Redbooks

ibm.com/redbooks

Implementing OpenStack SwiftHLM with IBM Spectrum Archive EE or IBM Spectrum Protect for Space Management

Khanh Ngo

Harald Seipp

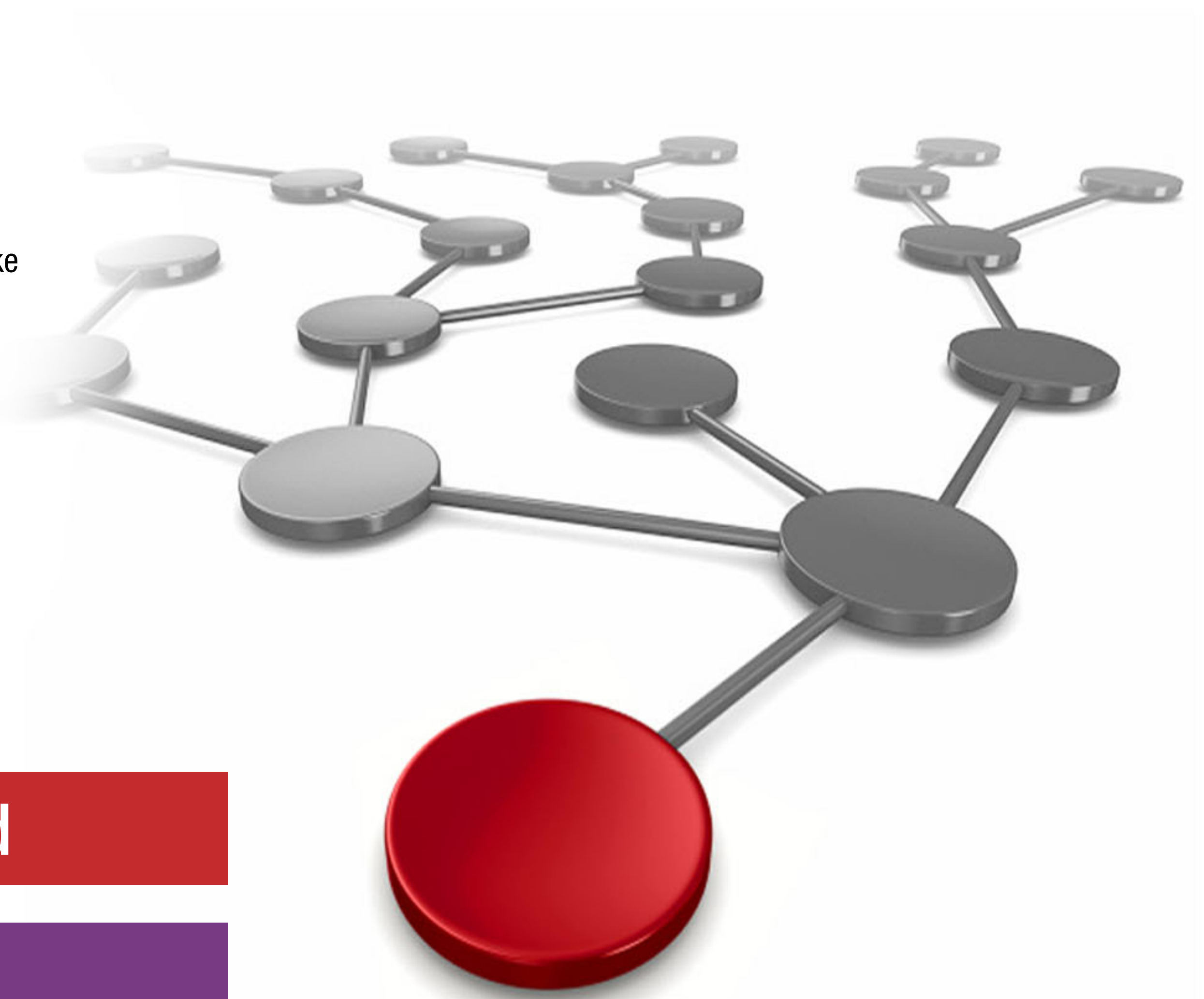
Slavisa Sarafijanovic

Dominic Müller-Wicke

Simon Lorenz

Takeshi Ishimoto

Larry Coyne



Cloud

Storage

Redpaper

Introduction

This IBM® Redpaper™ publication describes the Swift High Latency Media project and provides guidance for installation and configuration.

The Swift High Latency Media project seeks to create a high-latency storage back end that makes it easier for users to perform bulk operations of data tiering within a Swift data ring.

In today’s world, data is produced at significantly higher rates than a decade ago. The storage and data management solutions of the past can no longer keep up with the data demands of today. The policies and structures that decide and execute how that data is used, discarded, or retained determines how efficiently the data is used. The need for intelligent data management and storage is more critical now than ever before.

Traditional management approaches hide cost-effective, high-latency media (HLM) storage, such as tape or optical disk archive back ends, underneath a traditional file system. The lack of HLM-aware file system interfaces and software makes it difficult for users to understand and control data access on HLM storage. Coupled with data-access latency, this lack of understanding results in slow responses and potential time-outs that affect the user experience.

The Swift HLM project addresses this challenge. Running OpenStack Swift on top of HLM storage allows you to cheaply store and efficiently access large amounts of infrequently used object data. Data that is stored on tape storage can be easily adopted to an Object Storage data interface.

SwiftHLM can be added to OpenStack Swift (without modifying Swift) to extend Swift’s interface. This ability allows users to explicitly control and query the state (on disk or on HLM) of Swift object data, including efficient pre-fetch of bulk objects from HLM to disk when those objects must be accessed. This function, previously missing in Swift, provides similar functions as Amazon Glacier does through the Glacier API or the Amazon S3 Lifecycle Management API.

BDT Tape Library Connector (open source) and IBM Spectrum™ Archive or IBM Spectrum Protect™ are examples of HLM back ends that provide important and complex functions to manage HLM resources (tape mounts and unmounts to drives, serialization of requests for tape media, and tape drive resources). They can use SwiftHLM functions for a proper integration with Swift.

Although access to data that is stored on HLM can be done transparently without the use of SwiftHLM, this process does not work well in practice for many important use cases and other reasons. SwiftHLM function can be orthogonal and complementary to Swift (ring to ring) tiering (source).

For more information, see the Swift extensions for Tape Storage or other High-Latency Media technical overview. The high-level architecture of the low cost, high-latency media storage solution is shown in Figure 1.

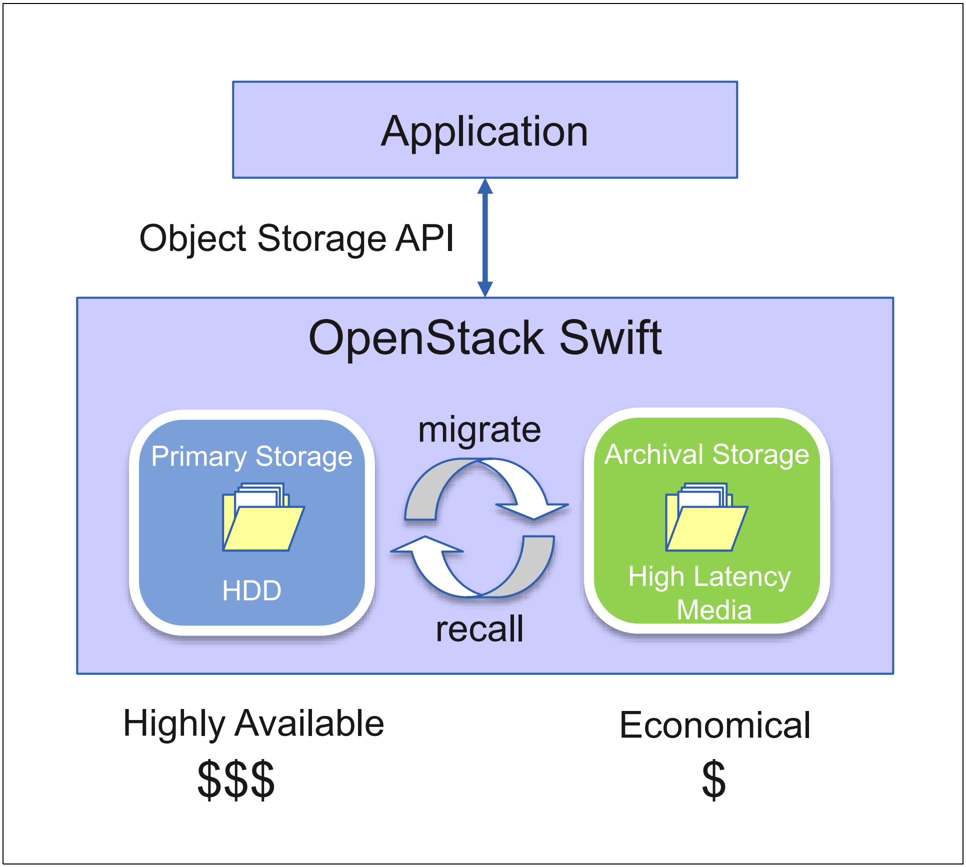


Figure 1   High-level architecture for high-latency, low-cost media storage Solution

HSM solution and tiering terms

A hierarchical storage management (HSM) solution often is a virtual expansion of the file system space because it introduces more hierarchical storage tiers. The reason for introducing storage tiers is to store large amounts of data at lower cost and higher efficiency than traditional rotating magnetic disks. In many cases, HSM-tiered environments are composed (from high to low) of different SSD classes, HDD classes, and tape as the last tier in the hierarchy.

The process of moving files between tiers is called *migration* and *recall*. *Migration* refers to moving the file data to the next lower tier and replace the data on the higher tier with only the metadata that is required to identify the new location of the file data in the lower tier for later recall. The file placeholder that contains the metadata is called a *stub file* and uses nearly no space.

The term *recall* refers to moving the data back to the higher tier in the storage hierarchy. The physical location of the file data is not apparent. An HSM solution does not distinguish between different file versions. A change to file data requires the recall from the lower tiers to a higher tier.

The tiered data flow when SwiftHLM is used in an OpenStack environment is shown in Figure 2.

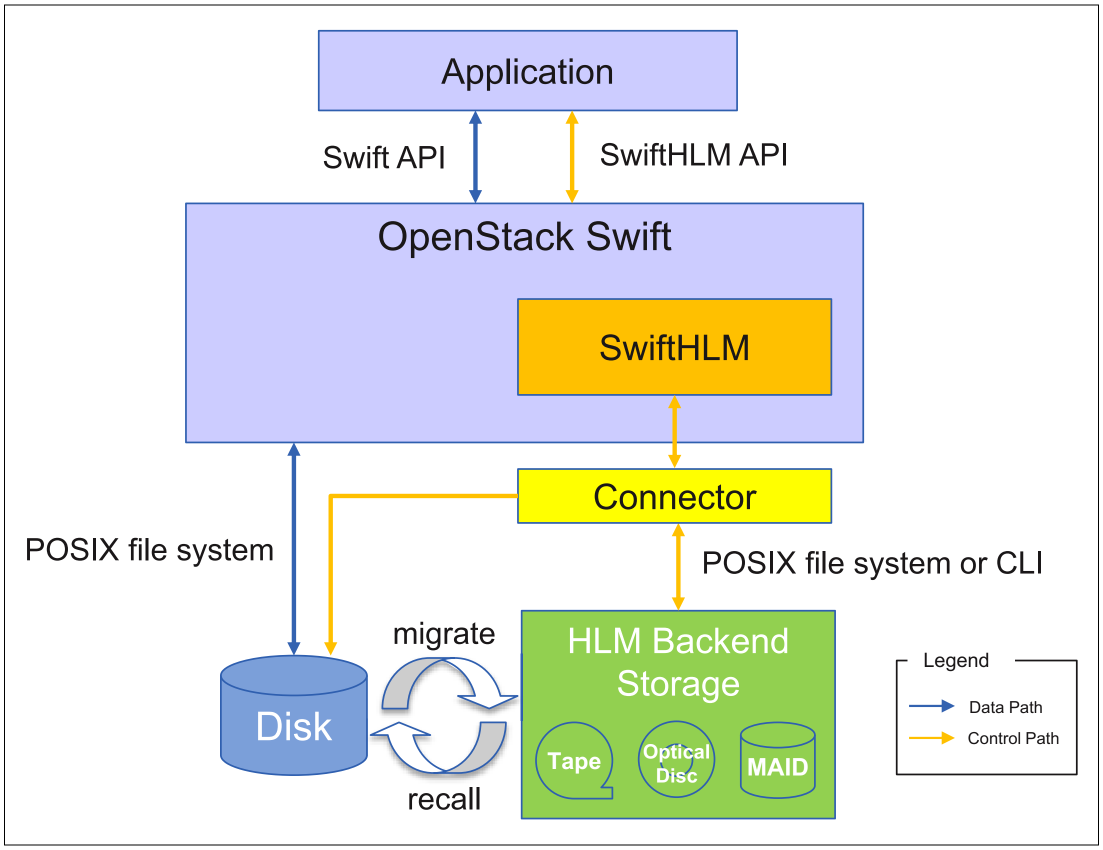


Figure 2 Tiered data flow that uses SwiftHLM in an OpenStack environment

Introduction to Spectrum Archive EE

A member of the IBM Spectrum Storage™ family, IBM Spectrum Archive™ enables direct, intuitive, and graphical access to data that is stored in IBM tape drives and libraries by incorporating the LTFS format standard for reading, writing, and exchanging descriptive metadata on formatted tape cartridges.

IBM Spectrum Archive eliminates the need for more tape management and software to access data. IBM Spectrum Archive offers the following software solutions for managing your digital files with the LTFS format:

- Single Drive Edition (SDE)
- Library Edition (LE)
- Enterprise Edition (EE)

IBM Spectrum Archive EE for the IBM TS4500, IBM TS3500, and IBM TS3310 tape libraries provides seamless integration of LTFS with IBM Spectrum Scale™, which is another member of the IBM Spectrum Storage family (formerly the IBM General Parallel File System) by creating a tape-based storage tier. You can run any application that is designed for disk files on tape by using IBM Spectrum Archive EE because it is transparent and integrates in the Spectrum Scale file system. IBM Spectrum Archive EE can play a major role in reducing the cost of storage for data that does not need the access performance of primary disk.

With IBM Spectrum Archive EE, you can enable the use of LTFS for the policy management of tape as a storage tier in an IBM Spectrum Scale environment and use tape as a critical tier in the storage environment. IBM Spectrum Archive EE supports IBM Linear Tape-Open (LTO) Ultrium 7, 6, and 5 tape drives, and IBM TS1140 and TS1150 tape drives that are installed in TS4500 and TS3500 tape libraries. LTO Ultrium 7, 6, and 5 tape drives that are installed in a TS3310 tape library also are supported.

The use of IBM Spectrum Archive EE to replace online disk storage with tape in tier 2 and tier 3 storage can improve data access over other storage solutions because it improves efficiency and streamlines management for files on tape. IBM Spectrum Archive EE simplifies the use of physical tape by making it not apparent to the user and manageable by the administrator under a single infrastructure.

IBM Spectrum Archive EE provides the following benefits (IBM, 2016):

- A low-cost storage tier in an IBM Spectrum Scale environment.
- An active archive or big data repository for long-term storage of data that requires file system access to that content.
- File-based storage in the IBM Linear Tape File System™ (LTFS) tape format that is open, self-describing, portable, and interchangeable across platforms.
- Lowers capital expenditure and operational expenditure costs by using cost-effective and energy-efficient tape media without dependencies on external server hardware or software.
- Supports the highly scalable automated TS4500, TS3500, and TS3310 tape libraries.
- Allows the retention of data on tape media for long-term preservation (10+ years).
- Provides the portability of large amounts of data by bulk transfer of tape cartridges between sites for disaster recovery and the initial synchronization of two IBM Spectrum Scale sites by using open-format, portable, self-describing tapes.
- Migration of data to newer tape or newer technology that is managed by IBM Spectrum Scale.
- Ease of management for operational and active archive storage.
- Expands archive capacity by adding and provisioning media without affecting the availability of data that is in the pool.

With IBM Spectrum Archive EE, you can perform the following management tasks on your systems (IBM, 2016):

- Create and define tape cartridge pools for file migrations.
- Migrate files in the IBM Spectrum Scale namespace to the IBM Spectrum Archive tape tier.
- Recall files that were migrated to the IBM Spectrum Archive tape tier back into IBM Spectrum Scale.
- Reconcile file inconsistencies between files in IBM Spectrum Scale and their equivalents in IBM Spectrum Archive.
- Reclaim tape space that is occupied by non-referenced files and non-referenced content that is present on the physical tapes.
- Export tape cartridges to remove them from IBM Spectrum Archive EE system.
- Import tape cartridges to add them to IBM Spectrum Archive EE system.

- Add tape cartridges to IBM Spectrum Archive EE system to expand the tape cartridge pool with no disruption to your system.
- Obtain inventory, job, and scan status of IBM Spectrum Archive EE solution.

Introduction to IBM Spectrum Protect for Space Management

IBM Spectrum Protect for Space Management (also known as UNIX HSM) is the tiering solution that is provided by IBM Spectrum Protect for the IBM Spectrum Scale file system. IBM Spectrum Protect for Space Management can be installed on multiple compute nodes in the IBM Spectrum Scale cluster. IBM Spectrum Protect for Space Management requires the IBM Spectrum Protect server that manages the backend storage media. The backend storage media can be one of the media types optical, disk, object (on or off premises), virtual tape or tape, or a combination of them. All nodes that have UNIX HSM installed participate in file system management. HSM activities, such as migration and recall of files, are distributed to all cluster nodes to use the parallelism inherent in the IBM Spectrum Scale cluster. Also, the processing is parallelized within each cluster node.

IBM Spectrum Scale supports the industry standard mechanism for coupling HSM functions to the file system’s internal data. This standard is called the Data Management application programming interface (DMAPI). DMAPI is provided by IBM Spectrum Scale and IBM Spectrum Protect for Space Management links to the IBM Spectrum Scale DMAPI library. This integration point permits the UNIX HSM to be a qualified data management application to IBM Spectrum Scale.

In terms of migrating files from a higher tier to a lower tier, UNIX HSM features the following file migration states:

- Resident: The file data is only on the live file system (tier 0). No valid copy of the file is in the IBM Spectrum Protect server (tier 1). This state is the migration state for new or changed files.
- Premigrated: The file data has a valid copy in the file system (tier 0) and on the IBM Spectrum Protect server (tier 1). The premigrated state applies for a file that was opened for read and is recalled from the IBM Spectrum Protect server.
- Migrated: The file in tier 0 was replaced by a stub file and is only on the IBM Spectrum Protect server (tier 1). This migration state persists until a data access is attempted on the file data.

When a file data access is performed, a recall is started. The data is staged to tier 0 while the data access system call is blocked, waiting for it to complete. You can recall files from a lower tier to a higher tier by explicitly starting an HSM command to recall a file or list of files, or by using the IBM Spectrum Scale `mmapplypolicy` command to recall files whose attributes match a set of policy rule criteria. UNIX HSM provides several methods to recall a file. The following recall styles can be used for manual or transparent recall (the tape-optimized recall method can be used only for a user-initiated, list-based recall):

- Normal: The application that triggered the recall by accessing the file must wait until the complete file data was recalled from the IBM Spectrum Protect server to the file system. The type of the access (which can be read, write, or truncate) has no effect on the recall mode.
- Partial: Only the portion of the file that was accessed is recalled. Depending on the access type, the recall can be a single block of the file or several collocated blocks. The recall mode can be applied for read access to the file only. For write or truncate access, the recall mode switches back to normal recall automatically.

- Streaming: The application can access the file in streaming mode beginning at byte 0. The recall occurs in the background while the application can access the file sequentially. The recall mode can be applied for read access to the file only. For write or truncate access, the recall mode switches back to normal recall automatically.
- Tape optimized: You generate a list of files to be recalled from tape. The use of the **`recall`** command preprocesses the file list and creates ordered file lists, one for each tape. The tape ordered file lists can then be submitted to the HSM **`recall`** command and the files are recalled.

  Each file system that is enabled for UNIX HSM has one management node. In cases of node or network failure, the file system management automatically fails over to another UNIX HSM node in the cluster. Reconciliation methods ensure that the file system content and the content at the IBM Spectrum Protect server are synchronized.

  Note: The integration of IBM Spectrum Protect for Space Management into the SwiftHLM software stack permits UNIX HSM to perform tape-optimized recalls only.

The IBM Spectrum Protect for Space Management client is integrated with IBM Spectrum Scale to provide more hierarchical storage tiers, such as disk pools and tape pools, which are provided by the IBM Spectrum Protect server. IBM Spectrum Protect for Space Management acts as an external storage pool and integrates seamlessly with the IBM Spectrum Scale storage pool concept. The integration between IBM Spectrum Scale and IBM Spectrum Protect for Space Management provides the following functions:

- Threshold migration: IBM Spectrum Protect for Space Management uses the IBM Spectrum Scale policy engine to monitor the file system thresholds when the user-defined thresholds are reached. The IBM Spectrum Scale policy engine scans the file system and generates candidate lists for migration based on user-defined, fine-grain policy rules. Typically, the rules nominate the large and less frequently used files. Compared to standard file system scan methods, the policy engine significantly increases the overall performance of the threshold migration.

  Note: The integration of IBM Spectrum Protect for Space Management into the SwiftHLM software stack permits UNIX HSM to perform tape optimized migrations only.

- Reconciliation: Data is synchronized between the file system and the IBM Spectrum Protect server. The IBM Spectrum Protect for Space Management client uses the IBM Spectrum Scale policy engine to generate a list of all migrated files in the file system and performs a comparison of the files that are listed and the files that are stored on the IBM Spectrum Protect server.
- High availability: The IBM Spectrum Protect for Space Management failover function ensures the high availability of the HSM services. Typically, one node in the IBM Spectrum Protect or IBM Spectrum Scale cluster is responsible for the HSM activities for a file system. If this node fails (for example, because of network failures or if the system is powered off) the HSM service automatically fails over to another node in the cluster. IBM Spectrum Scale user exit callbacks are used for this function of the IBM Spectrum Protect for Space Management Disaster recovery.

For more information, see the IBM Spectrum Protect for Space Management documentation at IBM Knowledge Center.

Product configurations

With this SwiftHLM solution, many possible configurations with IBM Spectrum Scale Object, IBM Spectrum Scale, and IBM Spectrum Archive EE or IBM Spectrum Protect for Space Management are available. The simplest configuration is a single node with all the software components. For scalability purposes, more nodes can be added. In this section, a high-level view is provided of the software stack that is needed for a proper implementation.

A high-level architecture that uses IBM Spectrum Archive Enterprise Edition is shown in Figure 3, Figure 4 on page 8, and Figure 5 on page 9. The same architecture that uses IBM Spectrum Protect is shown in Figure 6 on page 10, Figure 7 on page 11, and Figure 8 on page 12.

The common configuration in which all software components are installed on each node and each node has direct access to the disk and the tape is shown in Figure 3.

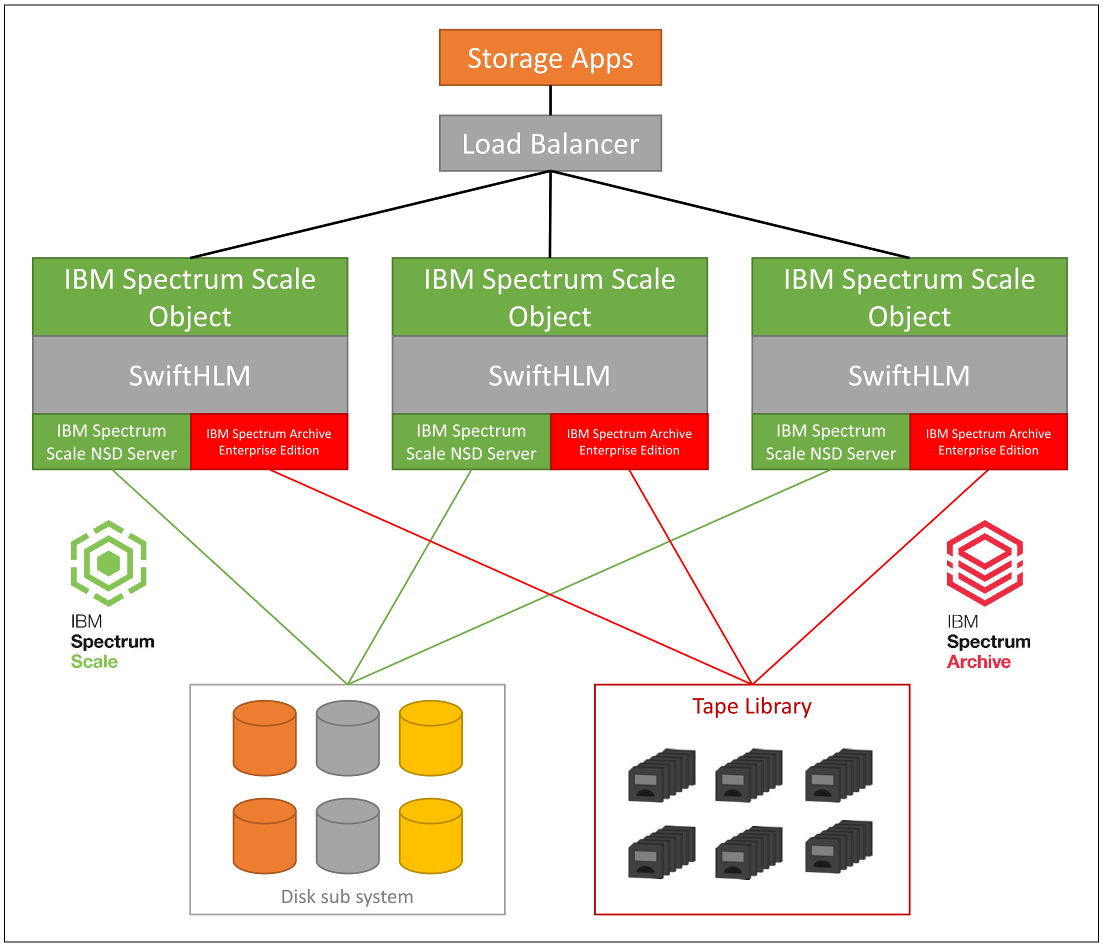

Storage Apps
Load Balancer
IBM Spectrum Scale Object
SwiftHLM
IBM Spectrum Scale NSD Server
IBM Spectrum Archive Enterprise Edition
IBM Spectrum Scale Object
SwiftHLM
IBM Spectrum Scale NSD Server
IBM Spectrum Archive Enterprise Edition
IBM Spectrum Scale Object
SwiftHLM
IBM Spectrum Scale NSD Server
IBM Spectrum Archive Enterprise Edition
IBM Spectrum Scale
IBM Spectrum Archive
Disk sub system
Tape Library

Figure 3 Common configuration of IBM Spectrum Scale and IBM Spectrum Archive

All software components are on all cluster nodes, as shown in Figure 4. All cluster nodes have access to the disk subsystem by way of NSD protocol and to the tape library from IBM Spectrum Archive EE.

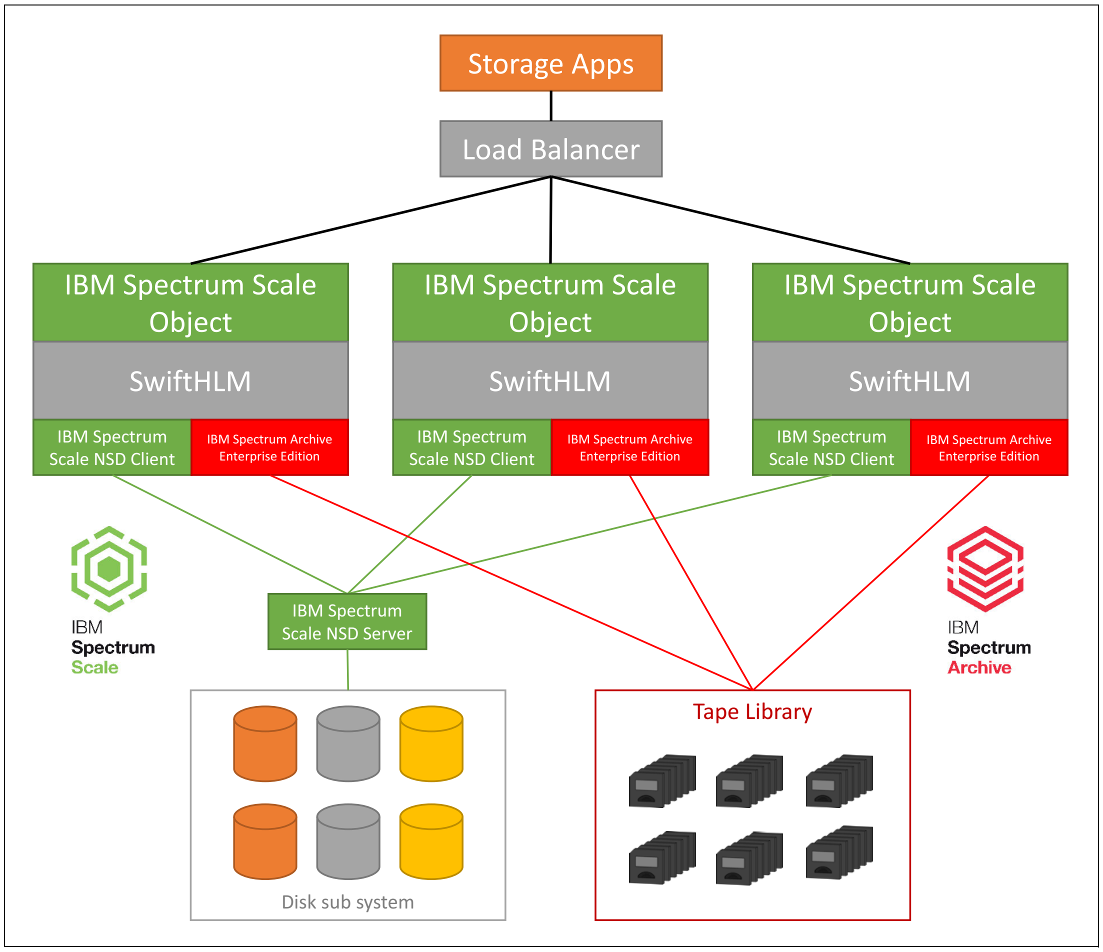

Storage Apps
Load Balancer
IBM Spectrum Scale Object
SwiftHLM
IBM Spectrum Scale NSD Client
IBM Spectrum Archive Enterprise Edition
IBM Spectrum Scale Object
SwiftHLM
IBM Spectrum Scale NSD Client
IBM Spectrum Archive Enterprise Edition
IBM Spectrum Scale Object
SwiftHLM
IBM Spectrum Scale NSD Client
IBM Spectrum Archive Enterprise Edition
IBM Spectrum Scale
IBM Spectrum Scale NSD Server
IBM Spectrum Archive
Tape Library
Disk sub system

Figure 4   IBM Spectrum Scale NSD servers are separated out

All software components are on all cluster nodes. All cluster nodes have access to ESS systems by using NSD protocol and to the tape library by using IBM Spectrum Archive EE, as shown in Figure 5.

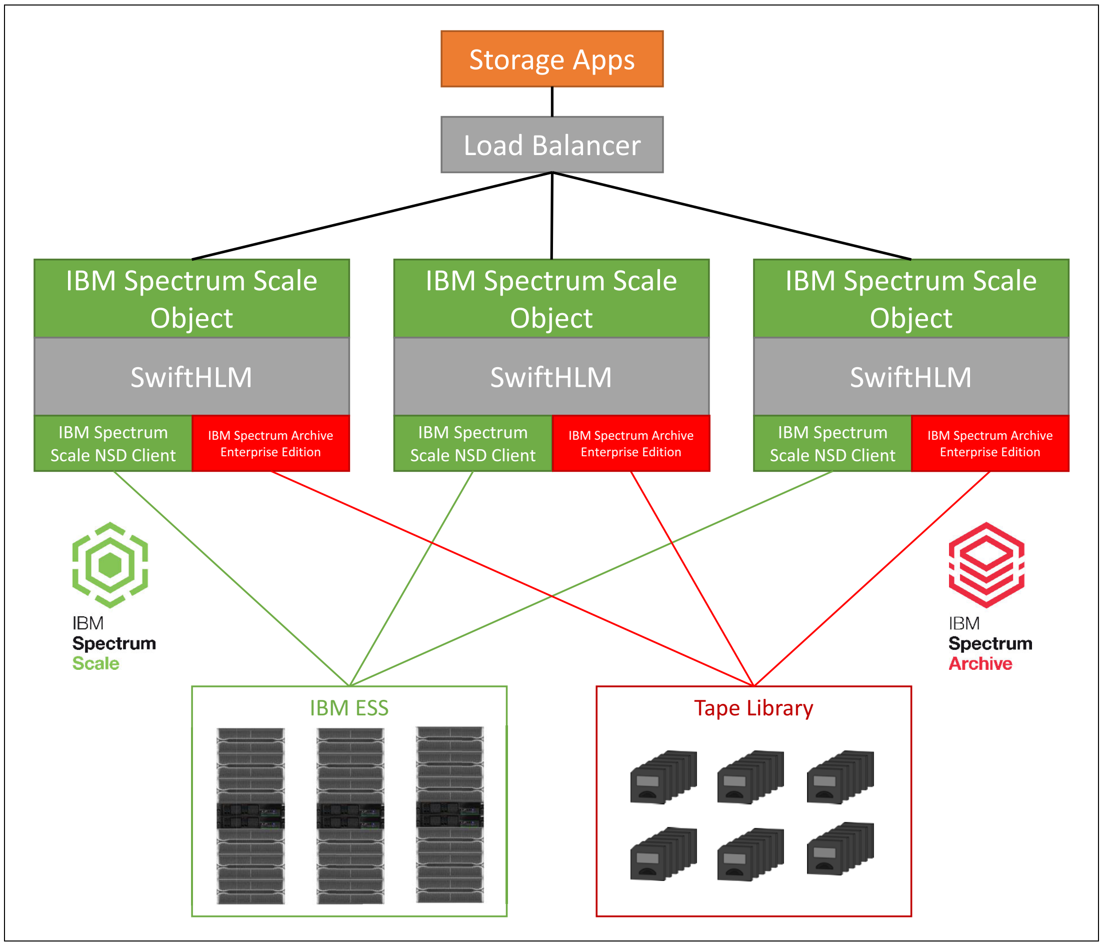

Storage Apps
Load Balancer
IBM Spectrum Scale Object
SwiftHLM
IBM Spectrum Scale NSD Client
IBM Spectrum Archive Enterprise Edition
IBM Spectrum Scale Object
SwiftHLM
IBM Spectrum Scale NSD Client
IBM Spectrum Archive Enterprise Edition
IBM Spectrum Scale Object
SwiftHLM
IBM Spectrum Scale NSD Client
IBM Spectrum Archive Enterprise Edition
IBM Spectrum Scale
IBM Spectrum Archive
IBM ESS
Tape Library

Figure 5 IBM Spectrum Archive configuration that uses ESS as the disk subsystem

All software components are on all cluster nodes, as shown in Figure 6. All cluster nodes have direct access to the disk subsystem and to the IBM Spectrum Protect server.

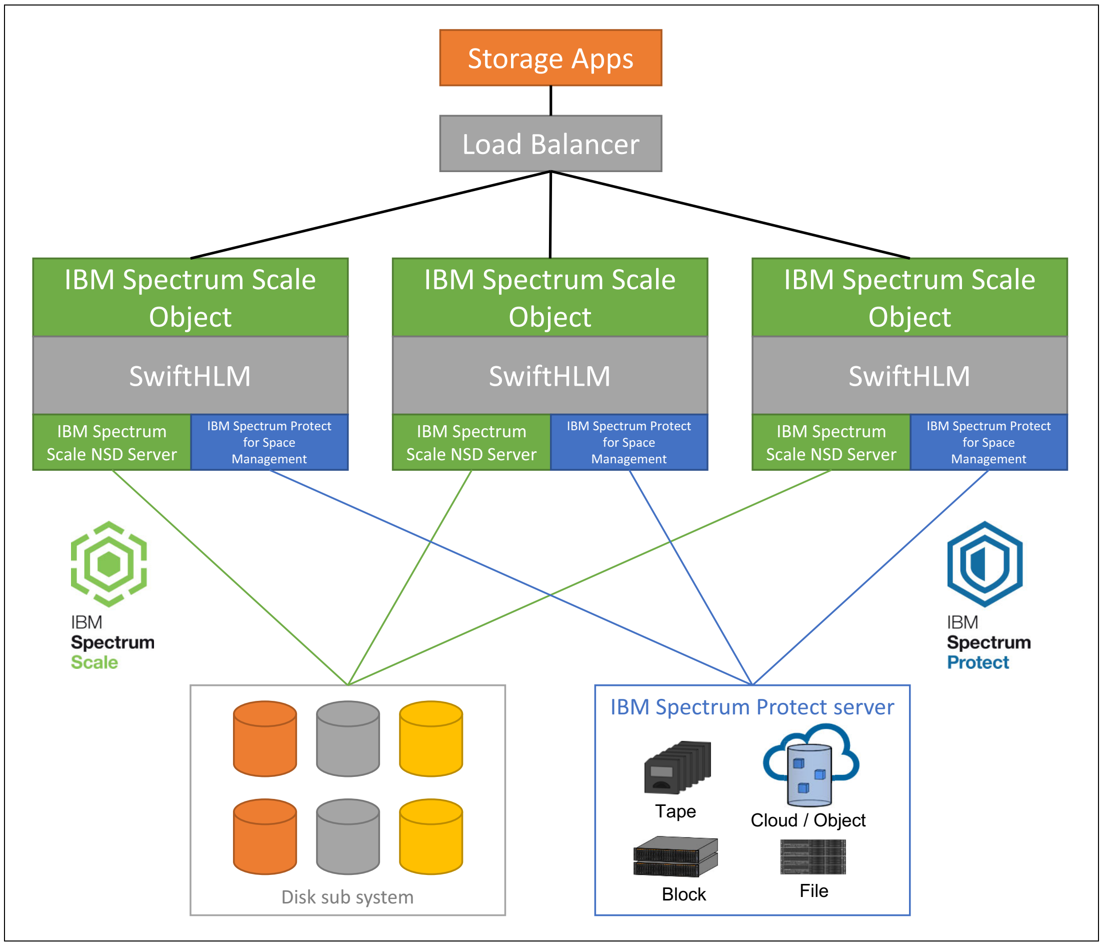

Storage Apps
Load Balancer
IBM Spectrum Scale Object
SwiftHLM
IBM Spectrum Scale NSD Server
IBM Spectrum Protect for Space Management
IBM Spectrum Scale Object
SwiftHLM
IBM Spectrum Scale NSD Server
IBM Spectrum Protect for Space Management
IBM Spectrum Scale Object
SwiftHLM
IBM Spectrum Scale NSD Server
IBM Spectrum Protect for Space Management
IBM Spectrum Scale
IBM Spectrum Protect
Disk sub system
IBM Spectrum Protect server
Tape
Cloud / Object
Block
File

Figure 6 All software components are on all cluster nodes

All software components are on all cluster nodes, as shown in Figure 7. All cluster nodes have access to the disk subsystem by way of NSD protocol and to the IBM Spectrum Protect server.

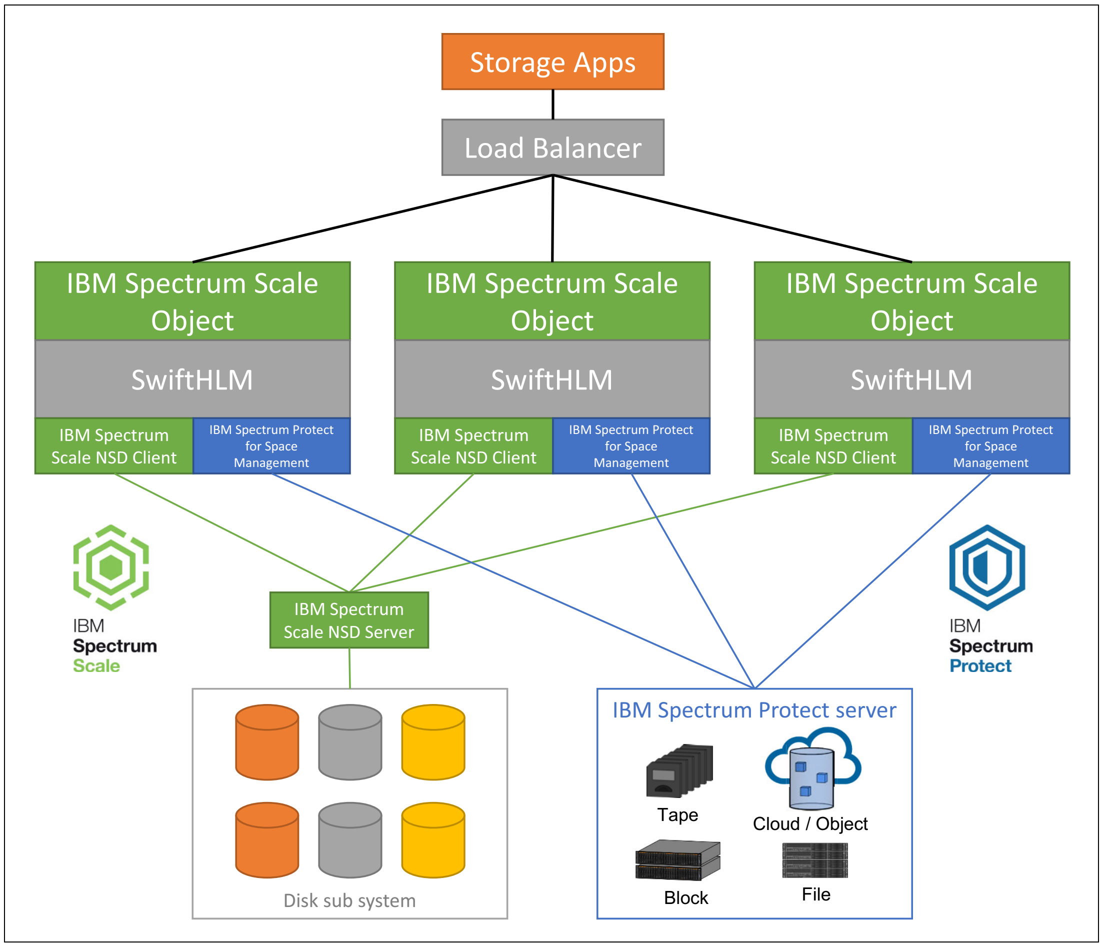

Storage Apps
Load Balancer
IBM Spectrum Scale Object
SwiftHLM
IBM Spectrum Scale NSD Client
IBM Spectrum Protect for Space Management
IBM Spectrum Scale Object
SwiftHLM
IBM Spectrum Scale NSD Client
IBM Spectrum Protect for Space Management
IBM Spectrum Scale Object
SwiftHLM
IBM Spectrum Scale NSD Client
IBM Spectrum Protect for Space Management
IBM Spectrum Scale
IBM Spectrum Scale NSD Server
IBM Spectrum Protect
Disk sub system
IBM Spectrum Protect server
Tape
Cloud / Object
Block
File

Figure 7 All software components are on all cluster nodes with disk access via NSD protocols

All software components are on all cluster nodes. All cluster nodes have access to ESS systems by using NSD protocol and to the IBM Spectrum Protect server, as shown in Figure 8.

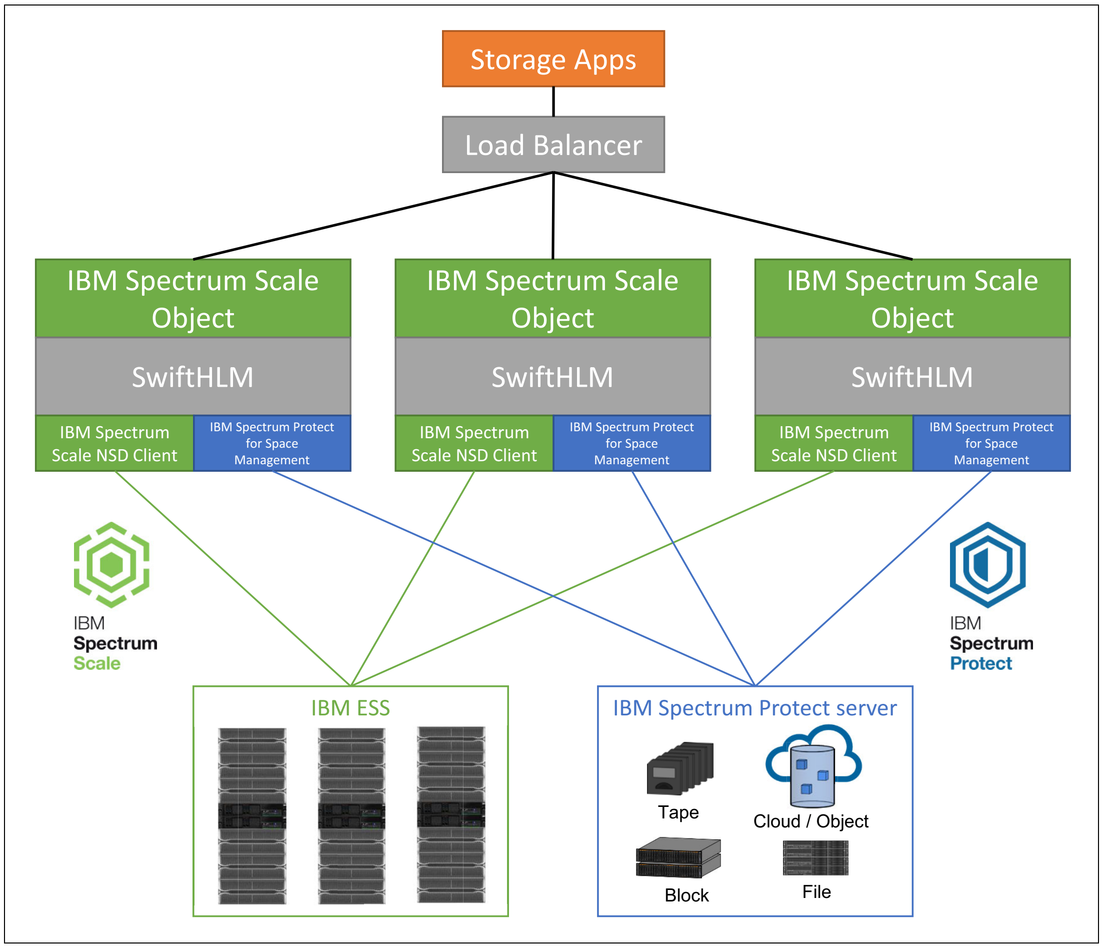

Storage Apps
Load Balancer
IBM Spectrum Scale Object
SwiftHLM
IBM Spectrum Scale NSD Client
IBM Spectrum Protect for Space Management
IBM Spectrum Scale Object
SwiftHLM
IBM Spectrum Scale NSD Client
IBM Spectrum Protect for Space Management
IBM Spectrum Scale Object
SwiftHLM
IBM Spectrum Scale NSD Client
IBM Spectrum Protect for Space Management
IBM Spectrum Scale
IBM Spectrum Protect
IBM ESS
IBM Spectrum Protect server
Tape
Cloud / Object
Block
File

Figure 8 IBM Spectrum Protect configuration that uses ESS as the disk subsystem

Assumptions

The purpose of this document is to describe the benefits of the use of SwiftHLM with IBM Spectrum Archive Enterprise Edition (EE) or IBM Spectrum Protect. This document also guides the administrator through the installation and configuration of SwiftHLM with the IBM Spectrum Archive EE or IBM Spectrum Protect backend, and describes the general set of configurations and scenarios that were validated.

Consider the following points:

- The audience should be familiar with IBM Spectrum Scale, IBM Spectrum Scale Object, and IBM Spectrum Archive EE, IBM Spectrum Protect, and OpenStack Swift. This paper is *not* intended to serve as a comprehensive IBM Spectrum Storage overview. For a comprehensive understanding of IBM Spectrum Storage, see “Related publications” on page 35.
- This paper focuses on the benefits of SwiftHLM with IBM Spectrum Archive EE or IBM Spectrum Protect and does not make any direct comparison to other Object Storage solutions.

IBM Spectrum Scale Object prerequisites

IBM Spectrum Scale Object includes the following prerequisites:

- IBM Spectrum Scale cluster is deployed.
- IBM Spectrum Scale CES (protocol) nodes are deployed on the same nodes as the SwiftHLM/IBM Spectrum Archive EE or IBM Spectrum Protect for Space Management nodes.
- IBM Spectrum Scale Object is enabled and running on CES (protocol) nodes.

For more information about high availability considerations, see section 2.1.6 “High availability and CES monitoring” of the IBM Redpaper publication, *A Deployment Guide for IBM Spectrum Scale Unified File and Object Storage*, REDP-5113.

IBM Spectrum Archive EE prerequisites

If IBM Spectrum Archive is used as storage backend, the following prerequisites must be met:

- The `rpcbind` service is running. If the service is not running, start the service by using the following command:

  ```
  # systemctl start rpcbind
  ```

- SELinux is disabled (preferred) or in permissive mode.
- IBM Spectrum Archive EE is installed and configured on all cluster nodes attending the SwiftHLM activity.
- IBM Spectrum Archive EE pools are created and tapes are formatted or assigned to pools.

IBM Spectrum Protect prerequisites

If IBM Spectrum Protect is used as storage backend, the following prerequisites must be met:

- The rpcbind service is running. If the service is not running, start it by using the following command:

  ```
  # systemctl start rpcbind
  ```

- SELinux is disabled (preferred) or in permissive mode.
- An IBM Spectrum Protect server is configured for Space Management migration and recall activity. Storage media was configured and assigned to the server.
- IBM Spectrum Protect for Space Management is installed and configured on all cluster nodes attending the SwiftHLM activity.
- IBM Spectrum Protect for Space Management is connected by using LAN or SAN to the IBM Spectrum Protect server.

SwiftHLM component overview

The components of the SwiftHLM solution are described in this section. A high-level SwiftHLM component diagram is shown in Figure 9.

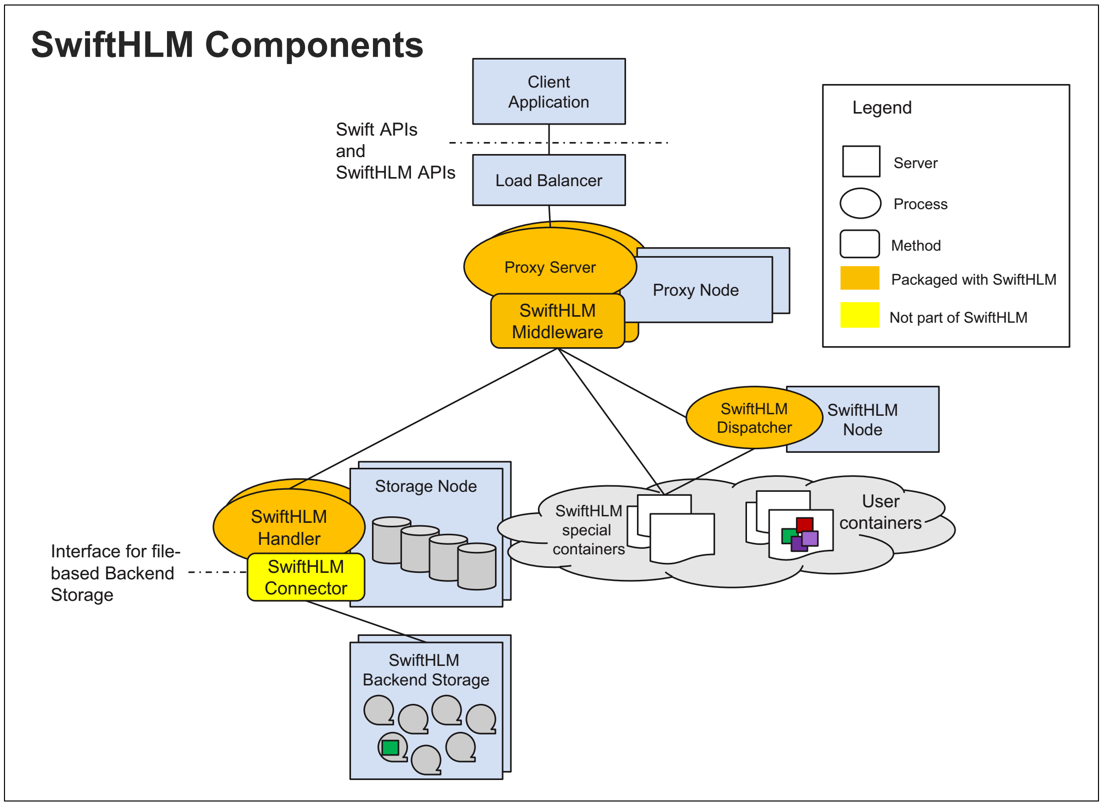


Figure 9 SwiftHLM components

SwiftHLM consists of the following components:

- SwiftHLM middleware

  The SwiftHLM middleware is added to Swift proxy nodes and it extends OpenStack Swift API with new requests to perform the following tasks:

  – Explicitly query object status (is object data on disk or HLM?)
  – Manipulate object state (migrate from disk to tape or recall from tape to disk)
  – Query status of previously submitted requests

  These new requests are also supported on container level, in which they apply for all the objects from the container.

  A container or an object status request is processed synchronously. For that process, the middleware lists object replicas, identifies the storage nodes for each replica, dispatches the subset list asynchronously to the appropriate Swift storage nodes, and collects and merges the results that are obtained from the storage nodes.

  A migration or recall request is processed asynchronously. For that process, the middleware queues the request by storing (queuing) it as an empty object in a special SwiftHLM dedicated container, where the name of the object encodes the request information, and returns response to user that it accepted the request.

  A query of the status for submitted requests is processed synchronously by the middleware for which it queries the requests queue (that is, the content of the special container) for the requests that are related to a container or an object.

- SwiftHLM Dispatcher

  The SwiftHLM Dispatcher is a background daemon that pulls the user request, creates a list of objects, identifies the storage nodes for each object, and dispatches the subset of list asynchronously to the appropriate Swift storage node. The Dispatcher runs on a Swift proxy node, Swift storage node, or dedicated server. Initial release of SwiftHLM allows the user to run single instance of Dispatcher (it might be multiple instances in a future release).

- SwiftHLM Handler

  Installed on each Swift storage node, this handler provides or starts the generic interface for the SwiftHLM backend storage. It also maps the objects to files and submits the mapped list to the backend (Connector for SwiftHLM).

- Connector for SwiftHLM

  SwiftHLM requires a hardware-specific Connector module for SwiftHLM, which is supplied by a vendor (the Connector is not a part of SwiftHLM). In the implementation that is described in this IBM Redpaper publication, the Connector module for SwiftHLM is supplied by IBM.

Installing SwiftHLM on the CES (protocol) nodes

Complete the following steps to install SwiftHLM on the CES (protocol) nodes:

1. Ensure that the `git` rpm is installed by running the following command. If it is not found, install the `git` rpm:

   ```
   # rpm -q git
   git-1.8.3.1-5.el7.x86_64
   ```

2. Issue the **`git clone https://github.com/ibm-research/swifthlm.git`** command to download the `swifthlm` package and store in `/tmp`, as shown in Example 1.

   Example 1 Downloading the swifthlm package

   ```
   # cd /tmp
   # git clone https://github.com/ibm-research/swifthlm.git
   Cloning into 'swifthlm'...
   remote: Counting objects: 200, done.
   remote: Total 200 (delta 0), reused 0 (delta 0), pack-reused 200
   Receiving objects: 100% (200/200), 80.65 KiB | 0 bytes/s, done.
   Resolving deltas: 100% (104/104), done.

   # ls swifthlm
   config  LICENSE.txt  README.txt  setup.py  swifthlm  tests
   ```

3. Complete the following steps (for more information about a list of the distribution packages, see the RedHat's community OpenStack (RDO) website):

   a. Download the following packages to the SwiftHLM nodes:

      - `python-ecdsa`
      - `python-paramiko`
      - `python-pip`

   b. Use **`yum`** to install the three packages.

4. If SwiftHLM was installed, run the following command to uninstall it:

```
# pip uninstall swifthlm
```

5. Change directories into the `swifthlm` directory by running the following command:

```
# cd /tmp/swifthlm/
```

6. Run the **`python setup.py install`** command to install SwiftHLM, as shown in Example 2.

Example 2 Installing the swifthlm package

```
# python setup.py install
running install
running bdist_egg
running egg_info
creating swifthlm.egg-info
writing requirements to swifthlm.egg-info/requires.txt
writing swifthlm.egg-info/PKG-INFO
writing top-level names to swifthlm.egg-info/top_level.txt
writing dependency_links to swifthlm.egg-info/dependency_links.txt
writing entry points to swifthlm.egg-info/entry_points.txt
writing manifest file 'swifthlm.egg-info/SOURCES.txt'
reading manifest file 'swifthlm.egg-info/SOURCES.txt'
writing manifest file 'swifthlm.egg-info/SOURCES.txt'
installing library code to build/bdist.linux-x86_64/egg
running install_lib
running build_py
creating build
creating build/lib
creating build/lib/swifthlm
copying swifthlm/__init__.py -> build/lib/swifthlm
copying swifthlm/dispatcher.py -> build/lib/swifthlm
copying swifthlm/dummy_connector.py -> build/lib/swifthlm
copying swifthlm/handler.py -> build/lib/swifthlm
copying swifthlm/middleware.py -> build/lib/swifthlm
creating build/lib/tests
copying tests/__init__.py -> build/lib/tests
copying tests/test_hlm.py -> build/lib/tests
creating build/bdist.linux-x86_64
creating build/bdist.linux-x86_64/egg
creating build/bdist.linux-x86_64/egg/swifthlm
copying build/lib/swifthlm/__init__.py -> build/bdist.linux-x86_64/egg/swifthlm
copying build/lib/swifthlm/dispatcher.py ->
build/bdist.linux-x86_64/egg/swifthlm
copying build/lib/swifthlm/dummy_connector.py ->
build/bdist.linux-x86_64/egg/swifthlm
copying build/lib/swifthlm/handler.py -> build/bdist.linux-x86_64/egg/swifthlm
copying build/lib/swifthlm/middleware.py ->
build/bdist.linux-x86_64/egg/swifthlm
creating build/bdist.linux-x86_64/egg/tests
copying build/lib/tests/__init__.py -> build/bdist.linux-x86_64/egg/tests
copying build/lib/tests/test_hlm.py -> build/bdist.linux-x86_64/egg/tests
byte-compiling build/bdist.linux-x86_64/egg/swifthlm/__init__.py to
__init__.pyc
byte-compiling build/bdist.linux-x86_64/egg/swifthlm/dispatcher.py to
dispatcher.pyc
```

```
byte-compiling build/bdist.linux-x86_64/egg/swifthlm/dummy_connector.py to
dummy_connector.pyc
byte-compiling build/bdist.linux-x86_64/egg/swifthlm/handler.py to handler.pyc
byte-compiling build/bdist.linux-x86_64/egg/swifthlm/middleware.py to
middleware.pyc
byte-compiling build/bdist.linux-x86_64/egg/tests/__init__.py to __init__.pyc
byte-compiling build/bdist.linux-x86_64/egg/tests/test_hlm.py to test_hlm.pyc
creating build/bdist.linux-x86_64/egg/EGG-INFO
copying swifthlm.egg-info/PKG-INFO -> build/bdist.linux-x86_64/egg/EGG-INFO
copying swifthlm.egg-info/SOURCES.txt -> build/bdist.linux-x86_64/egg/EGG-INFO
copying swifthlm.egg-info/dependency_links.txt ->
build/bdist.linux-x86_64/egg/EGG-INFO
copying swifthlm.egg-info/entry_points.txt ->
build/bdist.linux-x86_64/egg/EGG-INFO
copying swifthlm.egg-info/requires.txt -> build/bdist.linux-x86_64/egg/EGG-INFO
copying swifthlm.egg-info/top_level.txt ->
build/bdist.linux-x86_64/egg/EGG-INFO
zip_safe flag not set; analyzing archive contents...
creating dist
creating 'dist/swifthlm-0.2.1-py2.7.egg' and adding
'build/bdist.linux-x86_64/egg' to it
removing 'build/bdist.linux-x86_64/egg' (and everything under it)
Processing swifthlm-0.2.1-py2.7.egg
Copying swifthlm-0.2.1-py2.7.egg to /usr/lib/python2.7/site-packages
Adding swifthlm 0.2.1 to easy-install.pth file

Installed /usr/lib/python2.7/site-packages/swifthlm-0.2.1-py2.7.egg
Processing dependencies for swifthlm==0.2.1
Searching for swift==2.5.1.dev2
Best match: swift 2.5.1.dev2
Adding swift 2.5.1.dev2 to easy-install.pth file

Using /usr/lib/python2.7/site-packages
Finished processing dependencies for swifthlm==0.2.1
```

7. For a first-time configuration of SwiftHLM, continue with the next steps; otherwise, proceed to step 15.
8. Run the **`mmces service stop OBJ --all`** command to stop the Swift services.
9. Run the **`mmobj config list --ccrfile proxy-server.conf --section pipeline:main --property pipeline`** command to retrieve the current Swift middleware pipeline setting and use the output with a small modification for the next step, as shown in the following example:

   ```
   # mmobj config list --ccrfile proxy-server.conf --section pipeline:main
   --property pipeline'
   ```

   The command output is shown in the following example:

   ```
   pipeline = healthcheck cache formpost tempurl s3token authtoken
   keystoneauth container-quotas account-quotas staticweb bulk slo dlo
   proxy-logging proxy-server
   ```

10. Add the SwiftHLM middleware into the proxy server pipeline:

    a. Change directory to where the SwiftHLM package was extracted (`/tmp/swifthlm`):

    ```
    # cd /tmp/swifthlm/config
    ```

b. Edit the `/tmp/swifthlm/config/proxy-server.conf.merge` file. Although your pipeline statement might not be the same as this example, ensure that you insert **`hlm`** before the final proxy-logging entrances, as shown in the following example:

```
# cat /tmp/swifthlm/config/proxy-server.conf.merge
[pipeline:main]
pipeline =  healthcheck cache formpost tempurl swift3 s3token authtoken
keystoneauth container-quotas account-quotas staticweb bulk slo dlo hlm
proxy-logging proxy-server
[filter:hlm]
use = egg:swifthlm#swifthlm
set log_level = INFO
```

11. Run the **`mmobj config change --ccrfile proxy-server.conf --merge-file /tmp/swifthlm/config/proxy-server.conf.merge`** command to merge the configuration and register the SwiftHLM middleware:

```
# mmobj config change --ccrfile proxy-server.conf --merge-file
/tmp/swifthlm/config/proxy-server.conf.merge
```

12. Configure SwiftHLM to use a specific connector. This example uses the IBM Spectrum Archive Enterprise Edition SwiftHLM connector:

a. Edit the `/tmp/swifthlm/config/object-server.conf.merge` file and uncomment the following lines. Ensure that the `swifthlm_tmp_dir` value matches your IBM GPFS™ file system mount point:

```
set log_level = INFO
swifthlm_connector_module = swifthlmibmsa.ibmsa_swifthlm_connector
```

b. Run the **`mmobj config change --ccrfile object-server.conf --merge-file /tmp/swifthlm/config/object-server.conf.merge`** command to merge the configuration and register the SwiftHLM middleware.

13. Issue the **`mmces service start OBJ --all`** command to activate the middleware by restarting Swift services:

```
# mmces service start OBJ --all
```

```
# mmhealth node show ces -N all
```

14. The Swift user must be configured with passwordless SSH between the IBM Spectrum Scale Object nodes. The Swift user is not a privileged user and cannot run privileged operations. Complete the following steps:

a. Enable the Swift user to be able to log in:

```
# usermod -s /bin/bash swift
```

b. Set the password for the Swift user:

```
# passwd swift
```

c. Run the following command to generate the RSA keys for the Swift user:

```
# su - swift
# ssh-keygen -t rsa
```

The results of the command are shown in the following example:

```
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/lib/swift/.ssh/id_rsa.
Your public key has been saved in /var/lib/swift/.ssh/id_rsa.pub.
```

```
The key fingerprint is:
91:13:29:f4:cd:74:26:90:2e:52:33:45:6c:74:98:39 swift@senpai.abc.labs.com
The key's randomart image is:
+--[ RSA 2048]----+
|      ..=*B+ o   |
|       =.EB +    |
|      . *=.o     |
|     . . .o      |
|      . .S       |
|                 |
|                 |
|                 |
|                 |
+-----------------+
# exit
```

d. Add a list of host keys for all hosts within the IBM Spectrum Scale Object cluster into the `known_hosts` file by running the following command. This example uses `senpai` as the host name:

```
# sudo -u swift ssh-keyscan -t ecdsa senpai,senpai.abc.labs.com,9.1.2.3 >>
/var/lib/swift/.ssh/known_hosts
# senpai SSH-2.0-OpenSSH_6.6.1
```

Note: The list of hosts contains the short name (`senpai`), full host name (`senpai.abc.labs.com`), and the IP address (`9.1.2.3`). The command should be repeated for the rest of the IBM Spectrum Scale Object nodes.

e. Run the following command to add the list of public keys into the `authorized_keys` file to permit the user to log in to the IBM Spectrum Scale Object nodes by using Swift user without supplying a password:

```
cat /var/lib/swift/.ssh/id_rsa.pub >> /var/lib/swift/.ssh/authorized_keys
```

Note: For multiple nodes, ensure the Swift user public key from all IBM Spectrum Scale Object nodes is included in the `authorized_keys` file of each IBM Spectrum Scale Object node.

f. The `authorized_keys` file must be `chmod 600`; otherwise, password-less SSH does not work when SELinux is in permissive mode. Therefore, add the following line:

```
# chmod 600 /var/lib/swift/.ssh/authorized_keys
```

g. Run the following commands to change ownership to the Swift user:

```
# chown swift:swift /var/lib/swift/.ssh/authorized_keys
# chown swift:swift /var/lib/swift/.ssh/known_hosts
```

15. To start the SwiftHLM Dispatcher service, select one of the IBM Spectrum Scale Object nodes and run the following commands:

```
# cp /tmp/swifthlm/config/swifthlm.dispatcher.service /etc/systemd/system

# systemctl daemon-reload

# systemctl start swifthlm.dispatcher

# systemctl status swifthlm-dispatcher
```

Installing the SwiftHLM backend connector

A SwiftHLM backend connector is available for each product: IBM Spectrum Archive EE or IBM Spectrum Protect for Space Management.

Complete the following steps to install the appropriate SwiftHLM connector:

1. Ensure the attr rpm is installed (install the attr rpm if it is missing):

   ```
   # rpm -q attr
   attr-2.4.46-12.el7.x86_64
   ```

2. Choose one of the following options:

   – For an IBM Spectrum Archive EE implementation:

      i. Download IBM Spectrum Archive EE V1.2.4.x package or later from IBM Fix Central.

      ii. Extract the IBM Spectrum Archive EE package. It is automatically placed in the `/root/rpm` directory. The file is named `swifthlmconnector.tgz`.

      iii. Extract the `swifthlmconnector.tgz` package into the `/tmp` directory.

      iv. Run the following command.

      ```
      # tar -xzvf swifthlmconnector.tgz -C /tmp
      ```

   – For an IBM Spectrum Protect implementation:

      i. Download the IBM Spectrum Protect for Space Management RPM packages.

      Follow the installation procedure that is described in IBM Knowledge Center.

      ii. Download the SwiftHLM connector.

3. If SwiftHLM connector was installed, run the following command to uninstall it:

   ```
   # pip uninstall swifthlmibmsa
   ```

4. Change the directories to the `ibmsa-swifthlm-connector-master` directory:

   ```
   # cd /tmp/ibmsa-swifthlm-connector-master/
   ```

5. The appropriate backend must be changed with the `swifthlmibmsa/ibmsa_swifthlm_connector.py` file:

   a. For an IBM Spectrum Archive implementation, change BACKEND to:

   ```
   BACKEND = 'ibmsa'
   #BACKEND = 'ibmsp'
   #BACKEND = 'ibmsp-dsmls'
   ```

   b. For an IBM Spectrum Protect implementation, change BACKEND to:

   ```
   #BACKEND = 'ibmsa'
   BACKEND = 'ibmsp'
   #BACKEND = 'ibmsp-dsmls'
   ```

6. Run the **`python setup.py install`** command to install the SwiftHLM connector, as shown in Example 3.

Example 3 Installing SwiftHLM connector

```
# python setup.py install
running install
running bdist_egg
running egg_info
writing requirements to swifthlmibmsa.egg-info/requires.txt
```

```
writing swifthlmibmsa.egg-info/PKG-INFO
writing top-level names to swifthlmibmsa.egg-info/top_level.txt
writing dependency_links to swifthlmibmsa.egg-info/dependency_links.txt
reading manifest file 'swifthlmibmsa.egg-info/SOURCES.txt'
writing manifest file 'swifthlmibmsa.egg-info/SOURCES.txt'
installing library code to build/bdist.linux-x86_64/egg
running install_lib
running build_py
creating build/bdist.linux-x86_64/egg
creating build/bdist.linux-x86_64/egg/swifthlmibmsa
copying build/lib/swifthlmibmsa/ibmsa_swifthlm_connector.py ->
build/bdist.linux-x86_64/egg/swifthlmibmsa
copying build/lib/swifthlmibmsa/__init__.py ->
build/bdist.linux-x86_64/egg/swifthlmibmsa
byte-compiling
build/bdist.linux-x86_64/egg/swifthlmibmsa/ibmsa_swifthlm_connector.py to
ibmsa_swifthlm_connector.pyc
byte-compiling build/bdist.linux-x86_64/egg/swifthlmibmsa/__init__.py to
__init__.pyc
creating build/bdist.linux-x86_64/egg/EGG-INFO
copying swifthlmibmsa.egg-info/PKG-INFO ->
build/bdist.linux-x86_64/egg/EGG-INFO
copying swifthlmibmsa.egg-info/SOURCES.txt ->
build/bdist.linux-x86_64/egg/EGG-INFO
copying swifthlmibmsa.egg-info/dependency_links.txt ->
build/bdist.linux-x86_64/egg/EGG-INFO
copying swifthlmibmsa.egg-info/requires.txt ->
build/bdist.linux-x86_64/egg/EGG-INFO
copying swifthlmibmsa.egg-info/top_level.txt ->
build/bdist.linux-x86_64/egg/EGG-INFO
zip_safe flag not set; analyzing archive contents...
creating 'dist/swifthlmibmsa-0.2.1-py2.7.egg' and adding
'build/bdist.linux-x86_64/egg' to it
removing 'build/bdist.linux-x86_64/egg' (and everything under it)
Processing swifthlmibmsa-0.2.1-py2.7.egg
Removing /usr/lib/python2.7/site-packages/swifthlmibmsa-0.2.1-py2.7.egg
Copying swifthlmibmsa-0.2.1-py2.7.egg to /usr/lib/python2.7/site-packages
swifthlmibmsa 0.2.1 is already the active version in easy-install.pth

Installed /usr/lib/python2.7/site-packages/swifthlmibmsa-0.2.1-py2.7.egg
Processing dependencies for swifthlmibmsa==0.2.1
Searching for swifthlm==0.2.1
Best match: swifthlm 0.2.1
Processing swifthlm-0.2.1-py2.7.egg
swifthlm 0.2.1 is already the active version in easy-install.pth

Using /usr/lib/python2.7/site-packages/swifthlm-0.2.1-py2.7.egg
Searching for swift==2.5.1.dev2
Best match: swift 2.5.1.dev2
Adding swift 2.5.1.dev2 to easy-install.pth file

Using /usr/lib/python2.7/site-packages
Finished processing dependencies for swifthlmibmsa==0.2.1
```

7. Run one of the following **cp** commands:
   - If IBM Spectrum Archive (SA) backend is used, run:

     ```
     cp ibmsa-swifthlm-connector-master/conf/swifthlm.sa /etc/sudoers.d/swifthlm
     ```
   - If IBM Spectrum Protect backend is used, run:

     ```
     cp ibmsa-swifthlm-connector-master/conf/swifthlm.sp /etc/sudoers.d/swifthlm
     ```
8. Run the **chmod 440 /etc/sudoers.d/swifthlm** command:

   ```
   # chmod 440 /etc/sudoers.d/swifthlm
   ```
9. Run the **mmces service stop OBJ --all** command to stop the Swift services:

   ```
   # mmces service stop OBJ --all
   ```
10. Edit the template file called `/tmp/ibmsa-swifthlm-connector-master/config/object-server.conf.merge` and modify the content of the file to match your IBM Spectrum Archive EE or IBM Spectrum Protect configuration. Ensure that the `connector_tmp_dir` and `gpfs_filesystem_or_fileset` values match your GPFS file system mount point.

    If IBM Spectrum Archive is the connector, the library and tape cartridge pool configuration must also be configured, which is the `library` and `tape_storage_pool` fields. The tape storage pool configuration can have one or up to three `pool@library` parameters, separated by a single space character. The following example includes the two required lines for the IBM Spectrum Archive `library` and `tape_storage_pool` fields:

    ```
    # Configure IBM Spectrum Archive/Protect Connector/Backend for SwiftHLM
    [ibmsasp]
    # IBM Spectrum Archive/Protect Connector configuration
    connector_tmp_dir = /ibm/gpfs/tmp/swifthlm
    # IBM Spectrum Archive/Protect Backend configuration
    gpfs_filesystem_or_fileset = /ibm/gpfs
    library = lib0
    tape_storage_pool = pool0@lib0
    ```
11. Run the **mmobj config change --ccrfile object-server.conf --merge-file /tmp/ibmsa-swifthlm-connector-master/config/object-server.conf.merge** command to merge the configuration and register the SwiftHLM middleware.
12. Run the **mmces service start OBJ --all** command to activate the middleware by restarting Swift services:

    ```
    # mmces service start OBJ --all
    ```

    ```
    # mmhealth node show ces -N all
    ```
13. Ensure that the Connector temporary directory exists and is accessible to the Swift user. Also, ensure that your directory name matches what you specified for the `connector_tmp_dir` value:

    ```
    # mkdir -p /ibm/gpfs/tmp/swifthlm
    ```

    ```
    # chown swift:swift -R /ibm/gpfs/tmp/swifthlm
    ```

Usage examples

This section describes the functions of SwiftHLM, including command usage examples.

SwiftHLM provides the in-band control method for the data placement between the disk and economical HLM storage devices. Swift API and SwiftHLM use the same access point (`hostname` and `port #`).

The following methods are available for placement control and status monitoring:

- `migrate`: Move an object (or all objects in a container) to HLM tier.
- `recall`: Restore an object (or all objects in a container) from HLM tier.
- `status`: Query the current placement of object (or objects in a container).
- `requests`: Checks if any pending operation exists for the specified object (or container).

Swift APIs work as is, except that GET method fails if the object is placed in the HLM tier.

The SwiftHLM solution is defined by the following interfaces:

- Frontend: SwiftHLM APIs are the control method for the user.
- Backend: This interface is the communication protocol between SwiftHLM Handler and Connector for SwiftHLM.

The following elements are needed to use SwiftHLM commands:

- Authorization token

  IBM Spectrum Scale Object Storage relies on the Keystone service for validating an incoming user that is associated with a request before processing the object access request. Keystone is the identity validation service that is used by different OpenStack services.

  The Identity back end is the registry of a user name and password. Before sending a request to the IBM Spectrum Scale object store, the user sends a request to the Keystone service to obtain a token that is required for accessing the object service. The request to the Keystone service contains the user name and password. The Keystone service validates the user name and password with the configured Identity back end. On successful validation of the user name and password, Keystone returns a token to the user. The obtained token is used for subsequent object store access requests.

  After a user is authenticated, a token is generated for authorization and access to an OpenStack environment, which includes SwiftHLM.

  Use the following command to acquire the TOKEN:

  ```
  TOKEN=`swift stat -v | awk '/Auth Token:/ {print $3}'`
  ```

- Account information

  The account information is the top-level element in the IBM Spectrum Scale Object Storage system hierarchy. An account contains a list of the containers in the account. In the OpenStack environment, *account* is synonymous with *project* or *tenant*, as used by Keystone.

  Use the following command to acquire the account information:

  ```
  ACCT=`swift stat -v | awk '/Account:/ {print $2}'`
  ```

  After you have the Authorization token and the account information, you can run the SwiftHLM commands.

Requests command

Querying requests status (REQUESTS) for an object or a container are processed by SwiftHLM middleware by reading listing of the special HLM-dedicated container or containers. If there are no pending (incomplete or failed) requests for a container, the previously submitted operations for that container might be considered completed. This process is more efficient than to query the state for each object of a container.

The **requests** command uses the following format:

```
curl -H "X-Auth-Token: $TOKEN" -X GET
"http://<CES node>:8080/hlm/v1/requests/$ACCT/<container>”
```

The **requests** command where `<CES node>` is **tora-ces** and the `<container>` is **hlmc** is shown in the following example:

```
curl -X GET -H "X-Storage-Token: $TOKEN"
"http://tora-ces:8080/hlm/v1/requests/$ACCT/hlmc"
```

The result of the **requests** command when no requests are on the queue is shown in the following example:

```
["There are no pending or failed SwiftHLM requests."]
```

The result of the **requests** command when there is a pending request for object `test_object_4` in the Redpaper container on the queue is shown in the following example:

```
[
"20170316172233.418--migrate--AUTH_f3013cd87a264a8b87a44b831bfc7579--Redpaper--0--
test_object_4--pending"
]
```

Status command

The **status** command uses the following format:

```
curl -H "X-Auth-Token: $TOKEN" -X GET
"http://<CES node>:8080/hlm/v1/status/$ACCT/<container>”
```

The **status** command where `<CES node>` is **tora-ces** and the `<container>` is **Redpaper** is shown in the following example:

```
# curl -X GET -H "X-Storage-Token: $TOKEN"
"http://tora-ces:8080/hlm/v1/status/$ACCT/Redpaper" | python -m json.tool
```

The output of the status command can result in one of the following states for the object:

- Resident: The object is on disk only (takes up disk space).
- Premigrated: The object is on disk and tape (takes up disk space).
- Migrated: The object is on tape only (no disk space is used).

The result of the **status** command where the object are resident is shown in the following example:

```
{
    "/AUTH_f3013cd87a264a8b87a44b831bfc7579/Redpaper/test_object_1": "resident",
    "/AUTH_f3013cd87a264a8b87a44b831bfc7579/Redpaper/test_object_2": "resident",
    "/AUTH_f3013cd87a264a8b87a44b831bfc7579/Redpaper/test_object_3": "resident"
}
```

The **status** command for the object `test_object_1` is shown in the following example:

```
# curl -X GET -H “X-Storage-Token: $TOKEN”
“http://tora-ces:8080/hlm/v1/status/$ACCT/Redpaper/test_object_1” | python -m
json.tool
```

The results of the **status** command for the object `test_object_1` are shown in the following example:

```
{
    "/AUTH_f3013cd87a264a8b87a44b831bfc7579/Redpaper/test_object_1": "resident"
}
```

Migrate command

Migrating an object that is named `test_object_4` in the Redpaper container (including the acceptance of the request) is shown in the following example:

```
# curl -X POST -H “X-Storage-Token: $TOKEN”
“http://tora-ces:8080/hlm/v1/migrate/$ACCT/Redpaper/test_object_4”

Accepted migrate request.
```

Recall command

Recalling an object that is named `test_object_4` in the Redpaper container (including the results output stating the acceptance of the request) is shown in the following example:

```
# curl -X POST -H “X-Storage-Token: $TOKEN”
“http://tora-ces:8080/hlm/v1/recall/$ACCT/Redpaper/test_object_4”

Accepted recall request.
```

The use of the **requests** command for the Redpaper container to show that the migration of `test_object_4` is pending is shown in the following example. If there was a problem, the requests output show “failed” instead of “pending”:

```
# curl -X GET -H “X-Storage-Token: $TOKEN”
“http://tora-ces:8080/hlm/v1/requests/$ACCT/Redpaper” | python -mjson.tool

[
“20170316180259.938--recall--AUTH_f3013cd87a264a8b87a44b831bfc7579--Redpaper--0--t
est_object_4--pending”
]
```

End-to-end example of uploading, migrating, recalling, and downloading files as objects

This use case shows uploading a file as an object to IBM Spectrum Scale Object Storage, migrating the object to tape, recalling the object back to IBM Spectrum Scale Object Storage, and then, downloading the object as a file.

The **curl** command that is used to upload a file as an object from your system into IBM Spectrum Scale Object Storage is shown in the following example. In this example, the file name after `-T` (`movie1.mpg`) is uploaded as an object that is named `test_object_4`:

```
# curl -X PUT -H "X-Storage-Token: $TOKEN" -T movie1.mpg
"http://tora-ces:8080/v1/$ACCT/Redpaper/test_object_4"
```

Use the **status** command to check the state of `test_object_4`:

```
# curl -X GET -H "X-Storage-Token: $TOKEN"
"http://tora-ces:8080/hlm/v1/status/$ACCT/Redpaper" | python -m json.tool
{
    "/AUTH_f3013cd87a264a8b87a44b831bfc7579/Redpaper/test_object_1": "resident",
    "/AUTH_f3013cd87a264a8b87a44b831bfc7579/Redpaper/test_object_2": "resident",
    "/AUTH_f3013cd87a264a8b87a44b831bfc7579/Redpaper/test_object_3": "resident",
    "/AUTH_f3013cd87a264a8b87a44b831bfc7579/Redpaper/test_object_4": "resident"
}
```

Migrate the object `test_object_4` by using the **migrate** command, as shown in the following example:

```
# curl -X POST -H “X-Storage-Token: $TOKEN”
“http://tora-ces:8080/hlm/v1/migrate/$ACCT/Redpaper/test_object_4”

Accepted migrate request.
```

Use the **status** command to check the state of the object `test_object_4`, as shown in the following command:

```
# curl -X GET -H "X-Storage-Token: $TOKEN"
"http://tora-ces:8080/hlm/v1/status/$ACCT/Redpaper" | python -m json.tool
{
    "/AUTH_f3013cd87a264a8b87a44b831bfc7579/Redpaper/test_object_1": "resident",
    "/AUTH_f3013cd87a264a8b87a44b831bfc7579/Redpaper/test_object_2": "resident",
    "/AUTH_f3013cd87a264a8b87a44b831bfc7579/Redpaper/test_object_3": "resident",
    "/AUTH_f3013cd87a264a8b87a44b831bfc7579/Redpaper/test_object_4": "migrated"
}
```

Recall `test_object_4` to IBM Spectrum Scale Object Storage by using the **recall** command, as shown in the following example:

```
# curl -X POST -H “X-Storage-Token: $TOKEN”
“http://tora-ces:8080/hlm/v1/recall/$ACCT/Redpaper/test_object_4”

Accepted recall request.
```

Use the **status** command to check the state of the recalled object `test_object_4`, as shown in the following example:

```
# curl -X GET -H "X-Storage-Token: $TOKEN"
"http://tora-ces:8080/hlm/v1/status/$ACCT/Redpaper" | python -m json.tool
{
    "/AUTH_f3013cd87a264a8b87a44b831bfc7579/Redpaper/test_object_1": "resident",
    "/AUTH_f3013cd87a264a8b87a44b831bfc7579/Redpaper/test_object_2": "resident",
    "/AUTH_f3013cd87a264a8b87a44b831bfc7579/Redpaper/test_object_3": "resident",
    "/AUTH_f3013cd87a264a8b87a44b831bfc7579/Redpaper/test_object_4": "premigrated"
}
```

The following **curl** command downloads the recalled object as a file:

```
# curl -X GET -H "X-Storage-Token: $TOKEN" -o movie1.mpg
"http://tora-ces:8080/v1/$ACCT/Redpaper/test_object_4"
```

If you attempt to download an object that is still in the migrated state, you receive the following precondition failure:

```
HTTP/1.1 412 Precondition Failed
Content-Length: 118
Content-Type: text/plain
X-Trans-Id: tx783ae4a6288a4cfe872f0-0058cb0a08
Date: Thu, 16 Mar 2017 21:56:25 GMT

Object /AUTH_f3013cd87a264a8b87a44b831bfc7579/Redpaper/test_object_4 needs to be
RECALL-ed before it can be accessed.
```

Note: You can upload multiple objects first and then migrate all the objects in a container by using one **migrate** command. Also, you can recall all of the objects back to the container by using one **recall** command.

Attempting to migrate with invalid object or invalid container

The following examples show the use of the **migrate** command with an invalid object and an invalid container name. This output is similar if an invalid object or invalid container name is used by all SwiftHLM commands (migrate, recall, requests, and status):

```
# curl -X POST -H “X-Storage-Token: $TOKEN”
“http://tora-ces:8080/hlm/v1/migrate/$ACCT/Redpaper/invalid_test_object”

Object AUTH_f3013cd87a264a8b87a44b831bfc7579/Redpaper/invalid_test_object does not
exist.

# curl -X POST -H “X-Storage-Token: $TOKEN”
“http://tora-ces:8080/hlm/v1/migrate/$ACCT/Invalid_Redpaper”

Account/Container AUTH_f3013cd87a264a8b87a44b831bfc7579/Invalid_Redpaper does not
exist.
```

SwiftHLM best practices and limitations

This section describes the following best practices and limitations for implementing SwitHLM with IBM Spectrum Scale and IBM Spectrum Archive:

- Do not use active **`status`** polling to determine whether object is recalled

  After issuing a **`recall`** command, it is suggested that a **`get`** command is run to download objects after checking that the command was removed from the request queue. There is no need to run a **`status`** command to check status of each object. This suggestion assumes that there are no pending **`migrate`** requests for the object or its container that were issues after the **`recall`** request.

- SwiftHLM migration and recall requests

  SwiftHLM migration and recall requests are run in a FIFO manner. The submission time information is stored (as part of empty object name) in the queue of requests, and the dispatcher pulls the oldest request and processes one request at a time, by distributing it in parallel across the storage nodes. After the request is completed, it is removed from the queue and the next oldest request is pulled and processed. (Future options should allow prioritizing recalls or removing previous migration requests for the same object and container if a recall for an object or container is submitted.)

- SwiftHLM **`migrate`** command

  The **`migrate`** command should not be issued immediately after object upload. The delay should be greater than 2 minutes before issuing the **`migrate`** command.

- SwiftHLM **`status`** command

  Not more than 10 `status` commands should be submitted in parallel (the limitation to be removed in the future).

- SwiftHLM requests command

  The **`requests`** command is issued at the container level. Future considerations include extending the command to the account level.

IBM Spectrum Scale and IBM Spectrum Archive considerations

This section describes the following IBM Spectrum Scale and IBM Spectrum Archive considerations:

- Swift Replication

  IBM Spectrum Scale Object Storage does not replicate data within a region. The combination of having all ring devices available on all nodes and the dynamic network address management of IBM Spectrum Scale Clustered Export Services (CES), which ensures that all ring devices are always accessible. Data protection of the underlying file system can be provided through the IBM Spectrum Scale RAID capabilities, storage controller RAID, or IBM Spectrum Scale replication.

  A standard OpenStack Swift deployment depends on replication because the deployment is made up of disparate nodes with private storage. The loss of a node means that the ring devices and the object data that is contained on that node are no longer available. A standard Swift deployment replicates the data so that it can acquire it from another node if the primary node is unavailable. IBM Spectrum Scale Object Storage is built upon IBM Spectrum Scale and the CES framework. Therefore, it does not need replication to ensure data access.

However, replication is required for Object Storage in a multi-region environment. Each region has one copy of the data. If there are two or three regions, there are two or three copies of the data. However, the data is not duplicated within a region.

It is recommended to use IBM Spectrum Archive EE to create multiple replicas of the object on tape during the migration process. The purpose of the replica function is to enable creating multiple LTFS copies of each object file during migration that can be used for disaster recovery, including across two tape libraries at two different locations. All copies of the object file on tape is called a *replica*. With IBM Spectrum Archive EE, up to three replicas can be created (for a total of three copies of the object file on various tapes).

- Do not use GPFS external pool policies on the object fileset

  IBM Spectrum Scale Object provides more Swift Storage Policies that are based on file system functions, such as Compression and Encryption. It uses ILM policies that are run scheduled (Compression migration rule) or are installed (Encryption rule). If it is needed to install your own IBM Spectrum Scale ILM Policy rules, validate if for the fileset and file system on which you want to apply your rules (other rules exist).

  Use the command **`mmobj policy list`** to see other created compression and encryption policies and the **`mmlspolicy`** command to list policies. Validate that policy rules do not contradict with rules that you want to apply.

  Also, ensure that no migration and placement rules are created for filesets on which SwiftHLM is used. For more information about IBM Spectrum Scale ILM Policies, see the Information lifecycle management for IBM Spectrum Scale topic at IBM Knowledge Center.

  HSM policies and commands of IBM Spectrum Archive EE and IBM Spectrum Protect for Space Management should not be used against the object fileset so that SwiftHLM can fully control data placement. To prevent SwiftHLM-managed objects from being listed as the migration candidates, use the rule statement in the migration policy for excluding the object fileset.

- Handoff nodes

  Swift uses so-called handoff nodes if a primary node is not available during an object put request because of a power failure, network issues, or other reasons. After the primary node becomes available, the object that is written to the handoff node is moved to the primary node by the object-replicator automatically. This function causes issues for the migration process. At the same time, an object is migrated to Tape from the handoff node location, object-replicator might correct the object location as the primary node becomes available again. The Object data file attempts to move to another node while it is in the migration process. Then, objects that are stored on a handoff location are skipped for migration until the object is at the primary location.

Troubleshooting HLM

For more information about IBM Spectrum Scale Object related errors (REST Request shows an error; for example, 4xx or 5xx), see the Object issues topic in IBM Knowledge Center.

For more information about how to view or change log levels on any of the object-related services and about where to find the log files, see the Object logs topic in IBM Knowledge Center.

For SwiftHLM proxy middleware, the `/var/log/swift/proxy-server.log` file is the most important file for debugging purposes. To filter for specific SwiftHLM traces, create a file that is named `/etc/rsyslog.d/00-swifthlmlog.conf` with the following content:

```
if $programname contains 'hlm-middleware' then /var/log/swift/hlm.log
if $programname contains 'hlm-middleware' and $syslogseverity <= 3 then
/var/log/swift/hlm.error
if $programname contains 'hlm-middleware' then ~
if $programname contains 'dispatcher' then /var/log/swift/hlm.log
if $programname contains 'dispatcher' and $syslogseverity <= 3 then /var/log/swift/hlm.error
if $programname contains 'dispatcher' then ~
if $programname contains 'handler' then /var/log/swift/hlm.log
if $programname contains 'handler' and $syslogseverity <= 3 then /var/log/swift/hlm.error
if $programname contains 'handler' then ~
if $programname contains 'connector' then /var/log/swift/hlm.log
if $programname contains 'connector' and $syslogseverity <= 3 then /var/log/swift/hlm.error
if $programname contains 'connector' then ~
```

After adding the file, restart the rsyslog daemon to make the change effective, as shown in the following example:

```
# systemctl restart rsyslog
```

For more information, see the following resources:

- IBM Spectrum Scale Object related errors (for example, node unhealthy), see the CES monitoring and troubleshooting topic at IBM Knowledge Center.
- Troubleshooting with IBM Spectrum Archive EE, see Chapter 10 of the IBM Redbooks® publication, *IBM Spectrum Archive Enterprise Edition V1.2.2 Installation and Configuration Guide*, SG24-8333.
- Troubleshooting IBM Spectrum Protect for Space Management, see the Troubleshooting the space management client topic at IBM Knowledge Center.

To improve the serviceability of the SwiftHLM software stack, the best practice is to configure the IBM Spectrum Protect error logging and HSM logging.

Note: The UNIX HSM activities in the cluster can be distributed to different nodes in the cluster. The best practice for cluster-wide error logging and HSM logging is to define a global work directory. The global work directory can be a dedicated IBM Spectrum Scale file system that is mounted on all nodes in the cluster that is attending the IBM Spectrum Protect for Space Management activities.

For more information about configuring error logging and HSM logging for IBM Spectrum Protect for Space Management, see the Logs for HSM activity and error messages topic at IBM Knowledge Center.

For more information about troubleshooting IBM Spectrum Protect server, see the Troubleshooting topic at IBM Knowledge Center.

Authors

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Khanh Ngo is an IBM Senior Technical Staff Member (STSM) in Tucson, Arizona serving as the Tape Storage Test Architect. He sets test strategies, designs tests, and tests solutions for the tape drive, tape library, and IBM Spectrum Archive product lines. He joined IBM in 2000 with a Bachelor of Science degree in Electrical Engineering and a Bachelor of Science in Computer Science. Later, he received a Master of Science degree in Engineering Management. He is also an IBM Master Inventor and on his IBM 5th patent plateau. Because of his work with various IBM Spectrum Archive Enterprise Edition (EE) customers across multiple industries, Khanh is often sought out for his expertise to lead, execute, and successfully complete proof of concepts and custom engineering solutions to integrate IBM Spectrum Archive EE into customers' production environments.

Harald Seipp is the founder and technical leader of the Center of Competence for OpenStack Storage as part of the EMEA Storage Competence Center in Frankfurt, Germany. He leads a virtual team and provides guidance to worldwide IBM teams across organizations and works with customers and Business Partners across EMEA to create and implement complex storage cloud architectures. By using his strong storage, software development, and open source technology background, he recently created innovative solutions that combine Object Storage with IBM Bluemix® based IoT and Watson technologies in addition to working with IBM Research and the OpenStack Swift community to create an innovative Object Storage solution that uses high-latency media to provide a cost-effective cloud-based archive. In more than 25 years of experience in IT technology, he has held roles as software developer, software development leader, lead developer and architect of successful software products. He also is a co-inventor of an IBM storage product. He holds seven patents on storage and networking technology.

Slavisa Sarafijanovic is a researcher at IBM Research Zurich, Switzerland. He joined IBM in 2009 and has been working mainly on the design and development of tiered storage solutions. He created the initial prototype and contributed development of IBM Spectrum Archive, a clustered file system that combines disk and tape storage. He then started SwiftHLM open source project for enabling OpenStack Swift-based Object Storage to be used with high-latency media, such as tape and optical, and he has been the main design and implementation contributor since other people joined the project. His other work includes performance modeling and optimization of storage as function of storage cost, sizing and data placement, and network data transfer optimization.

Dominic Müller-Wicke joined IBM in 2006 and works at the IBM R&D Lab in Frankfurt, Germany. He leads the IBM Spectrum Protect client development team in Frankfurt. In addition, he is responsible for the integration of the IBM Spectrum Protect clients with IBM Spectrum Scale and IBM Spectrum Archive. Dominic is known for his expertise in HSM and scalable backup solutions and works close with many IBM Spectrum Protect enterprise customers across multiple industries for proof of concepts and custom solutions.

Simon Lorenz is an IT Architect in IBM Research and Development in Frankfurt, Germany. He joined IBM Germany in 1993 and worked on productivity and manufacturing quality improvements within the IBM Disk Drive Manufacturing Management software. During international assignments, he helped to improve fully automated chip factories in Asia and the US. Simon has held various positions within IBM Research and Development. Since 2009, he worked on Storage Systems Management software and has been responsible for subcomponents, such as system health reporting, cluster configuration management, and recovery. Simon joined the IBM Spectrum Scale development team in 2014. During the last three years, he worked on IBM Spectrum Scale and OpenStack Swift integration.

Takeshi Ishimoto is the storage software developer in the IBM Systems, Tokyo Software & Systems Development Laboratory. Since 2009, he has led the architecture design of IBM Spectrum Archive software, which is known as Linear Tape File System (LTFS). He has been in storage development since 1999, and worked on various IBM storage subsystems, including IBM DS8000®, IBM TS7700, IBM XIV®, and Scale Out Network Attached Storage. He joined IBM in 1987 after graduating with a Bachelor of Science degree in Computer Science at The University of Electro-Communications in Tokyo, and started his career as a RAS software developer. Before storage development, he developed firmware and device drivers for mobile devices and their operating systems, such as ThinkPad, WorkPad, Apple's PowerBook, Windows, and Linux for IBM PowerPC®.

Larry Coyne is a Project Leader at the International Technical Support Organization, Tucson Arizona center. He has 35 years of IBM experience with 23 in IBM storage software management. He holds degrees in Software Engineering from the University of Texas at El Paso and Project Management from George Washington University. His areas of expertise include client relationship management, quality assurance, development management, and support management for IBM Storage Software.

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

Illarion Borisevich
IBM Systems

Related publications

The publications that are listed in this section are considered particularly suitable for a more detailed discussion of the topics that are covered in this paper.

IBM Redbooks

The following IBM Redbooks publications provide more information about the topic in this document. Note that some publications that are referenced in this list might be available in softcopy only:

- *A Deployment Guide for IBM Spectrum Scale Unified File and Object Storage*, REDP-5113
- *Benefits of Spectrum Scale with OpenStack Deployments*, REDP-5358
- *IBM Spectrum Archive Enterprise Edition V1.2.2: Installation and Configuration Guide*, SG24-8333
- *Introduction Guide to the IBM Elastic Storage Server*, REDP-5253

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Other publications

The following publications are also relevant as further information sources:

- *IBM Spectrum Scale Version 4 Release 2.3 Concepts, Planning, and Installation Guide*, GA76-0441
- *IBM Spectrum Scale Version 4 Release 2.3 Administration Guide*, SA23-1455
- *IBM Spectrum Scale Version 4 Release 2.3 Problem Determination Guide,* GA76-0443
- *IBM Spectrum Scale Version 4 Release 2.3 Command and Programming Reference*, SA23-1456
- *IBM Tivoli Storage Manager Space Management for UNIX and Linux: User's Guide,* SC23-9794

Online resources

The following websites are also relevant as further information sources:

- IBM Elastic Storage Sever:

  `http://www.ibm.com/systems/storage/spectrum/ess`
- IBM Elastic Storage Sever Knowledge Center:

  `https://www.ibm.com/support/knowledgecenter/SSYSP8/sts_welcome.html`
- IBM Spectrum Protect for Space Management Knowledge Center:

  `https://www.ibm.com/support/knowledgecenter/SSERBH/landing/welcome_sserbh.html`
- IBM Spectrum Scale:

  `http://www.ibm.com/systems/storage/spectrum/scale`
- IBM Spectrum Scale Knowledge Center:

  `https://ibm.biz/Bdinhb`
- IBM Spectrum Scale Wiki:

  `https://ibm.biz/BdFymB`
- IBM Spectrum Archive Enterprise Edition Knowledge Center:

  `https://www.ibm.com/support/knowledgecenter/en/ST9MBR_1.2.3/ltfs_ee_ichome.html`

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Notices

Trademarks

REDP-5430-00

ISBN 0738456098