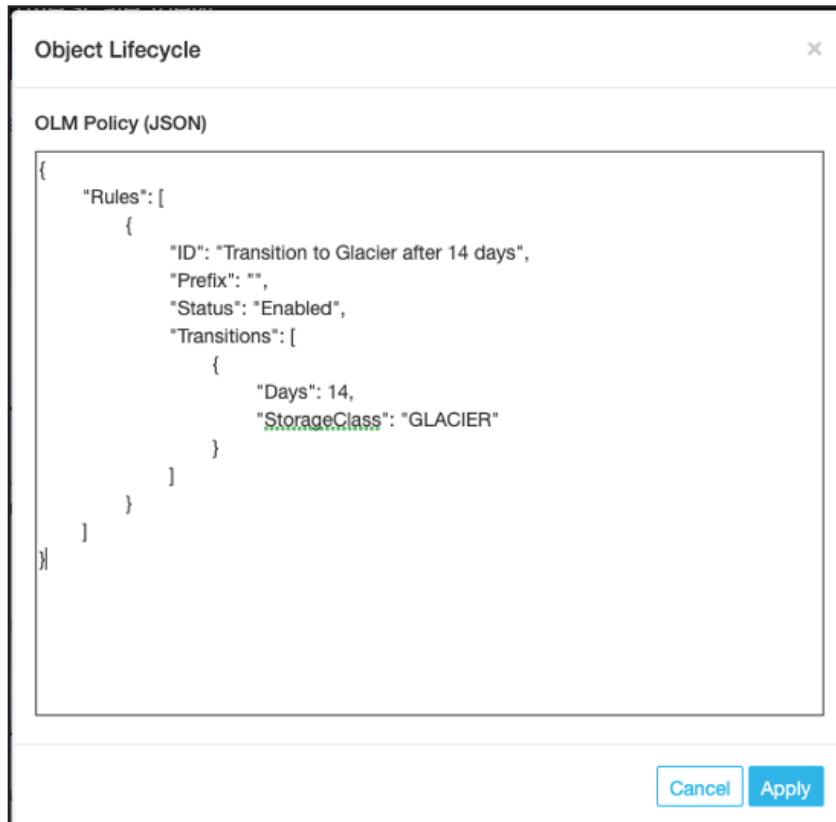
Figure 19 - OLM Policy JSON Example

The following policy actions are supported:

Policy Action	Purpose
Expiration	Expire objects or versions after a specified period
NoncurrentVersionExpiration	Permanently delete noncurrent object versions at a specific period in the object's lifetime
Transition	Transition objects or versions to another storage class, such as GLACIER, after a specified period
NoncurrentVersionTransition	Transition noncurrent object versions to another storage class, such as GLACIER

Table 4 - Object Lifecycle Policy Actions

ENCRYPTION

ActiveScale supports both encryption of data in transit, and at rest. Data being transferred to and from the system is always encrypted when using standard HTTPS/SSL.

Activation of data at rest encryption is controlled via an accessory license key. This is purely to satisfy export restrictions. The license key is available at no charge in eligible regions and is not available for systems sold into restricted countries. The encryption license key is unique to each ActiveScale system and is cryptographically signed by Quantum. Before applying the key and enabling encryption, the ActiveScale system will validate that the submitted license key is correct and properly signed.

Two modes of encryption of data at rest are supported by ActiveScale: system-wide and object level. Both modes encrypt objects before they are erasure coded and stored to disk. Object data and object metadata are encrypted using a 256-bit key, with a unique encryption key generated for each object. Unlike AWS, ActiveScale encrypts all object metadata, including custom object metadata.

System-wide encryption automatically encrypts all incoming objects and is transparent to applications. Object level encryption only encrypts objects sent with the S3 request header `x-amz-server-side-encryption:AES256`, and thus is reliant on application support. Once encryption is enabled it may not be disabled, but it can be toggled between modes if needed. Changing the encryption mode does not change the status of objects already written to disk – encrypted objects remain encrypted, and vice versa. ActiveScale supports only server-side encryption with Amazon S3-managed keys (SSE-S3), not alternative methods of server or client-side encryption.

Each object is protected with a unique key auto-generated by the system and stored in the object metadata. Object metadata is encrypted using a system-generated bucket encryption key. Bucket encryption keys are stored in the bucket metadata and are encrypted using the system master encryption key. The master encryption key is generated when the encryption at rest feature is enabled and is based in part on an administrator-provided passphrase. Neither this passphrase nor any of the encryption keys are ever stored in clear text.
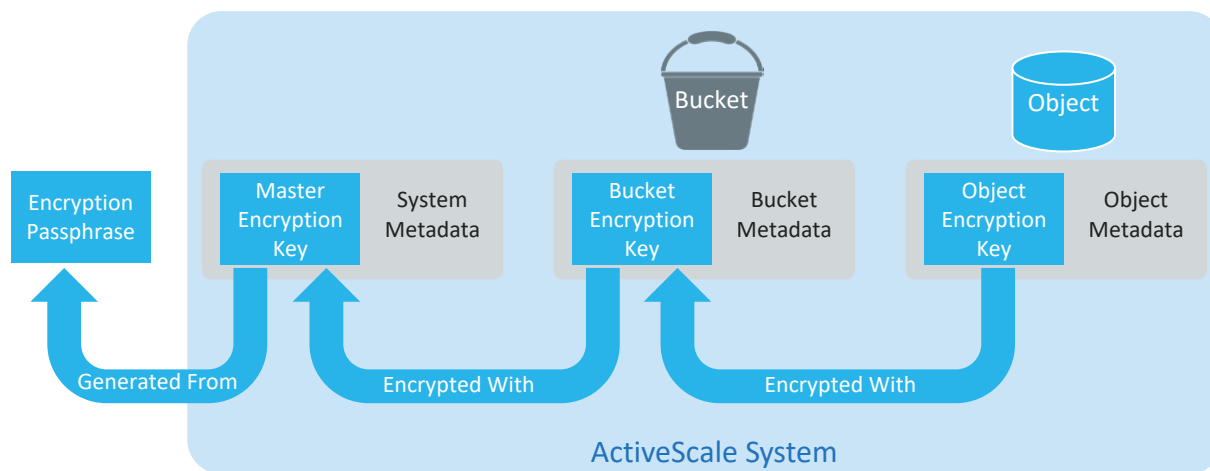

Figure 20 - Encryption Chain

OBJECT LOCK

Object Lock provides the ability to prevent an object version from being overwritten or deleted for a specified time. The S3 API defines two modes, governance mode and compliance mode. Governance mode enables administrative users to lock and unlock objects at will. Compliance mode locks objects until a specified time in the future and may not be overridden or disabled once set. This can help meet regulatory requirements that mandate a WORM (write once read many) model for storage. ActiveScale supports Compliance mode.

When the retention period is set on an object version, ActiveScale stores a time stamp in the object version's metadata to indicate when the retention period expires. After the retention period

expires, the object can be deleted. Once set, the retention period cannot be shortened, but it can be extended.

A retention period can be set one of two ways, either explicitly through the S3 API by specifying a 'retain until' date, or implicitly through a bucket default setting that specifies a duration in days or years. When an object is placed into a bucket with a default retention setting, ActiveScale calculates and applies the 'retain until' date automatically. The bucket default may be overridden by explicitly specifying a different retention when storing an object.

To use Object Lock, it must be enabled when a bucket is created, and it requires that versioning also be enabled. Once enabled, it may not be disabled. Setting the default duration is optional and changing the bucket default does not affect objects that have already been written.

Figure 21 - AS View Create New Bucket Dialog

ACCOUNT QUOTAS

Accounts in ActiveScale are the owners and creators of S3 buckets and NFS volumes. To facilitate capacity management, an account quota function is provided (patent US11157205B2). Quotas are especially useful when multiple users and / or applications have access to an ActiveScale system, to ensure that one user or application cannot monopolize the available capacity of the system at the expense of the others. An account can be the owner of multiple S3 buckets and multiple NFS volumes. Because quotas apply at the account level, they apply to the sum of the capacity in all buckets and volumes owned by an account.

Warning Threshold	Soft Quota Action	Hard Quota Action
100%	Alert	Alert, buckets & vols become read-only, `PUT` & `INIT` Denied
95%	Alert	Alert
85%	Alert	Alert

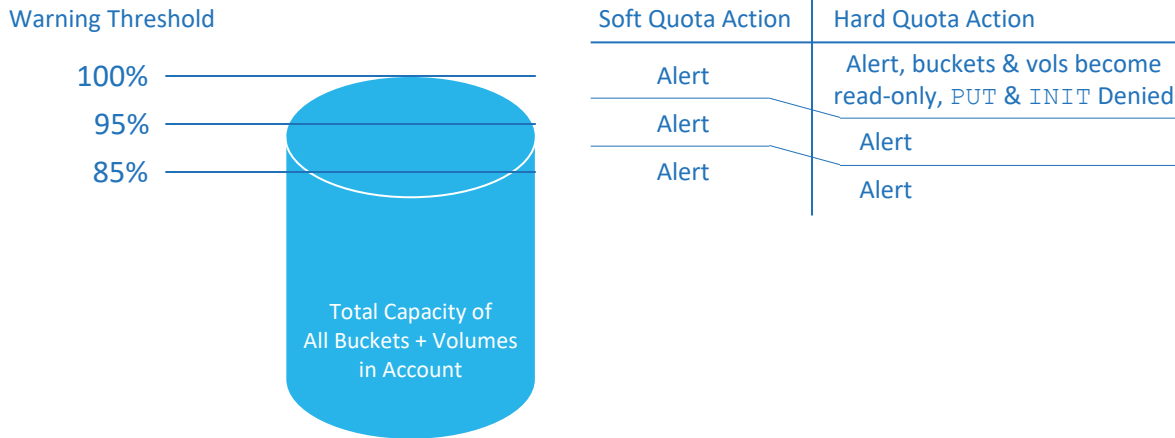Total Capacity of All Buckets + Volumes in Account

Figure 22 - Account Quota Threshold Actions

Both hard and soft quotas are supported, both with configurable low and high warning thresholds, which are set to 85% and 95% by default. Exceeding a warning threshold causes alerts to be sent to the account email address and system administrator. No enforcement action is taken. Similarly, exceeding a soft quota (reaching 100% utilization) only results in additional alerts. Exceeding a hard quota, however, results in future `PUT` operations being denied. Affected buckets and volumes are set to read only and the account and system administrator are notified the limit has been reached. Applications attempting object `PUT` or `INIT` operations will receive a `403 Forbidden (Quota Exceeded)` response. When enough capacity has been freed or the quota has been increased by an administrator, writes may resume.

System administrators may monitor usage via automatically generated reports that reside in the system bucket. System-wide and per-account reports are available, which are retained for a configurable duration.

DATA PIPELINE SERVICE

Data Pipeline Service (DPS), formerly known as Object Notification Service (ONS), enables real-time notifications of events that occur in a configured bucket. It works much like Amazon's Simple Notification Service (SNS).

DPS works by sending asynchronous notifications to a topic on an external Apache Kafka® server. Notifications can trigger actions, such as workflow processing steps. These are useful in many use cases and business contexts, including real-time analytics, IoT, mobile applications, rich media workflows, and business processes. For example, when a new video is `PUT` into a configured ActiveScale bucket, the notification generated by DPS could trigger a process to read that object, transcode it into a different format, and write the result to another bucket. The receipt of a new purchase order into a bucket could trigger a notification that results in a product being released to the shipping department and an invoice generated.
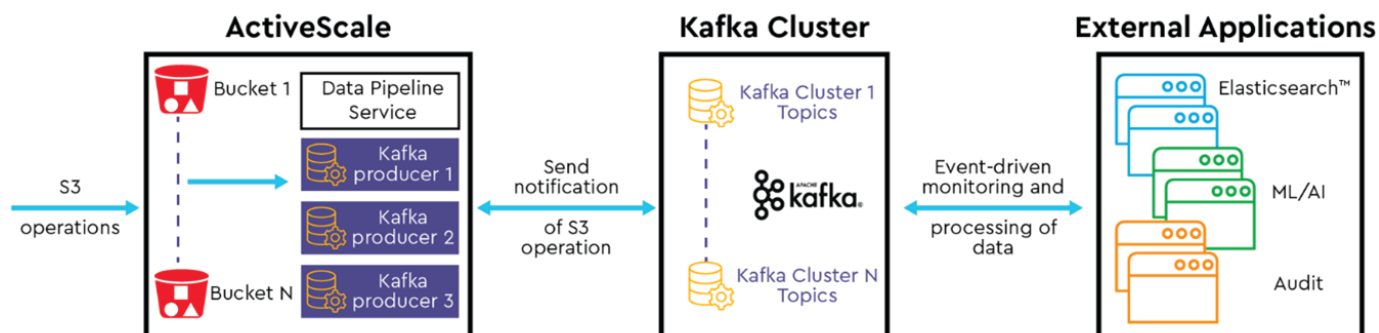
Figure 23 ActiveScale DPS Workflow

To enable DPS, the ActiveScale system administrator must first enable the DPS feature on the system, specifying the list of Kafka brokers and topics that will be used. Next, individual buckets are configured, which involves selecting the topics to be targeted, and the specific S3 events that will trigger notifications to those topics. Table 4 lists the supported event notification types.

S3 Operation	A Notification is Sent When...
`S3:ObjectCreated:Put`	An object is written to ActiveScale. An object copy will also trigger this notification.
`S3:ObjectCreated:CompleteMultipartUpload`	A multipart upload is complete
`S3:ObjectCreated:*`	An object is `PUT` or copied, including multipart uploads
`S3:ObjectRemoved:Delete`	An object is deleted
`S3:ObjectRemoved:DeleteMarkerCreated`	An object is marked for deletion
`S3:ObjectRemoved:*`	An object is marked for deletion or deleted

Table 5 - DPS Event Notifications

BUCKET REPLICATION / HYBRID CLOUD SUPPORT

Data replication is useful for data protection, data distribution, and workflows that must cross geographic boundaries. ActiveScale provides the ability to automatically, asynchronously replicate the contents of an ActiveScale bucket to a bucket on a separate ActiveScale system, or to a bucket in Amazon AWS or other S3-compatible targets. The functionality is very similar to that offered by AWS Multi-Region Asynchronous Object Replication. When enabled, the ActiveScale OS continually monitors the contents of the source bucket for changes and immediately forwards these changes to the destination bucket, using dedicated replication accounts on the source and target. Replication can be unidirectional or bidirectional. Because it is asynchronous, it works well with high latencies and unstable networks, with automatic retries guaranteeing eventual consistency between the source and target buckets. ActiveScale also supports replication of Object Lock enabled buckets, employing the same immutability policies of the source bucket for replicated objects in the destination bucket. The ActiveScale SM UI includes graphs that make it easy to monitor the status of replication, some of which are shown in Figure 24.

In addition to standard S3 replication features, ActiveScale provides the unique ability to replicate between S3 Glacier cold storage classes on independent systems, including ActiveScale-to-ActiveScale replication of cold data, and ActiveScale-to-AWS S3 Glacier Service replication. This

delivers a durable and cost-effective multi-copy solution for the long-term retention of data, both within a private storage cloud environment or as a hybrid cloud solution. Replication between storage classes is also supported, for example, from S3 Standard to S3 Glacier class across two sites or in a hub and spoke model, providing for secondary, offsite copies to be stored on lower cost tape media.
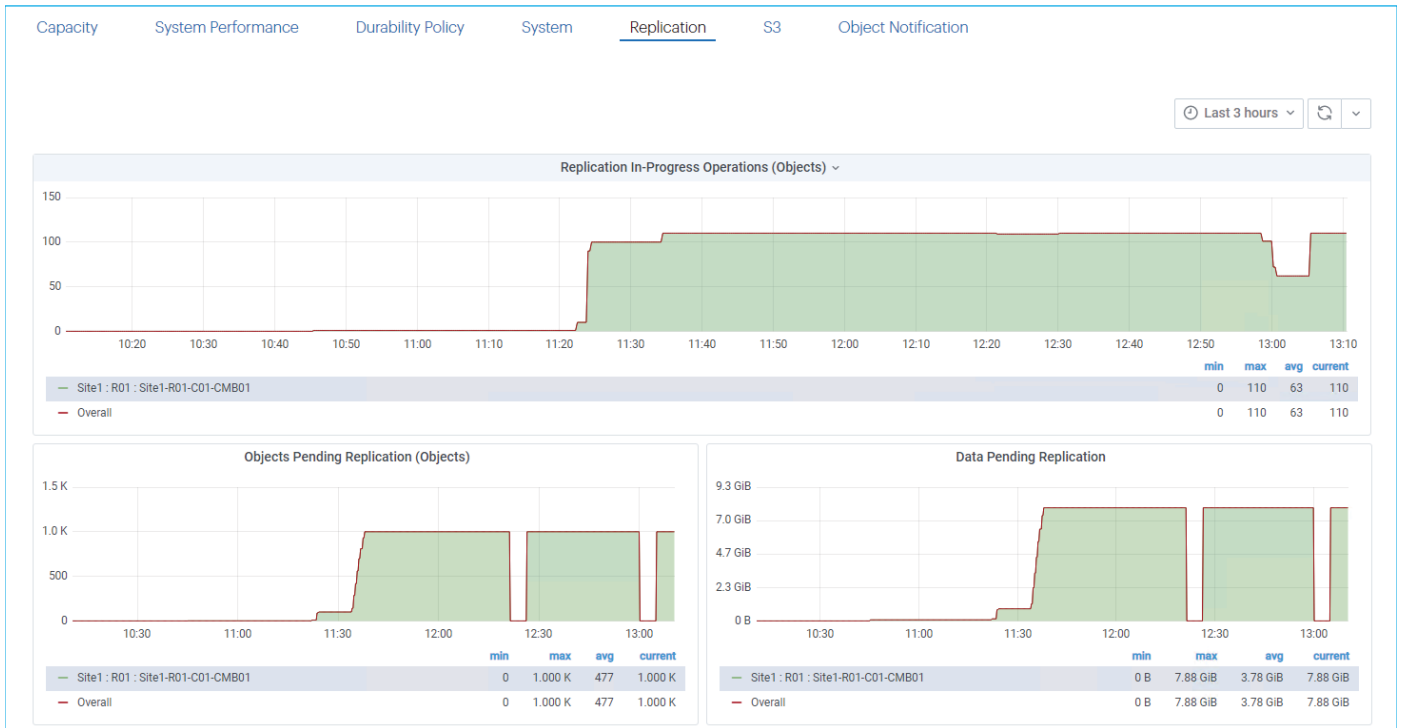

Figure 24 - ActiveScale SM Replication Monitoring Page

Replication may be configured via the ActiveScale management GUI, CLI, or S3 API. Configurable warning thresholds enable monitoring the replication pipeline so that action may be taken if excessive queuing occurs at the source, and replication pipeline status may be monitored graphically.
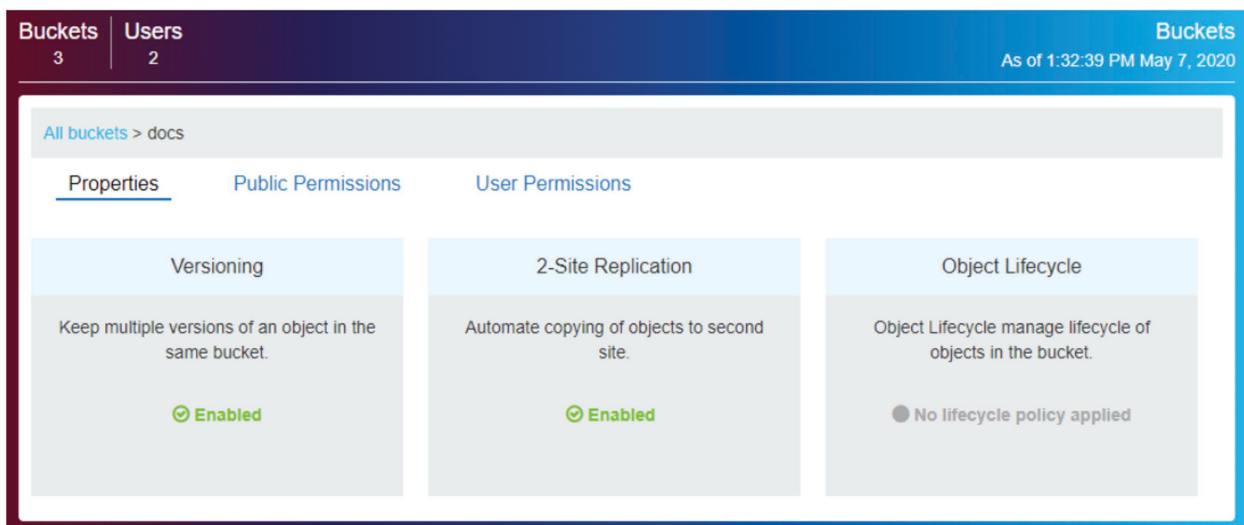

Figure 25 - Bucket Replication Enabled

MANAGEMENT & MONITORING

As described in the previous section, ActiveScale systems are supplied with two GUIs, ActiveScale System Manager (ActiveScale SM) and ActiveScale View. These UIs are supplemented by a variety of other tools for management, monitoring, and alerting.

ACTIVESCALE SM

ActiveScale SM is a web-based management tool used exclusively by System Administrators. Some of the capabilities available through ActiveScale SM include:

- Creating and managing S3 Accounts, Users, and API Keys
- Managing encryption, replication, and other system settings
- Monitoring system health, capacity, and performance
- Viewing and resolving system events
- Viewing logs and reports, including capacity reports and metering logs

Perhaps the most important feature of ActiveScale SM, however, is it's use for installing software upgrades and performing system expansions. Some object storage systems require administrators to SSH into individual nodes and manually edit configuration files and apply updates. This is untenable at scale.

Software upgrades on an entire ActiveScale cluster are accomplished with one click. ActiveScale SM coordinates upgrades across all nodes so that the services to users and applications are never interrupted. Each node simply undergoes a brief outage as the upgrade is applied, while the rest of the cluster remains on-line and available.

System expansions happen in a similar way. After the new nodes are racked and cabled into the cluster, the system expansion is completed within ActiveScale SM. As with software upgrades, users of the system experience no interruption of services. This coordination and automation are part of what makes it possible for ActiveScale systems to scale to 100 PB+ in practice, not just on paper.
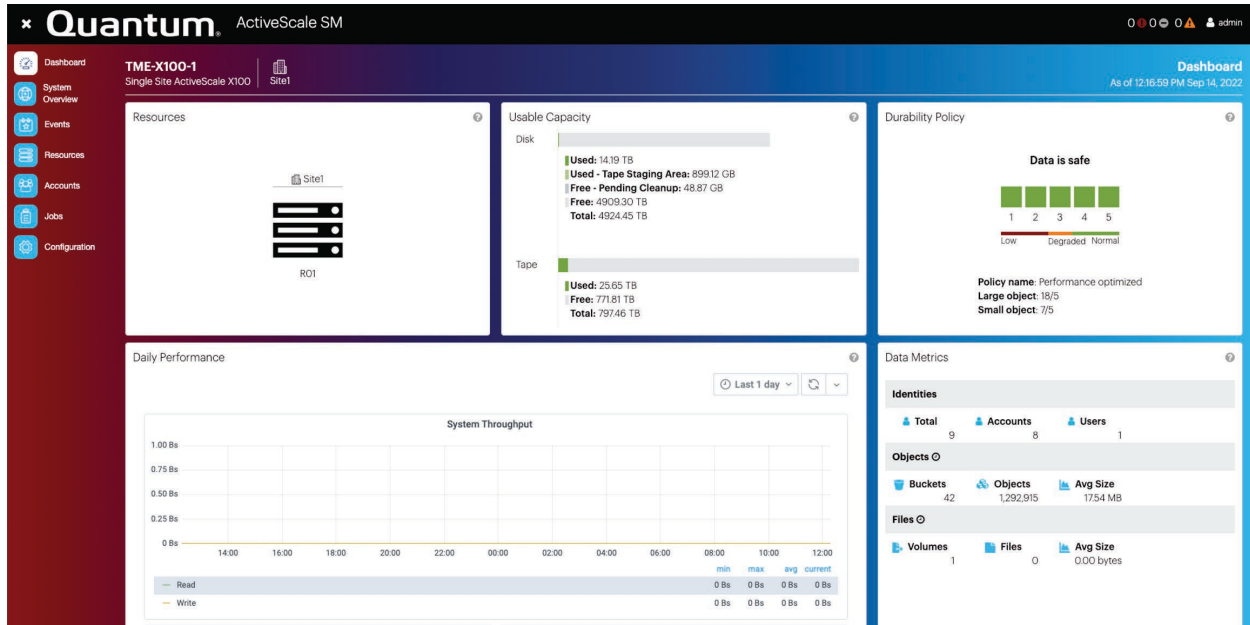
Figure 26 - ActiveScale SM Dashboard

ACTIVESCALE VIEW

ActiveScale View is a web-based tool primarily designed for use by the owners of S3 accounts, but it can also be used by the system administrator to take actions on behalf of S3 accounts. ActiveScale View enables functions like these:

- Create and manage S3 buckets
- Enable versioning
- Enable replication
- Manage bucket lifecycles
- Create S3 users and manage their credentials
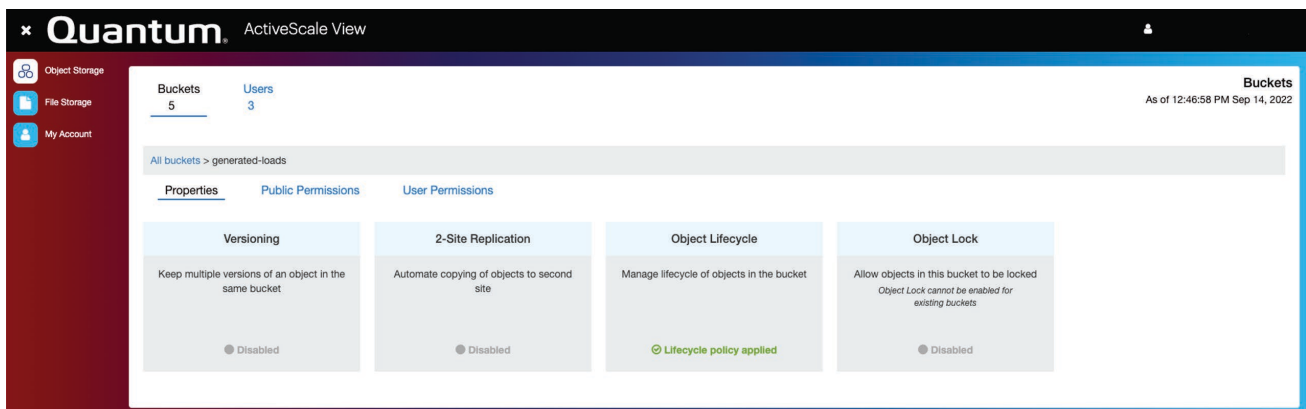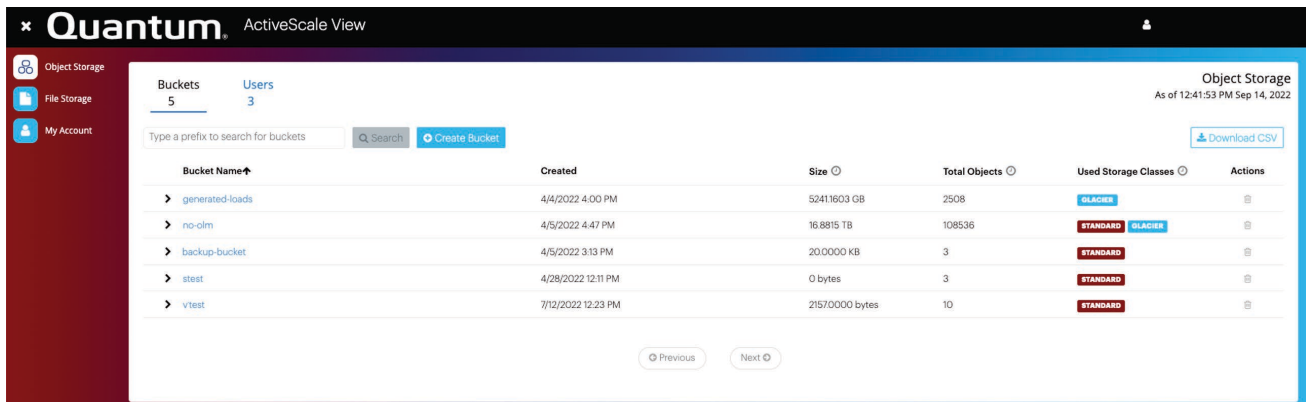- Create and manage NFS volumes and exports

Figure 27 - ActiveScale View GUI Examples

REST API AND CLI TOOL

Both ActiveScale SM and ActiveScale View are web-based front ends that consume a REST API. The REST API is fully documented to make it easy for customers to integrate ActiveScale management functions into their own management, monitoring, provisioning, or billing systems. Virtually all management actions may be completed with the API, using secure authenticated connections.

For those requiring a more traditional CLI, the ActiveScale CLI tool, known as `ascli`, is a Linux-only utility that acts as a wrapper for the ActiveScale management REST API. The tool must be downloaded from the ActiveScale SM GUI, and full documentation is located here.

SNMP, SMTP, AND SYSLOG STREAMING

In addition to the interactive and programmatic methods for monitoring and managing the system, more traditional methods are also supported, including SNMP v2c, SMTP, and Syslog streaming.

A comprehensive SNMP MIB may be downloaded from the ActiveScale SM GUI. When enabled, the ActiveScale system will send traps to the requested destination, as well as respond to polling for performance and status information. This is a great way to integrate ActiveScale into traditional enterprise data center monitoring frameworks.

Figure 28 - SNMP Configuration UI

Individual events, or a daily digest of events, may also be sent via email using SMTP. For multiple-site installations (such as 3GEO configurations), separate addresses may be specified for each site, if desired.

Figure 29 - Syslog Configuration UI

By default, ActiveScale retains log events for 90 days. If longer retention is desired, or there is a need for outside auditing or analysis of logs, ActiveScale can be configured to push log events to any RFC 5424 compliant syslog server. UDP, TCP, and TCP with TLS protocols are supported.

CLOUD MONITORING USING QUANTUM CLOUD-BASED ANALYTICS

To facilitate management and support, Quantum also offers Quantum Cloud-Based Analytics (CBA), a cloud-based monitoring and management tool, that enables administrators, Quantum support personnel, and authorized service providers to monitor ActiveScale system health remotely, with the system owner's permission.

Conclusion

Today's businesses are built on data, and every organization wants to keep more data, longer. Object storage has emerged as the preferred paradigm for storing massive unstructured data sets because of its scalability, flexibility, and friendliness to application developers - but all object stores are not equal. While they each address data resiliency, durability, and performance to some degree, the devil is in the details. Different implementations provide greater or lesser value.

Public cloud storage options are only cost-effective if you rarely access the data, but many unstructured data repositories are active, with portions regularly re-used for historical analysis, training new AI/ML models, and other activities. On premises object stores eliminate access fees, but keeping everything on spinning hard drives when only portions of the data are "hot" at any time doesn't make sense, especially when facility costs are factored in. Tape is attractive as a reliable, low-cost "green" medium for nearline storage, but most integrations of tape with object storage squander this promise.

Quantum ActiveScale is the answer. ActiveScale's patented technologies enable best-in-class data access, data protection, and storage efficiency. As described in this paper, erasure coding, automated monitoring, and proactive repair processes ensure that stored objects are always safe and recoverable at scale. Failures in any storage system are inevitable and increase with size. ActiveScale is designed with resiliency in mind, maintaining high availability and performance through disk, node, and site failures.

ActiveScale's robust options and data services allow customers to tailor each implementation to provide added value and manage cost. This is particularly true with the innovative Cold Storage option that seamlessly and intelligently integrates tape, fully leveraging its strengths while minimizing or eliminating the overhead and management traditionally required.

In summary, Quantum ActiveScale meets the need for enterprise-class object storage with the data durability, security, and availability required of massive unstructured data sets and use cases that range from terabytes to exabytes in scale. As the only object store system with an integrated class of storage for cold data, ActiveScale delivers up to 80% cost savings relative to any other on-prem object store. With S3 APIs, S3 Standard and Glacier storage class compatibility, and an optimized multi-tiered storage architecture, ActiveScale simply and cost-effectively scales performance and capacity without bound in support of analytical workloads, active archiving, long term retention, and cold data storage.

Resources

ActiveScale Product Page
www.quantum.com/object-storage

Quantum Object Storage Services Page
www.quantum.com/object-storage-services

Documentation Center (click the 'Object' tab)
https://qsupport.quantum.com/kb/flare/Content/doc_portal/Content/docs-portal/docs_portal.html

ActiveScale Features Tutorial Video Series
https://quantum.gallery.video/portal2/category/videos/quantum-activescale-features

Primary Author:

Dan Duperron

Technical Marketing Architect

Quantum Corporation

Quantum.

Quantum technology, software, and services provide the solutions that today's organizations need to make video and other unstructured data smarter — so their data works for them and not the other way around. With over 40 years of innovation, Quantum's end-to-end platform is uniquely equipped to orchestrate, protect, and enrich data across its lifecycle, providing enhanced intelligence and actionable insights. Leading organizations in cloud services, entertainment, government, research, education, transportation, and enterprise IT trust Quantum to bring their data to life, because data makes life better, safer, and smarter. Quantum is listed on Nasdaq (QMCO) and the Russell 2000® Index. For more information visit www.quantum.com.

www.quantum.com | 800-677-6268

©2023 Quantum Corporation. All rights reserved. Quantum and the Quantum logo are registered trademarks, and ActiveScale is a trademark, of Quantum Corporation and its affiliates in the United States and/or other countries. All other trademarks are the property of their respective owners.