SCALITY | RING

WHITE PAPER

Unbreakable cloud storage for data centers

Table of contents

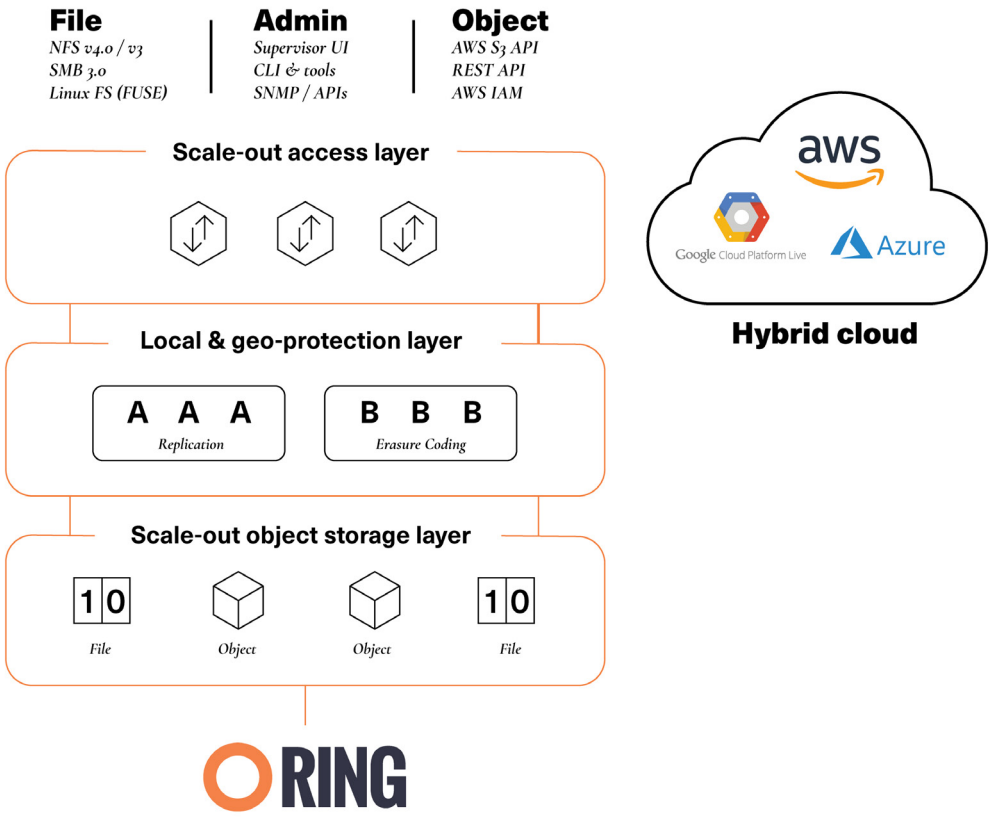
I. Unbreakable cloud storage for private and hybrid data centers

Scality RING has become widely deployed as a cornerstone of enterprise private clouds and service provider cloud storage infrastructures. RING is a mature and proven cloud storage infrastructure solution for petabyte-scale application data from a wide range of workloads, including long-term AI data lakes, analytics, backups and data archives, video content delivery, surveillance data and more.

RING also provides integrated hybrid-cloud data management capabilities to make it possible for enterprises to leverage public cloud services for data stored on RING. These capabilities accommodate a new class of hybrid-cloud data workflows to support use cases such as cloud archiving, bursting and cloud disaster recovery (D/R).

RING is a software-based scale-out storage solution deployed on standard x64 storage server platforms to create a single, elastic and resilient pool of storage. RING uses a cloud-scale distributed architecture designed to create virtually unbounded storage for both legacy and cloud-native applications. By distributing user data and associated metadata across underlying storage nodes, RING eliminates typical bottlenecks and single points of failure to achieve continuous availability.

RING integrates modern Amazon S3 API-compatible object storage capabilities and a POSIX-compatible file system with access through standard NFS v4.0/v3, SMB 3.0 and Linux FUSE protocols. Both object and file access provide full scale-out capabilities in performance and storage capacity that grows with increasing workload requirements over time.



By offering file and object storage capabilities on equal footing in terms of scale-out capabilities, RING can fully support digital transformation as enterprises move from legacy to new cloud applications with hybrid-cloud data management. This is key for supporting applications in industries such as healthcare for medical imaging, in the enterprise for large-scale analytics, and for video-on-demand applications in broadcasters and service providers — all of whom need to manage unstructured data at petabyte scale.

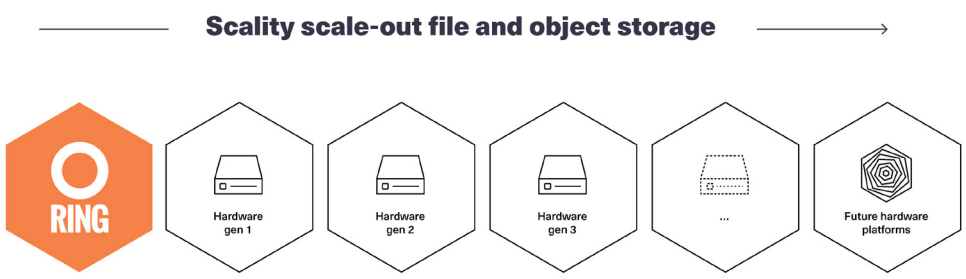
RING deployments

RING is a distributed system deployed on industry-standard hardware, starting with a minimum of three (3) storage servers. The system can be seamlessly scaled out to hundreds (or more) physical servers, with 10s to 100s of petabytes of storage capacity. RING provides data protection and resiliency through local and geo-distributed erasure coding and replication, with services for continuous self-healing to resolve expected failures in platform components such as servers and disk drives.

RING has no single points of failure and requires no downtime during system upgrades, scaling, planned maintenance or unplanned system events. It continues to operate normally throughout these events, and can automate rebuild or repair (self-heal) processes for data affected in the event of a disk or server error or failure.

RING can independently scale out its access layer of protocol endpoints ("connectors") to enable higher aggregate performance for the application workload. RING software can be hosted on standard x64 servers with a choice of popular Linux distributions, with no kernel modifications. This approach also decouples the need to maintain hardware compatibility lists (HCLs) and paves the way for bringing support of new platforms to market much faster.

The use of commodity components also extends to the network elements with 10/25/40/100GbE interfaces acceptable for both the external connector interfaces and the internal RING interconnect fabric. This flexibility makes it possible to deploy capacity-optimized RINGs or performance-optimized RINGs as needed to fit specific business requirements. In all cases, the RING software abstracts the underlying physical servers and hard disk drives and can exploit the lower-latency access characteristics of SSD (flash) storage to maintain its internal metadata. RING is designed to scale out over time across various hardware vendors, server generations and densities expected as a normal part of the RING platform lifecycle.



Hybrid-cloud, core and edge data

RING has introduced multi-cloud data management capabilities to address the oncoming need for unified storage management across object storage solutions and public cloud services. Support for cloud storage services from AWS, Azure, Google and others provide opportunities for customers to have their choice of the most appropriate cloud for their applications and data.

RING provides intelligent data and metadata management services that can unify these disparate and heterogeneous storage silos and clouds under a single control plane and namespace. It also provides advanced data mobility, ease-of-use and global visibility to protect, search and process data across RING and cloud storage.

RING integrates multi-cloud technologies with capabilities for managing and mobilizing data across multiple RINGs and public clouds. These capabilities are designed to support emerging hybrid-cloud data management use cases including:

RING to cloud for data protection and business continuity:

- Mirror RING data cloud (all or partial) for disaster recovery

RING to cloud for compute bursting or data analysis:

- Move RING data to the cloud for use with services such as compute bursting, analysis, etc.

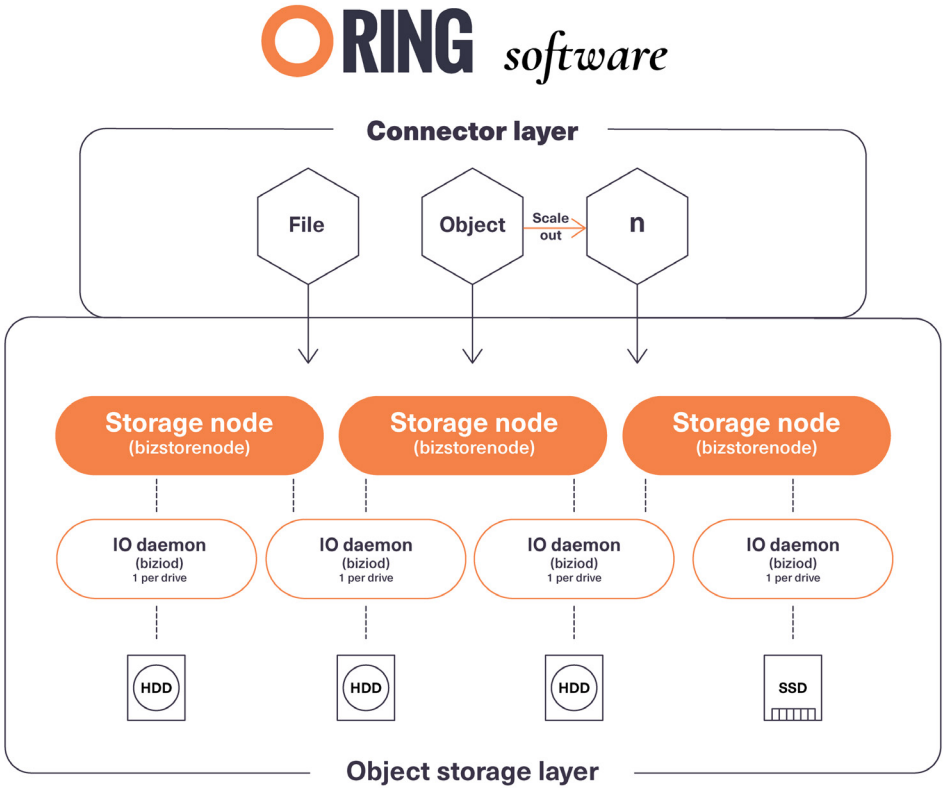
RING to cloud archive:

- Free up RING capacity while maintaining data for compliance or regulatory purposes
- Offload old/unused RING data to inexpensive cloud cold storage archive

RING hybrid-cloud data management capabilities are described in further detail in this paper.

II. RING architecture

RING provides a set of distributed software services that abstract the underlying physical servers and disks into a single, scalable and resilient storage service. At the top of the architecture stack, a layer of stateless access protocol services (connectors) provides storage interfaces for applications. Middle layers consist of distributed metadata services for both the S3 object storage and for the virtual file system layer, data protection mechanisms to ensure data durability and integrity, self-healing processes, and a set of systems management and monitoring services. At the bottom of the stack, the system is built on a distributed storage layer composed of virtual storage nodes and underlying IO daemons that abstract the physical storage servers and disk drive interfaces. RING management and monitoring interfaces include the Supervisor UI management portal, command line interface (CLI) and a family of APIs.



At the heart of the storage layer is a scalable, distributed object key/value store based on an advanced peer-to-peer routing protocol. This routing protocol ensures that store and lookup operations scale efficiently to very high numbers of nodes. The underlying protocol and related keyspace mechanisms are described in this paper.

Design principles

Scality has designed RING in alignment with the design criteria spearheaded by hyperscale cloud-scale service providers, such as AWS, Azure and Google. RING leverages distributed system designs that are hosted on commodity, mainstream hardware along with the following key tenets:

- 100% parallel design for metadata and data to enable scaling of capacity and performance to unbounded numbers of objects — with no single points of failure, service disruptions, or forklift upgrades as the system grows
- Multi-protocol data access to enable the widest variety of object, file and host-based applications to leverage RING storage
- Flexible mechanisms to efficiently and reliably protect a wide range of data types and sizes
- Self-healing from component failures to provide high levels of data durability — the system expects, tolerates and automatically resolves failures
- Hardware freedom to provide optimal platform flexibility, eliminate lock-in and reduce TCO

RING connectors

RING connectors provide data access endpoints and protocol services for applications that use RING for data storage. As a scale-out system, RING supports any number of connectors and endpoints to support large and growing application workloads.

RING provides the following integrated object and file protocol interfaces:

- **AWS S3 API:** A comprehensive implementation of AWS S3 REST API with support for the bucket and object data model, AWS-style access key pairs, Signature v4/v2 authentication and an AWS compatible model of identity and access management (IAM)
- **REST (sproxyd):** RING's native key/value REST API provides a flat object storage namespace with direct access to RING objects
- **NFS v4.0 and v3:** Volumes presented as standard mount points
- **SMB 3.0:** Volumes presented through Samba shares to Microsoft Windows clients
- **FUSE:** Volumes presented as a local Linux file system

To support multiple use cases, a RING deployment may simultaneously use a mix of file and object connectors, and any number of them needed to deliver the desired performance.

Connectors provide storage services for read, write, delete and lookup for objects or files stored into the RING based on either object or POSIX (file) semantics. S3 connectors are stateless, so applications can use multiple connectors in parallel to scale out the number of operations per second or the aggregate throughput of the RING. For further details on load balancing for both S3 and file system access, see the *RING deployment* section below.

RING software stack and requests flow

I/O requests from applications first access the RING at the connector level. Connectors dispatch these requests to the RING storage nodes. Connectors are also responsible for applying the configured data protection storage policy (replication or erasure coding). When a new object is created (written), the connectors will divide objects that are above a configurable size into smaller objects (chunks) before they are sent to the storage nodes. The storage nodes, in turn, write these data chunks to the underlying storage nodes and IO daemons.

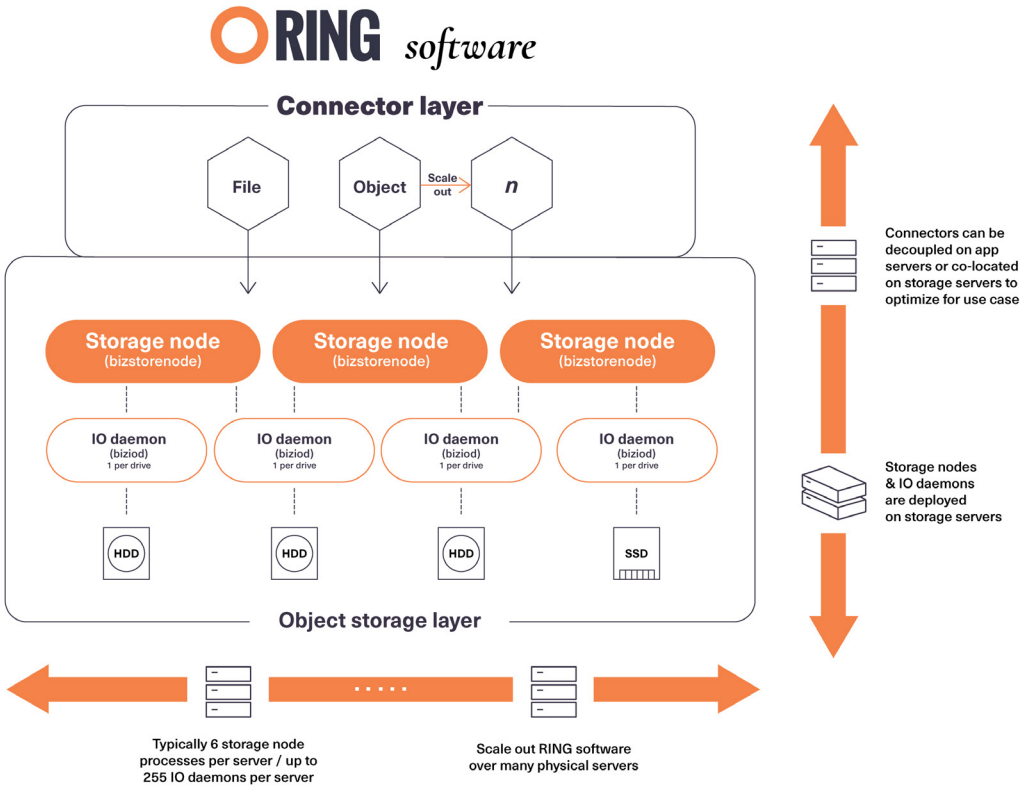
Storage nodes and IO daemons

At the heart of the RING are the storage nodes — virtual processes that own and store a range of objects associated with their portion of the RING's keyspace (described in the next section).

Each physical storage server (host) is typically configured with six (6) virtual storage nodes (bizstorenode). Under the storage nodes are the storage daemons (biziod), which are responsible for persistence of the data on disk in an underlying local standard disk file system. Each biziod instance is a low-level software process that manages the IO operations to a particular physical disk drive and maintains the mapping of object keys to the actual object locations on disk. Biziod processes are local to a server, managing only local, direct-attached storage and communicating only with storage nodes on the same server. The typical configuration is one biziod per physical disk drive, with support for up to hundreds of daemons (up to 255) per server, so the system can support very large, high-density storage servers.

Each biziod stores object payloads and metadata in a set of fixed size container files on the disk it is managing. With such containerization, the system can maintain high-performance access, even to small files, without fragmentation or over-allocation of file system resources. The bizoid daemons leverage low-latency flash (SSD or NVMe) devices to store index files for faster lookup performance. The system provides data integrity assurance and validation through the use of stored checksums on the index and data container files, which are validated upon read access to the data. The use of a standard file system underneath biziod ensures that administrators can use normal operating system utilities and tools to copy, migrate, repair and maintain the disk files if required.

The recommended deployment for systems that have both HDD and SSD media on the storage servers is to deploy a data RING on HDD and the associated metadata in a separate RING on SSD. Metadata is typically 1-2% of the capacity of the actual data, but it is dependent on application-specific metrics such as number and average file size (Scality tech support will provide actual sizing recommendations).



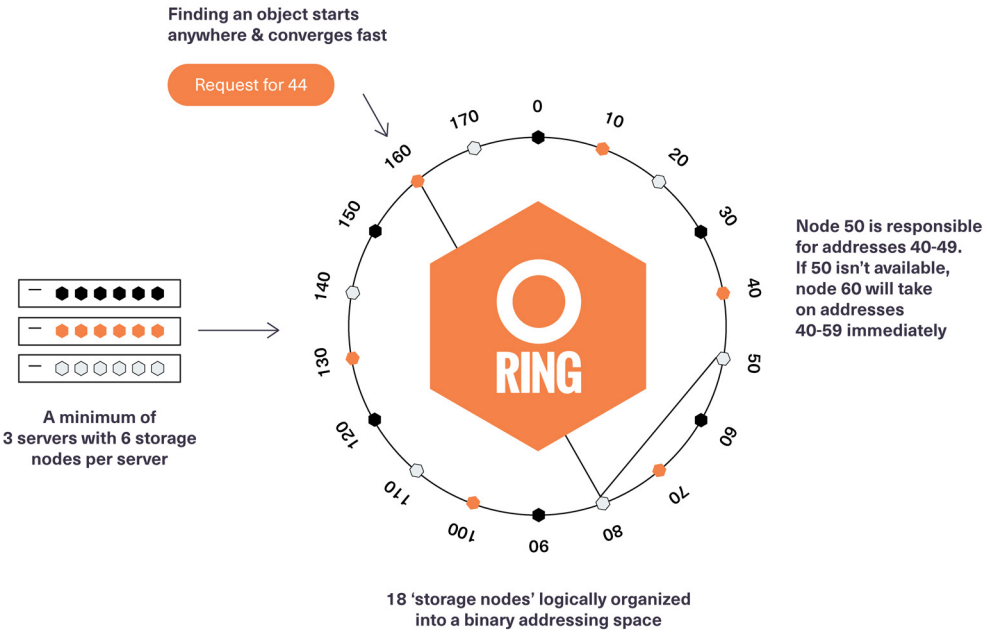
Distributed routing protocol and RING keyspace

Large distributed systems, such as RING, depend on fast and efficient routing of requests among high numbers of nodes. Many mechanisms exist for performing these operations — including centralized routing approaches that can optimize locking and conflict detection — but these do not scale effectively and can present bottlenecks in performance and suffer from central points of failure. The opposite approach — a distributed broadcast model — can partially eliminate these bottlenecks but is limited in practice due to the number of changes that must be reflected in the system's topology.

In response to these issues, the research community has proposed a set of efficient routing protocols, including a set of second-generation peer-to-peer protocols like MIT's Chord protocol. Chord is highly responsive to changes in system topology, such that these changes do not require broadcasting to all nodes, but only to a few relevant nodes. This enables the algorithm to work efficiently in very large clusters.

RING architecture is based on Chord, a foundation for distributed storage that can hyper-scale to hundreds of billions of objects. Scality has augmented and patented the basic Chord protocol to enable high levels of data durability, high performance, self-healing and simplified management. The basic algorithm arranges storage nodes along a logically circular keyspace ("RING") with each node being assigned a fraction of this keyspace. Each node then owns the range of keys bounded by its own key (up to the key before its successor node). RING is able to quickly and efficiently route requests for a given key from any node to the node that owns the key. The number of lookups scales sub-linearly and deterministically for very large numbers of nodes and massive storage capacity. For example, in a 1000-node system, a maximum of five lookup "hops" are required to find a key, and caching will reduce the number after the first lookup to the key.

The figure below shows an example with a RING on three (3) physical servers. To subdivide the keyspace more effectively across physical capacity, each physical server is assigned a set of at least six (6) virtual storage nodes. These storage nodes are then logically arranged into the circular keyspace according to their assigned key value. In this simplified example, storage node 50 is responsible for storing keys ranging from 40 to 49. If storage node 50 departs the RING (either intentionally or due to a failure), its keys will be automatically reassigned and rebalanced to its successor, the storage node with key 60 (which then assumes responsibility for keys in the range 40 to 59).



RING is dynamic, with the ability to adapt rapidly to changes in the keyspace as a result of new nodes joining or departing the cluster. Without service disruption, the system automatically rebalances keys in the keyspace on node additions and departures. Rebalancing can move the set of keys owned by a node to the new node(s) now assigned to the affected addresses in the keyspace, or it can move data that was owned by a departing node to its previous neighbor node. During rebalancing, the system maintains full data access by routing around changes in the keyspace, establishing proxies or alternate paths to data on other nodes until the rebalancing process is complete.

RING keyspace and the distributed hash table

Underpinning the keyspace algorithm is Scality's Distributed Hash Table (DHT keyspace table, in references below) implementation, which provides the routing mechanism for locating objects on RING.

The keyspace table is distributed among the nodes assigned to the keyspace. An important aspect of the table is that it is decentralized — the partition of the keyspace table on a node only knows its own key range, a few neighboring nodes (its immediate predecessors and successors), and a few additional nodes at well-known geometric projections across RING.

Importantly, the keyspace table does not represent a centrally shared metadata store of the keyspace — it merely captures the local node's knowledge of a subset of RING so that lookup operations can efficiently compute the next-best estimate of the location of a key on other nodes, until it is found. While multiple hops may occur during key lookups, the algorithm uses knowledge of predecessor and successor nodes to locate the node deterministically and with low latency (typically in milliseconds). These node lookups do not require disk operations; they merely require navigating the keyspace table across a sequence of nodes.

Key format and class of service

Scality organizes the keyspace using its own key format, consisting of 160-bit keys:

- The first 24 bits of each key serve as a randomly generated dispersion field (to avoid inadvertent collisions or convergence of a set of related keys)
- The next 128 bits represent the location of the object payload
- The last 8 bits represent the class of service (CoS - or replication number) of the object associated with the key

RING provides highly flexible object data protection through:

- CoS: Classes of Service, for the number of stored object copies from 1 to 6 (CoS 0 to 5)
- EC: Flexible erasure-coding implementation

For example, replication CoS=5 allows the system to tolerate up to 5 simultaneous disk failures, while still preserving access and storage of the original object. Any failure causes the system to self-heal the lost replica from one of the surviving copies, to automatically bring the object back up to its original class of service as fast as possible.

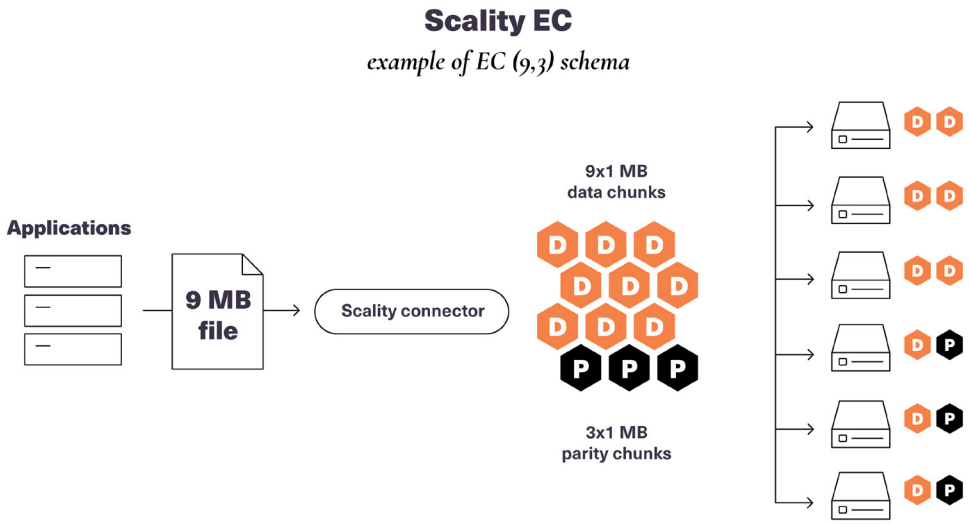
While replication is optimal for many use cases where the objects are small and access performance is critical, it does impose high storage overhead compared to the original data. For example, a 100 KB object being stored with a CoS=2 (2 extra copies so 3 total) consumes 3 x 100 KB = 300 KB of actual physical capacity on RING.

This overhead is acceptable in many cases for small objects, but it can become a costly burden for megabyte- or gigabyte-level video and image objects — in this case, a 200% penalty to store a 1 GB object, since it will require 3 GB of underlying raw storage capacity for 3 replicas. When measured across petabytes of objects, the cost burden is substantial for many businesses, necessitating a more efficient data protection mechanism.

RING erasure coding

RING supports erasure coding (EC) to provide an alternative data protection mechanism to replication (CoS) that is much more space efficient for large objects and files. RING implements Reed–Solomon erasure-coding techniques to store large objects with an extended set of parity stripes, instead of multiple copies of the original object.

The basic idea with erasure coding is to break an object into multiple stripes (m) and apply a mathematical encoding to produce an additional set of parity stripes (k). A description of the mathematical encoding is beyond the scope of this paper but can be conceptually understood as an extension of the XOR parity calculations used in traditional RAID. The resulting set of stripes (m + k) are then distributed across the RING nodes, providing the ability to access the original object as long as any subset of m data or parity stripes are available. Stated another way, this provides a means to store an object with protection against k failures, with only k / m overhead in storage space.



To provide a simplified example, assume a 9 MB (object is to be stored using an EC 9+3 erasure coding schema. RING will calculate 3 additional 1MB EC parity stripes to be stored. This is, therefore, 33% space overhead (3MB/9MB) with protection against three simultaneous disk failures — significantly less than the 200% space overhead that would be required to store 3 replicas of the same object using CoS=3 replication.

More specific to the actual implementation in RING, large objects will be decomposed into 4MB stripes that are individually erasure-coded as per the configured EC schema. So, in the case of a 100MB object, it is decomposed into 25 stripes (25 x 4MB), and each 4MB stripe has EC applied to calculate the parity stripes. Based on the configured EC schema, the net space overhead is the same as the conceptual example above, but it is performed at a more granular level for data distribution and protection across the system.

With RING EC, the data stripes are stored in the clear, without any encoding, so there is no performance penalty from decoding during normal read accesses. This means that EC data can be accessed as fast as other data unless a data stripe is missing, requiring a parity stripe to be accessed and decoded.

Note that other commercial storage solutions impose a performance penalty on reading objects stored through erasure coding because all of the stripes, including the original data, are encoded before they are stored. This approach requires mandatory decoding on all access to objects all the time, even when there are no failure conditions on the data stripes.

In summary, RING replication and EC data protection mechanisms provide very high levels of data durability with the ability to trade off performance and space characteristics for different data types.

Intelligent data durability, fast rebuilds and self-healing

For flexibility in storing mixed-size workloads, RING is usually configured with a mixed CoS/EC data protection policy, with an object-size threshold to determine CoS (for objects below the size threshold) and EC (for objects above the size). By default, this size threshold is 60 Kbytes, but can be configured to optimize for specific file size distributions and durability requirements. Note that each connector may also have an independent CoS/EC policy threshold to optimize for workloads with different object sizes.

RING is designed to expect and maintain service during a wide range of failures — including disk, server, network and even entire data center failures — while ensuring that data remains protected during these conditions. RING provides self-healing processes that monitor and automatically resolve component failures. This includes rebuilding missing data stripes due to disk drive or server failures, rebalancing data when nodes leave and join the RING, and proxying requests around component failures. If a disk drive or even a full server fails, background rebuild operations are spawned to restore the missing object data from its surviving replicas or EC stripes. The rebuild process completes when it has restored the original class of service or the original number of EC data and parity stripes.

To optimize rebuilds as well as mainline IO performance during rebuilds, RING uses the distributed power of the entire storage pool. The parallelism of the underlying architecture eliminates central bottlenecks that might otherwise limit performance or cause contention between servicing application requests and normal background operations, such as rebuilds, especially when the system is under load.

To further optimize rebuild operations, the system only repairs the affected object data, not the entire set of disk blocks, as is commonly the case in RAID arrays. This is impactful in practice, especially on new or partially utilized systems since RING would only repair a small amount of data in seconds/minutes while a RAID array would repair an entire disk drive sector-by-sector (which may require hours to days in the case of a multi-TB disk drive).

Rebuilds are distributed across multiple servers and disks in the system, harnessing the aggregate processing power and available IO of multiple resources in parallel, rather than serializing the rebuilds onto a single disk drive. By leveraging the entire pool, the impact of rebuilding data stored either with replication or EC is minimized since there are relatively small degrees of overlap between disks involved in servicing data requests and those involved in the rebuilds.

A local disk failure can also be repaired quickly on a storage node (distinct from a full distributed rebuild) through the use of an in-memory key map maintained on each node. Storage nodes are also responsible for automatically detecting mismatches in their own keyspace, for rebalancing keys, and for establishing and removing proxies during node addition and departure operations. Self-healing provides RING with the resiliency required to maintain data availability and durability in the face of many failure conditions, including multiple simultaneous component failures at the hardware and software process levels.

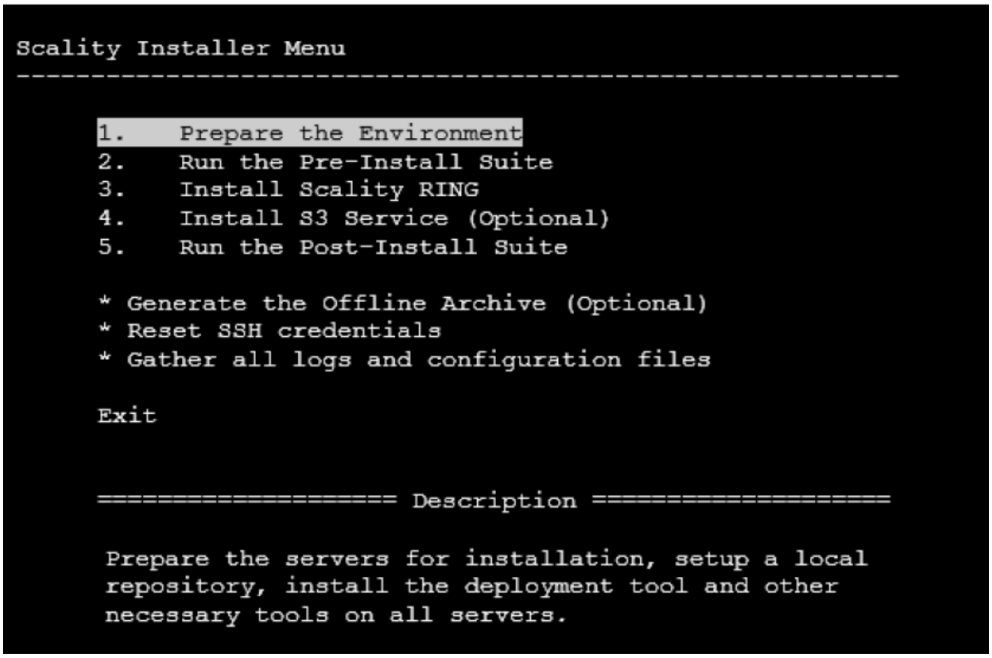
III. RING deployment

Scality software deployment tools are designed to be flexible for multi-platform environments. The software is designed for standard x64 storage server hardware platforms and the most popular modern Linux flavors: Redhat Enterprise Linux (RHEL) and CentOS-compatible distributions Rocky and Alma.

The installer provides a centralized and automated approach for RING deployments. The installer uses a simple JSON-type configuration and a step by-step approach, leading the user through the different steps of the installation process. The installer uses a *platform description file*, which contains the most important configuration information about the target environment, such as IP addresses, each host's role assignments (connector, storage node or both), and hardware details. The platform description file is an extract of Scality sizing tools that are used to describe the full architecture. The file contains the host bill-of-materials (BOM) information, such as the amount of RAM, NICs, disks and RAID controllers. Internally, the system configuration is captured in an internal database that maintains a full inventory of the platform including servers, disks, connectors and other components. This inventory is used to install, configure and manage the system throughout its lifecycle.

The installer follows five steps:

1. **Prepare the environment:** Verifies all host servers and deploys the cluster management tooling on all servers
2. **Pre-install suite:** Verifies that the server's operating systems configurations follow Scality's deployment best practices and are specified according to the hardware components listed in the platform description file
3. **Install Scality RING:** Deploys all RING software components, such as the supervisor, the SOFS connectors and the storage components
4. **Install S3 services:** Deploys all S3 components of the configuration
5. **Post-install suite:** Validates that the installation has been successful, including checking that all components are started and working well together



Special note for 3-server RING deployments:

RING is supported on an entrypoint three-server deployment, which can be expanded to six nodes. Such RINGs offer significant hardware cost savings, while still providing extremely high data durability and protection. Three-server RING deployments have some specific software and hardware platform configurations as described in the table:

Component	Description / Notes
Platform	Three storage servers and one virtual machine (for Supervisor)
Operating system	Popular Linux distributions supported
Connector types	S3, SMB or NFS
Number of connectors	One active (default) plus N passive (optional)

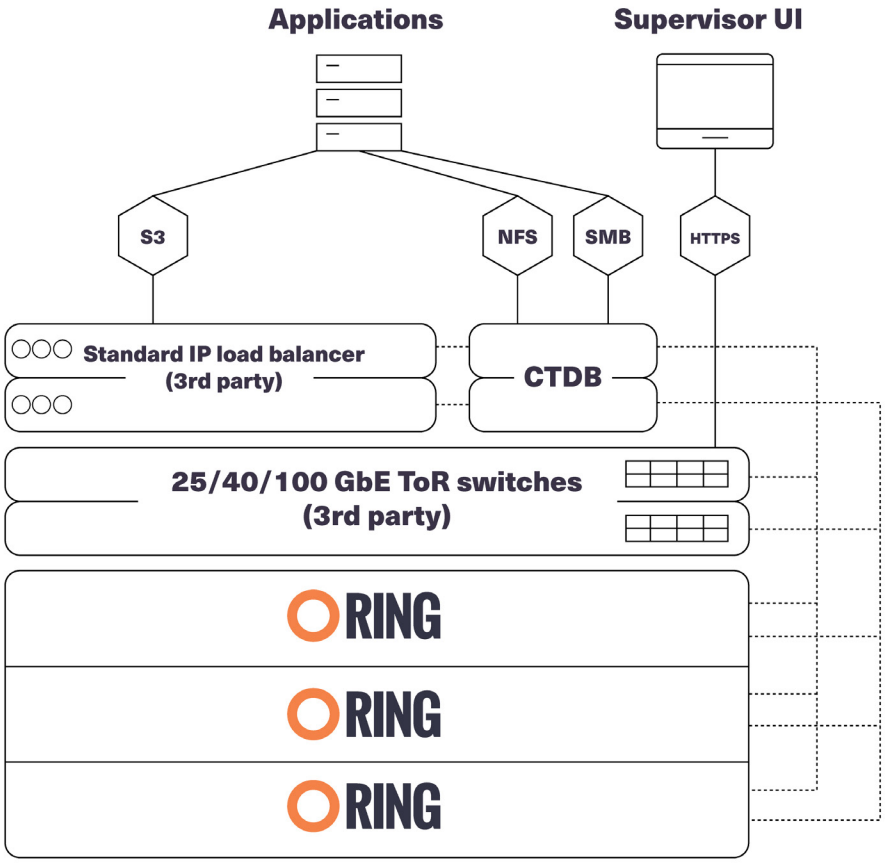
Function	Description	Supported
Multi-site options	1 site	Yes
-	2-site stretched	No
-	3-site stretched	No
-	2-site asynchronous replication	Yes
Data durability	Replication (CoS)	CoS=2 recommended
-	Erasure coding (EC)	4+2 recommended
Expansion	Grow capacity within cluster	Add drives to storage servers
-	Grow capacity for a full cluster	Scale to 6 storage servers (contact Scality Support)

Network and load balancers

RING uses IP network connectivity for access from client/user applications and for cluster inter-communications. Network configurations are customer deployment-specific, and details will be specified in supporting technical documents. However, a typical network configuration uses top-of-rack 25/40/100GbE switches for connectivity to the user network (LAN), for connectivity of storage servers to the cluster and, optionally, for connectivity to other racks for additional capacity.

S3 connectors are stateless so that client requests can be load balanced across multiple endpoints. For S3, RING is typically deployed with standard third-party software or hardware load balancers to balance traffic across multiple S3 endpoints for scale-out and high-availability/failover. In addition, load balancers can provide SSL termination for secure S3 connections, or they can be configured for SSL passthrough for unencrypted object operations. Some load balancers have a QOS function, if needed, which is especially useful from source IP.

NFS and SMB connectors can also scale-out for higher aggregate throughput. Scality recommends the use of CTDB (clustered configuration database on Linux) for NFS/SMB Virtual IP (VIP) management and the integrated Distributed Lock Manager (DLM) as the mechanisms for load balancing and arbitrating shared access for RING file system interfaces.



IV. RING systems management and monitoring

Managing and monitoring RING is enabled through a cohesive suite of user interfaces, built on top of a family of RESTful interfaces termed the Supervisor API (SupAPI). SupAPI provides an API-based method that may be accessed from scripts, tools and frameworks for gathering statistics, metrics, health check probes and alerts, and for provisioning new services on RING.

SupAPI also enables role-based access control (RBAC) by supporting an administrator identity to provide access control privileges for super admin and monitor admin user roles. The system supports real-time systems monitoring through an internal database for capturing events, alerts and more. Monitoring alerts can be configured to notify customers through a variety of mechanisms, including email, messaging, pagers and SNMP traps.

RING provides a family of tools that use SupAPI for accessing the same information and services. RING includes Scality Supervisor, a browser-based portal for both systems monitoring and management of Scality components. The Supervisor supplies capabilities across object (S3) and file (NFS, SMB, FUSE) connectors, and it provides integrated dashboards that include such key performance indicators (KPIs) as: global health, performance, availability and forecast. The Supervisor also includes provisioning capabilities to add new servers in the system and a new zone management module to handle customer failure domains for multi-site deployments.

A scriptable command line interface (CLI) called RingSH is provided, as well as an SNMP-compatible MIB and traps interface for monitoring from standard consoles. RING is designed to be self-managing and autonomous so administrators are freed up to work on other value-added tasks, without worry about the component-level management tasks common with traditional array-based storage.w

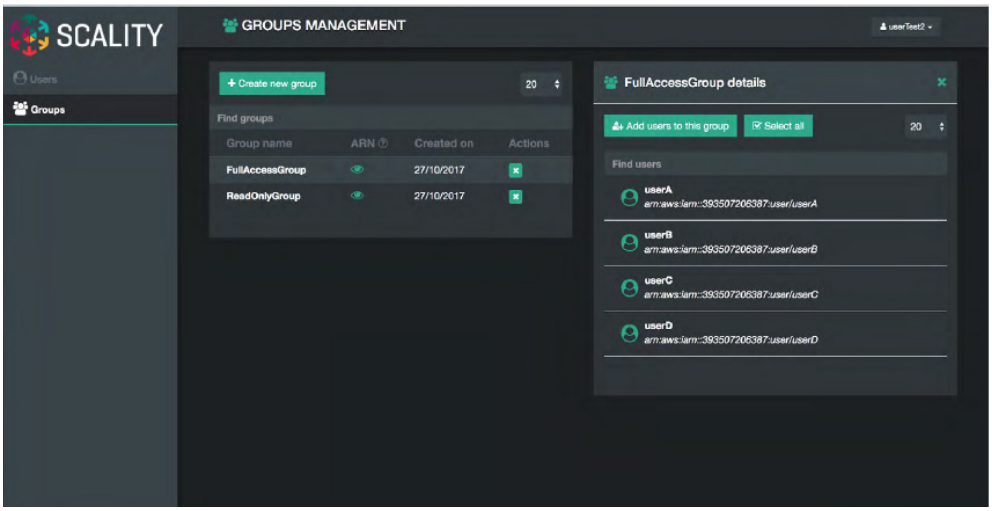
RING Supervisor provides a top level dashboard with a view of three logical layers:

- Storage services (S3 object storage or SOFS file view)
- RING view (logical data and metadata RING view with usable/free capacity view)
- Hardware (platform view, with drill down access to server, disk health metrics and KPIs)

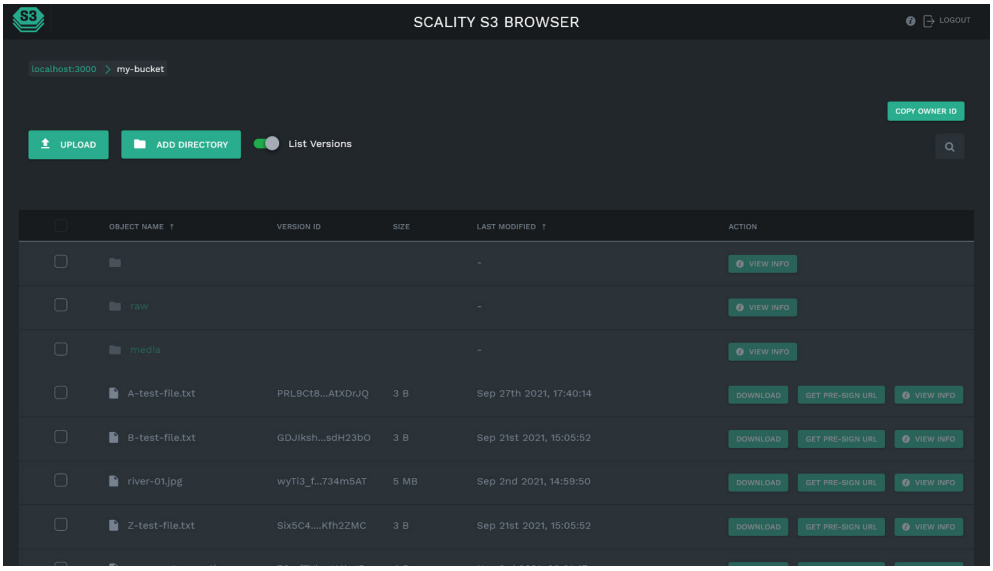
Each layer provides sub-pages with additional views and details. For example, the S3 service view (above) shows the status of the services, its global health, distribution of S3 API requests, response codes and more.

Supervisor UI also includes an advanced monitoring dashboard where all collected metrics can be graphed and analyzed on a component-level (per server and per disk) This dashboard is based on a powerful graphing engine (leveraging Grafana/Kibana & ElasticSearch) that has access to detailed metrics with graphing/trending capabilities, including network statistics, and S3 service-specific information related to metadata, S3 response and error codes, and asynchronous replication.

An S3 Service Management portal is provided to manage the integrated AWS identity and access management (IAM) model of S3 multi-tenancy for RING. This provides tiered (two-level) management of accounts, users/groups and IAM access control policies. The S3 Console can also be customized/ rebranded for white-labeling by service providers.



The S3 Browser is an S3 API client that runs on the S3 user browser and is accessible to both the storage administrator and the S3 end user. It enables browsing S3 buckets, uploading and downloading of object data, and managing key S3 features such as bucket versioning, replication, object locking, editing of metadata attributes and tagging. The browser also provides an end-user accessible metadata search capability to locate objects within a bucket based on attribute values.



V. S3 object storage

RING S3 Connector provides a modern and comprehensive Amazon S3-compatible application interface to the Scality RING. The S3 API has become the industry's default cloud storage API and emerged as the standard RESTful dialect for object storage (as NFS was for the NAS generation).

S3 Connector uses a scale-out architecture to support very high levels of application workloads and concurrent user access. Underlying the S3 Connector is the S3 metadata service built as a reliable consensus cluster, designed for high-performance S3 access, high availability and strong consistency. The S3 metadata service is optimized for fast S3 bucket and object PUT, GET, DELETE and bucket listing operations at massive scale to billions of objects.

Core S3 API operations supported include:

- Bucket and object level PUT, GET, DELETE, HEAD operations
- Efficient multi-part upload (MPU) for large objects
- Bucket versioning

- Bucket lifecycle management (expiration policies)
- Bucket cross-region replication (asynchronous replication)
- Bucket object locking (WORM/data immutability)
- Bucket CORS (cross-origin resource sharing)
- Bucket website
- Bucket/object level encryption
- Bucket notification (asynchronous notifiers on bucket updates)

The list of supported S3 APIs is updated regularly and available in the RING technical publications API guide.

RING S3 deployments can be geo-replicated to enable highly available disaster recovery solutions for both metro-area network environments (synchronous stretched deployments), as well as asynchronous replication (AWS CRR) of individual S3 buckets or a full site (see full details in the *multi-site RING deployments* section below).

To provide the capabilities and user experience of AWS S3 more fully, RING S3 also provides an implementation of the AWS multi-tenancy and identity management (IAM) model, including Accounts, Users, Groups, Roles and User/Group/Bucket policies for access control. RING S3 capabilities use AWS-style key/secret key pairs, and authentication of S3 API requests is implemented using AWS Signature V4 (or V2)-style HMAC authentication (see more details in the *RING security* section below).

For large enterprises that commonly leverage popular directory services, RING S3 IAM supports federated authentication to LDAP and Active Directory to integrate into the enterprise IT security environment.

RING S3 provides utilization tracking to enable reporting, billing, and chargeback. RING provides an API (UTAPI or Utilization API) to provide extended metrics tracking. UTAPI can track metrics/KPIs at four levels: RING, Account, User and per-Bucket level.

At each level, it tracks:

- Utilized capacity (TB), number of objects
- Bandwidth in/out (MB/sec per unit time)
- S3 operations (GETs, PUTs, DELETEs per unit time)

RING S3 enables administrators to set account-level quotas on their S3 storage space. This feature enhances the flexibility of storage system space management by allowing multiple users or business entities to use the same storage platform without overconsuming space. Simply specify the amount of space to be allocated to an account, then allow each account to use storage resources as needed — without fear of user overconsumption. Reporting and visualization of S3 Utilization metrics are provided through the Supervisor UI.

S3 scale-out architecture

RING S3 is implemented and deployed as a distributed set of containerized services, including the S3 Cloud Server (API Service), IAM Vault (IAM), S3 Metadata (clustered database) and other services, such as the utilization API (UATPI). These distributed services are primarily stateless, allowing the system to scale out with standard IP/HTTP style load balancing across external service interfaces. The key exception to these stateless services is the S3 Metadata service, which is a stateful replicated database for storing bucket and object metadata.

RING S3 is distributed and redundant, with no single points of failure. The key service that determines availability is the S3 Metadata service — a stateful service that provides lookups to bucket and object locations (as well as persistent storage for S3 Vault objects such as accounts, users, groups and policies). Because the metadata service is critical to service availability, it is implemented using a distributed consensus protocol to ensure strong consistency with failure tolerance and restart of servers in the cluster.

To understand the high availability and failover model, we must understand the interactions of the other services with the metadata service, by outlining the data flows through the S3 Connector. Note that S3 services (containers) may be deployed on the same hosts, with localhost connections between the S3 Cloud Server, S3 Vault, S3 Metadata and Sproxyd containers, or can be decoupled across hosts to provide additional flexibility and performance.

In all cases, IO commands through the S3 Connector can be characterized with the following general data flows:

1. A client application sends an http API request to load balancer (in most production deployments this will be a dedicated hardware LB).
2. Load balancer routes the request to one of the S3 Cloud Server API endpoints.

   a. request is an S3 API PUT, GET, HEAD or DELETE command.

   b. request will also require authentication, based on an HMAC signature computed as per AWS Signature v4 or v2
3. S3 API service receives the request and calls the S3 Vault service

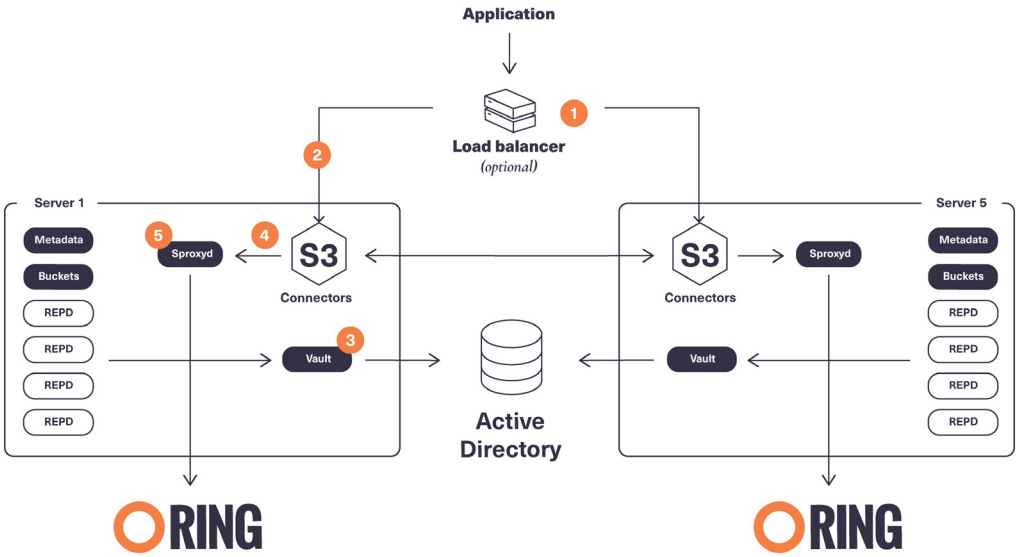
   a. Vault service performs authentication and access control (policy) checks.
4. The S3 API service sends the request to the S3 Metadata service to perform bucket and object lookups.

   a. Metadata service locates the bucket through the bucket daemon (bucketd), which has cached information about the bucket leader.

   b. The bucket daemon contacts the metadata leader daemon (repd) for the specified bucket

5. The S3 API service calls the RING REST interface (sproxyd) service to process the underlying PUT, GET or DELETE action on the data RING.
6. The S3 API service returns a response code to the requesting application

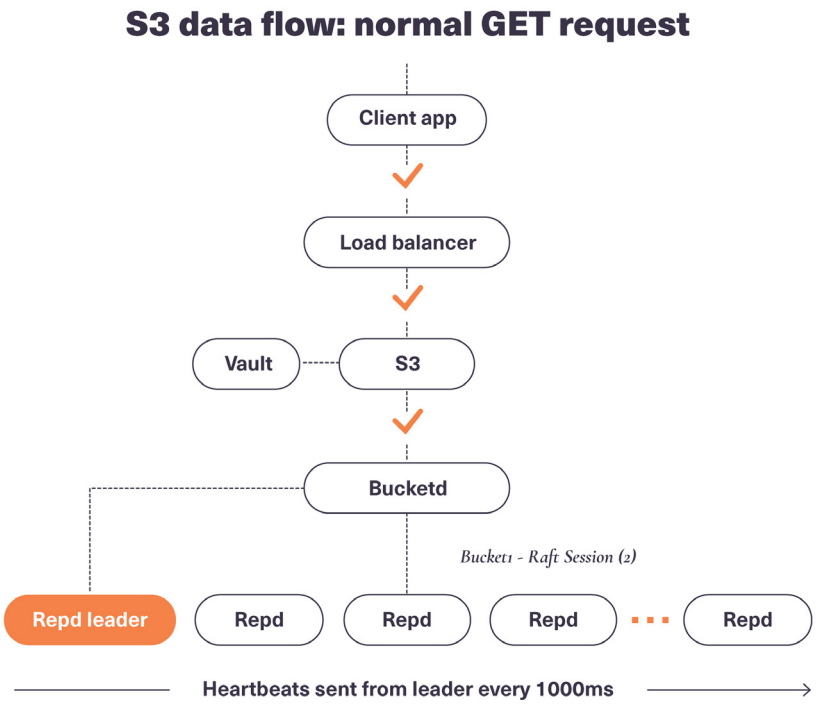
   a. http 200 OK, or http 500 for an error



S3 metadata service

The metadata service internally consists of two daemon services, termed *bucketd* and *repd*. Bucketd maintain and cache the mapping of buckets to metadata clusters (internally termed “Raft sessions”) and track the leader (repd) for each given session for requests by the servers.

The following diagram shows a normal GET Request flow from the client application, through the load balancer, and to the S3 server to the leader. Bucket IO requests flow through bucketd to the current metadata leader.



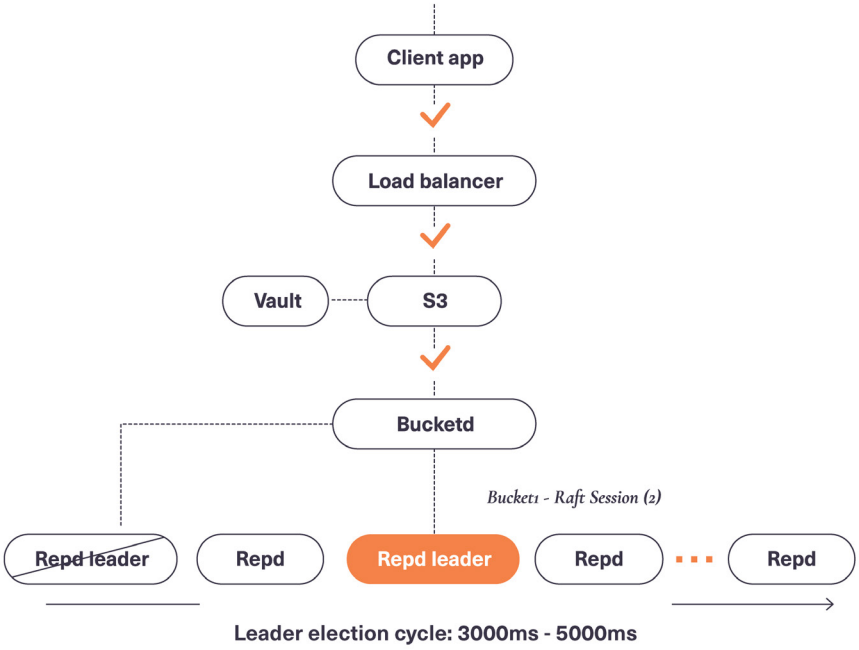
Standard GET request flow through the S3 Connector stack

The leader repd receives requests to the specified bucket and performs the following key tasks:

- **GET operations:** The leader retrieves the metadata for the operation and returns it to the requesting S3 server for data access and to return results "up the stack" to the client.
- **PUT operations:** The leader updates metadata to its local key/value database and replicates all updates to the other repds in the cluster. To ensure strong consistency is maintained, a majority of 3 repds (leader plus at least two other repds must reply to the update request) before the leader returns control to the S3 server.
- The leader periodically checks the heartbeat between the repds in its session with a default frequency of 1000 ms (1 sec). A sequence of missed heartbeats from the leader will invoke a repd to participate in a new leader election, as described further below.

When enough buckets are managed, all repds will be assigned the leader role at some point. Leader election in the metadata cluster is a dynamic, recurring, standard operation to the underlying consensus protocol. Any failures of the current leader are resolved quickly and automatically through subsequent leader election cycles, maintaining service availability. Knowledge of newly elected leaders is pushed to the bucketd daemons through an event-driven mechanism, eliminating the need for polling (pull mechanisms) and resulting timeouts that can extend failover time. This ensures that bucketd daemons are quickly aware of the new leader for a given bucket.

New leader notification via event-driven mechanism

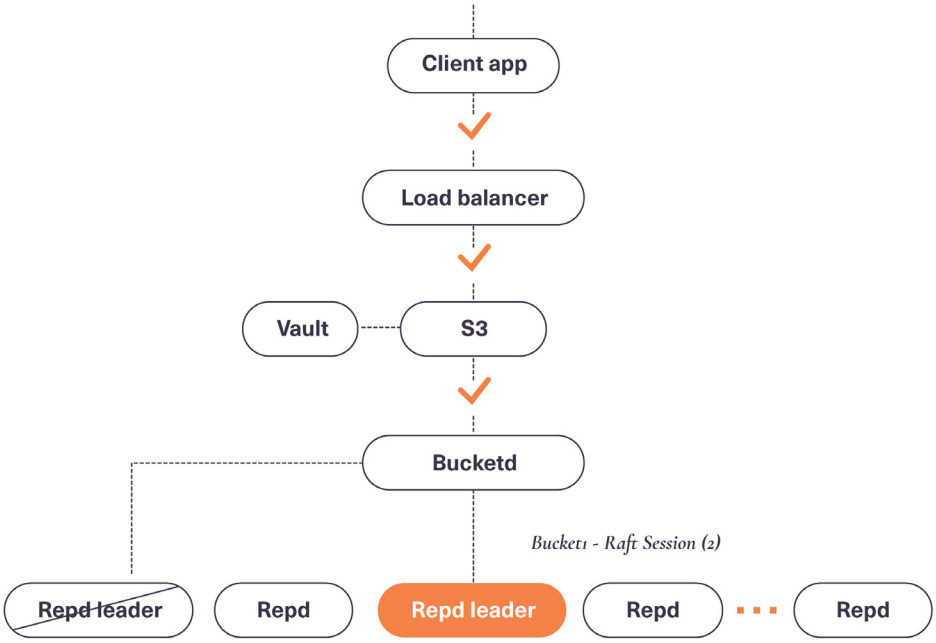


New leader election and push-alert mechanism

The repds in a session expect regular heartbeats from the leader as described above. If a repd does not receive a heartbeat for a defined time interval (defaulted to 3000 ms), the repd attempts to become elected the new leader. However, to resolve a condition where two repds are vying to become the leader at the same time, a second interval of 5000 ms is defined to allow another election cycle to conclude.

Therefore, a leader election cycle averages slightly over three seconds with an upper bound of five seconds, which you can expect as the overall failover response time of S3 Connector requests if the leader fails during an IO request. Once the new leader has been elected, IO continues as usual.

Normal IO continuation after new leader election



Through this distributed architecture, RING S3 delivers the scale and performance needed to address enterprise and service provider requirements, including support for millions of accounts, buckets, users and objects. Performance can be scaled out to provide higher aggregate performance as the number of buckets increases.

VI. Scale-out file system (SOFS)

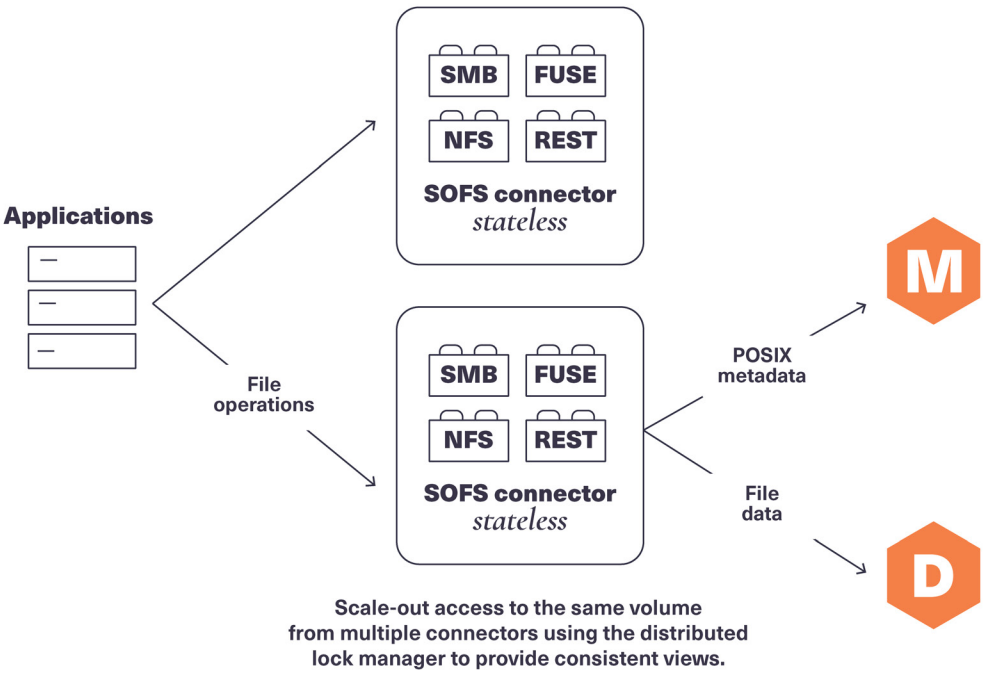
A major advantage of RING is the combination of a scale-out object store with an integrated scale-out file system (SOFS). This union allows customers to deploy a single RING with both object and file presentations simultaneously, which can help minimize multiple storage silos — for example, a separate NAS and object storage solution — and simplify storage management.

SOFS is a POSIX-compatible, parallel file system that is accessed using popular and standard NFS (v4.0 and v3), SMB 3.0 and FUSE file interfaces. SOFS provides file system services that can scale seamlessly without any

predefined limits on the amount of physical resources (storage servers), endpoint (NFS, SMB, FUSE) or storage services that can be presented and consumed by client applications.

Technically, SOFS is implemented as a virtual file system that uses a distributed metadata database directly integrated into the core RING P2P architecture and services. The database provides the file system hierarchical view (directories/sub-directories) and stores file system structures such as inodes, links and POSIX metadata attributes. Through the database, the file system is always consistent in that updates are atomic (either committed or rolled back entirely). This guarantees the file system is never left in an intermediate or inconsistent state.

By integrating SOFS into the core architecture, file lookups are performed using the standard RING routing protocol. For fast access performance, SOFS metadata is stored on flash/SSD drives in the storage servers. SOFS file payloads are stored in the data RING typically on hard disk drives/HDDs (flash is supported for data as well for performance intensive workloads). SOFS works directly with the same RING data protection and durability mechanisms described previously, including replication (CoS) and erasure coding schemas.



To simplify management and configuration, SOFS can be provisioned into one or more volumes and scaled in capacity as needed to support application requirements. Each volume can be accessed by any number of connectors to support the incoming workload, even with mixed protocols (NFS, SMB or FUSE).

One RING can support an enormous number of volumes (up to 2^{32}), and can grow to hundreds of billions of files per volume. There is no need to pre-configure volumes for capacity (the RING effectively supports thin-provisioning of volumes). Volumes use the entire storage pool to expand as needed when files are created and updated. For efficient storage of very large files, RING supports the concept of sparse files (files combined from multiple individual data stripes).

SOFS permissions and directory services integration

SOFS provides POSIX- and industry-standard access control, through Microsoft Windows-style Access Control Lists (ACLs) for access over the SMB protocol. RING integrates with standard security servers for authentication of SOFS users. Active Directory (AD) access is enabled for SMB user access, and Kerberos Domain Controllers (KDC) for NFS users. This provides a way for administrators to consolidate user and group identities and permissions in these well-known services.

Key capabilities include

- For NFS v4.0/v3, SOFS implements standard POSIX file permissions (r/w/x) that provide read, write, and execute permissions at the directory and file level.
- NFSv4 ACLs are mapped on POSIX (RWX on many users and groups)
- POSIX ACLs are shared across all connectors (NFSv3, NFSv4, SMB) with ACL inheritance
- Kerberos authentication supports RPC_SECGSS (with krb5, krb5i, krb5p export/mount options).

Concurrent access and the distributed lock manager

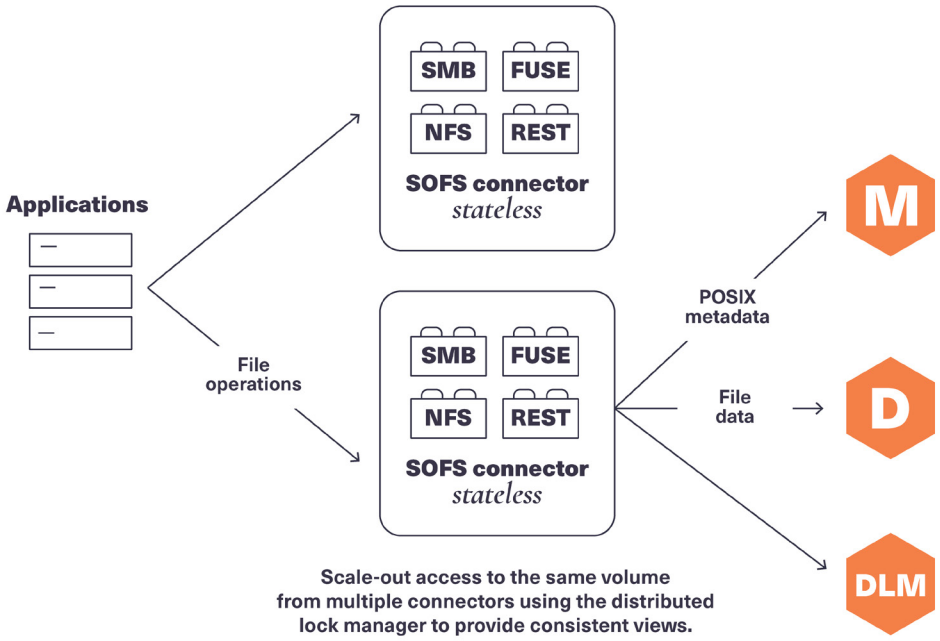
SOFS supports scale out access across any number of NFS, SMB or FUSE connectors. By supporting scale out across any number of connectors,

SOFS throughput can be scaled to support increasing workload demands. When performance saturation is reached, it is always possible to add more connectors, storage nodes and underlying physical resources to increase throughput in and out of the file system.

A single SOFS volume may have multiple connectors simultaneously accessing the volume, the same directory or the same files within a directory. To maintain file system consistency and arbitrate sharing of files, SOFS provides an integrated distributed lock manager (DLM), that provides automated and transparent locking services.

The DLM provides internal locking across connectors, to ensure proper synchronization of views and serialization of file updates. DLM services communicate with each other to maintain safe and coherent lock states on inodes (parts of files), and can provide read, write and update (intent to write) lock states. The connectors leverage the well-known Zookeeper service for joining and leaving the DLM cluster for a specific volume, and connectors directly exchange locking information with each other.

Note that DLM services are part of the RING software (it is distinct from the Linux kernel DLM) and are typically deployed co-located on the same hosts as the connectors and storage nodes, thereby minimizing hardware resource requirements. As will be described later, DLM services can also be deployed in a synchronous, stretched architecture across multiple zones/sites



An important point is related to network file protocol locking, such as NFS advisory locks (through NLM) and SMB mandatory locks. These types of protocol layer locks operate at a higher level than SOFS DLM locking. Therefore, NFS or SMB protocol locks can be implemented but the locks are only within the scope of a single connector. This is, therefore, distinct from SOFS DLM locks, which are cross-connector, internal locks for directory and inode structures and independent of NFS or SMB protocol layer locks. Scality support can provide additional guidance for the use of NFS and SMB level-locking best practices.

For optimal performance, SOFS connectors will cache file data locally. As client connections are routed back to the connector (through the CTDB VIP mapping), this ensures that frequently used data is present in the cache for fast access. The DLM also helps to ensure that cached data remains consistent (flushed and refreshed accordingly) when the same data is accessed from multiple connectors.

When multiple connectors attempt to write to the same directory, the DLM is used to maintain view consistency of the volume across the connectors. For example, if a new file is created in a directory from one connector, clients connected to other connectors will see the new file present in directory listings performed thereafter. This prevents different (inconsistent) views of a directory from multiple connectors.

At the file level, multiple connectors may access the same file with the following lock states and behavior:

- **Read locks:** Read locks are shared, so that multiple connectors may simultaneously access the same file (inodes) for read access. The DLM will automatically and transparently acquire read locks on the relevant inodes. If a connector attempts to write to the same inode while there is a read lock, the DLM will temporarily block the writer to ensure the write request is properly serialized after the read locks are released.
- **Update locks:** Connectors that receive asynchronous writes buffer the data in memory and rely on an update lock, which indicates an intent to write the inode. This locking mode prevents any other connector from writing but allows readers to access the file.

- **Write locks:** When a file is to be committed — either because data was buffered or because a synchronous change is requested — a write lock is obtained, which is an exclusive lock so that other connector reads and writes are properly serialized.
- The DLM provides a built-in notification mechanism to avoid long-term blocking behavior when connectors compete on the same inode in a conflicting way.

SOFS Storage Accelerator

A powerful new internal tiering capability, termed the Storage Accelerator, is now supported. It provides the ability to configure multiple tiers of storage within a single RING, with each tier defined as a combination of storage media and data protection policy. For example, a tier could be defined as replication (CoS) on flash media, and another tier could be erasure-coding on HDD, and so on.

The initial implementation provides configurable age-based movement of files across tiers. In other words, as files get older they can move from faster to more cost-effective storage. Note that tiers are not caches. They are indeed fully reliable RING storage but with the addition of a defined media, protection schema and age-based policy. For data movement across tiers, data is flushed to the next level tier asynchronously — after a configurable time interval — to allow tuning for different types of data and workloads.

This capability also helps workloads that require stable and reliable file write performance — such as live TV capture (recording) or image ingestion in healthcare. SOFS can leverage flash drives to initially store data using replication for highest performance. Later, when the files are less frequently accessed, they can be moved (aged off) to more cost-efficient HDD storage with an erasure-coding durability policy.

SOFS utilization and quotas

SOFS provides volume-level utilization metering and quota support, in addition to user and group (uid/gid) quotas. Administrators may meter, report and limit space (capacity) usage at the volume level. This feature is useful in a multi-tenant environment where multiple applications or use cases are sharing the same RING, but accessing data stored in their respective volumes. The Supervisor UI allows managing of quota limits, viewing and sorting users/groups by user capacity, inodes and soft/hard limits and comparing usage of users and groups.

SOFS volume protection

An immutable file system volume is key to protecting against ransomware attacks as it prevents data from being maliciously modified or deleted. For this purpose, SOFS provides a volume protection capability that enables a per-volume policy to convert files to a “read-only” state after a specified time window, to protect it from being modified or deleted (and more specifically, encrypted). This protection is enforced regardless of the application or administrator credentials and regardless of the file permissions set on the files, making the files truly immutable.

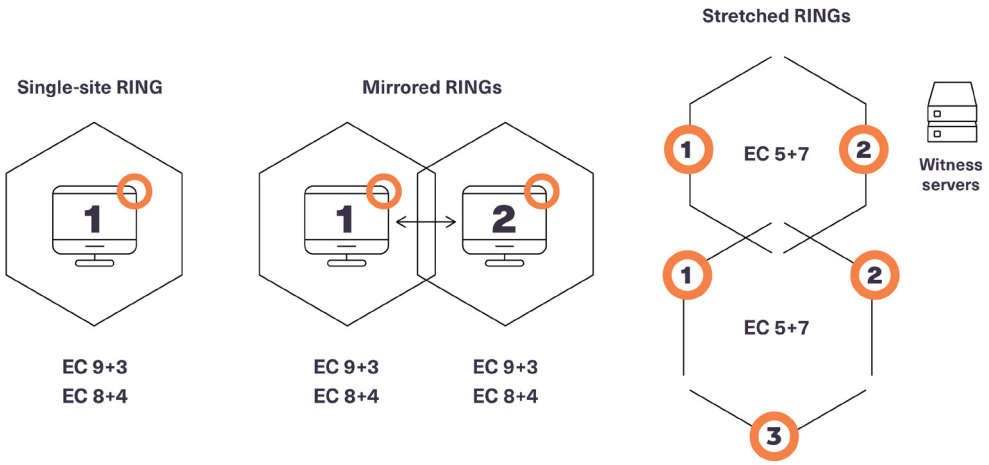
For example, in a RING deployed as a digital archive, a file could be created (written) into the storage layer, and after 30 minutes the system will prevent file changes, modifications and deletions (the period is configurable as specific to business requirements). In this example, any ransomware attack may compromise the files that were created within the last 30 minutes, but any files that were at least 30 minutes old when the attack began are guaranteed to be unmodified. In such a case, it is possible to safely provide a guaranteed recovery point objective (RPO) of 30 minutes prior to when the attack started. As the file system is protected in-place, the recovery time objective (RTO) is zero.

Because this protection is enforced on top of any access control that the application may leverage, it is resilient to privilege escalations that ransomware attacks often leverage by compromising user credentials from an Active Directory server.

By preventing updates/deletes to specific volumes, RING is able to protect against common ransomware attacks. This approach can be implemented as a standalone measure or in combination with asynchronous replication to remote sites, as is described in the following section.

VII. Multi-site RING deployments

RING natively supports both asynchronous replication (mirroring) and synchronous (stretched) mode for multi-site deployments. The synchronous stretched mode provides site failure protection and complete data consistency between all sites. In this mode, a single logical RING is deployed across multiple data centers with all nodes participating in the standard RING protocols as if they were local to one site, with synchronous writes to all sites.



Stretched modes are supported for both SOFS and S3 access, across two or three sites (data centers), or logical zones. RING maintains service availability even in the event that one of the sites/zones becomes inaccessible (due to a connectivity issue or site failure). Once connectivity is restored, RING will resynchronize data back to the storage nodes on the restored site to bring the overall system back into balance from a data protection perspective.

In this manner, stretched RING provides a real advantage in overall system availability, with site-level fault tolerance and effectively zero RPO (recovery point objective) and RTO (recovery time objective) after a site failure, due to the synchronous nature of these deployments.

Stretched deployments provide multiple key benefits including:

- Active/active access across all sites for both reads and writes to the data
- Full site-level failure protection, with service continuing even during such an event
- Continuous access to data in the event of a full site failure, which provides effectively immediate failover and zero RPO/RTO recovery

These advantages for stretched RINGs are gained from the use of synchronous writes across multiple sites, ensuring that all data is written to all sites for data consistency. Because synchronous writes will incur network latencies, best practice is to use this mode in lower-latency, metro-area networks. Scality recommends network latencies under 15 milliseconds roundtrip as a maximum for stretched RINGs.

Additional notes and considerations for stretched RING deployments:

- 2- and 3-site stretched deployments are supported for both S3 object storage and file system (SOFS) access.
- To tolerate the failure (outage or other) of a full site, data is spread across the sites to ensure the system can continue to operate, protect and serve data during the outage. In practice, Scality recommends erasure coding (EC) policies that allow one full site failure, plus one additional drive failure, and still keep the service available. Specific details for 2 and 3-site EC schemas and storage overheads are outlined below.
- If a failed site in a stretched deployment is restored (brought back online), RING will automatically restore (self-heal) the missing data to the storage nodes on that site. This brings data protection back to the state of the original configuration.

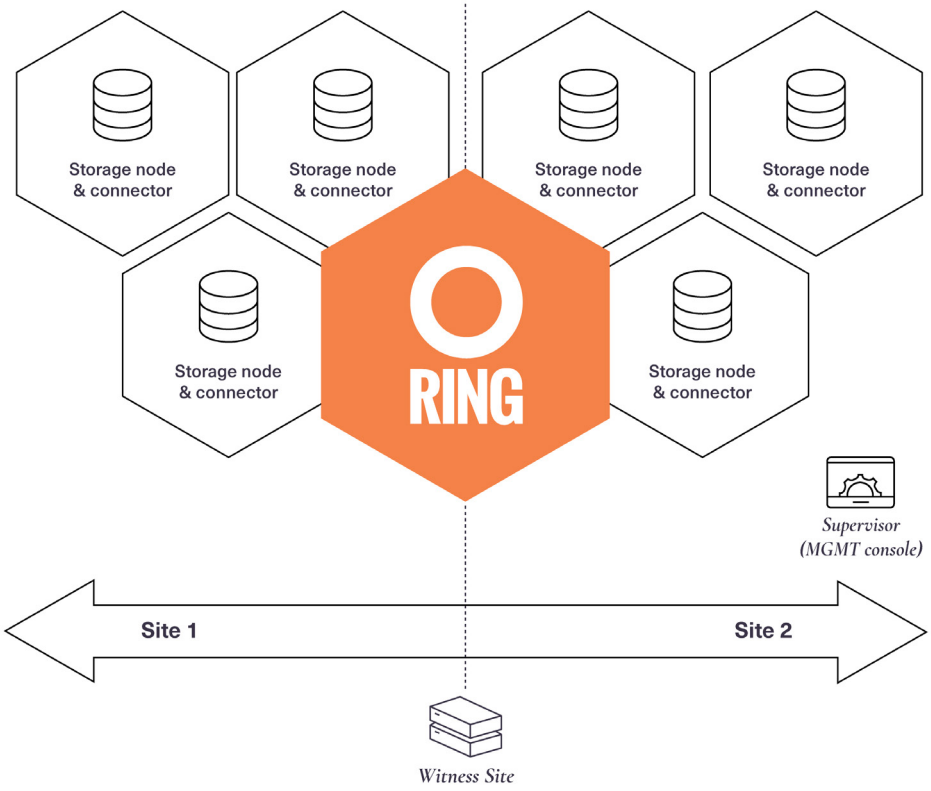
The next sections describe the different deployment options for stretched and replicated (mirrored) deployments for file system (SOFS) and object (S3)-based RING solutions.

Two-site stretched deployments

In this two-site scenario, the system needs to be able to serve data in the event one of the two sites suffers an outage. Scality typically recommends an EC schema of 5+7 (5 data stripes + 7 parity stripes), with 6 stripes stored on each site. This approach protects against the loss of one full site, plus provides added protection for one additional disk failure on that site. This implies the space overhead for two-site stretched deployments is approximately 2.4 times the original data (or stated otherwise, 140% overhead) to attain the level of protection and availability.

For two-site stretched SOFS deployments, two additional “witness servers’’ (physical or virtual machines are supported) are used to manage and arbitrate file system consistency in the event of a network partition between the two sites. The witness servers are used to establish a quorum (majority consensus) of file system state, which would otherwise not be assured in the case of two sites.

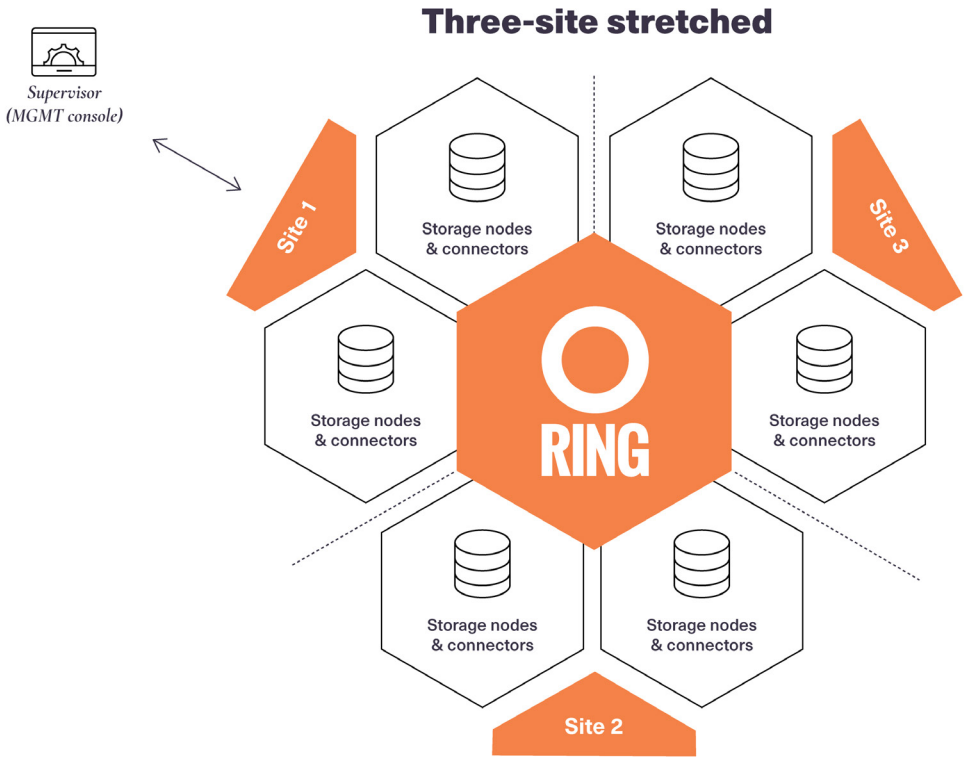
Two-site stretched with witness



Three-site stretched deployments

In this three-site scenario, the system needs to be able to serve data in the event one of the three sites suffers an outage. RING intelligently distributes data across site domains to ensure data is available in the event one of the sites suffers an outage or total failure. Scality recommends an EC schema of 7+5 (7 data stripes + 5 parity stripes), with 4 stripes spread and stored on each site. Since only 7 stripes are needed to reconstruct data, service can continue in the event of one full site failure (4 stripes), plus one additional disk failure can be tolerated on the two surviving sites.

The three-site stretched RING maintains the same advantages as described earlier, but with significantly reduced storage overhead of only 1.71 the original data (71% space overhead). This model is highly-efficient compared to the two-site model, and also more efficient than full volume or site mirroring (as with asynchronous replication) to be described next.



Multi-site asynchronous replication for SOFS

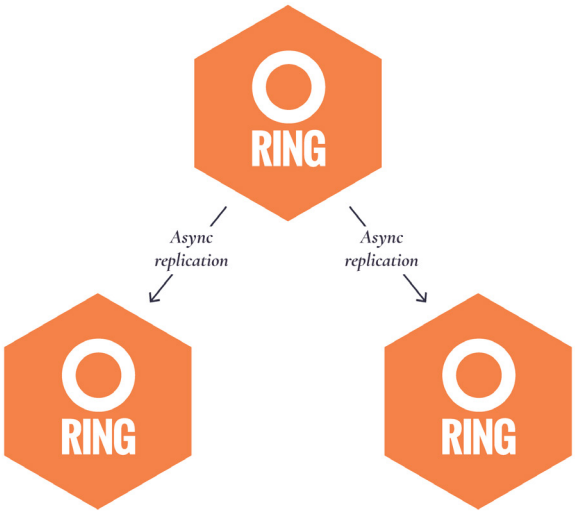
For RING deployments across multiple sites connected on a higher-latency, wide area network (WAN), SOFS supports volume-level asynchronous replication. This allows individual SOFS (source) volumes to be replicated to one or more remote (target) volumes, providing remote disaster recovery protection or an additional layer of recovery against threats such as ransomware.

Advancements in SOFS volume replication now include:

- The ability to replicate a SOFS volume consistently to more than one remote volume (and on different sites)
- Scale-out for high throughput workloads requiring multiple NFS or SMB connectors for ingest on the source and target volumes
- Consistency to ensure that write ordering is preserved on the target (remote) volumes as it was on the primary (source) volume, in time order
- Ensured consistency even in the event of multi-connector access to a replicated volume

Maintaining file system consistency in a scale-out, multi-connector replication scenario represents a major advancement in the industry state-of-the-art for file system replication. This also creates opportunities for new use cases for enterprises or service providers with multiple global data centers, such as for data distribution and disaster recovery.

Multi-site asynchronous replication



Asynchronous replication for S3

For higher-latency wide area network (WAN) deployments requiring multi-site data, RING supports an S3 asynchronous replication mechanism. This mechanism is designed and proven at scale to enable the replication of massive amounts of S3 object data, and with high throughput rates. Asynchronous replication can enable disaster recovery by providing a second copy of data in a remote site, and it enables continuous access to data on the remote site after a failure (or outage) of the primary site.

RING S3 asynchronous replication is based on the S3 CRR (cross-region replication) API to enable replication on a bucket between two sites/buckets (one source bucket to one target bucket). In addition to S3 style bucket-level replication, RING also offers a mode to support site-level replication of all buckets to a remote site.

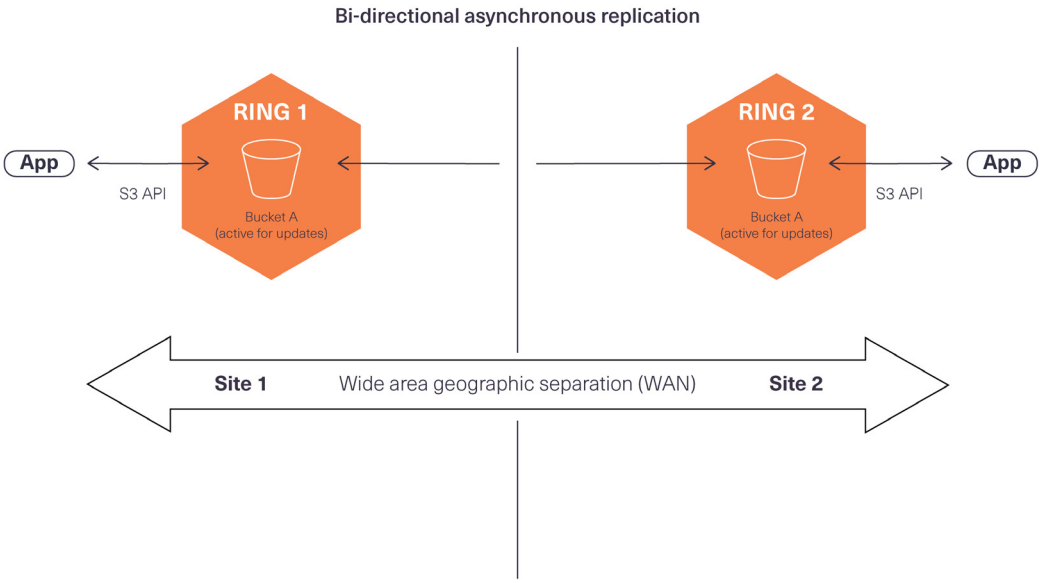
RING S3 replication can enable both active/passive or active/active access to buckets on the two sides of the replication configuration. Comprehensive metrics, performance KPIs and status monitoring are provided via APIs and through the RING supervisor GUI.

Active/active asynchronous replication for S3

Asynchronous replication for S3 can be configured to work in a bidirectional manner, enabling two sites to update the same bucket. In this model, the bucket will maintain the same object and version state, except for the time of the asynchronous replication window.

In terms of recovery point objective (RPO) and recovery time objective (RTO), this means it is possible to deploy two RINGs in two geographically separate data centers, with very small (near zero) RPO/RTO from a disaster recovery perspective in the event one of the two sites fails. This is a powerful model that eliminates the inefficiency of the “hot standby” active/passive model traditionally deployed for disaster recovery purposes.

S3 active/active asynchronous replication



RING consistency models

At the storage layer, consistency determines the minimum number of replicas of an object stored for a given class of service. For example, for CoS=3, strong consistency requires all three replicas to be committed to storage before the write is deemed complete. Relaxed (or eventual) consistency rules allow only one or two of the three replicas to be committed with the write deemed complete. The remaining replicas are eventually written to the storage layer, but there may be a delay until this occurs. At the core layer, RING policies can be configured to relax strong consistency if it is deemed acceptable to an application. However, at the protocol connector layer for SOFS and S3, strong consistency is enforced as described next.

The RING scale-out file system (SOFS) layer implements a virtual POSIX file system abstraction, which enforces strong consistency semantics. To ensure that the file system is always consistent, SOFS is implemented with the MESA database, using full ACID (atomic) database transactions for writes to the file system. RING also applies distributed two-phase commit9 protocols to ensure that data is on stable storage across all storage nodes involved in the write before proceeding. This implies that writes to SOFS will always require all file replicas (or data and parity stripes, in the case of erasure coding) to be written to the storage layer before the write is acknowledged. While this ensures file system consistency, in return it requires that multiple concurrent writers be aware of the serialization constraints implied when writing to the same file system structures, such as a single directory.

For RING S3 object storage, strong consistency is enforced at the metadata level, through the use of a majority consensus model. All new object write (PUT) requests to S3 must be acknowledged by at least a majority of the metadata service nodes, and all read (GET) requests are serviced by the metadata master node (the leader), to ensure the latest state of the object is retrieved. In the event that the metadata leader does not respond or has failed for some period of time, a new metadata leader is elected through the consensus algorithm. In either case, the S3 service will not respond to a request that fails to ensure strong consistency.

VIII. Multi-cloud extended data management

RING provides the ability to manage data across storage locations, both for multiple RING deployment instances and public cloud storage services. Multi-cloud extended data management requires one or more additional servers to host additional processing services (referred to as the Backbeat service in the product documentation). Multi-cloud services can be deployed on a single server or can be scaled-out on many servers as needed to address larger workloads.

Multi-cloud extended data management includes the following key capabilities:

- A global namespace that can manage S3 buckets across local and remote RING storage and public cloud storage services
- Asynchronous replication from RING to cloud storage
- Metadata search across S3 buckets spanning across RING and cloud storage

Multi-cloud services (Backbeat) provide an asynchronous processing engine based on a scale-out design using Kafka queues. Multiple job queues can be dynamically created and managed — one for each configured replication. This enables asynchronous new events to be published, consumed and processed simultaneously and in-parallel to sustain large replication workloads or multiple parallel replication streams.

S3 global namespace and cloud replication

The RING global namespace makes use of the S3 bucket *location* attribute to assign default storage locations to S3 buckets (in AWS, these define an AWS data center region, such as US-West-1, US-East-1, etc). For RING multi-cloud, locations are defined as a combination of endpoint URL, bucket name and access keys for a local or remote RING or cloud service endpoint.

Locations can be used to assign a default location to a new bucket, meaning that S3 PUT operations will then store the object data in that target location, and subsequent operations will also access the data from the specified target. Locations are also used for RING bucket-to-bucket asynchronous bucket replication, which was previously described in the *RING multi-site deployments section.*

Locations are also used to configure asynchronous replication (supported through AWS S3 CRR API) policies from local RING buckets to cloud buckets. In general, this requires two buckets (source and target) to be created and configured, and enabling CRR on the local (source) bucket:

- Bucket on local RING with versioning enabled
- Bucket on remote cloud target (AWS S3 for example) created through the AWS S3 console, also with versioning enabled
- Enable cross-region replication on the RING bucket and assign the remote cloud bucket as the target for the replication (this step can be done via CLI or S3 API as described in the RING product documentation).

In addition to the two buckets, a specific set of bucket policies and IAM Roles must be configured locally and on the cloud target to allow the RING multi-cloud service access to the cloud service with the required permissions. These policies and roles are detailed further in the RING product documentation.

Once enabled, any new object updates (PUT or DELETE) operations on the local bucket will be automatically replicated to the target bucket. Note that existing objects in the local bucket before enabling cross-region replication will not be replicated to the target bucket. Cloud replication can be enabled/disabled/suspended/resumed individually on any bucket (replication will stop once it is disabled). Multi-cloud services provide intelligent retry capabilities on replication in the event of a connectivity error or failure. Replication status and performance can be monitored through provided REST APIs

S3 metadata search

The metadata search feature allows users with access to object metadata to perform powerful searches on objects and easily find the data they need when they need it. Metadata search operates on the following types of metadata:

- System defined metadata attributes (S3 protocol assigned such as Data, Content-Length, Content-Type, Last-Modified and more)
- User-defined metadata key/value pairs assigned to an object through the optional "x-amz-meta-" headers on PUT Object API operations. Up to 2 Kbytes of space is reserved for metadata per object.
- S3 object tags: Key/value pairs applied to S3 objects which can be created, updated or deleted any time during the lifecycle of the object

In a standard S3 GET bucket API call, RING returns a listing of the object keys in that bucket back to the requesting client application. Bucket listings can also use a prefix filter to only return those keys matching the prefix back to the client.

RING S3 metadata searches are exposed via an extended S3 GET bucket API, to allow the search condition to be expressed in a pseudo-SQL like syntax. Similarly, RING will process the GET bucket operation but will return the object keys where the metadata attributes matched the search condition. The S3 Browser UI provides a user-friendly interface to S3 metadata search to allow interactive searches of buckets on attributes. These are displayed in the S3 console as a listing of matching objects.

IX. Security and ransomware protection

The security threat landscape has increased markedly in the last several years. Scality believes that a cyber-resilient storage solution is a key to protecting data from ransomware, and that a cornerstone of defense begins with immutability. RING provides integrated immutability through S3 Object Locking, as well as an intrinsically immutable storage engine. Immutability is augmented through end-to-end cyber-resiliency capabilities that protect against threats at all levels of the architecture (see *S3 Ransomware Protection* below).

RING security guidelines

These guidelines lay out best practices for deploying RING software to protect the system against common security threats:

- **Data loss:** One of the best-known forms of attack that can lead to data loss is a ransomware attack.
- **Data leakage:** Unwanted exposure of sensitive information, either publicly or to a third party.
- **Compromised services:** Multiple forms, from denial of service (DoS) to sophisticated data corruption, that lead to falsifications or even the corruption of authentication mechanisms.

Three attack vectors on RING — multiple forms from multiple origins — are of primary concern:

- Remote attacks from an external or client network
- Internal network attacks (from an organization's internal trusted network)
- Physical access attacks (from a direct physical access on machines)

Data protection and security must be treated with a holistic approach. The most carefully architected system can be compromised by poor maintenance, human error or negligence. Scality recommends that careful attention be paid to the credentials of those who have the greatest level of system access and to changes that occur over time (e.g., changes in personnel, the addition of new equipment and services to the platform). While it is impossible to create an entirely secure system, one must create a robust, attack-resistant environment.

A couple of new features aimed at enhancing RING and overall data security include:

- **Non-root execution:** RING components are deployed and run under a non-root user, which helps safeguard root privileges further.
- **Kerberos authentication:** The NFS v4.0 stack can be used with Kerberos to manage user authentication and application access. Kerberos provides strong cryptography and third-party ticket authorization.

RING also provides extensive logging of administrative, user and internal actions and activities. RING logs are in an open JSON format for easy ingestion into third-party logging, event, alert management tools. For ransomware detection, audit logs are also available in LEEF format for consumption by popular SIEM tools

S3 multi-tenancy and security

RING provides a secure multi-tenancy model compatible with AWS style of identity and access management (IAM) for S3 operations:

- Multiple accounts within one system, with role-based access control (RBAC) for super admin, account admin and monitoring roles.
- Per accounts: Users, groups, roles, plus user/group policies and bucket policies with granular allow/deny style access control
- AWS-style key/secret key pairs, with Signature V4 (or older v2) style HMAC authentication for each API operation
- RING Supervisor UI provides custom access control policy editor, which enables a) import of user or bucket policies from AWS for reuse in RING, or b) editing of these policies into custom ones.

S3 ransomware protection

Digital extortion schemes are on the rise, and they're getting more sophisticated. No matter how hardened your environment is, there is no guarantee a ransomware attack won't be successful. When one does manage to succeed, the only option is to restore — provided, of course, there is a clean, uninfected backup copy to restore from, which is exactly why ransomware attacks target backup data.

RING is typically deployed in a data center behind other security layers and services, including secure firewalls, network security, application servers (which also provide authentication and access control at this layer), and, ultimately, the data storage. RING is, therefore, part of an overall, comprehensive security and ransomware protection infrastructure within the data center.

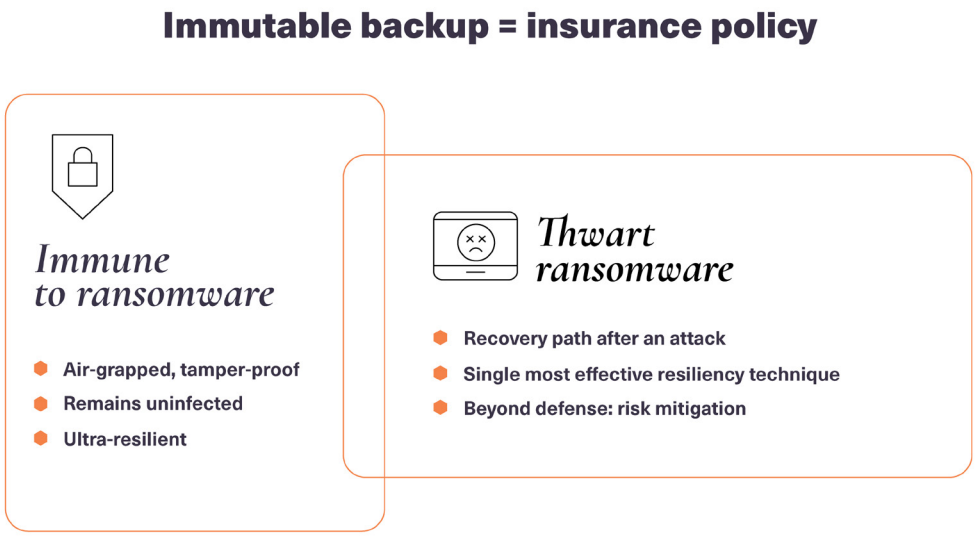
That's where data immutability comes in, keeping backup data securely and effectively "air-gapped" and tamper-proof, so it remains uninfected, immune and essentially out of reach to ransomware. With the recovery path

kept open, ransomware attacks are thwarted. It's the single most effective resiliency technique — a critical insurance policy to have for when (not if) ransomware strikes.

As stated above, immutability is a cornerstone of ransomware protection. In RING, immutability is augmented through end-to-end, cyber-resiliency capabilities that protect against threats at all levels of the architecture. These capabilities are collectively termed CORE5 cyber resilience:

- API-level resilience
- Data-level resilience
- Storage-level resilience
- Geographic-level resilience
- Architecture-level resilience

At each level, RING provides a set of capabilities that provide cyber-resilience, which makes RING especially suitable for protecting large-scale and long-term backup data from ransomware.



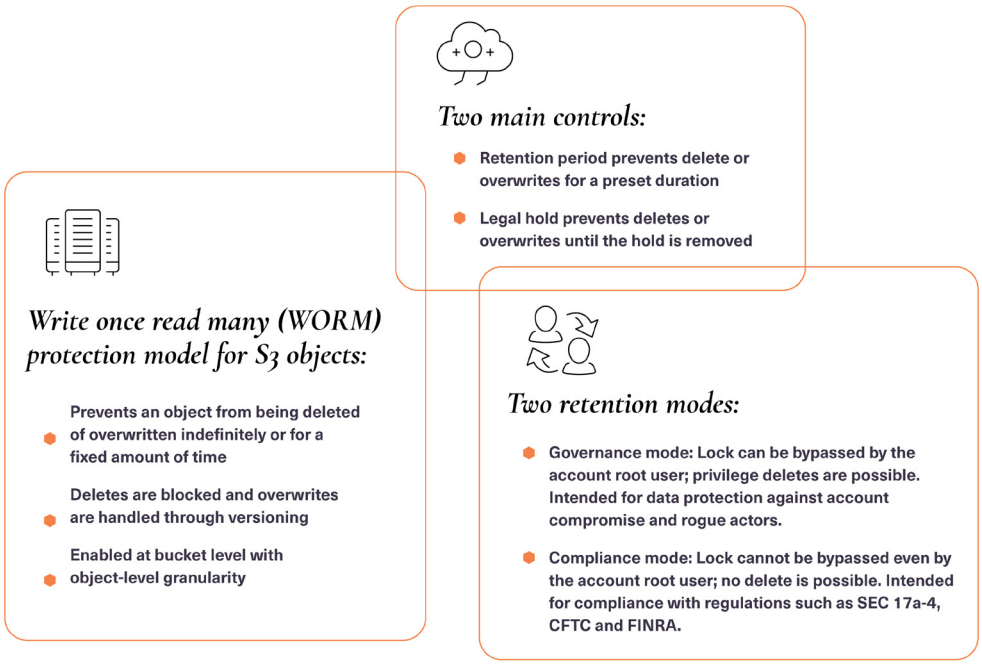
For S3 application data, RING provides data immutability capabilities through the S3 Object Locking. Immutability ensures that data that is stored can be protected against unauthorized or malicious updates (deletes, overwrites such as through encryption, which is common in ransomware attacks). S3 Object Lock is an Amazon S3 feature that blocks object version deletion during a customer-defined retention period. You can use it to prevent an object from being deleted or overwritten for a fixed amount of time or indefinitely. In conjunction with S3 versioning, which protects objects from being overwritten, it ensures that objects remain immutable for as long as S3 Object Lock protection is applied.

S3 Object Lock protection can be applied by either assigning a “retain until” date or a “legal hold” to an object. Users can apply retention settings within a PUT request or apply them to an existing object after it has been created, provided the object lock flag was set at bucket creation time.

S3 Object Lock can be configured in one of two modes, both of which are supported by RING. In governance mode, accounts with specific privileges granted by IAM permissions are able to remove object locks from objects. By contrast, in compliance mode the protection cannot be removed by any user — not even the root account — thereby offering stronger immutability to comply with regulations. Both governance and compliance modes retain the lock on an object until the configured retention period expires.

The retention period defines the term during which the object is protected from deletes. In addition, legal hold can be enabled on an object version. Once a legal hold is enabled, regardless of the object’s retention date or retention mode, the object version cannot be deleted until the legal hold is removed. For more details, please refer to the *Object Lock* sections in the *S3 Operation* and *S3 Connector Reference* books in Scality product documentation.

Immutability via S3 Object Lock

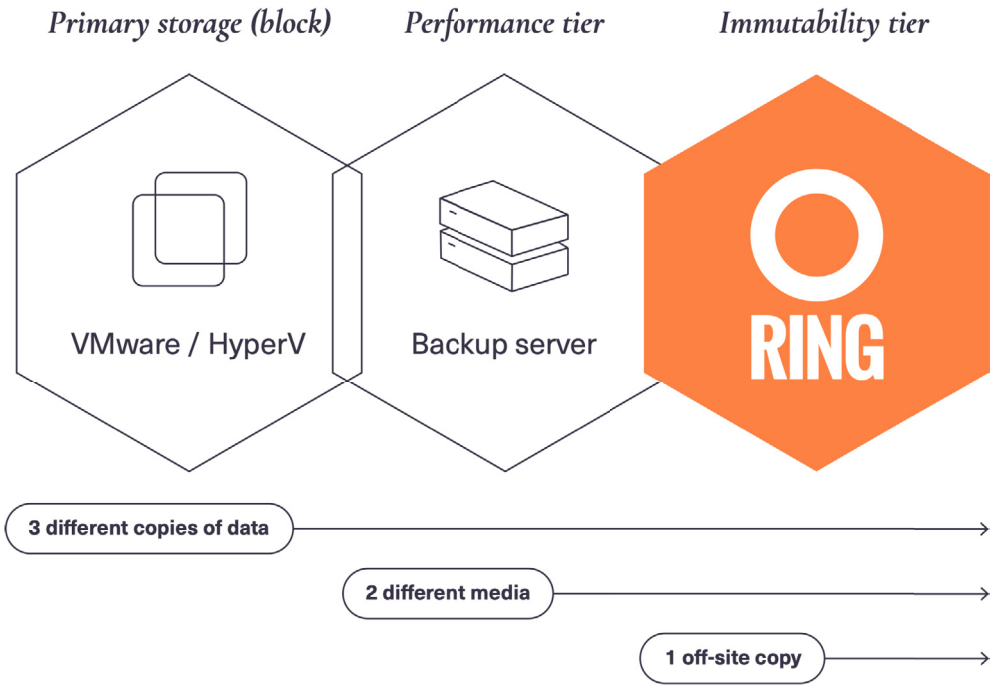


A set of popular data protection (data backup) applications now support data immutability through the S3 Object Lock API and can configure RING buckets to provide this functionality.

For example, Veeam and Commvault are two vendors that now support this functionality and have certified RING for data immutability for their customers. This makes it possible to deploy RING in environments using the well-known "3-2-1" (or 3-2-1-1) data protection guidelines, which recommend that customers store:

- 3 copies of data (live, backup and remote)
- On at least two types of media (local application storage and a secondary capacity tier)
- A third "air-gapped" or remote copy (such as on tape or cloud storage), which can now be RING with data immutability enabled

This approach offers multiple advantages over the use of tape or cloud storage. A fast, scalable and online copy of backups is immediately available in the event a ransomware attack occurs, requiring data restores with the lowest recovery time objective (RTO).



S3 encryption and key management

RING S3 supports bucket level encryption, specified as a policy at the time a bucket is created (S3 PUT Bucket API compatible with AWS specifications). Encryption is implemented with AES256-bit OpenSSL libraries.

Bucket encryption is compatible with AWS SSE-S3 (Server Side Encryption) with the following behavior:

- S3 will accept the write of an object with an SSE request if the bucket on which the object is written is encrypted.
- S3 will not accept the write of an object with an SSE request if the bucket has not been configured with encryption.

For secure key management, RING supports the OASIS key management interoperability protocol (KMIP) standard that allows users to interact with popular key management systems (KMS). KMIP protocol version 1.2 (and later) allows for streamlined data encryption by simplifying encryption key management. The KMIP driver is vendor-agnostic and supports all KMIP appliances with basic encryption profile support out of the box.

SOFS ransomware protection: Volume protection and multi-site asynchronous replication

As described earlier, SOFS provides volume protection capability, which enables a per-volume policy to convert files to a "read-only" state after a specified time window, effectively protecting it from being modified or deleted (and, more specifically, encrypted). This protection is enforced regardless of the application or administrator credentials and regardless of the file permissions set on the files, making the files truly immutable. This capability can be implemented as a standalone measure for ransomware protection, or in combination with asynchronous replication to remote sites.

As described previously (see section *RING multi-site deployments*), SOFS provides multi-site, asynchronous replication, enabling mirroring of file system volumes to multiple, remote RING deployment instances. For recovery purposes, it is desirable to have remote sites maintain stable previous (older) states of the file system in the event the primary site is impacted by a breach or attack. For comprehensive protection against ransomware, using replication with at least two remote target sites is recommended.

To maintain the previous (older) state of the file system on the remote sites, SOFS asynchronous replication can be periodically paused during normal operations. By pausing replication streams from the primary site to each remote site independently and at alternate times, remote data protection can be maintained for a desired period of time.

For ransomware protection, replication may be paused for a period of days to weeks (or more) to create a rotating window of recovery against a ransomware incident on the primary RING and file system. While replication is paused to one of the remote volumes, replication continues to a second remote volume. The following week, this order is reversed.

For instance, to provide at least 7 days of protection, RING at the source site and one of the destination sites would be synced every Sunday at 9:00 AM, while the other replication is paused for the following week. The following Sunday at 9:00 AM, the first replication is paused while the second one is resumed. Alternating replication in this manner ensures that at least one of the remote sites will be unaffected by a ransomware attack during a given week.

This approach provides a range of recovery options in the event of a ransomware incident, with an RPO between hours and days:

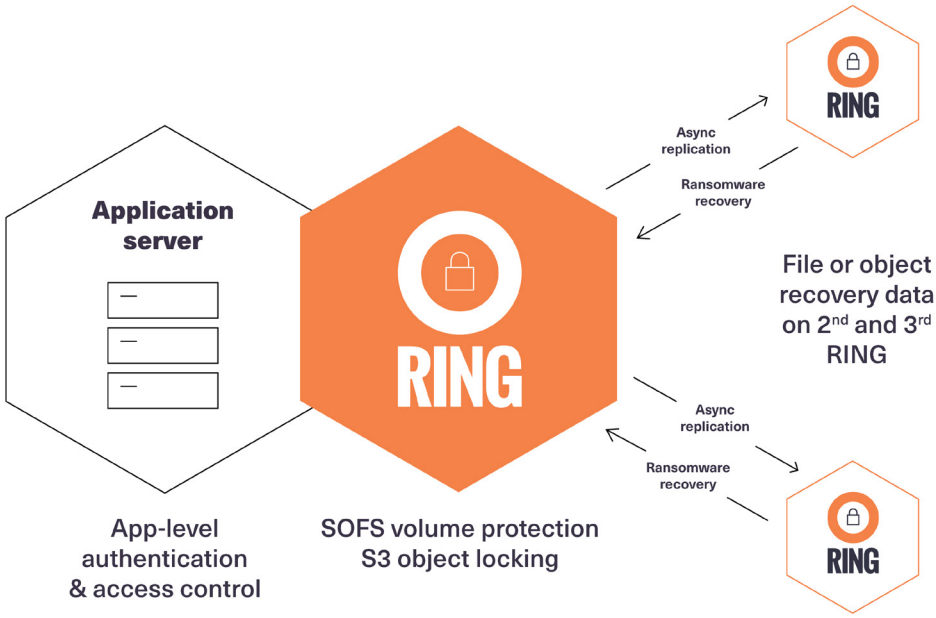
- If a ransomware incident strikes the primary RING early in the week (let's say just after 9am on Sunday), the "paused" remote site can be used for data recovery with an RPO measured in the range of a few hours.
- If a ransomware attack strikes the primary RING later in the week (say just before 9am on the following Sunday), the RPO will be up to 7 days from the paused remote site.

The period and the number of destination sites to leverage are fully configurable, also allowing periods of time when no site is receiving updates.

To ensure stronger isolation, it is possible for the target sites that have the replication paused to be shut down or physically disconnected from the network. The result is a pseudo "air-gapped" solution, which ensures the ransomware attack cannot propagate.

The main tradeoff with multi-site asynchronous replication is storage capacity overhead, which may be fully justified for businesses requiring recovery capabilities against attacks for an extended period. The timeframe can also be extended to support longer disaster recovery intervals if desired.

In addition, the solution is flexible in the sense that it is not necessary to have remote volumes leverage the same architecture, sizing, or even data protection (erasure coding vs. replication) configuration as the source volume. In fact, remote volumes may actually be used for production data for other applications or use cases, while at the same time providing target replication volumes for the primary volume.



X. Summary

Scality RING is proven as the industry's leading scale-out file and object storage solution. Ideal for both private and public cloud infrastructure, it provides unbreakable cloud storage.

RING is designed based on a core set of design principles to deliver true customer value — massive capacity scaling, consolidation of multiple storage silos with reduced operational management costs, always-on data availability and the highest levels of data durability — to protect data for the long-term at the economics of cloud-scale data centers. RING provides a comprehensive software-defined storage (SDS) solution on industry-standard platforms to enable these values.

Further information on RING is available at www.scality.com.