Redbooks
ibm.com/redbooks

IBM Cloud Object Storage Concepts and Architecture

System Edition

Bradley Leonard
Hao Jia
Johan Verstrepen
Jussi Lehtinen
Lars Lauber
Patrik Jelinko
Vasfi Gucer

Storage

IBM

Redpaper

Introduction

Object Storage is the primary storage solution that is used in the cloud and on-premises solutions as a central storage platform for unstructured data. Object Storage is growing more popular for the following reasons:

- It is designed for exabyte scale.
- It is easy to manage and yet meets the growing demands of enterprises for a broad set of applications and workloads.
- It allows users to balance storage cost, location, and compliance control requirements across data sets and essential applications.

IBM® Cloud Object Storage (IBM COS) system provides industry-leading flexibility that enables your organization to handle unpredictable but always changing needs of business and evolving workloads.

IBM COS system is a software-defined storage solution that is hardware aware. This awareness allows IBM COS to be an enterprise-grade storage solution that is highly available and reliable and uses commodity x86 servers. IBM COS takes full advantage of this hardware awareness by ensuring that the server performs optimally from a monitoring, management, and performance perspective.

The target audience for this IBM Redpaper publication is IBM COS architects, IT specialists, and technologists.

Summary of changes in this new revision (REDP5537-01):

This paper is the new edition of the paper *IBM Cloud Object Storage Concepts and Architecture*, REDP5537-00, that was originally published on May 29, 2019. The following new information is included in this revision:

- Retention enabled buckets
- Zone Slice Storage (ZSS)
- Performance improvements for SecureSlice Encryption
- A table on advantages and disadvantages of the mirror modes
- Hard synchronous option in mirror write threshold functionality
- Container and vault mode comparison table
- Data security enhancements
- Object expiration

IBM COS includes a rich set of features to match various use cases. Figure 1 on page 2 shows IBM COS main features and typical use cases.

© Copyright IBM Corp. 2020. All rights reserved.

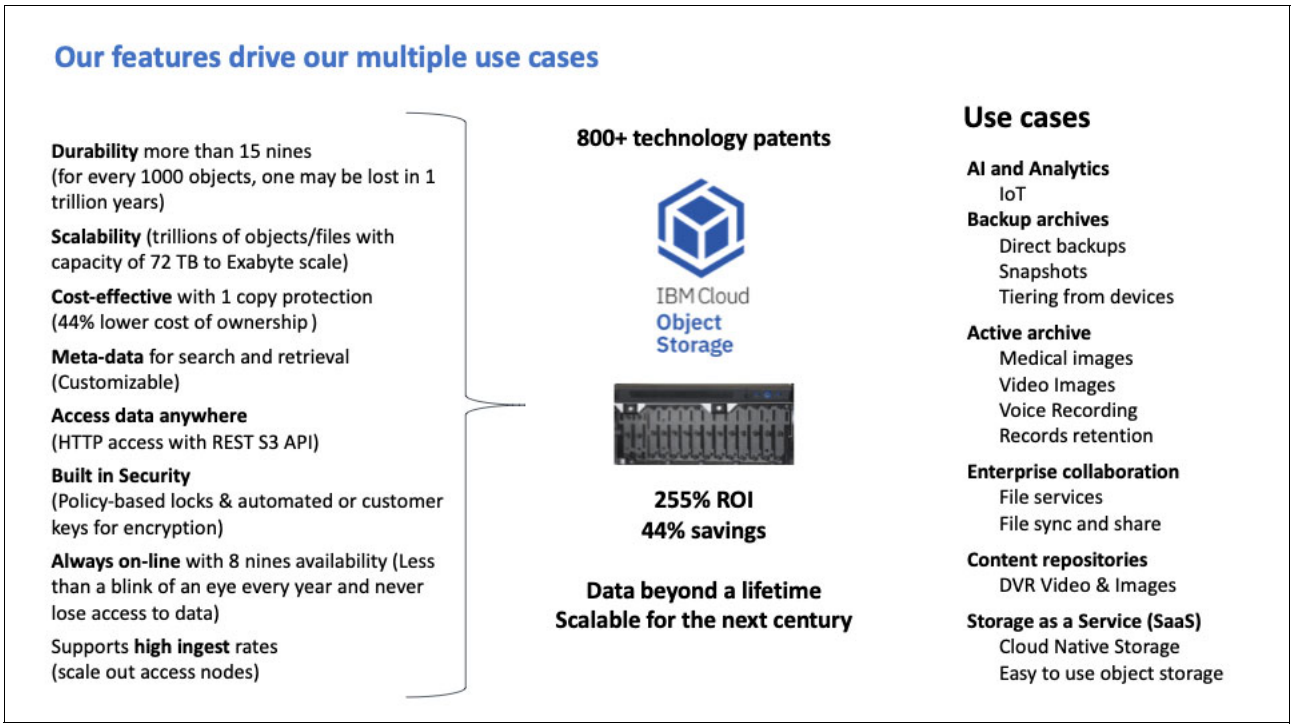

Figure 1 IBM COS main features and typical use cases

IBM validated more than 80 IBM and third-party applications with IBM COS and created extensive technical documents that describe their interoperability.

Validated applications per use case included the following examples:

- Backup:
  - IBM Spectrum® Protect, IBM Spectrum Protect Plus
  - Commvault
  - Veritas NetBackup
  - Rubrik
  - Veeam
  - Actifio
- Active archive:
  - Komprise
  - Veritas Enterprise Vault
  - Moonwalk
- Enterprise file services:
  - IBM Aspera®
  - Ctera
  - Nasuni
  - Panzura
- Enterprise content management:
  - IBM Filenet
  - IBM Content Manager On Demand
- Other:
  - IBM Spectrum Scale
  - Apache Spark

- Merge Healthcare®
- Splunk
- Nice

Tip: Most applications that support S3 API can use IBM COS for storage.

For more information, see the IBM Cloud® Object Storage web page.

Differences between block, file, and Object Storage

The main difference between block, file, and Object Storage is who accesses the data:

- Block storage

  Block storage is visible to operating systems or hypervisors that are running on a bare metal server. Operating systems write blocks of data on to disk tracks and sectors.

- File storage

  File storage is often visible directly to users in the form of a directory by way of SMB or NFS storage protocol. Users must decide and know where to store files and remember where to find them.

- Object Storage

  Object Storage is accessed directly from applications by way of RESTful API. An object is stored in a flat namespace with all other objects in the same namespace. An object name is used to write and read objects from Object Storage.

  Object Storage provides the capability to add custom metadata to application data.

Figure 2 shows the differences between block, file, and Object Storage.

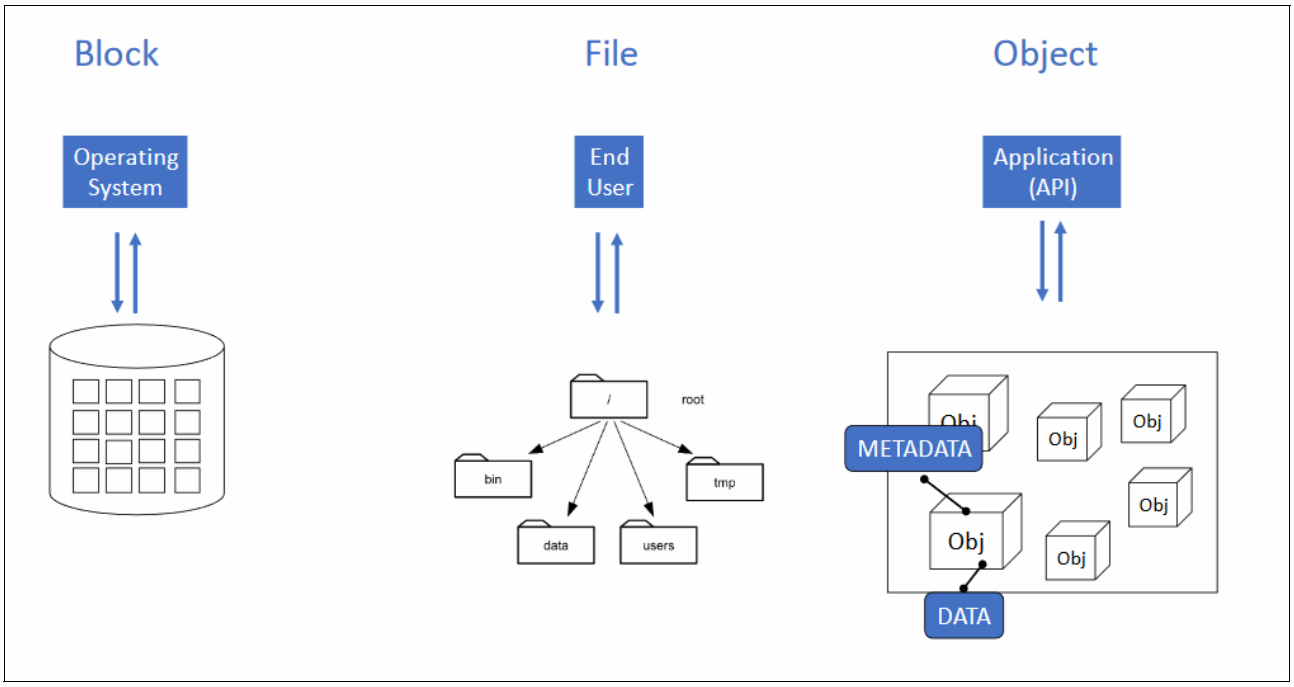

Figure 2 Differences between block, file, and Object Storage

Use cases for Object Storage

Typical application use cases for IBM COS across industries include the following examples:

- Analytics, artificial intelligence, and machine learning data repository; for example, Hadoop and Spark data lakes.
- IoT data repository; for example, Sensor data collection for autonomous driving.
- Secondary storage; for example:
  - Active archive: Tiering of inactive data from primary NAS filers.
  - Storage for backup data: Leading backup applications have native integration with Object Storage for longer term retention purposes.
- Storage for cloud native applications: Object Storage is the de-facto standard for cloud native applications.
- Storing data in tabular parquet format: Provides significant data compression and allows query operations against the data.

Industry-specific use cases for IBM COS include the following examples:

- Healthcare and Life Sciences:
  - Medical imaging, such as picture archiving and communication system (PACS) and magnetic resonance imaging (MRI)
  - Genomics research data
  - Health Insurance Portability and Accountability Act (HIPAA) of 1996 regulated data
- Media and entertainment; for example, audio and video
- Financial services; for example, regulated data that requires long-term retention or immutability

For information about use cases, see *IBM Cloud Object Storage System Product Guide*, SG24-8439.

Flexible deployment options of IBM Cloud Object Storage

IBM COS is available in the following modes:

- On-premises Object Storage:
  - IBM hardware appliances with IBM COS software
  - IBM certified third-party x86 servers with IBM COS software
- Public cloud Object Storage (multi-tenant)

Figure 3 on page 5 shows the various deployment options for IBM COS.

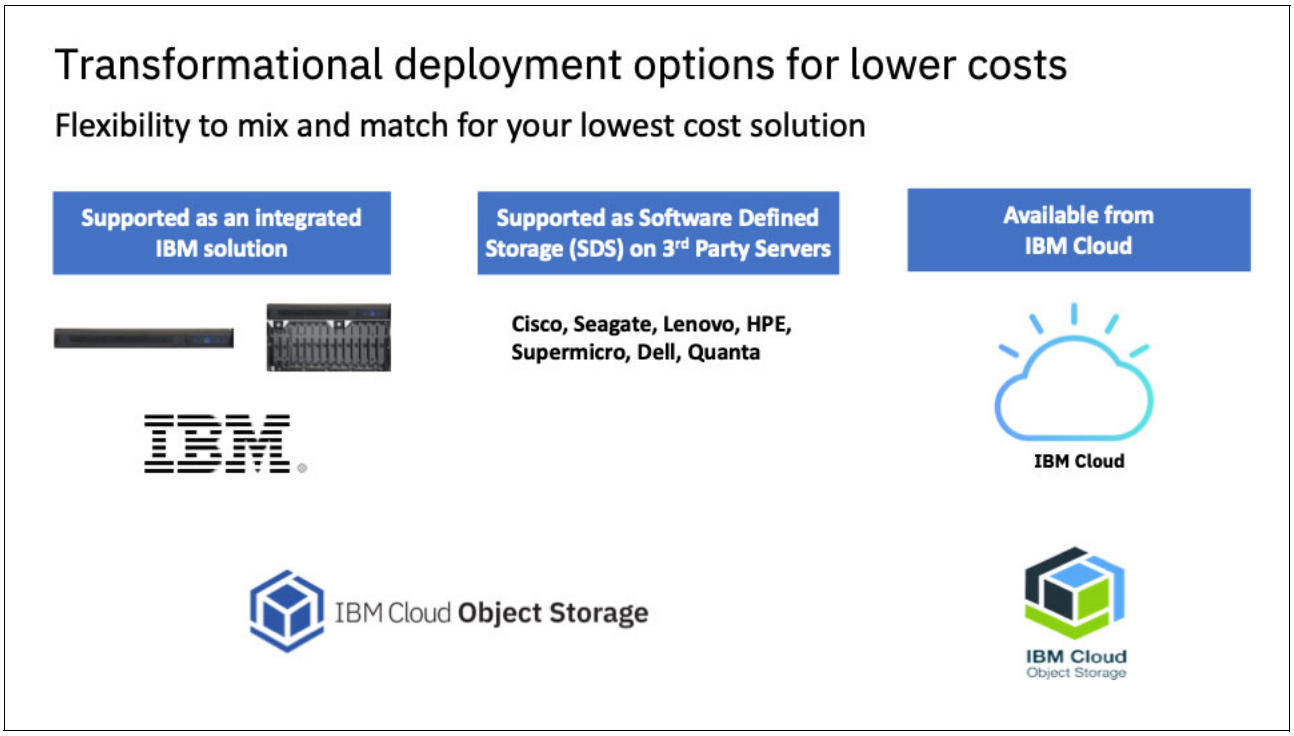

Figure 3 IBM COS deployment options

Note: IBM COS Software is available in several licensing models, including perpetual, subscription, or consumption.

This IBM Redpaper publication explains the architecture of IBM Cloud Object Storage *on-premises* offering and the technology behind the product. For more information about the IBM Cloud Object Storage use case scenarios and deployment options, see *IBM Cloud Object Storage System Product Guide*, SG24-8439.

For more information about the IBM Cloud Object Storage *public cloud offering*, see the following publications:

- *Cloud Object Storage as a Service: IBM Cloud Object Storage from Theory to Practice*, SG24-8385
- *How to Use IBM Cloud Object Storage When Building and Operating Cloud Native Applications*, REDP-5491

IBM Cloud Object Storage architecture

IBM COS is a dispersed storage system that uses several storage nodes to store pieces of the data across the available nodes. IBM COS uses an Information Dispersal Algorithm (IDA) to break objects into encoded and encrypted slices that are then distributed to the storage nodes.

No single node has all of the data. This configuration makes it safe and less susceptible to data breaches while needing only a subset of the storage nodes to be available to retrieve the stored data. This ability to reassemble all the data from a subset of the slices dramatically increases the tolerance to node and disk failures.

The IBM COS architecture is composed of the following functional components. Each of these components runs IBM COS software that can be deployed on certified, industry standard hardware:

- IBM Cloud Object Storage Manager

  IBM Cloud Object Storage Manager provides a management interface that is used for administrative tasks, such as system configuration, storage provisioning, and monitoring the health and performance of the system.

  The Manager can be deployed as a physical appliance, VMware virtual machine, or Docker container.

- IBM Cloud Object Storage Accesser® node

  IBM Cloud Object Storage Accesser node encrypts and encodes data on write and decodes and decrypts it on read. It is a stateless component that presents the storage interfaces to the client applications and transforms data by using an IDA.

  The Accesser node can be deployed as a physical appliance, VMware virtual machine, Docker container, or can run as an embedded Accesser node on the IBM Slicestor® appliance.

- IBM Cloud Object Storage Slicestor node

  The IBM Cloud Object Storage Slicestor node is responsible for storing the data slices. It receives data from the Accesser node on write and returns data to the Accesser node as required by reads. The Slicestor also ensures the integrity of the saved data and rebuilds if necessary.

  Slicestor nodes are deployed as physical appliances.

Figure 4 shows a simple architecture layout of the different components in IBM COS.

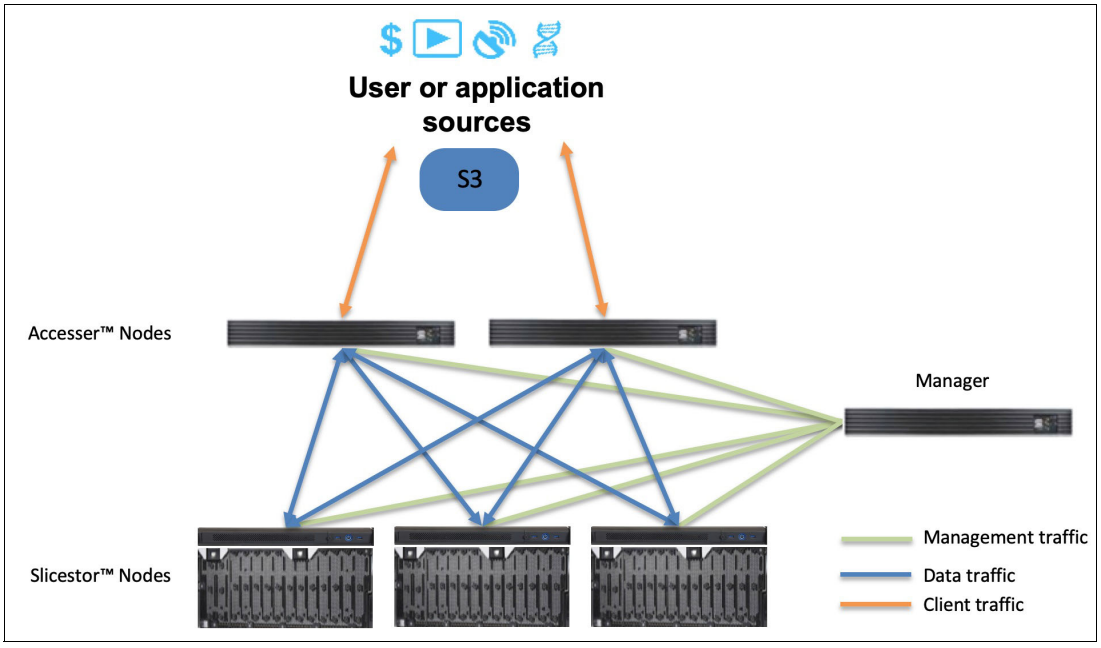

Figure 4 IBM Cloud Object Storage architecture

S3 interface: IBM COS uses the S3 interface for all storage operations; for example:

- PUT: Writes an object to the storage.
- GET: Reads an object from the storage.
- DELETE: Deletes an object from storage.
- LIST**:** Lists objects that are in a bucket.

All API calls are issued against an IBM COS Accesser node.

Core concepts

This section provides information about IBM COS core concepts. Figure 5 shows the major IBM COS logical concepts.

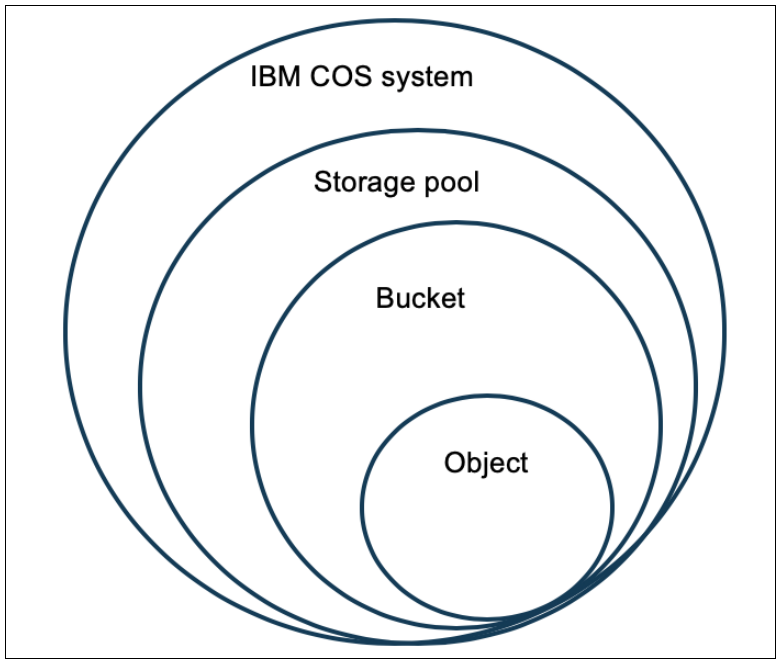

Figure 5 IBM Cloud Object Storage logical concepts

Device sets

IBM COS uses the concept of *device sets* to group Slicestor devices (see Figure 6 on page 8). Each device set consists of several Slicestor devices.

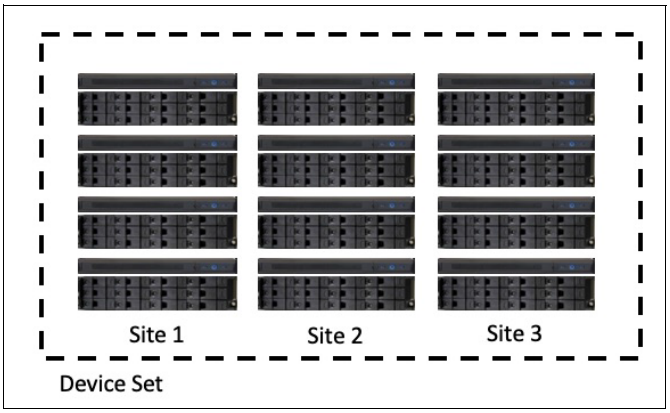

Figure 6 Device set: A set of Slicestor devices

Device sets can be spread across one or multiple data centers. All Slicestor nodes in one device set must have the same configuration (Slicestor node model, number of drives, and drive size).

Storage pools

A *storage pool* consists of one or more device sets that can be spread across multiple data centers, as shown in Figure 7.

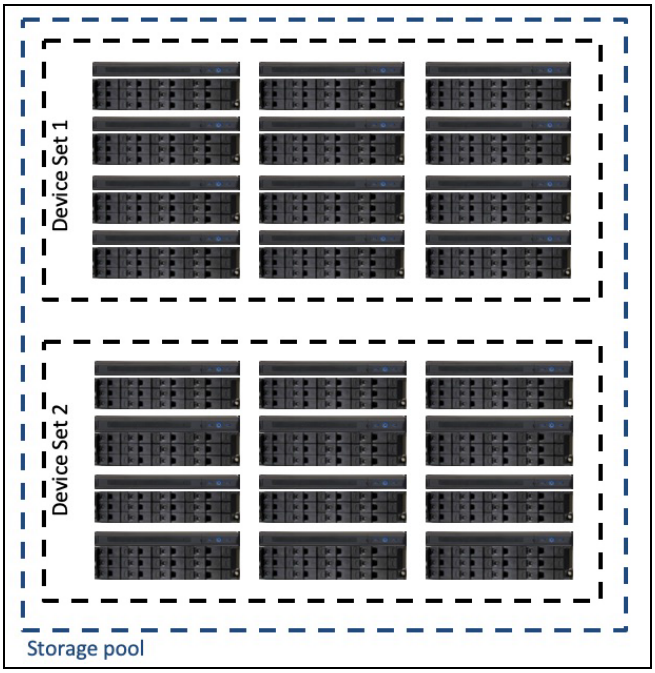

Figure 7 IBM Cloud Object Storage storage pools

Device sets in a storage pool can have different configurations. This configuration enables adding newer Slicestor nodes to a system without replacing older Slicestor nodes.

Note: Storage pool expansion must follow specific rules. For more information, see *IBM Cloud Object Storage System Product Guide*, SG24-8439.

Vaults

Vaults are logical storage containers for data objects that are contained in a storage pool, as shown in Figure 8.

Important: A vault in IBM COS features the same functionality as an S3 bucket.

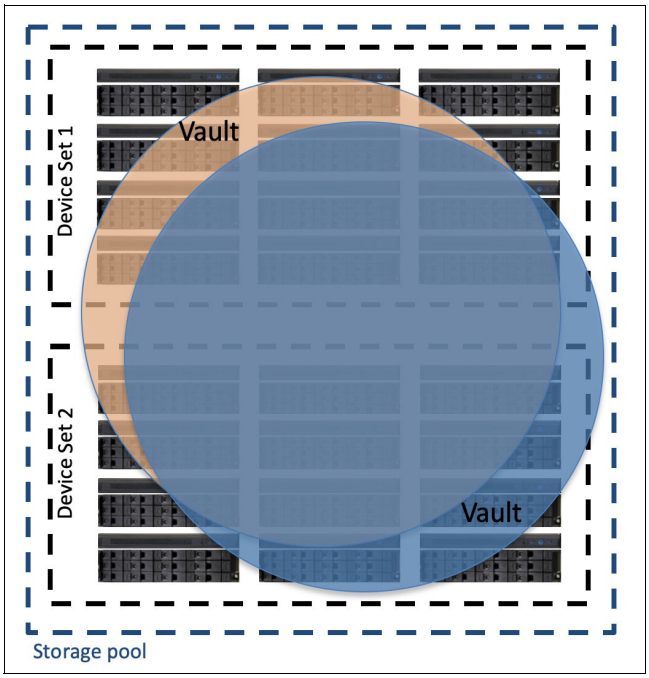

Figure 8 IBM Cloud Object Storage vault

Vaults are deployed on a storage pool and automatically spread across all the device sets. One or more vaults can be deployed to a storage pool.

Mirrored vaults

A vault that is on one storage pool can be mirrored to a vault on another storage pool, commonly in a different location. Both component vaults are controlled by a mirror and storage operations are issued against the mirror. All objects in the mirror are available on both vaults. This concept is usually seen in a two site deployment, but can be used for other use cases, such as *hub and spoke* design.

Recommended practice: Although vaults in a mirrored configuration can have different IDAs and protection settings, it is recommended to have the same usable capacity on both storage pools in a two site configuration.

A mirrored setup across two different sites protects the IBM COS system against a site failure. If one site is unavailable, reads and writes occur from the available vault automatically. A failover procedure is not required if the application can reach a functioning Accesser node at either site. A failback procedure is not required when the site comes back online.

Access pools

An *access pool* consists of one or more Accesser nodes, which present a vault to an application. More than one access pool can separate traffic or restrict access to certain vaults. This way, a tenant separation can be implemented.

The connection between access pools and vaults is a many-to-many connection. One vault can be deployed on many access pools and one access pool can have more than one vault deployed.

Information Dispersal Algorithm

The Information Dispersal Algorithm (IDA) is based on erasure coding and defines the reliability, availability, and storage efficiency of an IBM COS system. The IDA is defined at the vault level at the vault creation time. The IDA is written as *width/read threshold/write threshold*; for example: 12/6/8.

The IDA consists of the following components:

- Width: The *width* of the IDA is the total number of slices that is generated by erasure coding. For example, in a 12-wide storage pool all data has 12 slices.
- Read threshold: The *read threshold* of an IDA defines the number of slices of the width that must be available for the data to be readable. For example, if the read threshold of a 12-wide system is set to 6, the system needs only six slices to read the data.

  Tip: If the read threshold is set higher, the IBM COS system can survive fewer failures, but the storage efficiency is better.

- Write threshold: The *write threshold* of an IDA is the number of slices of the width that need to be written before the Accesser node returns the success to the client. *The write threshold always must be higher than the read threshold so that the data is available, even if a failure occurs right after the write is completed*. For example, if the write threshold of a 12-wide system is set to 8, the system musty successfully write eight slices to complete a write request.

  Tip: If the write threshold is set lower, the IBM COS system can survive more failures, but the storage efficiency suffers because of the higher redundancies.

Expansion factor: The expansion factor is calculated as the width divided by the read threshold. It also defines the ratio of raw capacity versus usable capacity. See Table 1 on page 12 for examples.

Dispersal modes

IBM COS can operate in two different dispersal modes, as shown in Figure 9.

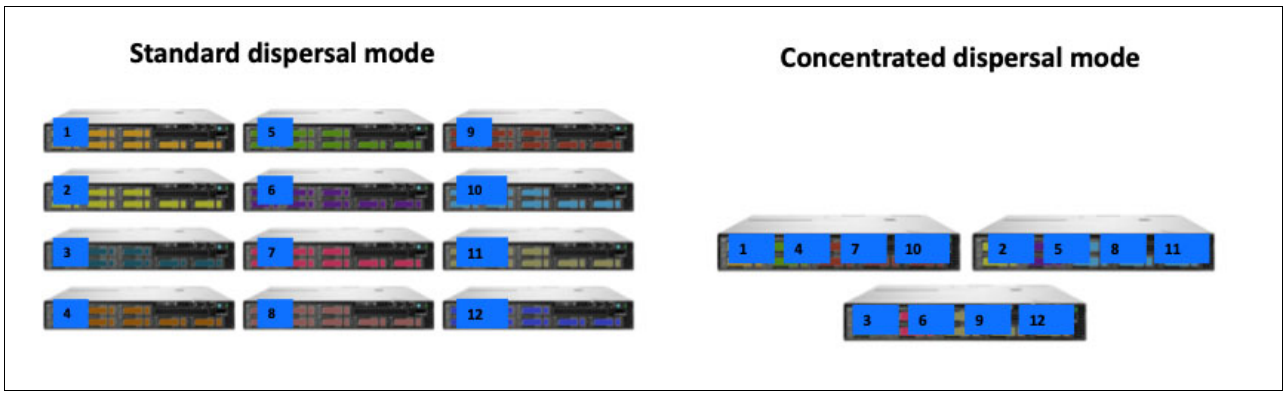

Figure 9 Dispersal modes in IBM COS

In Standard Dispersal Mode (SD Mode), which is also called non-Concentrated Dispersal Mode, each slice is written on a different Slicestor node. This mode ensures the highest performance and availability because one Slicestor node down means that only one slice is unavailable. The SD Mode is usually used in larger configurations and supported on systems with at least 12 nodes.

Note: SD Mode allows you to configure width, read, and write thresholds. For more information about the IDA configuration guidelines, see *IBM Cloud Object Storage System Product Guide*, SG24-8439.

In Concentrated Dispersal Mode (CD Mode), multiple slices of a single object segment are placed on a single Slicestor node, *but never on the same disk*. This mode enables cost efficient smaller systems starting from 72 terabytes to a few petabytes. If one Slicestor node goes down, more slices become unavailable. CD Mode is supported on systems starting with three Slicestor nodes.

Note: CD Mode defines preconfigured IDAs that are optimized for storage or performance.

Location options

IBM COS offers options to be deployed in one or more sites. A single site setup does not protect against a site failure, although it does provide the lowest possible overhead and the best latency. Two sites are typically set up as a mirrored configuration. IBM COS plays out all advantages in a minimum three sites geo-dispersed setup. Slicestor nodes are distributed across multiple sites for reliability and availability. In a geo-dispersed setup, IBM COS relies on a single copy of data that is protected by way of erasure coding against site failures.

The options are shown in Figure 10.

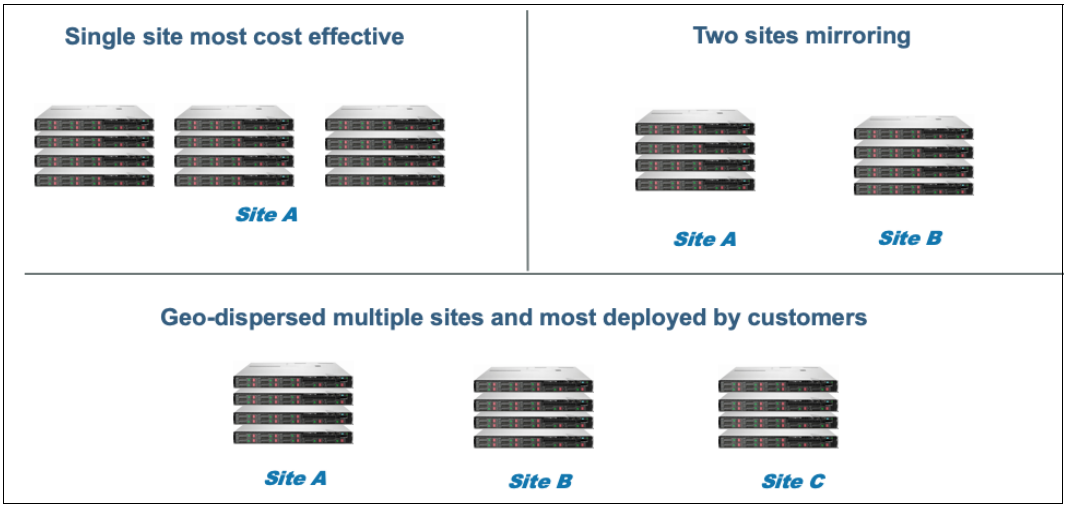

Figure 10 IBM Cloud Object Storage location options

The nodes of a single IBM COS system can be spread across distances of thousands of kilometers if the round-trip latency between nodes does not exceed 100 milliseconds.

Typical expansion factors are listed in Table 1.

Table 1 Typical expansion factors for IBM COS

Number of sites	Typical expansion factor range
Single site	1.3 - 1.5
Two sites	2.4 - 2.8
Three sites	1.8 - 2.0
More than three sites	1.4 - 1.8

Container mode

The default for an IBM COS system is vault mode, which is suitable for most customer deployments. For deployments that support thousands of buckets or tenants, IBM COS can be deployed in *container mode*.

Note: General term for a logical storage unit in S3 is a bucket. In vault mode, a bucket is referred to as a *vault*. In container mode, a bucket is referred to as a *container*.

Container mode brings the following capabilities to the IBM COS system:

- Support for millions of buckets
- Support for millions of users
- Self-provisioning capability for service by using RESTful API
- Support for billing users based on usage
- Isolation of objects between users and tenants

For more information, see the following documents:

- *IBM Cloud Object Storage System Container Mode Guide*
- *IBM Cloud Object Storage System Container Mode Feature Description Document*

Notes: Consider the following points:

- In the container mode deployment, you need a solution (for example, a portal or script-based procedures) for managing accounts, buckets, and so on.
- Vaults still exist in container mode and are defined by the operator as container vaults. Individual users can create containers within those container vaults, but the configuration of those containers is inherited from the container vault, which includes the IDA.

The main features of vault and container mode are compared in Table 2.

Table 2 Vault and container mode comparison

Feature	Vault mode	Container mode
Maximum number of buckets	1,000 vaults (1,500 with supported hardware and system configuration)	Millions of containers
Maximum number of users	Thousands	Millions
Bucket, user, permission management	Via GUI or REST API on the Manager node	Via Service API on the Accesser nodes

Feature	Vault mode	Container mode
User authentication for the bucket	S3 Access KeyID/Secret Access Key pair or username/password	S3 Access KeyID/Secret Access Key pair
Bucket deletion authorization	Yes	No
Bucket quota support	Yes	Yes
Bucket firewall support	Yes (IP whitelist)	Yes (IP allow/disallow)
Bucket indexing options	Name index/Recovery listing only/No index	Name index
Bucket tagging support	Yes	No
S3 versioning	Yes (up to 1000 versions per object)	No
Retention enabled buckets, legal hold	Yes	Yes
Mirroring (2 site data replication)	Yes	No
S3 proxy and migration support	Yes	No
Accesser deployment options	Physical, virtual machine, container, application	Physical server only
Supported dispersal modes	Standard and Concentrated Dispersal Mode	Standard Dispersal Mode

IBM Cloud Object Storage read and write operations

Data import and retrieval for IBM COS is achieved by using standard S3 (HTTP) PUT and GET commands. In the background, the data goes through several transformation steps across the IBM COS Accesser node before it is sent to the Slicestor nodes for storage. The process of reading and writing data to IBM COS is described next.

Writing data to IBM Cloud Object Storage

As shown in Figure 11 on page 14, client applications use the S3 API interface to write objects to IBM COS through an Accesser node. The location of the client application, Accesser node, and Slicestor nodes are independent; all can be at the same site or at different sites. The process of writing data includes the following steps:

1. The client application issues a write request and sends the data to an Accesser node.
2. The Accesser node segments the object.
3. The Accesser node encrypts and performs the erasure coding of the data.
4. The system stores data on the Slicestor nodes.
5. The Accesser node acknowledges the write to the client application.

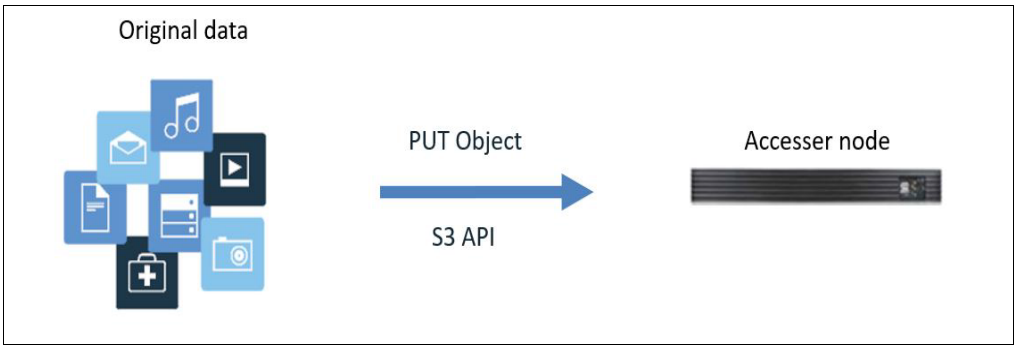

Figure 11 Client application sends the data to an IBM Cloud Object Storage Accesser node

Segmentation

If the object is larger than 4 MiB, the Accesser node splits the data into 4 MiB segments for optimal performance. For example, a 1 GiB object is split into 250 4 MiB segments, as shown in Figure 12.

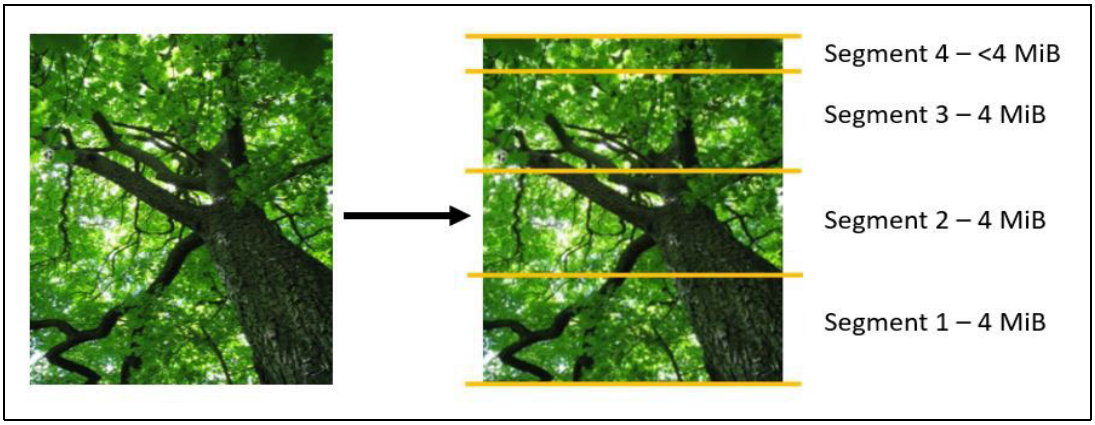

Figure 12 IBM Cloud Object Storage Accesser node creating a 4 MiB segments

Note: Smaller objects (less than 1 MiB) are stored together in *bin files*. This configuration enables better space management and faster read, write, and list operations.

Data-at-rest encryption and erasure coding

The individual segments go through a transformation that enables the system to store the data reliably and securely. This transformation includes two main steps:

1. Data-at-rest encryption with SecureSlice.
2. Erasure coding and data dispersal with the Information Dispersal Algorithm.

SecureSlice

SecureSlice uses an *all-or-nothing-transform* (AONT) to encrypt the data. AONT is a type of encryption in which the information can be deciphered only if all the content is known.

Figure 13 on page 15 shows the IBM COS SecureSlice AONT encryption process:

1. Append an integrity check value to the segment.
2. Generate random encryption key.
3. Encrypt data by using encryption key.
4. Calculate the hash of encrypted data.
5. Calculate exclusive-OR (XOR) of hash and encryption key.

6. Append the result to the encrypted data to create the AONT package.

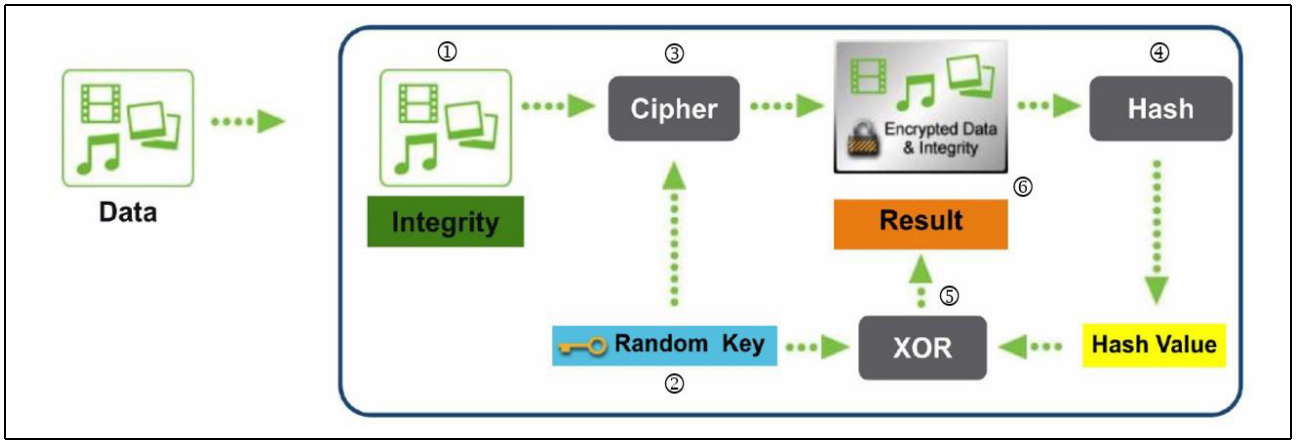

Figure 13 IBM Cloud Object Storage SecureSlice AONT encryption

Notes: Consider the following points:

- Integrity check values are always added to the data segments, but encryption is an optional feature. Encryption is enabled by default, and can be disabled on a per vault basis in the IBM COS Manager at the time of vault creation.
- IBM COS software version 3.14.6 introduced the following changes to the implementation and configuration of data-at-rest encryption with SecureSlice:
  - A new encryption algorithm was added, AES-GCM-256, which provides up to 15% performance improvement compared to the existing default AES-128.
  - After upgrading to 3.14.6 or later, AES-GCM-256 becomes the default and the recommended encryption algorithm for the vaults.
  - Administrators now can choose the preferred encryption algorithm for new vaults and they can also change the algorithm for the existing vaults.
  - If the encryption settings are changed for a vault, it is applied to only the newly written objects. Existing objects are *not* reencrypted, and they do not benefit from improved performance and better security that is provided by the new algorithm.
  - It is now possible to enable or disable the SecureSlice for existing vaults at any time.
  - All changes that are made to the encryption settings generate audit log messages for tracking purposes.

Information Dispersal Algorithm

In the next step, erasure coding is performed on the AONT package and data is distributed to the Slicestor nodes.

Note: In general, the erasure coding transforms a message of k symbols into a longer message with n symbols such that the original message can be recovered from a subset of the n symbols (only k symbols are needed to reconstruct the data). In the case of IBM COS erasure coding, k is always the read threshold and n is the width.

The IBM COS erasure coding process includes the following steps:

1. The AONT package is sliced to read threshold number of slices.
2. Erasure coding creates the width number of encoded slices from the slices that were created in step 1.
3. Data is sent to the Slicestor nodes.

Figure 14 shows the main steps of the data transformation.

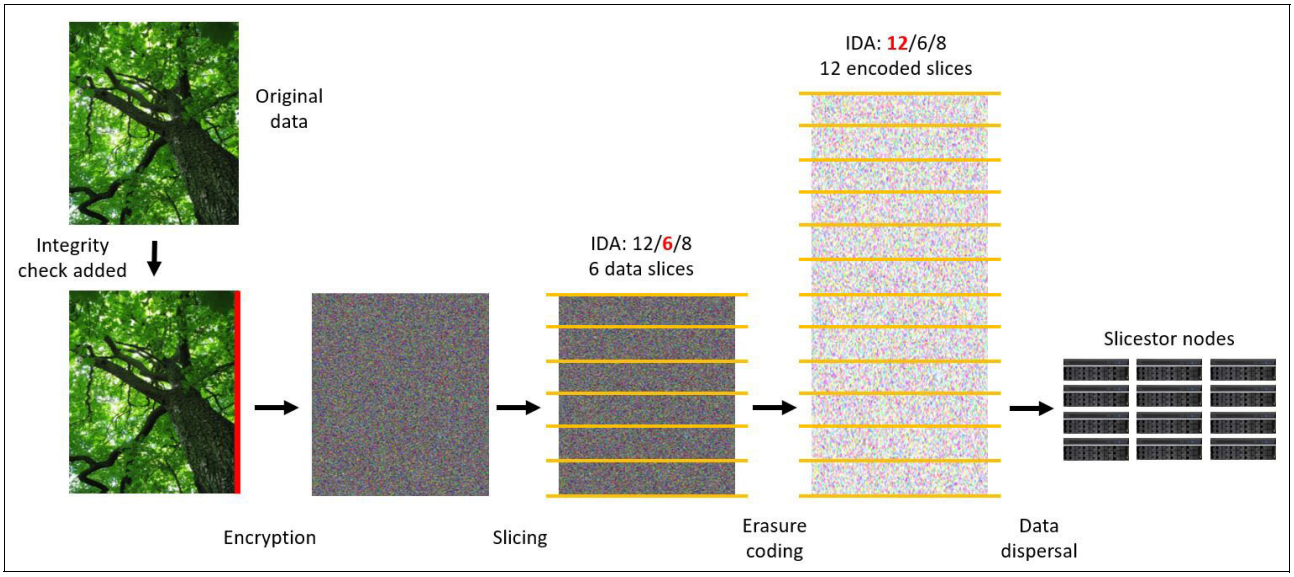

Figure 14 Data transformation in IBM COS

Storing the data on the Slicestor nodes

After erasure coding, data is distributed to the Slicestor nodes. The storage pool and the vault IDA configuration determines how many Slicestor nodes are used to store the data for a specific segment.

Writing data in Standard Dispersal Mode

If the storage pool is configured to use SD Mode, each Slicestor node in a device set stores a single slice, as shown in Figure 15.

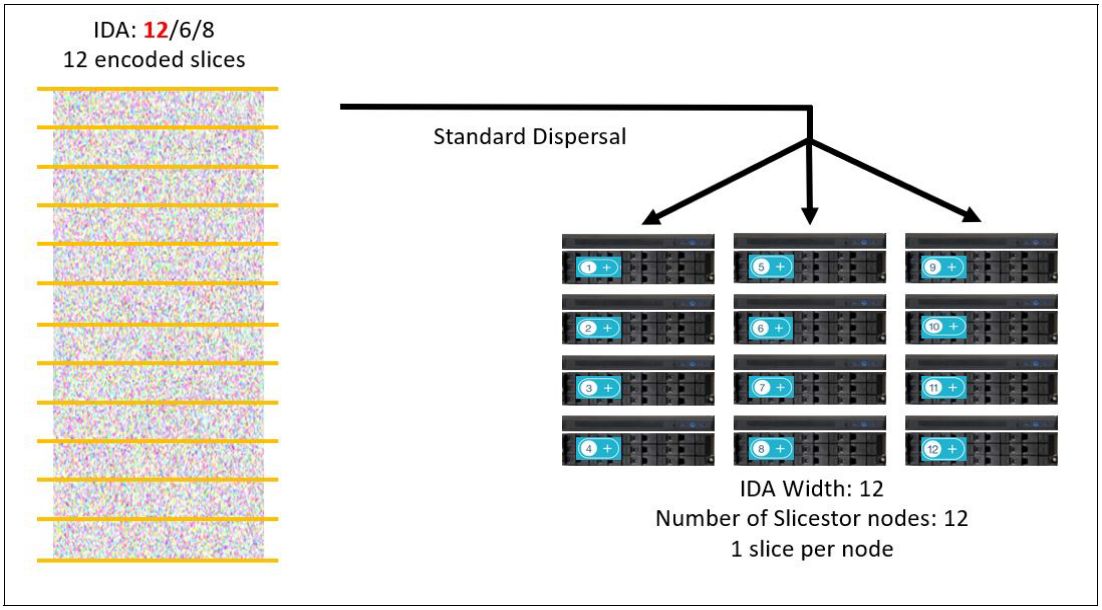

Figure 15 Writing to a Standard Dispersal Mode storage pool

Writing data in Concentrated Dispersal Mode

If the storage pool is configured to use CD Mode, each Slicestor node stores multiple encoded slices. The system ensures that the slices are not stored on the same drives within the same chassis. This configuration ensures high reliability and that a single drive failure does not cause loss of multiple slices. Figure 16 shows writing to a CD Mode storage pool.

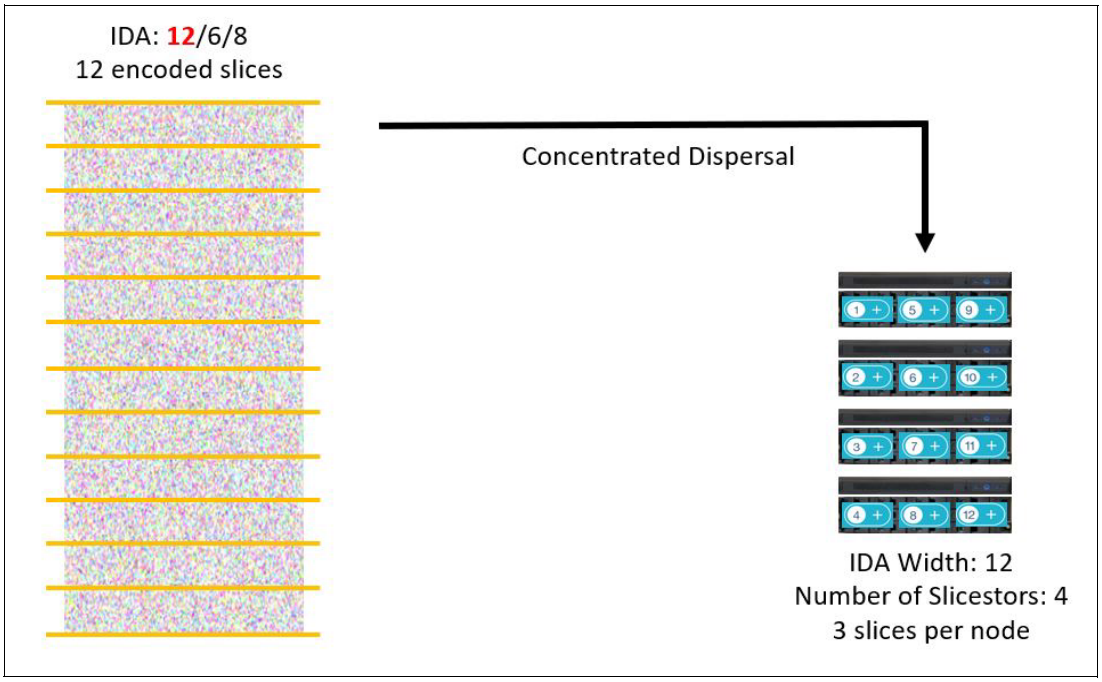

Figure 16 Writing to a Concentrated Dispersal Mode storage pool

Writing data to a mirrored vault

In the case of a mirrored vault, the Accesser node sends the slices to both component vaults that always are on different storage pools and typically on different sites. Figure 17 shows a write operation to a mirrored vault in CD Mode. Write operation to an SD Mode-based vault mirror works similarly, but only one slice is written to every Slicestor node.

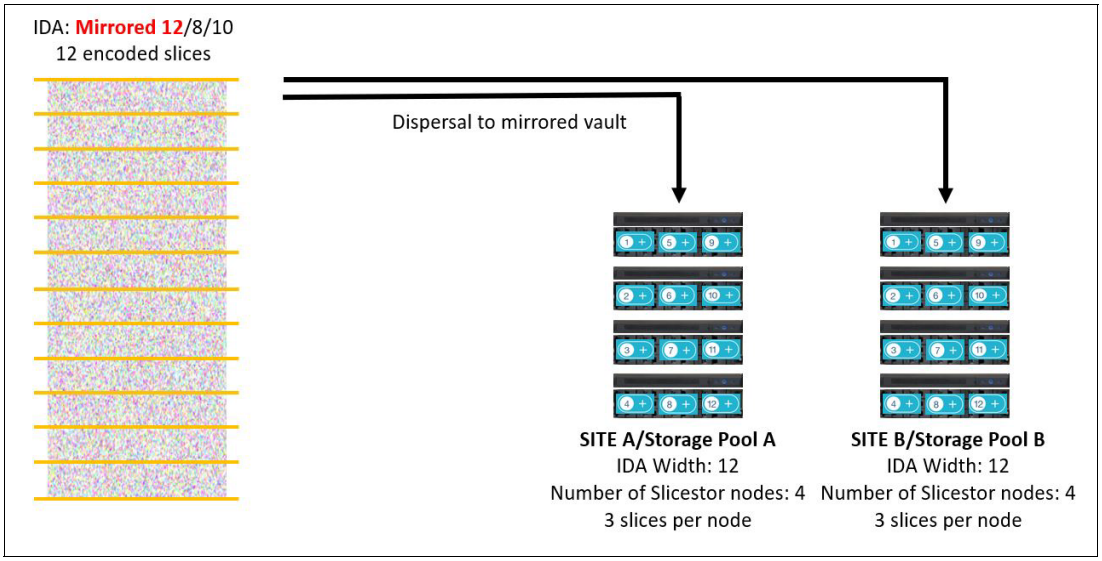

Figure 17 Writing data to a mirrored vault

Successful write operations

The write operation is successful and acknowledgment is sent back to the client when the minimum write threshold number of slices are written to the storage pool. The Accesser node continues to write the remaining slices to the Slicestor nodes asynchronously. If any of the slices cannot be written (for example, when a Slicestor node is unavailable), the distributed rebuilder process (which is running on the Slicestor nodes) automatically re-creates the missing slices.

For mirrored vaults, the following modes are available when the mirror is created:

- Asynchronous (default setting)

  The Accesser node sends acknowledgment to the client application after the write operation is confirmed on *either* side of the mirrors.

- Synchronous

  The Accesser node sends acknowledgment to the client application after the write operation completes on *both* sides of the mirror but both sides do not have to succeed in writing the data.

- Hard Synchronous

  The Accesser node sends acknowledgment to the client application after *both* sides of the mirror confirmed the successful write.

Figure 18 shows more information about each mirror mode's advantages and disadvantages.

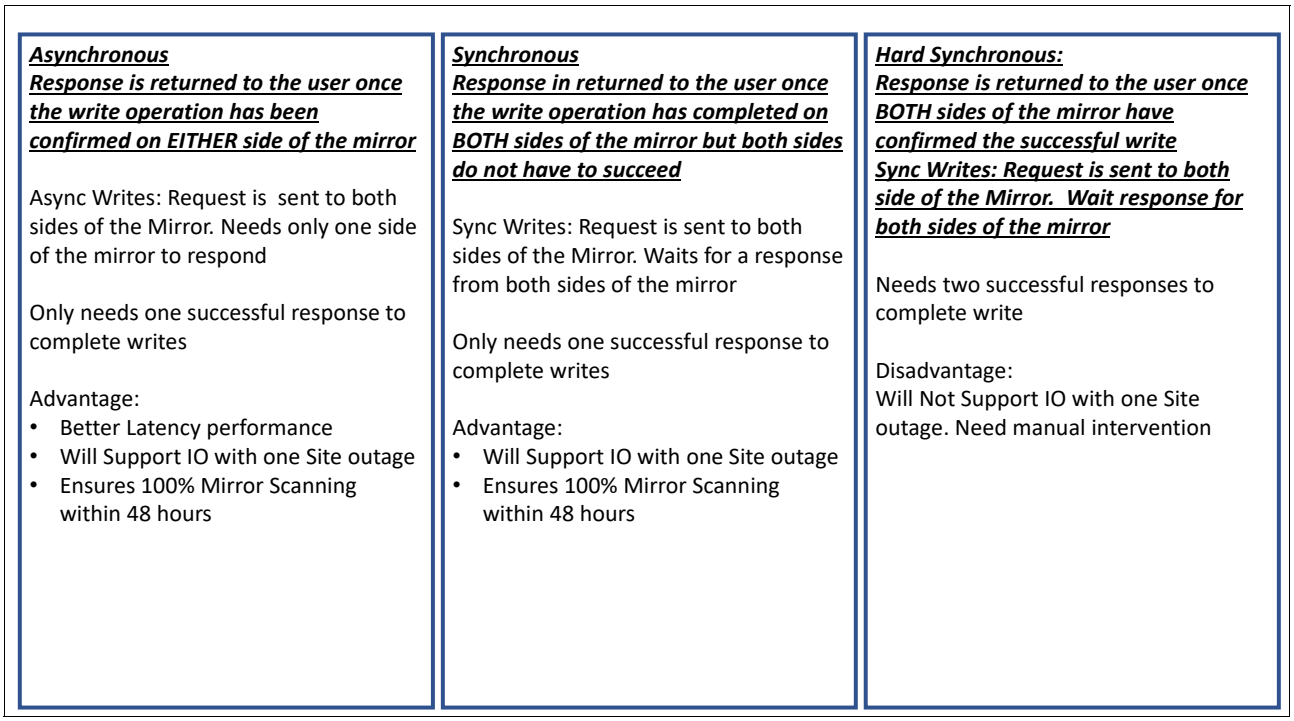

Figure 18 Advantages and disadvantages of the mirror modes

Note: Changing between mirroring modes is nondisruptive.

For more information about writing to mirrored vaults, see IBM Knowledge Center.

Performance optimizations for write operations

SmartWrite enables a successful write operation, even if the full width of slices cannot be written in a timely manner. This issue can be caused by a failure condition at a node or within the network.

SmartWrite optimistically attempts to write all slices. After the required write threshold of slices is achieved, SmartWrite considers the write successful. The Accesser node attempts to write the remaining slices asynchronously.

If a slice write operation times out on the Accesser node, no further attempts to write the slice are made. The SmartWrite feature is always active and increases overall system write performance in occasions; for example, when:

- Network connection to a site or a Slicestor node is slower or the connection is down
- A degraded (limping) Slicestor node is in the system

Figure 19 shows the SmartWrite operation.

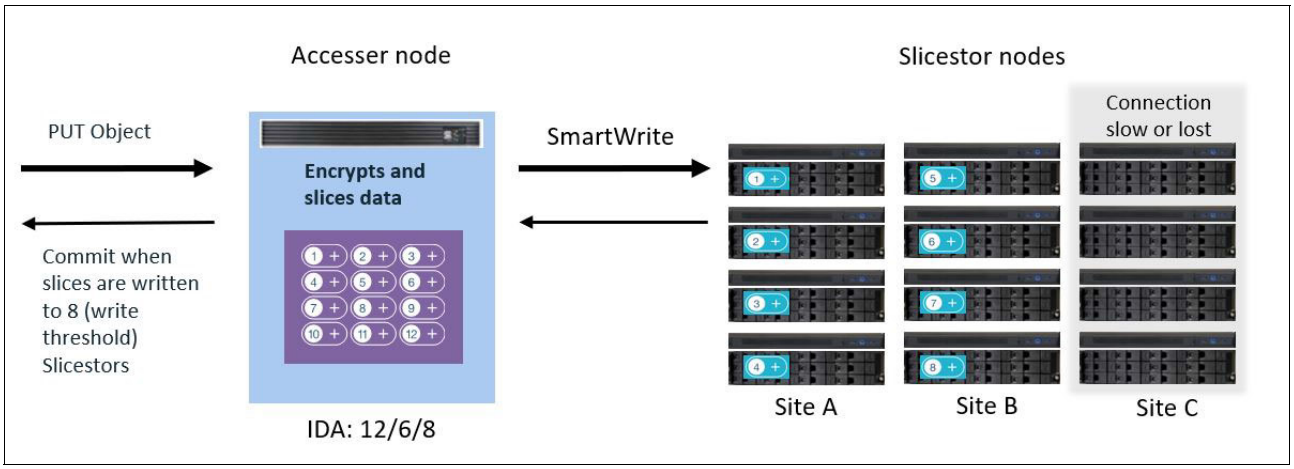

Figure 19 Performance optimization with SmartWrite

Reading data from IBM Cloud Object Storage

This section describes how data is read from IBM COS through the Accesser nodes. One of the advantages of IBM COS is data can be read by using any Accesser node within the system, even from ones on different sites.

The read operations include the following main steps:

1. The client application issues a read request that is sent to one of the Accesser nodes.
2. The Accesser node instructs the Slicestor nodes to send the required data slices.
3. After the read threshold number of slices is received, the Accesser node decodes the object and verifies its integrity.
4. The reconstructed data is sent to the client application by using the S3 protocol.

Tip: IBM COS supports the *S3 API range read feature*, which means that applications can request a full object or a part of an object.

Figure 20 on page 21 shows reading data from IBM COS. Reading is successful, even if multiple Slicestor nodes are down. IBM COS requires only the read threshold number of Slicestor nodes that are available to reconstruct the data.

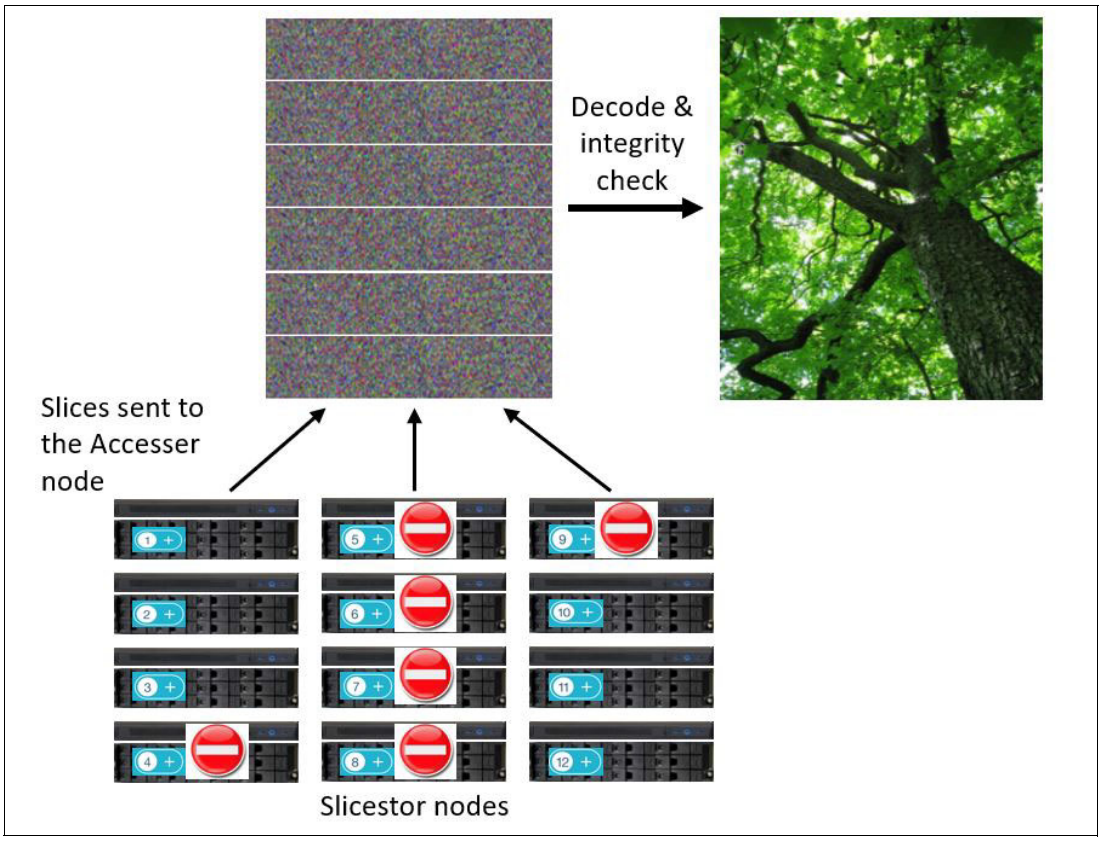

Figure 20 Reading data from IBM Cloud Object Storage

Decryption for reads

If the object was encrypted by SecureSlice, the following process is used to decrypt the slices during the data reconstruction:

1. Strip the appended result from the end of the encrypted data.
2. Calculate the hash of the encrypted data.
3. Exclusive OR (XOR) the hash with the result to recover the encryption key.
4. Use the key to decrypt the data.
5. Verify the integrity of the data with the previously added integrity check value.

Figure 21 shows IBM COS SecureSlice decryption.

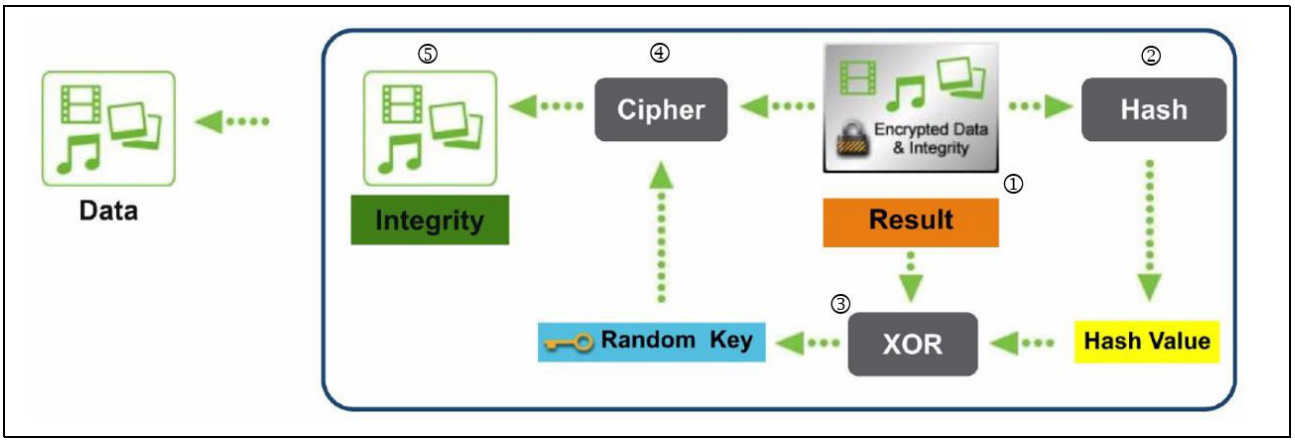

Figure 21 IBM Cloud Object Storage SecureSlice decryption

Performance optimizations for read operations

SmartRead predicts the optimal network routes and Slicestor nodes from which to most efficiently retrieve data. SmartRead ranks Slicestor nodes by real-time performance and requests the optimal combination (read threshold number) of slices to re-create the data. If a slice request is not performing, SmartRead requests the missing slice from another node. This feature is always on and significantly increases overall system read performance in cases when, for example, the network connection to some Slicestor nodes are slower or a limping Slicestor node in the system. Figure 22 shows the SmartRead operation details.

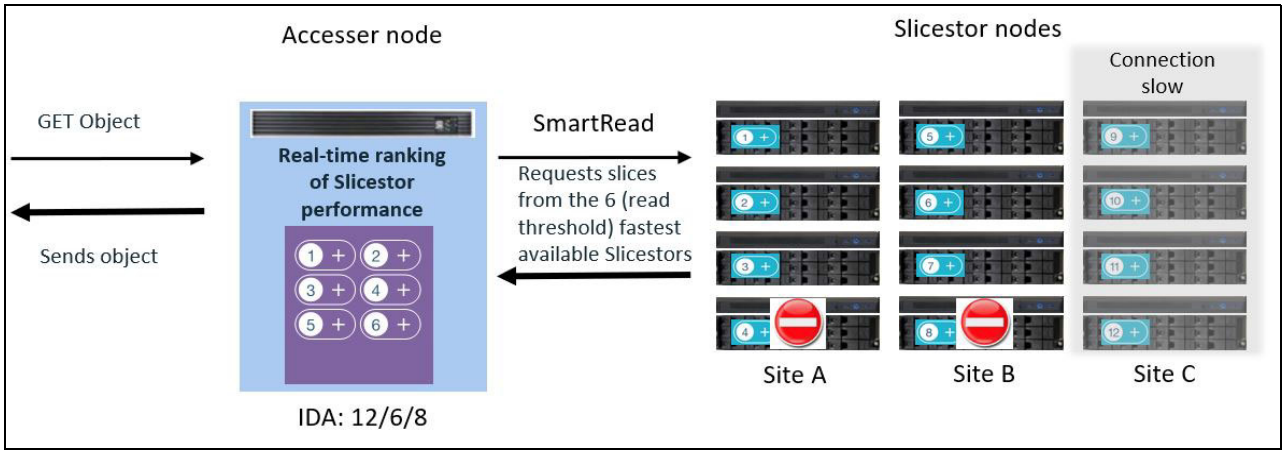

Figure 22 Performance optimization with SmartRead

Storing data within the Slicestor nodes

IBM COS uses Packed Slice Storage (PSS) or Zone Slice Storage (ZSS) as the backend storage engine to store the slices on the Slicestor nodes.

Packed Slice Storage

PSS uses 128 bin files per drive to store the object slices to optimize the underlying file system for performance and utilization.

PSS offers the following benefits on top of file system:

- Uses storage space efficiently, especially for small objects.
- Increases read performance and lowers read latency. In most cases, a single seek in to read a slice and no I/O occurs when trying to read a non-existent slice.
- Writes objects with greater efficiency. Data is appended to the end of one of a fixed number of bin files that PSS maintains.
- Improves listing operations in speed and system resources that are used.
- Reduces the burden on underlying file system.
- Performs at the same level for long periods at a wide range of fills because of the reduced load on underlying drives.

Figure 23 shows the benefits for PSS when storing small files.

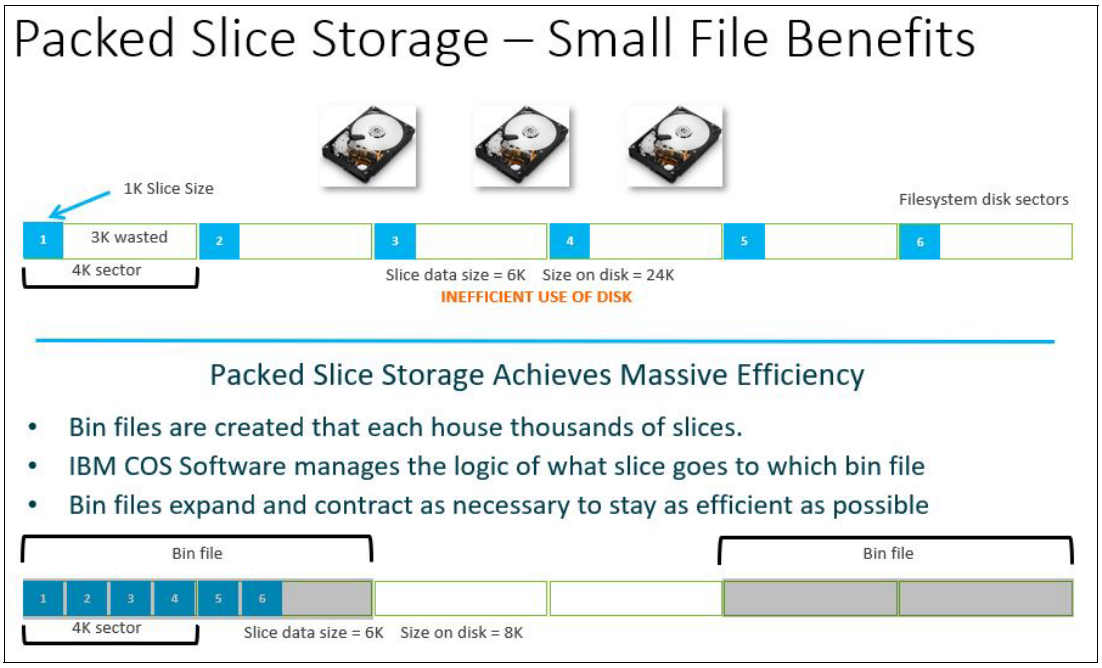

Figure 23 Packed Slice Storage benefits

What this means for you: Unlike other Object Storage systems that use space-consuming mirroring for small objects, IBM Cloud Object Storage uses advanced erasure coding for all object sizes, small or large. This feature enables an efficient and reliable way of storing data.

Zone Slice Storage

ZSS is the next generation of IBM COS-backend storage engine that provides more benefits on top of PSS. The following key changes of the new architecture are shown in Figure 24 on page 24:

- Access the raw disk at the block level instead of going through a file system.
- Perform all writes sequentially; that is, no overwrites.
- Convert the entire storage stack to asynchronous, which reduces the number of threads that are required to keep all disks saturated.
- Manage write cache at the application level rather than in kernel. Uncommitted data is never written to disk. Also, reads are cached based on type, recency, and frequency.

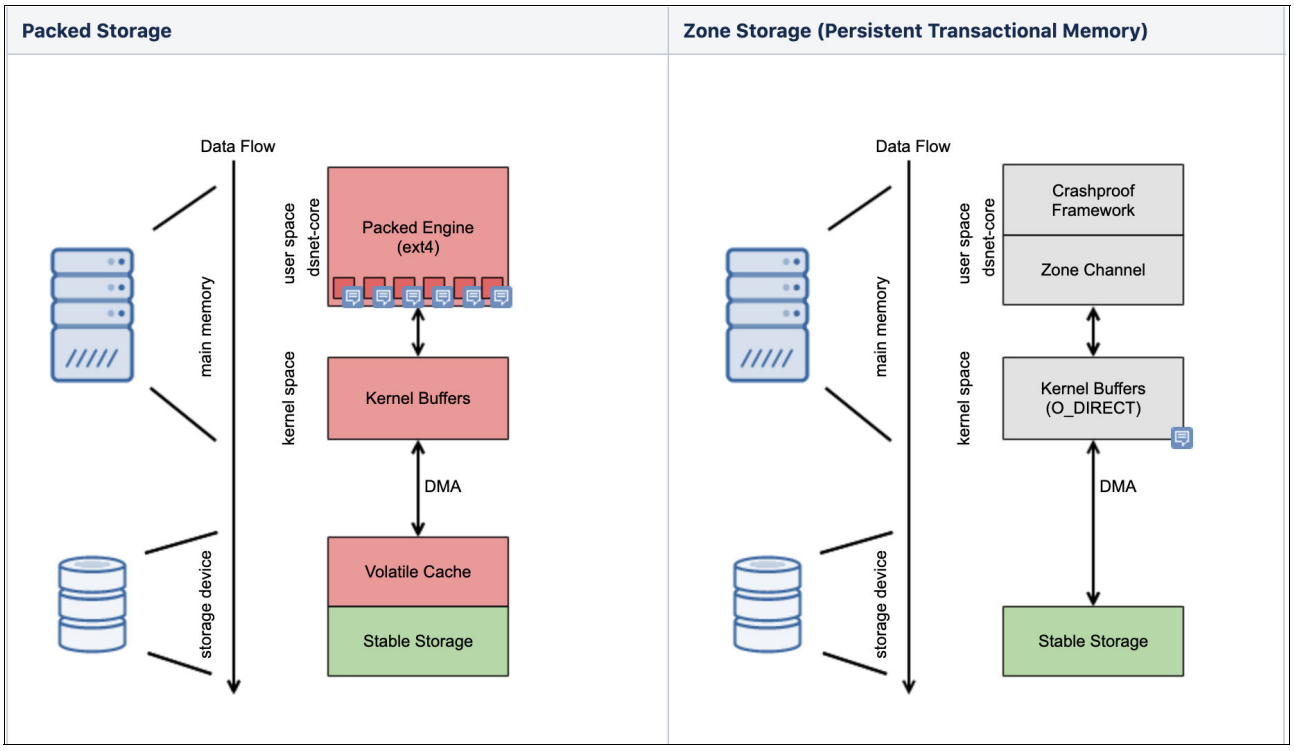

Figure 24 PSS and ZSS architecture comparison

The goals of the new architecture are to provide significant impact on the following features:

- Performance:
  - Efficient support of media that prefers or requires sequential writes, specifically shingled magnetic recording (SMR), hard disk drive (HDD), and solid-state drive (SSD).
  - I/O performance approaches the theoretical media limits for reads and writes for both throughput and latency.
  - High performance against a wide range of system loads without manual configuration.
- Resilience:
  - Full internal consistency after hardware or software crashes (no orphaned, dangling, or unaccounted usage) without the use of offline correction tools (for example, file system consistency check) or full journaling of writes.
  - Able to limit collateral damage from most unrecoverable read errors on disks to a small amount of data loss (which is then rebuilt from other nodes).
  - Object level atomicity (old or new version of object guaranteed to be available), even on a single power grid.
  - Synchronous mode in which success is not acknowledged to user until data is durable.
  - Predictability under adverse conditions, such as limping, failing, malfunctioning hardware.
  - Protection against many attacks based on naming or content of the data written.

- Space efficiency:
  - Allow filling the media much closer to 100%.
  - Uniform and stable performance that is largely independent of level of storage use and number of objects.
  - Small amount of overhead stored per object.
- Supportability:
  - Extensible architecture for future system customization and enhancements.
  - Shorter code stack to hardware, which allows better maintainability and easier root causing of issues.
  - Comprehensive specialized tooling to allow for easier debugging, state introspection, and improved root cause determination.
  - Extensive statistics to allow better introspection into system behavior.

Features and functionality

This section describes the key features and functionality of IBM COS.

Availability and reliability

IBM COS can operate without disruption during software upgrades, hardware refreshes, hardware failures (disk failure or node failure), or even when an entire site is unavailable.

IBM COS availability includes the following key characteristics:

- No scheduled maintenance windows required for expansion, upgrade, and data migration. All of these operations can be done while the solution is available to users and applications.
- IBM COS architecture is designed to tolerate multiple simultaneous failures:
  - Typical IBM COS systems are designed for 99.9999% availability for writes. This means that the system is unavailable for less than 32 seconds per year.
  - Typical IBM COS systems are designed for 99.999999% availability for reads. This means that the system is unavailable for less than 316 milliseconds per year.

    Fun fact: A planet-killing asteroid has a chance of striking the Earth 1 in 100,000,000 years. That is, it is just as likely for the IBM COS system to be unavailable for reads as it is for the Earth to be struck by a planet-killing asteroid.

- Automatic and continuous integrity checking and error correction for all slices and objects.
- An IBM COS system can be designed for 99.9999999999999% reliability (durability). To put it into perspective, this means you might expect to lose 1 byte of data in every 1 trillion years.

What this means for you: IBM Cloud Object Storage provides *always-on availability*, which means it can even tolerate catastrophic regional outage without downtime or intervention. Continuous availability and ultimate reliability is built into the architecture.

Disk Lifecycle Management

Disk Lifecyle Management (DLM) is a set of programs, libraries, and kernel modules that are part of IBM COS that provide administration and monitoring of disk drives.

IBM COS DLM provides the following key characteristics:

- Monitors disk medium errors.
- Monitors disk performance and error rates.
- Provides a configuration framework for disk policies.
- Collects the following information:
  - Disk model
  - Firmware information
  - Read and writes
  - Errors
  - Other metrics
- DLM provides multiple disk states that allow the administrator to determine the detailed status of a drive.
- Supports Self-Monitoring, Analysis, and Reporting Technology (SMART) for drive diagnostics.

What this means for you: IBM COS is designed to provide you with detailed information about the state of drives in the solution. From disk states to built-in monitoring, the IBM COS solution includes various capabilities that are designed to help with DLM. IBM COS improves data reliability by proactively monitoring for disk medium errors.

Rebuilder

The following software agents comprise the Rebuilder:

- Scanning agent

  This agent checks the consistency of the slice names and revision numbers that are held by the Slicestor nodes.

  All Slicestor nodes scan for missing slices and limit the number of listing requests that are processed at one time to throttle the scanning operations.

  A full scan of an IBM COS system should not take more than 48 hours, even for an exabyte scale system. Therefore, missing slices are detected within 48 hours.

- Rebuild agent

  This agent retrieves the new slice data from other Slicestor nodes to repair a slice on a Slicestor node.

  A rebuild agent on a Slicestor node addresses scenarios where slices are missing from their respective Slicestor nodes or which are corrupted on a drive.

  Each Slicestor node in a storage pool ensures data integrity across all of the Slicestor nodes that are in that storage pool. If a Slicestor node goes offline and misses some newly written slice, it rebuilds any missing slices when it comes back online. The Slicestor node reads the content of other Slicestor nodes and re-creates the missing slice.

- Integrity agent

  This agent checks the integrity of slices on the Slicestor nodes.

  Each Slicestor node runs an integrity agent, which reads every slice that is stored on that node and compares it against a 4-byte CRC32 checksum, which is stored alongside that slice. If the checksum does not match, the slice is deprecated and then rebuilt from the other slices for that object. This check occurs every time a client system attempts to read an object. If the check fails, the object and its component slices are rebuilt.

Scalability

IBM COS was tested at web-scale with production deployments that exceed hundreds of petabytes of capacity, and can scale to exabytes.

IBM COS scalability provides the following key characteristics:

- Internet-style, distributed, shared-nothing, and peer-to-peer architecture.
- Greater than Yottabyte-scale global namespace with 2^384 (10^115) unique object IDs.

  Fun fact: The observable universe has approximately 10^80 atoms.

- Support for millions of buckets.
- Support for 2^128 objects per bucket.
- Parallel access support:
  - Multiple Accesser nodes can access the same bucket simultaneously.
  - Applications can use the same Accesser node to access different buckets simultaneously.
  - Geo-dispersed Accesser nodes have access to all buckets and all Slicestor nodes.
- More Accesser nodes might be required to:
  - Support more applications
  - Improve performance
- Increase storage capacity and performance by adding Slicestor nodes in multiples of the IDA width.
- Scale to 65,536 appliances (any combination of Slicestor nodes and Accesser nodes) in a single IBM COS system.
- Near-linear increase in system throughput and HTTP operations per second as the system grows.

What this means for you: IBM Cloud Object Storage is built for large data sets and can scale to exabytes while maintaining availability, reliability, manageability, and cost-effectiveness without any compromise. The scalability is virtually unlimited.

Security

IBM COS security provides the following key characteristics:

- All crucial configuration information is digitally signed to avoid being compromised.
- Certificate-based authentication of every node (Manager, Accesser node, and Slicestor node) is provided.
- Transmission and storage of data is inherently private and secure.
- No single disk or Slicestor node contains a full object.
- Transport Layer Security (TLS) is supported for data-in-motion protection.
- All data is encrypted at rest by default.
- Supports retention enabled buckets.

IBM COS security is separated into appliance, user authentication, network, data, retention enabled vaults, client, and object security.

Appliance security

The underlying IBM COS Manager, IBM COS Accesser, and IBM COS Slicestor appliances have multiple levels of security:

- Stock open source Linux distribution and Linux kernel that is pared down to the minimal necessary functions enabled. National Security Agency (NSA) hardening guidelines are then applied.
- Industry standard 64-bit processors that support the NX (No eXecute) bit, which prevents a buffer overflow from turning into remote code execution vulnerabilities.
- Each appliance has its own internal firewall, which restricts network access to everything but the appliance's critical services.
- Each appliance is monitored in real time by the Manager, which receives SSL-encrypted system logs.

User authentication

Authentication allows users and storage administrators to identify themselves to other entities in a trusted manner. Consider the following points:

- Credentials that are exchanged between appliances are encrypted by using TLS.
- Customers can turn off TLS between the Accesser and Slicestor appliances if it is not required. For those customers, the confidentiality of credentials is still protected. The appliances encrypt the credentials by using a key negotiated that uses the Diffie-Hellman protocol. This configuration ensures that no one can sniff network traffic to intercept credentials.
- Passwords that are stored in the credential database are not stored in a raw form; rather, they are salted and hashed to prevent direct exposure or the application of rainbow tables if password database falls into the wrong hands.
- IBM COS supports direct integration with an Active Directory (AD) server. Passwords are encrypted in flight over Lightweight Directory Access Protocol (LDAP) if supported by the AD server.
- User authentication is supported by digital certificates. Authenticating by using digital certificates is superior to authentication with passwords because the private credential (the private key) is never divulged during the authentication process.

- All user operations that are performed in the GUI and on the appliances are audited to provide accountability and traceability.
- Supports local named accounts for GUI and all appliances for individual accountability.
- Supports user name, password, and SSH keys for SSH authentication.

Network security

All network traffic that flows within IBM COS is encrypted. Authentication with SSL/TLS requires the use of digital certificates and these certificates must be verified as belonging to a valid node. Consider the following points:

- Appliances are given a signed digital certificate at the time they are approved into IBM COS.
- Approval requires an IBM COS administrator to log in to the management interface, view the request, and authorize. The administrator can see the IP address, MAC address, fingerprint of the appliance and verify that each is valid before approval into IBM COS.
- After approval, the appliance is granted a certificate by the internal certificate authority (CA) for IBM COS. All appliances in IBM COS trust this CA and any appliance that owns a valid certificate that is signed by this CA.
- The use of external CAs is supported.
- Appliance certificates can be revoked by the Manager for any reason by using a client revocation list (CRL), which is periodically polled by every appliance in IBM COS.
- All configuration information, such as Access Control Lists (ACLs), vault quotas, device IP addresses, is published over HTTPS.

Data security

Securing data has three main requirements: confidentiality, integrity, and availability. All three of these requirements are satisfied by using SecureSlice. Consider the following points:

- SecureSlice is a standard product feature that is enabled by default and all data is encrypted.
- SecureSlice can be configured to use any of the following combinations of encryption and data integrity algorithms:
  - RC4-128 encryption with MD5-128 hash for data integrity
  - AES-128 encryption with MD5-128 hash for data integrity
  - AES-256 encryption with SHA-256 hash for data integrity
  - AES-GCM-256 encryption with SHA-256 hash for data integrity (*default and recommended*)
- SecureSlice does not require an external key management system.
- SecureSlice supports Server-Side Encryption with Customer Provided Keys (SSE-C).
- To prevent accidental deletion of data by a single administrator, vault deletion authorization can be enabled where a second administrator must approve the delete request.

For more information, see “SecureSlice” on page 14.

Retention enabled buckets

Retention enabled buckets introduce another level of protection for customers that are required to enable strict retention policies from regulatory entities or internal policies. This feature provides the customer with the ability to create retention enabled buckets.

Objects that are stored in retention enabled buckets are protected objects that include associated retention periods and optional legal holds. Protected objects cannot be deleted or modified until the retention period expires and all legal holds on the object are removed.

With this feature, IBM COS is natively compliant with the following key standard and compliance requirements:

- Securities and Exchange Commission (SEC) Rule 17a-4(f) (for more information, see this web page):
  - SEC 17a-4(f)(2)(ii)(A): Protect data from deletion and overwriting.
  - SEC 17a-4(f)(2)(ii)(B): Automatically verify that the storage system properly stored the data.
  - SEC 17a-4(f)(2)(ii)(C): Manage retention periods for the objects.
  - SEC 17a-4(f)(2)(ii)(D): Download indexes and records.
  - SEC 17a-4(f)(2)(iii/v): Store duplicate copies and provide audit capabilities.
- Financial Industry Regulatory Authority (FINRA) Rule 4511, which references requirements of SEC Rule 17a-4(f).
- Commodity Futures Trading Commission (CFTC) Rule 1.31(b)-(c).

Additional reference: For more information about deployment details and feature limitations, see IBM Knowledge Center.

Client and object security

IBM COS includes the following client and object security features:

- Object-level authentication is done by using an Access Key ID and Secret Access Key mechanism.
- TLS is supported between applications and the Accesser node by using HTTPS.
- Operations that are performed on user data are also audited to provide accountability and traceability.

What this means for you: IBM Cloud Object Storage is designed with a high level of security in mind to protect your data from security breaches. From built-in encryption of data at rest and in motion, to a range of authentication and access control options, the IBM Cloud Object Storage solution includes a wide range of capabilities that are designed to help meet your security requirements. These security capabilities are implemented to help enable better security, without compromising scalability, availability, ease of management, or economic efficiency.

Object expiration

Object expiration feature provides the capability of automatic deletion of objects, based on a lifecycle policy that is configured on the bucket level. This feature provides the following benefits:

- Allows the customer to manage their storage costs by scheduling periodic deletion of data that is no longer needed.

- Provides a better way to manage object deletion without external tools.
- It adds compatibility with S3 API features; that is, applications that use S3 object expiration can be used with IBM COS systems.
- Deleting objects that use this built-in object expiration feature has performance improvements over listing and deleting objects manually.
- IBM COS Manager provides the monitoring and alerting capability for this feature.

Additional reference: For more information about deployment details and feature limitations, see IBM Knowledge Center.

Authors

This paper was produced by a team of specialists from around the world working at the IBM Cloud Object Storage Office in Chicago.

Bradley Leonard is an IBM Certified IT Specialist with IBM Cloud And Cognitive Software in the United States. He has over 24 years of experience working in cloud computing, enterprise system management, testing, and DevOps. Bradley is focused on IBM Cloud Object Storage as a Solution Architect with the World Wide Cloud Object Storage Services team. He holds a Bachelor's degree in Computer Science from California Polytechnic State University, San Luis Obispo.

Hao Jia is a Lead Engineer within IBM Cloud Object Storage (COS) Level 3 Support organization based in Chicago, Illinois, US. He has 10 years of IT experience in Cloud Object Storage and Telecommunications. He holds a Master's degree in Electrical and Computer Engineering from Illinois Institute of Technology in Chicago. Hao has been supporting the integration of IBM COS into IBM Cloud since 2016. He is leading the effort to provide strategic customers of IBM Cloud and COS on-premises with the best technical support experience.

Johan Verstrepen is an IBM Certified IT Specialist. He worked 9 years in the AIX/VIOS and Linux Support team in IBM Belgium as a virtualization and networking specialist. Since then, he moved to IBM Switzerland and has been working in the IBM COS team for 3 years, providing IBM Cloud Object Storage Product Support and Services. Johan has a Bachelor Degree in Telecommunications, received at the Groep-T campus of the University of Leuven (Belgium).

Jussi Lehtinen is a Solutions Architect for IBM File and Object Storage working for IBM Systems in Europe. He has over 30 years of experience working in IT; last 20 years with Storage. He holds a Bachelor's degree in Management and Computer Studies from Webster University in Geneva, Switzerland.

Lars Lauber is a File and Object Storage Client Technical Specialist with IBM Cloud in Germany. He has 3 years of experience working with IBM Cloud Object Storage. He holds a Master's degree in Technology and Innovation Management from FOM University of Applied Sciences for Economics and Management in Stuttgart.

Patrik Jelinko is a Client Technical Specialist with IBM Australia. He has 20 years of experience as an IT infrastructure specialist, and started working with object-based storage systems more than 15 years ago. Patrik is focused on software-defined storage and modern data protection solutions. He holds a Master's degree in IT Engineering from Budapest University of Technology and Economics.

Vasfi Gucer is a project leader with the IBM Systems WW Client Experience Center. He has more than 20 years of experience in the areas of systems management, networking hardware, and software. He writes extensively and teaches IBM classes worldwide about IBM products. His focus has been primarily on storage and cloud computing for the last 8 years. Vasfi is also an IBM Certified Senior IT Specialist, Project Management Professional (PMP), IT Infrastructure Library (ITIL) V2 Manager, and ITIL V3 Expert.

Thanks to the following people for their contributions to this project:

Ann Lund
IBM Redbooks®, Poughkeepsie Center

Rob McCammon, David Wohlford, JT Wood, John K Butler, Sanjaya Kumar, Scott Horan, Praveen Viraraghavan, Timothy Ranttila, Latasha Smith, Michael Knieriemen
IBM USA

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an IBM Redbooks residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

`ibm.com/redbooks/residencies.html`

Stay connected to IBM Redbooks

- Find us on Facebook:

  `http://www.facebook.com/IBMRedbooks`
- Follow us on Twitter:

  `http://twitter.com/ibmredbooks`
- Look for us on LinkedIn:

  `http://www.linkedin.com/groups?home=&gid=2130806`
- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  `https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm`
- Stay current on recent Redbooks publications with RSS Feeds:

  `http://www.redbooks.ibm.com/rss.html`

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

© Copyright IBM Corp. 2020. All rights reserved.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

Accesser®
Aspera®
IBM®
IBM Cloud®
IBM Spectrum®
Redbooks®
Redbooks (logo)  ®
Slicestor®

The following terms are trademarks of other companies:

Merge Healthcare, are trademarks or registered trademarks of Merge Healthcare Inc., an IBM Company.

ITIL is a Registered Trade Mark of AXELOS Limited.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

VMware, and the VMware logo are registered trademarks or trademarks of VMware, Inc. or its subsidiaries in the United States and/or other jurisdictions.

Other company, product, or service names may be trademarks or service marks of others.

REDP-5537-01

ISBN 0738458937

Printed in U.S.A.