




Whitepaper

Exploring Object Storage in Conjunction with the Ascendancy of COSI

Copyright © 2023, Futurewei® Technologies, Inc. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Futurewei® Technologies.

Trademarks and Permissions

 and other Futurewei® trademarks are trademarks of Futurewei® Technologies. Huawei trademarks are trademarks of Huawei Technologies Co., Ltd. All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services, and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services, and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees, or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

FUTUREWEI® TECHNOLOGIES, INC.

Boston Research Center

Address: 111 Speen Street, Suite 114
Framingham, MA 01701
United States of America

Website: <http://www.futurewei.com/>

Table of Contents

Executive Summary	5
Object Storage.....	6
What is Object Storage	6
History	7
Comparison with Other types Of Storage.....	8
Trend for Enterprise Application Storage.....	8
Simple Storage Service.....	9
Introduction	9
S3 Storage Classes	9
S3 Standard	9
S3 Intelligent-Tiering	9
S3 Standard-Infrequent Access (S3 Standard-IA).....	9
S3 One Zone-Infrequent Access (S3 One Zone-IA).....	9
S3 Glacier Instant Retrieval	9
S3 Glacier Flexible Retrieval	9
S3 Glacier Deep Archive	9
COSI	10
Introduction	10
Why COSI.....	10
COSI Architecture	11
Architecture	12
API	12
Self-Service.....	13
Portability.....	14
COSI Adoption	14
Modern Day Applications.....	15
History of Modern Applications.....	15
Importance of Modern Applications.....	15
Requirements of Modern Applications.....	15
Role of Object Storage in Modern Applications.....	16
Object Storage Use Cases	17

Object Storage For Backup & Disaster Recovery	17
Object Storage For DATA LAKEHOUSE.....	17
Object Storage For Unlimited Databases.....	18
Object Storage for DB as a Query Engine.....	19
Object Storage For Streaming Data Platforms	19
Object Storage For Machine Learning and AI	20
Significance of Object Storage and COSI in Data Mobility and Data Gravity	21
Data Gravity.....	21
Data Mobility.....	21
Object Storage & Data Gravity	21
COSI & Data Mobility	21
Databases on Object Storage	23
MinIO®	23
Snowflake®	23
Polybase In SQL Server®	24
Competitors	25
DELLEMC®	25
NetApp®	25
Pure Storage®	26
Conclusion.....	27
References.....	28

Executive Summary

- **Emergence of Object Storage:** Object storage architecture originated in the 1990s as an efficient solution for storing large amounts of unstructured data, particularly suitable for archiving and compliance requirements. It differs from traditional storage methods by organizing data into objects with associated metadata, rather than using files or blocks.
- **Strengths of Object Storage:** Object storage excels in scalability, metadata searchability, and cost-effectiveness for managing extensive unstructured data. It finds application in various use cases, including cloud-native applications, analytics platforms, data lakes, backups, databases utilizing scalable object storage as a data repository, storage for machine learning models, and more.
- **COSI Standardization:** COSI, or Container Object Storage Interface, standardizes integration with Kubernetes for portable object storage consumption, streamlining the interaction between containerized applications and object storage resources.
- **Data Gravity & Data Mobility:** Object storage alleviates the impact of data gravity, while COSI enhances data mobility, enabling enterprises to effectively handle large-scale data across a variety of environments, including cloud, hybrid, and on-premises settings.
- **COSI Expansion:** COSI holds the promise of expanding standardized integration of object storage for containerized applications, further enhancing the compatibility and flexibility of storage solutions within the container ecosystem.
- **Enterprise-Grade Solutions:** Leading vendors are actively delivering enterprise-grade object storage solutions, although adoption continues to grow as the capabilities of these solutions evolve and improve to meet the demands of modern data management.

Object Storage

What is Object Storage

Object storage, also known as object-based storage, is a computer data storage architecture designed to handle large amounts of unstructured data. Object storage differs fundamentally from conventional file and block storage in its approach to data management. In an object storage system, data is stored as discrete objects, each comprising the data itself and a distinct identifier called an object ID. This object ID serves as a means to locate and retrieve the object, eliminating the need for hierarchical file structures or block mappings, thus enhancing data access efficiency.

The architecture of object storage typically encompasses three key components: the data storage layer, the metadata index, and the API layer. Let's delve deeper into these components and explore how they collaborate to create a robust and adaptable storage solution.

The data storage layer serves as the repository for the actual data objects. Within an object storage system, data is usually distributed across numerous storage nodes to ensure superior performance, durability, and redundancy. Each storage node typically incorporates a mix of hard disk drives (HDDs) and solid-state drives (SSDs) to strike the right balance between capacity, speed, and cost-efficiency. Data objects are automatically duplicated across multiple nodes to guarantee data availability and safeguard against hardware failures or unforeseen disruptions.

The metadata index stands as a pivotal element of object storage architecture, responsible for maintaining a comprehensive record of each object's unique identifier, alongside pertinent metadata like access controls, creation date, and size. This information is kept separately from the actual data, facilitating rapid and efficient object retrieval based on metadata attributes. The metadata index is engineered for scalability, accommodating vast quantities, potentially numbering in the millions or billions of objects, within a single object storage system.

The API layer assumes the role of granting access to the object storage system, enabling users and applications to store, retrieve, and manage data objects. Most object storage systems support a variety of standardized APIs, including the Simple Storage Service (S3) API, OpenStack Swift API, and the Cloud Data Management Interface (CDMI). These APIs empower developers to seamlessly integrate object storage into their applications, regardless of the underlying storage technology or vendor.

History

Object storage had its origins in the mid-1990s, originally conceived as an efficient means to store substantial quantities of unchanging content over extended periods, particularly for purposes such as data archiving and regulatory compliance.

Early pioneers in the field of object storage technology included companies like FilePool®, Bycast®, and Permabit®. In the early 2000s, major storage vendors such as EMC®, NetApp®, and Hitachi® acquired many of these pioneering startups, expanding their portfolios to include object storage solutions.

The turning point for object storage's widespread adoption came with the launch of Amazon® S3 in 2006. Amazon S3 revolutionized the landscape by making object storage readily accessible as a service, while also popularizing the S3 API, which simplified the integration of applications with object storage.

Following the success of Amazon S3, numerous other cloud providers introduced their own object storage services, including Microsoft® Azure Blob Storage, Google Cloud Storage, and OpenStack Swift. Concurrently, there has been a surge in open-source object storage projects, further diversifying the object storage ecosystem.

Object storage's appeal lies in its suitability for constructing cloud-native applications and analytics platforms. It excels in offering unlimited scalability, the ability to efficiently search metadata, and a cost-effective approach to managing vast quantities of unstructured data.

Comparison with Other types Of Storage

	File storage	Block storage	Object storage
Data organization	Hierarchical	Fixed-size blocks	Individual objects
Access	File path	Block address	Object identifier
Scalability	Good for small to medium datasets	Good for large datasets	Good for very large datasets
Efficiency	Good for small files	Good for large files	Good for large amounts of unstructured data
Management	Easier	More difficult	Easier
Protection	Easier	More difficult	Easier

Trend for Enterprise Application Storage

Object storage is increasingly favored in enterprise applications due to its scalability, flexibility, and cost-efficiency when handling substantial volumes of unstructured data. Meanwhile, file storage remains the preferred choice for performance-sensitive workloads, offering low latency, data consistency, and file locking—ideal for databases, virtual machines, and traditional enterprise applications. Notably, the rise of high-performance object storage is challenging the boundaries between file and object storage, as it combines scalability with improved throughput and reduced latency. Looking ahead, projections suggest that by 2026, enterprises are anticipated to triple their unstructured data capacity, stored in either file or object storage, compared to the levels of 2022. This evolution reflects the changing landscape of data storage solutions in enterprise environments.

Simple Storage Service

Introduction

Amazon Simple Storage Service (S3) offers various storage classes optimized for different access frequency patterns, durability needs, and costs ¹. Choosing the right S3 storage class can help optimize performance and lower storage costs. This report provides an overview of the different S3 storage classes available.

S3 Storage Classes

S3 Standard

- High performance, low latency storage for frequently accessed data
- Offers 99.999999999% durability and 99.99% availability
- Used for cloud applications, dynamic websites, big data analytics, etc.

S3 Intelligent-Tiering

- Automatically moves objects between access tiers based on usage patterns
- No retrieval fees or operational overhead
- Optimizes costs for data with changing or unknown access patterns

S3 Standard-Infrequent Access (S3 Standard-IA)

- For infrequently accessed data, but requires rapid access when needed
- Lower storage price and per GB retrieval fee compared to S3 Standard
- Used for long-term storage, backups, disaster recovery

S3 One Zone-Infrequent Access (S3 One Zone-IA)

- Stores data in a single AZ instead of minimum
- 20% cheaper than S3 Standard-IA
- For infrequent access when availability/resilience of S3 Standard is not needed

S3 Glacier Instant Retrieval

- Lowest cost storage for archive data needing instant retrieval
- Milliseconds retrieval time
- Use cases: medical images, media assets, genomics data

S3 Glacier Flexible Retrieval

- Archive storage with minutes retrieval and free bulk retrievals
- 10% lower cost than S3 Glacier Instant Retrieval
- Use cases: backup, disaster recovery

S3 Glacier Deep Archive

- Lowest cost storage class in S3
- For long term archive, digital preservation
- Data retrieval time is 12-48 hours

COSI

Introduction

In 2017, the Kubernetes community introduced the Container Storage Interface (CSI) as a solution to address the limitations imposed by in-tree storage plugins. The support for CSI achieved the status of General Availability (GA) with Kubernetes version 1.13. CSI stands as a standardized approach for exposing both block and file storage systems, regardless of their specific nature, to containerized workloads operating on Kubernetes or any other container orchestrator employing the CSI framework. This innovation renders the Kubernetes volume layer highly extensible, allowing third-party storage providers to develop and deploy volume plugins seamlessly, without the need for modifications to the Kubernetes codebase.

Nonetheless, there is another compelling storage option to consider: object storage. In today's data landscape, marked by an unprecedented surge in unstructured data generation, object storage is swiftly gaining well-deserved popularity. Many production applications now opt for object storage to persist their data, driven by several key advantages. Notably, object storage is renowned for its cost-effectiveness and remarkable scalability. Additionally, it empowers fine-grained access control permissions and provides convenient accessibility through network APIs.

Why COSI

- To provide a generic, dynamic provisioning API to consume object store: Aim was to provide a standard interface for provisioning and consuming object storage in Kubernetes. This would make it easier for developers to use object storage in their applications, and also makes it easier for Kubernetes administrators to manage object storage resources.
- To allow App Pods to access the bucket in the underlying object-store like a PVC: COSI would allow Kubernetes applications to access object storage buckets using the same PersistentVolumeClaim (PVC) interface that they use to access other types of storage.
- To implement k8s controller automation design with pluggable provisioners: COSI is designed to be implemented using pluggable provisioners. This would make it easy to add support for new object storage providers without having to modify the COSI core code.
- To present similar user/admin experience for new and existing buckets: COSI presents a similar user and admin experience for both new and existing object storage buckets. This

makes it easier for users to learn and use COSI, and it also makes it easier for Kubernetes administrators to manage object storage resources.

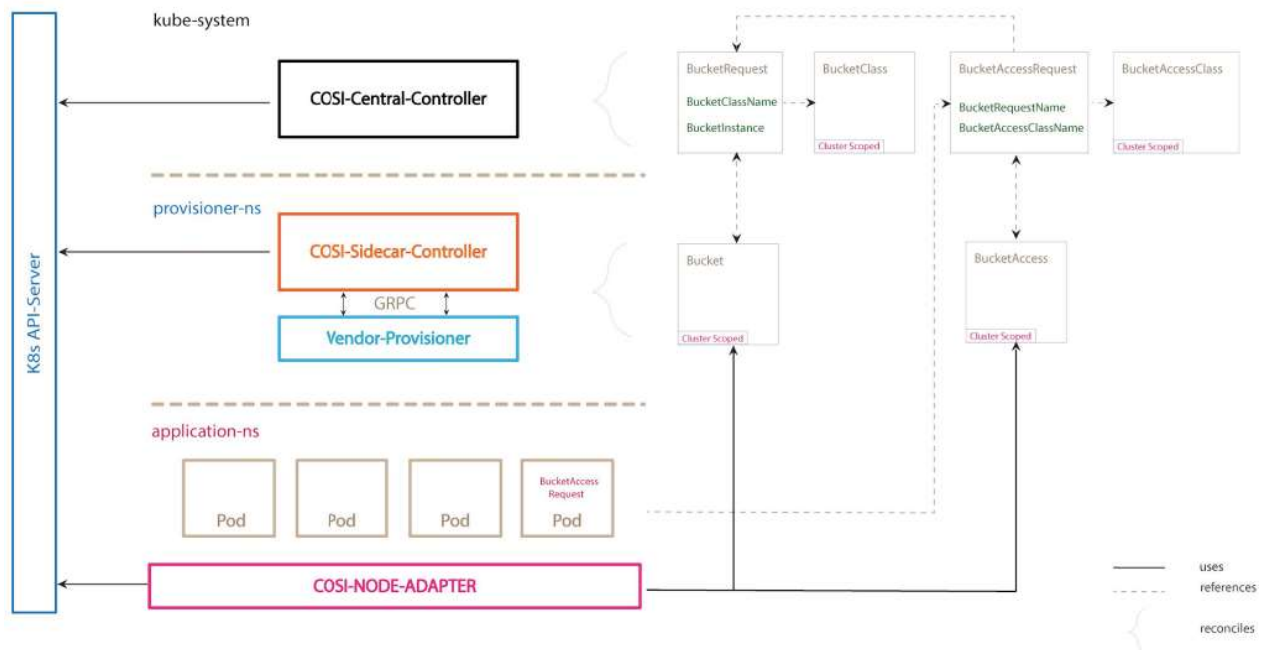
- To be vendor agnostic (S3, RGW, Swift, GCS, etc.): COSI is vendor agnostic, meaning that it can be used with any object storage provider that supports the S3 API. This makes it easy for users to choose the object storage provider that best meets their needs.

In summary, while it is possible to access object storage from inside a container without COSI, COSI provides a standardized, portable, and consistent way to interact with object storage systems, simplifying application development, enhancing security, and improving overall manageability, especially in complex and dynamic containerized environments. It helps bridge the gap between containerized applications and object storage resources, making them work together seamlessly.

COSI Architecture

COSI aims to standardize consumption of object storage to provide the following benefits:

- Kubernetes Native - Use the Kubernetes API to provision, configure and manage buckets.
- Self Service - A clear delineation between administration and operations (DevOps) to enable self-service capability for DevOps personnel.
- Portability - Vendor neutrality enabled through portability across Kubernetes Clusters and across Object Storage vendors.



Architecture

COSI is made up of three components:

- COSI Controller Manager
- COSI Sidecar
- COSI Driver

The COSI Controller Manager acts as the main controller that processes changes to COSI API objects. It is responsible for fielding requests for bucket creation, updates, deletion and access management. One instance of the controller manager is required per kubernetes cluster. Only one is needed even if multiple object storage providers are used in the cluster. The COSI Sidecar acts as a translator between COSI API requests and vendor specific COSI Drivers. This component uses a standardized gRPC protocol that vendor drivers are expected to satisfy.

The COSI Driver is the vendor specific component that receives requests from the sidecar and calls the appropriate vendor APIs to create buckets, manage their lifecycle and manage access to them.

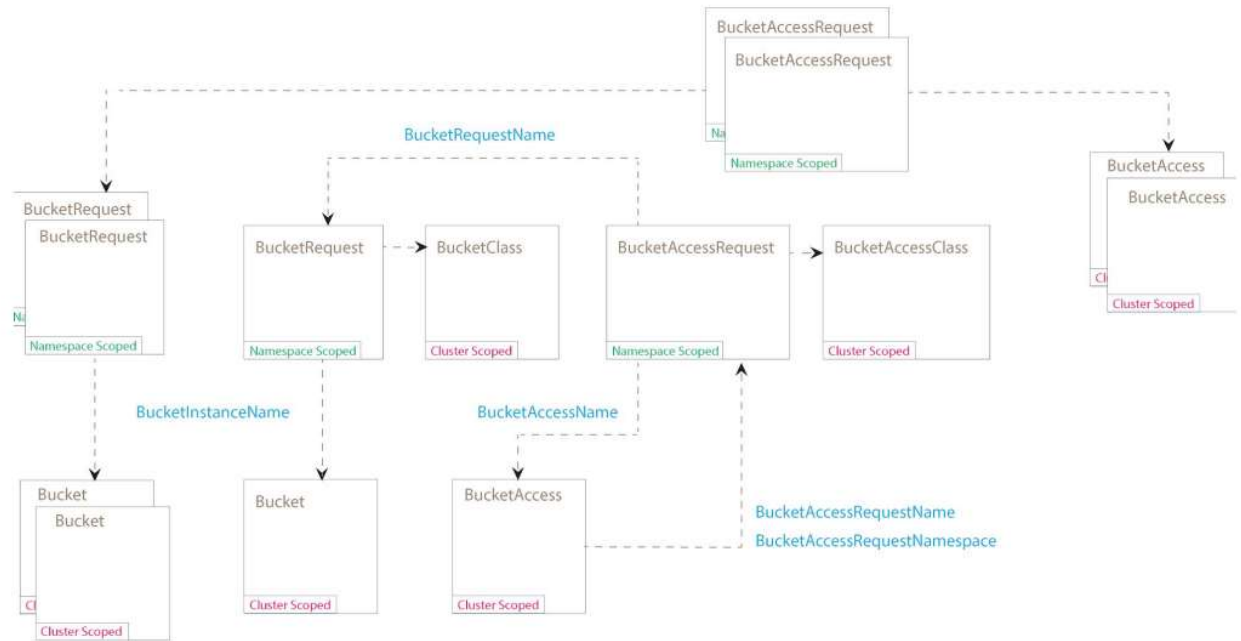
API

Object storage can't use the block and file storage primitives used in CSI. The unit of provisioning in object storage is a bucket (not a volume) and buckets are not mounted. Rather, they are accessed over the network. Moreover, object storage allows for more granular access control.

COSI introduces a set of new resources to work with buckets to implement object storage abstraction:

- **BucketClass:** A cluster-scoped resource containing fields defining the provisioner and a parameter set for configuring new buckets.
- **BucketRequest:** A namespaced resource representing a request for a new backend bucket or access to an existing bucket.
- **Bucket:** A cluster-scoped resource referenced by a BucketRequest and containing connection information and metadata for a backend bucket.
- **BucketAccessClass:** A cluster-scoped resource containing fields to specify policies that may be used to access buckets.
- **BucketAccessRequest:** A namespaced resource representing a request for access to an existing bucket.
- **BucketAccess:** A cluster-scoped resource for granting bucket access.

COSI Object Relationships



Self-Service

Other than providing kubernetes-API driven bucket management, COSI also aims to empower DevOps personnel to provision and manage buckets on their own, without admin intervention. This, further enabling dev teams to realize faster turn-around times and faster time-to-market.

COSI achieves this by dividing bucket provisioning steps among two different stakeholders, namely the administrator (admin), and the cluster operator. The administrator will be responsible for setting broad policies and limits on how buckets are provisioned, and how access is obtained for them. The cluster operator will be free to create and utilize buckets within the limits set by the admin.

For example, a cluster operator could use an admin policy could be used to restrict maximum provisioned capacity to 100GB, and developers would be allowed to create buckets and store data up to that limit. Similarly for access credentials, admins would be able to restrict who can access which buckets, and developers would be able to access all the buckets available to them.

Portability_

The third goal of COSI is to achieve vendor neutrality for bucket management. COSI enables two kinds of portability:

- Cross Cluster
- Cross Provider

Cross Cluster portability is allowing buckets provisioned in one cluster to be available in another cluster. This is only valid when the object storage backend itself is accessible from both clusters.

Cross-provider portability is about allowing organizations or teams to move from one object storage provider to another seamlessly, and without requiring changes to application definitions (PodTemplates, StatefulSets, Deployment and so on). This is only possible if the source and destination providers use the same data.

COSI does not handle data migration as it is outside of its scope. In case porting between providers requires data to be migrated as well, then other measures need to be taken to ensure data availability.

COSI Adoption

COSI adoption is in its infancy, but key vendors are making strides in driver development and support. Increased participation is anticipated as vendors refine their Kubernetes integration and overcome current limitations. Pure Storage's Portworx® Enterprise 2.12 added COSI support in late 2022. In October 2023, Dell unveiled a COSI driver, empowering containerized applications to dynamically provision and manage Dell Object Scale Storage. Notably, Netapp® has not yet announced production-level support for COSI integration at this time. The future holds promise for COSI expansion within the container ecosystem, with ongoing advancements by industry leaders.

Modern Day Applications

Modern day applications have evolved significantly from early desktop software and client-server models. The rise of the internet, mobile, and cloud computing led to new distributed application architectures, rapid release cycles, and data-driven products and services. Let's look into the history of modern applications, their importance today, key requirements, and the role object storage plays in enabling these next-generation workloads.

History of Modern Applications

In the early days of computing, applications were standalone software running locally on desktops and accessed data on the local disk. With the rise of networking and client-server architecture in the 1980s-90s, applications shifted to a distributed model with presentation layer on client devices and business logic and data on servers. The emergence of service-oriented architecture (SOA) in the 1990s led to decomposition of monolithic applications into reusable services that can interoperate to deliver functionality. The internet boom and web applications brought about multi-tier architectures and concepts like application servers and web services. The 2000s saw the rise of agile software development and DevOps culture which focused on faster iterations, continuous delivery, and infrastructure automation. Mobile and cloud computing led to new application models like software-as-a-service (SaaS). Microservices, containers, serverless computing and distributed data systems have become mainstream in the 2010s, enabling modern scalable and resilient applications. The shift from monoliths to microservices and event-driven architectures has been key to enabling rapid feature development.

Importance of Modern Applications

Modern applications are critical today for organizations to

- Enable digital transformation and adapt to technological disruptions.
- Improve customer experience and engagement across web, mobile, IoT etc.
- Quickly build innovative products, features, and services.
- Leverage data and analytics to generate business insights.
- Support distributed teams and agile development practices.
- Dynamically scale applications to meet fluctuating demand.
- Withstand failures and ensure high availability.

Requirements of Modern Applications

Some key requirements of modern applications include:

- **Microservices and APIs** - Decompose monoliths into independently deployable services to enable agility. Expose APIs for interoperability.
- **Containers and orchestration** - Package and deploy applications using containers and manage using orchestrators like Kubernetes.
- **Cloud-native design** - Build applications optimized for horizontal scalability and resilience on cloud platforms.
- **DevOps culture** - Adopt agile, lean, and continuous practices across teams to enable rapid releases.
- **Modular and distributed architecture** - Design modular components with loose coupling and high cohesion. Distribute across nodes.
- **Data-driven** - Incorporate variety of data sources. Apply analytics and machine learning.
- **Observability** - Monitor, trace and get insights into distributed applications.
- **Security** - Build in security across people, processes, and technology. Address increased surface area.

Role of Object Storage in Modern Applications

Object storage systems play a key role in modern application architectures:

- Provide highly scalable, durable, and cost-efficient storage for unstructured data like images, videos, logs and documents.
- Enable building data lakes for analytics use cases.
- S3-compatible APIs allow easy integration from various environments and frameworks.
- Built-in features like encryption, access control and replication provide data security and protection.
- Pay-as-you-go pricing matches fluctuating application demands. In summary, object storage is a critical data infrastructure component to build, deploy and run modern data-driven applications. It allows developers to focus on building application logic rather than storage management.

Object Storage Use Cases

In today's data-driven landscape, object storage has become a pivotal technology, finding diverse applications across various industries and scenarios.

Object Storage For Backup & Disaster Recovery

Object storage emerges as a prime candidate for backup and disaster recovery (DR) purposes, owing to its exceptional scalability, durability, accessibility, and integration capabilities. It empowers secure and cost-efficient data protection solutions across hybrid and multi-cloud environments.

The primary advantages of leveraging object storage for backup and DR encompass virtually boundless scalability, robust data durability, geographical dispersion of data, and substantial cost savings, notably in obviating the need for dedicated backup infrastructure.

Nevertheless, it's important to acknowledge certain challenges, including potential latency concerns when retrieving data, the absence of real-time synchronization, and the intricacies involved in managing object storage across various cloud providers.

Adhering to best practices is paramount in optimizing backup and DR with object storage. This includes adhering to the 3-2-1 rule for backup copies, logically air gapping backups, enforcing least privilege access controls, leveraging immutable backups, and implementing data encryption measures.

Object Storage For DATA LAKEHOUSE

A data lakehouse represents a fusion of data warehouse and data lake features within a unified architectural framework, offering the robust governance, performance, and reliability typically associated with data warehouses, while retaining the adaptability, agility, and scalability inherent in data lakes. Object storage has emerged as the preferred choice for housing data within data lakehouses due to its cost-effective handling of extensive unstructured data volumes.

There are several compelling reasons for object storage's suitability within data lakehouse setups. Firstly, it accommodates data in its native form, eliminating the need for upfront schema imposition, a characteristic often associated with traditional data warehouses. This feature aligns perfectly with the diverse range of structured, semi-structured, and unstructured data types commonly encountered in data lakehouses. Furthermore, object storage's capability for independent scaling of storage and compute resources facilitates optimized performance and cost management. Its robust design, tailored for web-scale workloads, ensures not only massive scalability but also data durability, making it an economical choice for storing vast data quantities within a data lakehouse.

Additionally, object storage's reliance on open APIs and formats mitigates the risk of vendor lock-in, which is often a concern with data warehouses. Utilizing object storage within a data lakehouse architecture yields several key advantages, including reduced storage costs, scalability to accommodate data growth without performance degradation, flexibility for adding new data sources without necessitating schema alterations, data durability through replication, and direct analysis of data in its native format, bypassing ETL (Extract, Transform, Load) overhead.

However, it's important to acknowledge some challenges associated with object storage in a data lakehouse. Object storage may exhibit lower performance compared to block storage for transactional workloads, requiring careful tuning. It also lacks built-in features like metadata management typically found in data warehouses, necessitating the use of additional tools. The persistence of deleted objects can impact performance until garbage collection occurs, and the presence of a large number of small files may degrade performance, necessitating optimization strategies. Despite these challenges, object storage remains a compelling choice for data lakehouse environments, enabling the seamless integration of structured and unstructured data while optimizing cost efficiency and scalability.

Object Storage For Unlimited Databases

Unlimited storage databases employ a distinctive approach, decoupling storage from computing resources by housing bulk data in cost-effective object storage. This separation enables databases to independently scale their storage capacity, eliminating the previous constraints imposed by local storage limitations.

In this setup, data is cached locally to ensure optimal performance, while the bulk of the data resides in object storage. To maintain efficient data access, Least Recently Used (LRU) cache replacement policies are employed to keep frequently accessed data locally available. Leading-edge databases like MongoDB®, CockroachDB®, SingleStore®, YugabyteDB®, and MariaDB® have embraced the concept of unlimited storage through object storage, providing numerous advantages, including nearly boundless scalability, integrated backup solutions, and point-in-time recovery capabilities. Unlimited storage has become a standard feature for cloud-native databases.

The fundamental design principle of unlimited storage databases revolves around harnessing object storage as a foundational data repository. This strategic choice delivers the scalability and cost-efficiency inherent to object storage, while preserving the high-performance characteristics of a local database. Data can be seamlessly cached locally and retrieved from object storage on demand.

Nonetheless, it's important to acknowledge some limitations inherent to this architectural approach. Initial data access may exhibit slower response times, as data retrieval from object storage may introduce latency. Achieving strong data consistency can also pose challenges, particularly in scenarios where the underlying object store follows an eventually

consistent model. Furthermore, specialized databases optimized for Online Transaction Processing (OLTP) workloads may not readily support object storage as their backend storage solution.

So object storage aligns naturally with the objectives of emerging cloud-native databases striving for limitless storage scalability. These technologies complement each other seamlessly, presenting a compelling solution for modern data management needs.

Object Storage for DB as a Query Engine

A growing trend among databases is to adopt object storage as their primary data repository, relegating the responsibilities of data querying and processing. This strategic shift empowers databases to concentrate on their core expertise: enhancing query optimization, all the while harnessing the boundless scalability offered by object storage.

By leveraging object storage, databases gain the capability to enhance query performance significantly, capitalizing on the limitless storage capacity it provides. Additionally, this approach obviates the need for labor-intensive Extract, Transform, Load (ETL) processes to load data into the database, streamlining data integration workflows.

Moreover, it enables databases to seamlessly access a multitude of diverse data sources without the necessity for data replication. This flexibility allows databases to prioritize their efforts on fine-tuning query performance rather than getting entangled in the intricacies of storage management.

As an added benefit, this paradigm shift can lead to substantial reductions in infrastructure and licensing costs, thanks to the cost-effectiveness of object storage. In essence, the adoption of object storage as a primary data store for databases represents a strategic move that enhances efficiency, scalability, and cost-effectiveness.

Object Storage For Streaming Data Platforms

Object storage offers limitless scalability and cost advantages for the storage of extensive volumes of unstructured data, making it an excellent choice for constructing data lakes. Nevertheless, handling streaming workloads necessitates specialized ingestion and processing capabilities.

While object storage can serve as a means to persist streams, it entails the conversion of streams into files and repetitive uploads. This approach may result in the creation of numerous small files and inefficient access, particularly when lacking stream processing capabilities.

Dedicated streaming platforms are better optimized for the high-volume ingestion and real-time processing of streams. Object storage can complement these streaming platforms by providing durable, long-term storage solutions.

So essentially, object storage delivers scalability and cost-efficiency benefits for data lakes, whereas streaming platforms excel in the ingestion and real-time processing of streaming data. These two components can be seamlessly integrated within a modern data architecture to harness the strengths of both paradigms.

Object Storage For Machine Learning and AI

Object storage stands as an ideal choice for machine learning and AI workloads, driven by its scalability, adaptability to unstructured data, and capacity to manage extensive metadata loads.

The key merits of object storage encompass nearly boundless scalability, impressive throughput performance, support for customizable metadata, and a cost-efficient approach. Its design facilitates seamless scalability, allowing for easy initiation at a smaller scale and effortless expansion as storage requirements grow. This scalability minimizes waste and reduces the total cost of ownership.

The S3 API has emerged as the prevailing standard for object storage, facilitating smooth integration with ML/AI platforms such as TensorFlow® and Spark®. Object storage excels at housing vast datasets required for model training and serves as a reliable repository for archiving trained models, bolstered by data protection through erasure coding and geo-replication.

However, it's worth noting that object storage may encounter potential latency challenges in real-time workloads. For low-latency and transactional workloads, file and block storage may present a more suitable alternative.

Significance of Object Storage and COSI in Data Mobility and Data Gravity

In our contemporary world of constantly expanding data, the issues of data gravity and data mobility are closely intertwined challenges when it comes to handling extensive datasets. Let's delve deeper to explore how object storage and COSI are effectively addressing and mitigating these challenges.

Data Gravity

Data gravity is the phenomenon where large datasets attract applications, services, and additional data. The larger the dataset, the greater its gravitational pull. This is because applications need to be located close to the data to achieve good performance and low latency. As more data and services cluster around large datasets, moving the data becomes difficult and expensive - the data becomes "heavy".

Data Mobility

Data mobility refers to the ability to efficiently move data between locations, systems, and applications. Good data mobility means being able to access, integrate, migrate, and analyze data across multiple environments with minimal latency or expense. Data gravity makes achieving data mobility more challenging. The larger the dataset, the more difficult and costly it becomes to move.

Object Storage & Data Gravity

Object storage is a scalable, cost-effective cloud storage solution optimized for storing and managing massive amounts of unstructured data. Object storage helps mitigate data gravity in several ways:

- It allows unlimited scalability, so storage can expand as data grows.
- Data can be distributed globally, bringing it closer to applications.
- Data accessibility is improved through standard HTTP-based APIs.
- Movement of objects between locations is fast and inexpensive. By distributing data globally while maintaining accessibility, object storage makes it easier to overcome data gravity.

COSI & Data Mobility

COSI improves data mobility by:

- Providing a consistent interface to access data across object storage systems.
- Allowing easy movement of data between object storage platforms.

- Enabling hybrid cloud architectures by integrating on-prem and cloud object storage. By abstracting the object storage implementation from applications, COSI makes it easier to move data and mitigate data gravity.

Databases on Object Storage

Databases on object storage represent a paradigm shift in data management, offering a robust and scalable solution for storing structured and semi-structured data in a highly flexible and cost-effective manner. Let's talk about a few of them.

MinIO®

MinIO stands as an open-source, high-performance object storage system, meticulously designed for the efficient storage of unstructured data on a grand scale. It organizes data within a structured framework comprising "buckets" and "objects." In this context, buckets serve as top-level directories, while objects encapsulate the actual data alongside associated metadata.

What sets MinIO apart is its remarkable ability to seamlessly manage objects at scale, all without the need for explicit partitioning. This feat is achieved through the adoption of innovative concepts such as micro-partitions, data clustering, and a columnar storage format, all geared toward optimizing the storage and retrieval processes.

MinIO incorporates robust data governance capabilities, including object locking, retention policies, and lifecycle management. These features empower users with valuable functionalities like data immutability, automated state transitions, and data expiration, facilitating effective control and management of their stored information.

Notably, MinIO is engineered to meet the demands of large-scale, multi-data center cloud storage services, offering the flexibility of independent tenant clusters. This architectural flexibility makes it a compelling choice for organizations seeking robust, scalable, and secure object storage solutions in the ever-evolving landscape of data management.

Snowflake®

Snowflake's storage layer relies on cloud object storage to house streamlined, compressed data objects effectively. In this architecture, compute and storage operate as distinct entities. SQL query processing is performed atop the storage layer through virtual warehouses, facilitating the autonomous scaling of both storage and compute resources.

Importantly, Snowflake seamlessly oversees the management of the storage layer, ensuring transparency for users. The genius of Snowflake's architecture lies in its fusion of shared disk (centralized data) and shared nothing (independent compute) paradigms. This harmonious blend yields numerous advantages, including exceptional concurrency, scalability, and

performance—qualities that traditional data warehouses struggle to match, especially when dealing with analytical workloads.

Polybase In SQL Server®

Polybase In SQL Server extends the capabilities of T-SQL by enabling seamless querying of data stored in Hadoop clusters or Azure Blob Storage. It leverages robust, scalable object storage systems such as Hadoop HDFS and Azure Blob Storage as external data sources, expanding the reach of your data.

Polybase empowers you to perform queries that seamlessly integrate external data with relational data, facilitating comprehensive analytics that encompass both sources. Additionally, it optimizes processing by offloading computation to Hadoop, harnessing its scalable processing power right where the data resides.

Furthermore, Polybase abstracts the intricacies of object storage implementation from the querying layer, simplifying your data access and analysis workflows.

Competitors

In the fiercely competitive world of data storage solutions, object storage has emerged as a contender that has disrupted traditional storage models. As competitors vie for dominance in this rapidly evolving field, they strive to offer innovative features and services that leverage the inherent advantages of object storage, such as scalability, cost-efficiency, and flexibility, to meet the evolving data storage needs of organizations across the globe.

DELLEMC®

Dell EMC® ECS stands as Dell's premier object storage platform, delivering enterprise-grade S3 compatibility, top-tier security, impressive scalability, and cost-effective solutions.

ECS can be deployed in various forms, including software, purpose-built appliances, or as a service. Key hardware options encompass the EX500, EX5000, and the high-performance all-flash EXF900.

In August 2022, ECS 3.8 was introduced, introducing pivotal features such as data mobility, enhanced security measures including air gapping, and expanded storage capacity, enhancing its capabilities significantly.

ECS boasts the capability to scale exabytes, catering to contemporary workloads like AI/ML, and ensuring compliance adherence. Remarkably, it offers the potential for up to a 76% reduction in Total Cost of Ownership (TCO) compared to public cloud object storage solutions.

Dell positions ECS as a compelling alternative to public cloud object storage, presenting cloud-like scalability and S3 compatibility while affording users the control and cost benefits of a private cloud environment.

NetApp®

NetApp StorageGRID represents NetApp's software-defined solution for object storage, designed with a core focus on scalability, security, and seamless cloud integration.

In May 2022, StorageGRID 11.7 was unveiled, introducing an array of security enhancements, including the governance mode for S3 Object Lock and automated upgrade verification, bolstering its security features.

StorageGRID seamlessly supports hybrid cloud configurations through its native integration capabilities with public cloud storage services like AWS S3. Furthermore, NetApp offers E-Series and EF-Series all-flash storage appliances that can effectively serve as object storage solutions, providing flexibility to users.

To ensure continued investment protection, NetApp offers the Storage Lifecycle Program, which includes StorageGRID upgrades at regular intervals of every 3 years, allowing users to stay current with evolving technology and features.

Pure Storage®

Pure Storage FlashBlade emerges as a high-performance, all-flash object storage solution, meticulously crafted to excel with contemporary workloads.

Specifically, FlashBlade//S is tailor-made to cater to analytics and AI/ML demands, offering ultra-low latency performance. Its seamless integration with Nvidia DGX servers and Snowflake further optimizes its suitability for these types of workloads.

In a 2022 survey conducted by Pure Storage®, it was revealed that 52% of IT buyers are prioritizing investments in AI/ML. However, to fully harness the potential of these emerging technologies, they recognize the need to modernize their infrastructure.

Pure Storage positions FlashBlade as a crucial component of a modernized data experience, simplifying complexity and empowering users with real-time analytics capabilities.

Conclusion

Object storage has cemented itself as a transformative technology in the modern data landscape. Its unique architecture provides unmatched scalability, metadata flexibility, and cost efficiency for vast amounts of unstructured data. Object storage empowers next-generation workloads like data analytics, AI/ML model training, and cloud-native applications. The COSI specification promises to expand integration into containerized environments through standardized abstractions.

Leading enterprise vendors offer robust object storage platforms with enterprise-grade durability, security, and performance. While adoption is still expanding, object storage undoubtedly disrupts traditional storage paradigms. Its web-scale capabilities align perfectly with contemporary data-driven applications.

As data volumes continue their unfettered growth, object storage emerges as the foremost solution for limitlessly scalable and economical data repositories. Its versatility caters to diverse modern workloads, from data lakes to streaming pipelines and machine learning. Object storage flexibly adapts to shifting requirements and runs the gamut from on-prem deployments to multi-cloud configurations. Its future shines bright as the foundational storage layer for the era of big data.

References

<https://objectfirst.com/blog/what-is-object-storage/>
<https://www.synopsys.com/cloud/insights/what-is-object-storage.html>
<https://davidalmazamendi.com/intro-to-data-lakehouses/>
<https://kubernetes.io/blog/2022/09/02/cosi-kubernetes-object-storage-management/>
<https://thenewstack.io/beyond-block-and-file-cosi-enables-object-storage-in-kubernetes/>
<http://storagegaga.com/project-cosi/>
<https://docs.singlestore.com/db/v8.1/manage-data/local-and-unlimited-database-storage-concepts/>
<https://www.datanami.com/2023/01/03/are-databases-becoming-just-query-engines-for-big-object-stores/>
<https://www.projectpro.io/article/snowflake-architecture-what-does-snowflake-do/556>
<https://min.io/>
<https://www.dell.com/en-us/dt/learn/data-storage/object-storage.htm>
<https://www.netapp.com/blog/storagegrid-object-storage-platform/>
<https://www.purestorage.com/products/unstructured-data-storage/flashblade-s.html>
<https://volumes.blog/2023/10/02/dell-objectscale-cosi-driver/>
<https://www.netapp.com/data-storage/storagegrid/>
<https://portworx.com/blog/portworx-scale-out-object-storage/>