PURESTORAGE®

TECHNICAL WHITE PAPER

Splunk Multisite SmartStore With Pure Storage FlashBlade

Disaster Recovery Guide.

Intellectual Property Notice

Contents

Executive Summary

Splunk SmartStore is an indexer capability to store master copy of the indexed data on S3-compliant object stores like Pure Storage FlashBlade or on the Cloud like Amazon S3, Azure Blob data, and Google GCS. SmartStore provides efficient compute and storage elasticity along with simpler and flexible data management capability. Pure Storage FlashBlade accelerates searches for real-time and historical data while reducing overhead costs, increasing availability, and improving operational efficiencies. A few examples of these searches include cyber breach, e-legal discovery, regulatory, and compliance requirements.

Though infrastructure components have become sophisticated and rarely fail, it is still prudent to be prepared for any failure. This is where Splunk Multisite SmartStore deployments prove useful.

An on-premises Splunk Multisite SmartStore deployment consists of an active-active multisite indexer cluster across two sites. Two Pure Storage FlashBlade systems, which are S3 compliant object stores, are used to host the Splunk SmartStore data on these two sites. This deployment provides site failover capability and data redundancy across geographically distributed locations.

This document describes the various possible failure scenarios with the Splunk Multisite SmartStore while using Pure Storage FlashBlade systems. In addition, the document also details the procedures that you can use to handle the failure scenarios. We tested these failure scenarios extensively in our lab to understand the impact, and the remedial actions to be taken.

Audience

The target audience for this document includes Splunk administrators, system and storage administrators, IT managers, system architects, sales engineers, field consultants, and professional service engineers. A working knowledge of Splunk and Linux, and an understanding of server, storage, and networking is assumed, but is not a prerequisite to read this document.

Scope and Objectives

The scope of this document is primarily limited to the failure scenarios of the object storage and does not include the indexer failure scenarios, except for the inter-site network failure. Also the failure scenarios are assumed to have happened when SmartStore was already in place and not during the migration from the Classical Splunk to the SmartStore deployment.

The purpose of the document is to list the various failure scenarios when using FlashBlade systems as the object stores with the Splunk Multisite SmartStore. The failure scenarios were based on the testing performed within our test environment and as such, might not cover all possible scenarios that you may encounter.

Overview

The Splunk Multisite SmartStore deployment in an on-premises setup involves two active read/writable FlashBlade systems (S3 compliant object stores) at the two sites, with data replication between them. Data is ingested into hot buckets and when rolled to warm it is uploaded to the local object store on that site. The data replication between the object stores enables higher data availability and protects the data against a site failure.

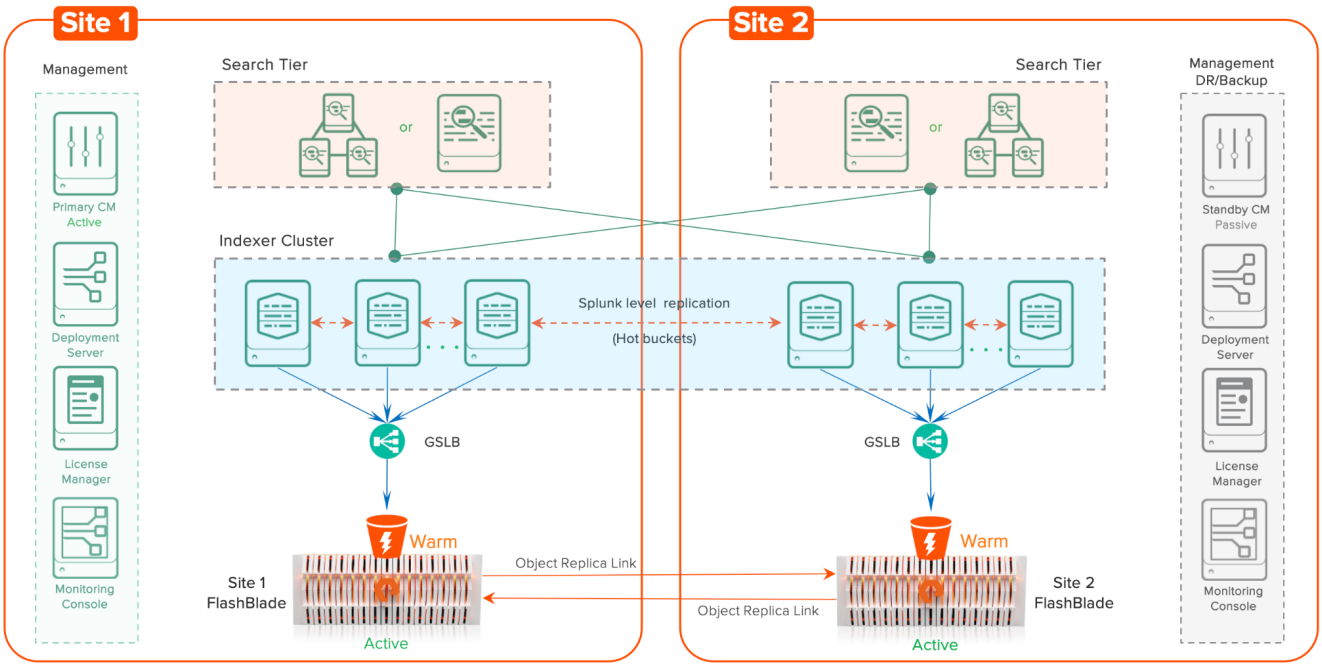
Consider the following requirements for deploying an on-premises Splunk Multisite indexer cluster with SmartStore.

- Each indexer cluster site should reside in an on-premises data center and be limited to two sites.
- One site location should host an active cluster manager, and the other site location should host a standby cluster manager node.
- The site locations should host two FlashBlade systems, in an active-active replicated relationship.
- Each peer node should point to the local remote object store URI through a third-party Virtual IP (VIP) or Global Server Load Balancing (GSLB).
- The VIP or GSLB should route traffic from each peer node to the FlashBlade hosted on the site. If one of the FlashBlade fails, the VIP or GSLB should reroute traffic to the other FlashBlade.
- Though not a hard requirement, search heads or search head clustering should be deployed on each site.
- Site affinity should be disabled by setting `site=site0` on `server.conf` of the search head.
- Network latency between the sites should be a maximum of 300ms. Splunk recommends a maximum of 100ms for better performance.

In addition to the above requirements, Pure Storage FlashBlade systems have the following requirement:

- The FlashBlade systems should be running Purity//FB version 3.3.3+/4.0.2+, and have multisite writable buckets enabled.

The following diagram illustrates the ideal Splunk Multisite SmartStore setup with Pure Storage FlashBlade systems in the Splunk Validated Architecture (SVA) topology using the above requirements.



The setup includes two FlashBlade systems, one at each site with bi-directional replication enabled between them. The setup also includes GSLB to route the traffic from the indexers in a site to the FlashBlade located within the same site. If a FlashBlade fails on that site, the GSLB enables seamless routing from the indexers to the FlashBlade on the other site.

For more information on the multisite SmartStore requirements for an on-prem deployment see Splunk documentation. Also, see the How to setup the FlashBlade systems for Splunk Multisite SmartStore article on the Pure Storage support site.

For more details on the various components like object storage, virtual infrastructure, Splunk servers, GSLB used in our test environment, see Appendix D.

Failure Scenarios

The following table lists the various possible failure scenarios that were tested on Splunk Multisite SmartStore using Pure Storage FlashBlade systems as the object stores.

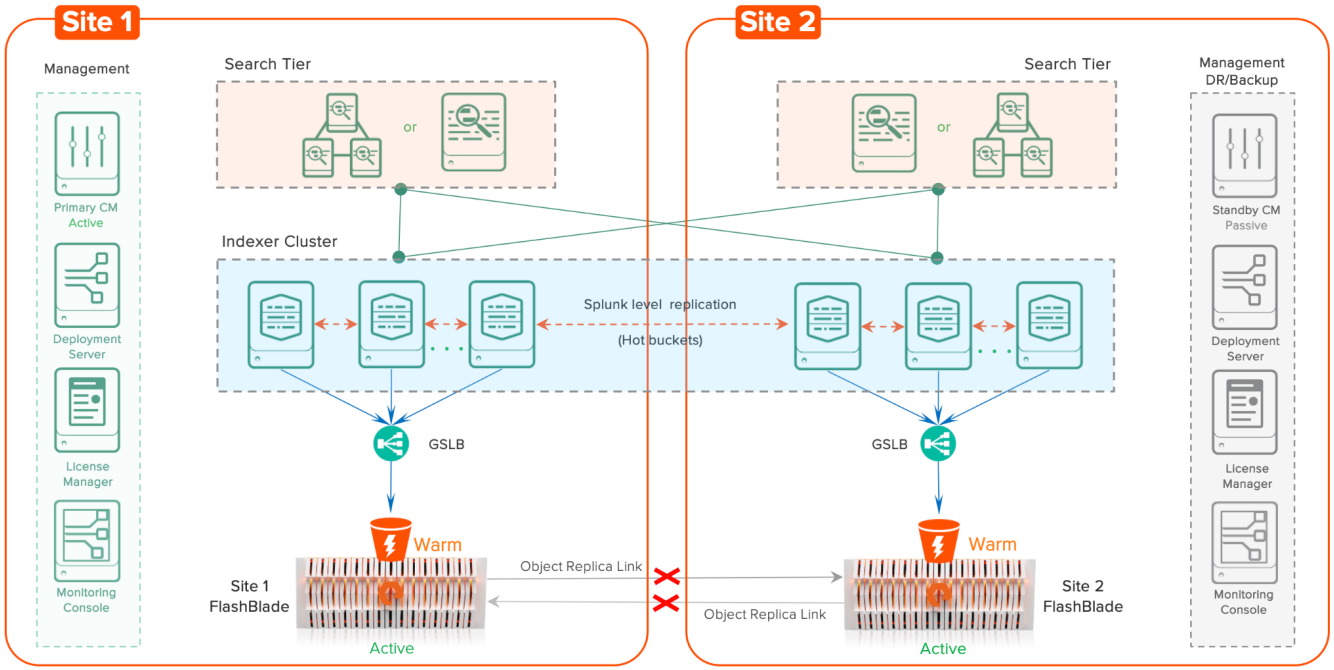
Scenarios	Description
Storage replication link failure	The storage replication link failure at the network layer stops both FlashBlade systems from replicating the changes with each other. However, all Splunk instances on both sites are operational, with peer replication functional between the indexers on both the sites.
Inter-site network connectivity failure	Network outage between the sites that disables any inter-site network connectivity. This impacts the storage replication between the two FlashBlade systems from replicating the changes with each other. While the Splunk instances on both sites are operational, they cannot reach the instances on the other site. Hence there is no peer replication across sites and the cluster manager does not see the indexers on the other site.
Object store failure on one site	FlashBlade failure on any one of the sites renders the indexers on the local site to point to the surviving FlashBlade for uploading warm buckets and to retrieve data for searches that are not in the cache. The Splunk instances on both sites are operational, with the peer replication functional between the indexers on both the sites.
Object store failure on both sites	FlashBlade failure on both the sites impacts the warm bucket uploads during the data ingest and searches that are not in the cache. The Splunk instances on both sites are operational, with peer replication functional between the indexers on both the sites.
Site failure	The Splunk instances on one of the sites including indexers, search heads, and FlashBlade fail.

Storage Replication Link Failure

Though the replication link failure is not a common scenario, this can create a network partition effect at the storage level as both FlashBlade systems cannot replicate the data between each other. The replication link failure can happen if there is a network level failure at the storage level or if the object replica link is paused on both the FlashBlade systems.

Assumptions

- The Splunk instances on both sites are operational.
- Inter-site network connectivity is operational between the Splunk instances including cluster manager, indexers and search heads.
- Peer replication between the indexers on both sites is operational.
- Failure is limited to the storage level replication between the two FlashBlade systems.



Failure impact

- Both the FlashBlade systems are not in sync after the replication link failure happens.
- If the failure lasts longer than the `remote_storage_upload_timeout` period, data might be duplicated when the replication link resumes.

Expected behavior

- Replication link failure does not impact the data ingestion as the indexer can still upload the warm buckets to the local FlashBlade and the peer replication is operational.
- Replication link failure does not impact the data availability for searches as the search peers are all available and any recently ingested data is in the indexer's cache. Any searched data that is not in the cache can still be downloaded from the local FlashBlade.

- When the replication link failure lasts longer than the `remote_storage_upload_timeout` period, a copy of the warm bucket that was uploaded to the FlashBlade by the primary indexer in one site is also uploaded by its peer on the other site to its local FlashBlade. This meets the high availability requirements at the storage level during the failure.

To understand the possibility of duplicate data when the replication link failure lasts longer than the `remote_storage_upload_timeout` period, you should understand how Splunk uploads the data to the remote object store in the SmartStore deployment.

The following sequence of events happen during an ingest on the Splunk SmartStore under normal operation:

NOTE: For ease of understanding, consider that the `site_replication_factor` is 2 with each site holding only one copy (Normally there might be more than one copy in the originating site based on your hot-bucket availability requirements.)

1. A primary indexer replicates the hot bucket with its peer across the site to meet the `site_replication_factor` and `site_search_factor.`
2. When a hot bucket is full the primary indexer rolls that bucket over to a warm bucket. The cachemanager then uploads this warm bucket to the local object store (FlashBlade1) in that site. The object keys include the GUID of the primary indexer that uploads the warm bucket.

   See Appendix C for the object key nomenclature.
3. The primary indexer also informs the peer on the other site holding a replica copy that it has uploaded the warm bucket to the object store.
4. As the FlashBlade systems are set up in a bi-directional replication, this data is replicated to the second site asynchronously.
5. After the `remote_storage_upload_timeout` period (which is generally set to 10 minutes to accommodate the replication lag between the sites), the peer in the remote site issues a GET request from its local object store (FlashBlade2) for the manifest file of the warm bucket.
6. If the GET request is successful, it confirms that the uploaded warm bucket is available on the object storage and the peer moves on.
7. If the GET request is not successful, the peer assumes that there was a problem with the primary indexer in uploading the warm bucket. The peer then uploads the content of the same warm bucket to the remote object store. When the peer uploads the data, the object keys include GUID of the peer which would be different from the object keys uploaded by the primary indexer. Even though the object keys might be physically different, the contents are logically the same.

Now consider the sequence of events during an ingest when the replication link failure happens:

1. A primary indexer replicates the hot bucket with its peer across the site to meet the `site_replication_factor` and `site_search_factor.`
2. When a hot bucket is full, the primary indexer rolls that bucket over to a warm bucket. The cachemanager then uploads the warm bucket to the local remote object store (FlashBlade1) in that site. The object keys include the GUID of the primary indexer that uploads the warm bucket.
3. The primary indexer also informs the peer on the other site holding a replica copy that it has uploaded the warm bucket to the remote object store.
4. Due to replication link failure, FlashBlade systems do not replicate the data asynchronously. The local FlashBlade1 holds the uploaded data.
5. After the `remote_storage_upload_timeout` period, the peer on the remote site issues a GET request from the local object store (FlashBlade2) for the manifest file of the warm bucket.
6. As the data is not synchronized, the GET request is unsuccessful. The peer on the remote site uploads the content of the same warm bucket onto the object store (FlashBlade2).

7. At this point, the FlashBlade systems in each site hold a copy of the same warm bucket. Even though the object keys for the same warm bucket are physically different, logically they hold the same data. Any searches (which require the download of the warm bucket back to the cache) to either site work as expected.

Recovery process

When the storage replication link resumes, both the FlashBlade systems start replicating the pending changes. If the failure is longer than the `remote_storage_upload_timeout` period, duplicate copies for the same warm bucket might probably exist on both FlashBlade systems. This is because the object keys are physically different on each FlashBlade.

The time required to resync the objects between the two FlashBlade systems depends on factors such as the duration of the failure and the replication throughput.

While the data availability is not impacted by the replication link failure, the storage usage might increase due to the duplicate data generated during the storage replication link failure.

Suggested fix

To eliminate the duplicate data generated after the replication link resumes and the pending replication completes, run the Splunk command **dedup** from one of the indexers.

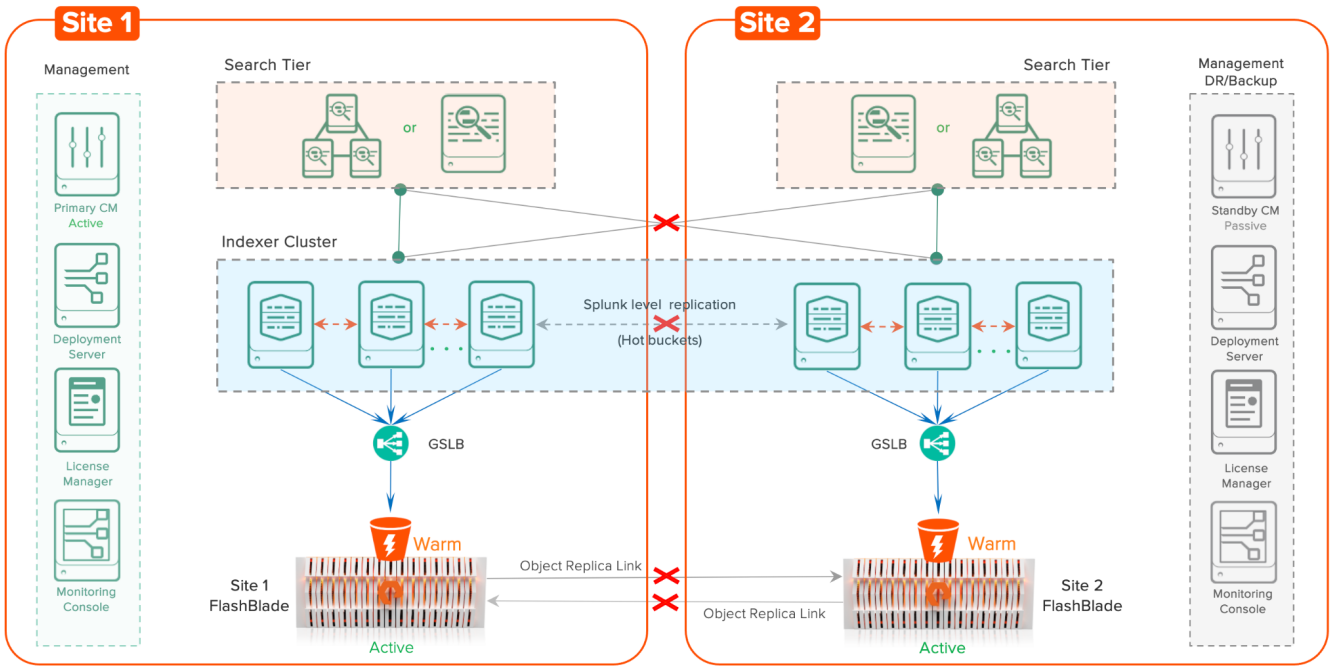
For more information on how to remove the duplicate data see Appendix A.

Inter-site Network Connectivity Failure

If the inter-site network connectivity between the two sites is lost, a network partition occurs. When this happens, Splunk instances from one site cannot access the instances on the other site. The object replication between the FlashBlade systems also fails.

Assumptions

- The Splunk instances on both sites are operational.
- Network connectivity is limited to a local site, meaning Splunk instances within the site can access each other but not across sites.
- As cluster manager can only reach the indexers and search heads on the local site, it marks the indexers and search heads on the remote site as unavailable.
- Peer replication between the indexers across sites is not-operational. It is operational only within the site.
- Both FlashBlade systems cannot replicate the changes between each other.



Failure impact

- Both the FlashBlade systems are not in sync when the inter-site network connectivity is lost. Hence the high availability of data at the storage level is not met until the inter-site network connectivity is restored.
- While the data ingestion from the local site is not impacted, the peer replication to the other site is not functional. Also, the site level replication factor/search factor is not met as the cluster manager doesn't have access to the remote peers.
- Cluster manager marks the indexers and search heads on the other site as "down" as it cannot access them.
- After the inter-site network connectivity failure, the data that are ingested in one of the sites are not available for searches from the other site.

Expected behavior

- Data ingestion behavior from the local site continues to work as the hot bucket is rolled over to warm when full. The warm bucket is then uploaded to the local FlashBlade.
- Any searches from any site for the data that was synchronized between the FlashBlade systems before the failure should work as expected.

Recovery process

When the inter-site network connectivity is restored, both the FlashBlade systems start replicating the pending changes. The time required to resync the objects between the two FlashBlade systems depends on factors such as the duration of the failure and the replication throughput.

During the inter-site network connectivity failure, any searches for the data that was ingested on the other site might still fail or not show the expected results, until the underlying splunk buckets are synchronized between the FlashBlade systems. Since the cluster manager now has access to the remote peers from the other site, it initiates the bucket-fixing process at the cluster level to meet `site_replication_factor` and `site_search_factor`.

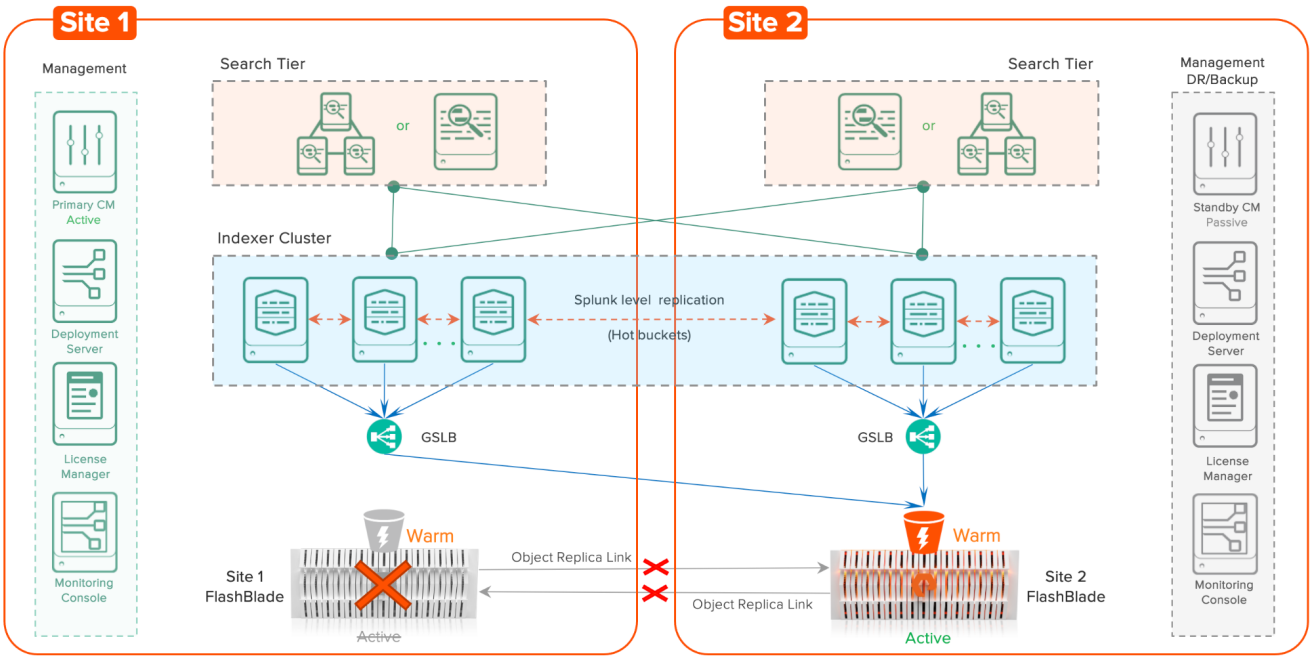
NOTE: Interestingly, inter-site network connectivity failure does not result in any duplicate bucket data being generated when the connection is restored. So you do not need to run the Splunk level **dedup** command. But we recommend that you periodically run the dedup command with --dry-run option to check for duplicates.

Object Store Failure on One Site

If the object store (FlashBlade) on one site fails, the GSLB or VIP should redirect the traffic automatically from the indexers on that site to the FlashBlade on the other site. This might result in increased WAN traffic across the data centers. Also the FlashBlade failure impacts the high availability of data for the Splunk indexer cluster until the failed FlashBlade comes back up and the objects are synchronized. Until then the Splunk Multisite SmartStore operates on a single FlashBlade.

Assumptions

- The Splunk instances on both sites are operational.
- Inter-site network connectivity is operational.
- Peer replication between the indexers across sites is operational.
- No storage level replication as a FlashBlade is down on one site.



Failure impact

- As one of the FlashBlade systems is down, the object replication is disabled. This impacts the high availability of data at the storage level.
- Due to the presence of GSLB, when a FlashBlade goes down for any reason, the indexers on that site will be routed automatically to the other FlashBlade on the remote site which results in increased WAN traffic across the data centers.
- During the FlashBlade failure, any in-flight warm bucket uploads to that FlashBlade fails. Splunk then retries to upload them to the other FlashBlade as the GSLB would have redirected the traffic to the FlashBlade on the other site.
- Due to replication lag, the surviving FlashBlade might be missing data recently uploaded to the failed FlashBlade. This could possibly result in incomplete or missing search results if the search couldn't be fulfilled by the cached data. This is a temporary issue and is resolved when the failed FlashBlade returns to service and the data is synchronized. When the failure lasts longer than the `remote_storage_upload_timeout`, the remote peer uploads the same warm bucket to the surviving FlashBlade.

- If the failure lasts longer than the `remote_storage_upload_timeout` period, there is a possibility of duplicate data when the failed FlashBlade comes back online. This is limited to the warm buckets that were uploaded to the FlashBlade before it failed, but were not replicated to the other FlashBlade.

Expected behavior

- Data ingestion continues to be operational from both sites. The indexers on the site that has the failed FlashBlade take longer to upload the warm buckets as they have to go across the WAN to upload to the surviving FlashBlade.
- Searches on both sites continue to work on all data except for those which were ingested recently, uploaded to the failed FlashBlade but not replicated to the other FlashBlade, and possibly evicted from the indexers on the same site as the failed FlashBlade. Even that data would be available for search, when the failure lasts longer than the `remote_storage_upload_timeout` period.

Recovery process

When the failed FlashBlade comes back online, both the FlashBlade systems start replicating the pending changes. If the FlashBlade failure lasted more than the `remote_storage_upload_timeout` period, the replication can result in duplicate data for the warm buckets that were uploaded to the failed FlashBlade but were not replicated to the surviving FlashBlade.

The time required to resync the objects between the two FlashBlade systems depends on factors such as the duration of the failure and the replication throughput. The higher the inter-site latency, the longer it takes for the replication.

NOTE: If a FlashBlade fails permanently, the data that was not replicated might be permanently lost. The cluster displays incomplete search results warnings and bucket fixup errors, in response to attempts to access the lost buckets. Contact Splunk Customer Support for assistance in removing the corresponding bucket metadata.

Suggested fix

To eliminate the duplicate data generated after the failed FlashBlade is online and replicates the pending changes with the other FlashBlade, run the **dedup** command from one of the indexers.

For information on how to remove the duplicate data, see Appendix A.

Recovery process after a catastrophic failure

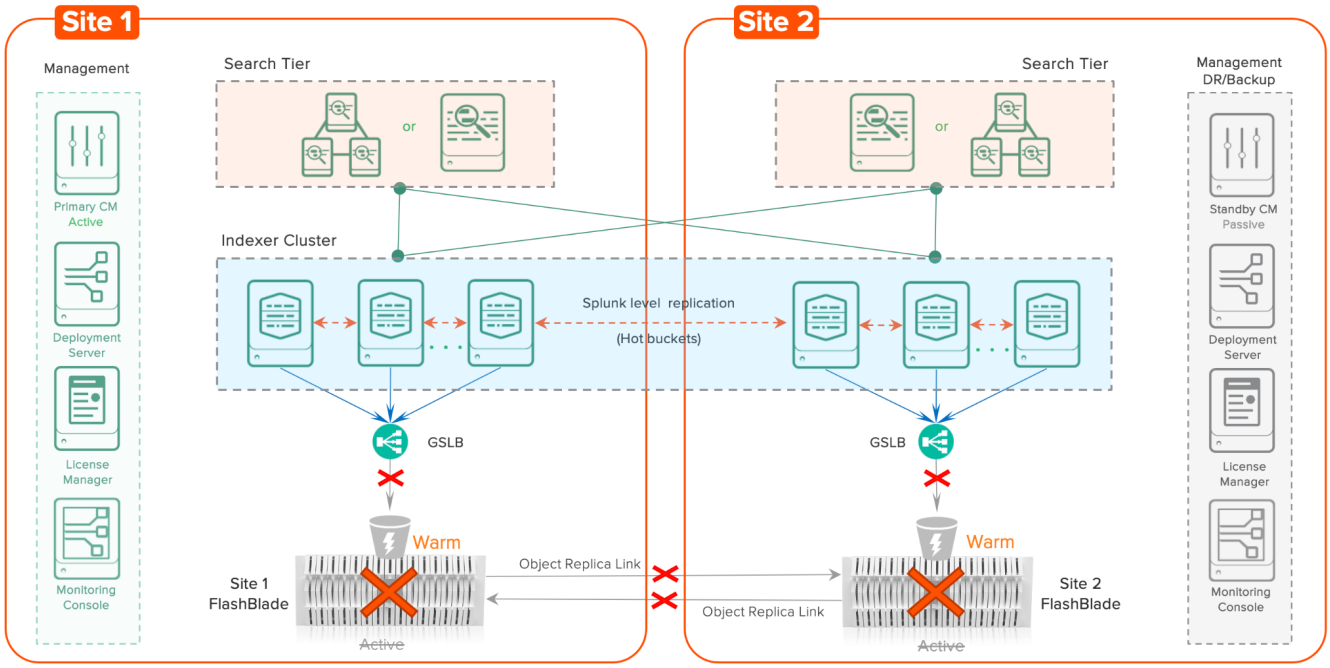
In the rare event that a FlashBlade fails permanently, work with your account team to get a replacement FlashBlade. Connect the new FlashBlade with the surviving FlashBlade, and then set up the object replication between them. Establish the baseline to synchronize all the data from the surviving FlashBlade to the replacement FlashBlade. Work with the Pure Storage Support team as enabling the baseline on a new FlashBlade is a support driven activity.

Object Store Failure on Both Sites

If the object store (FlashBlade) on both the sites fail, the GSLB or VIP cannot redirect the traffic from the indexers. Any uploads to the FlashBlade systems fail and Splunk constantly tries to upload the rolled warm buckets. The searches on either site work only when the data is served from the cache. Any searches that require the data to be downloaded from the object store result in an error.

Assumptions

- The Splunk instances on both sites are operational.
- Inter-site network connectivity is operational.
- Peer replication between the indexers across sites is operational.
- No storage level replication or high availability, as both FlashBlade systems are down.



Failure impact

- Both the FlashBlade systems are down, which impacts the availability of data at the storage level.
- Indexers from either site cannot upload the rolled warm buckets to the FlashBlade systems and Splunk continuously attempts to upload them.
- During the FlashBlade failures, any in-flight warm bucket uploads to the FlashBlade systems fail and Splunk continuously attempts to upload them.
- As both FlashBlade systems are not available, any searches on the sites for the data that is not in the cache result in an error. The searches have to be rerun when one or both FlashBlade systems are back online.
- If the failure lasts longer than the `remote_storage_upload_timeout` period, the primary indexer on one site and the peer on the other site try to upload their copy of the same warm buckets when the FlashBlade systems are back online. This can cause duplicate data when the pending replication finishes.

Expected behavior

- Data ingestion continues to be operational from both sites until the local tier is out of space to hold the warm buckets. These buckets are still not uploaded to the FlashBlade systems because of the storage failure.
- Searches on both sites continue to work on the data that are available on the cache. Any historical searches that require a download from the FlashBlade systems fail.

Recovery process

When the FlashBlade systems comes back online, Splunk Indexers from both the sites start uploading the pending warm buckets to their local FlashBlade systems. The FlashBlade systems start replicating the pending changes before the failure. The time required to upload the pending warm buckets from either site to their respective FlashBlade systems depends on the factors such as the duration of the failure and the upload throughput.

Any searches for data that was not in the cache, need to be rerun as those searches would have failed.

Suggested fix

To eliminate the duplicate data generated after the failed FlashBlade systems are online and replicate the pending changes, run the **dedup** command from one of the indexers.

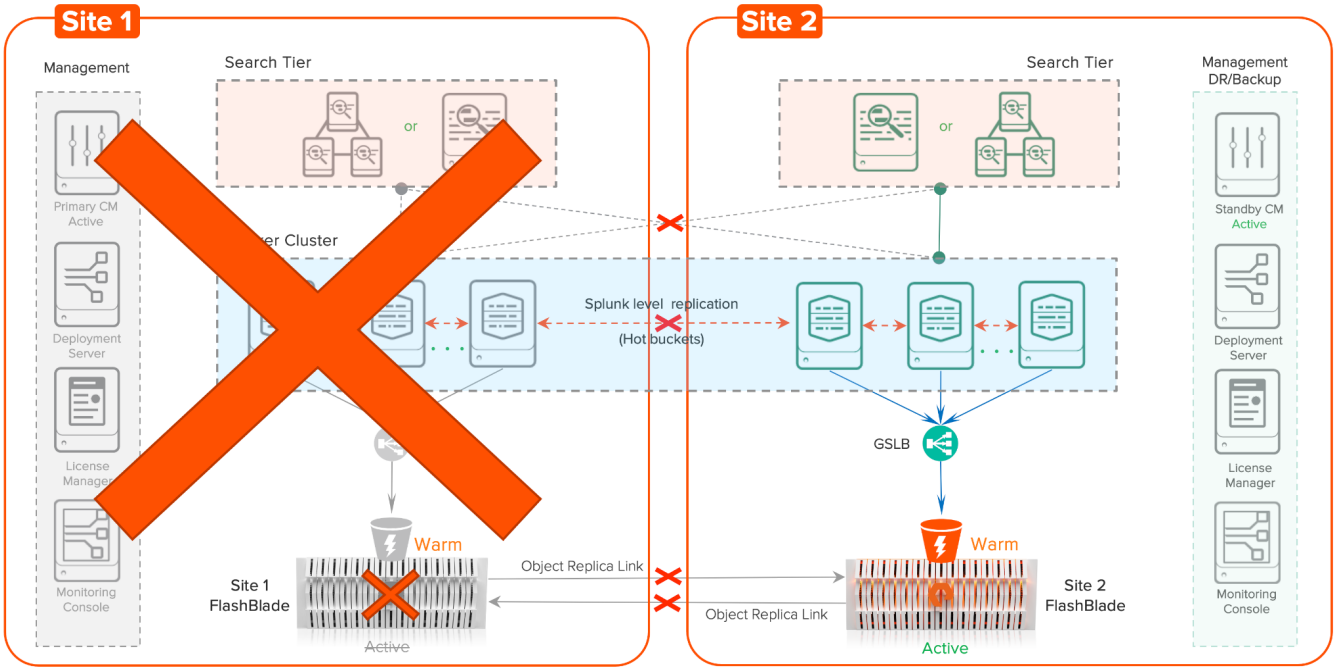
For more information on how to remove the duplicate data, see Appendix A.

SmartStore Site Failure

A site failure reduces the Splunk Multisite SmartStore to operate from the surviving site. A site failure results in loss of high availability of data until the failed site is back online and synchronized.

Assumptions

- The Splunk instances on one site are operational.
- No Inter-site network connectivity as one of the sites is not accessible.
- Peer replication is limited to the local site.
- No storage level replication, as one FlashBlade is down.



Failure impact

- The Splunk Multisite SmartStore is reduced to a single site with the cluster not meeting `site_replication_factor` and `site_search_factor`.
- If the active cluster manager is on the failed site, it is down as well. Manually switch to the standby cluster manager on the surviving site immediately. See Appendix B for the site failover process.
- The indexer peer replication is limited to the local surviving site.
- Due to replication lag, the FlashBlade on the surviving site might be missing data recently uploaded to the failed FlashBlade. If the search is not fulfilled by the cached data, the search results might be incomplete. This issue is resolved when the failed site returns to service and the data is synchronized between the FlashBlade systems.
- In-flight searches that access peers on the failed site return incomplete results. New searches from the surviving site only be redirected to the peers on that site.
- Data uploaded to the surviving FlashBlade is not replicated to the FlashBlade on the failed site until the site comes back online with all the components.

- If the failure lasts longer than the `remote_storage_upload_timeout` period, there is a possibility of duplicate data when the FlashBlade on the failed site comes back online. This is limited to the warm buckets that were uploaded to the failed FlashBlade before the site failure but were not replicated to the other FlashBlade.

Expected behavior

- Forwarders configured for load balancing across the cluster automatically redirect to the indexers on the surviving site. If the forwarders are not configured for load balancing, update them to point to the surviving indexers.
- Data ingestion continues to be operational from the surviving site and the rolled warm buckets are uploaded to the FlashBlade on that site.
- Any warm buckets that were recently uploaded to the FlashBlade on the failed site prior to the failure might not have replicated to the surviving FlashBlade. If so, the peer indexer on the surviving site uploads its copy to the FlashBlade after the `remote_storage_upload_timeout` period.
- Searches on the surviving site continue to work on all the data ingested from either site. This is because the indexers on the surviving site hold a copy of the hot and the warm buckets that were recently uploaded to the FlashBlade in the failed site.

Recovery process

When the failed site comes back online and returns to the cluster, perform the following activities manually:

1. Make sure that the indexers and search heads on the recovered site point to the new active cluster manager on the other site (which was the standby manager before the site failure).
2. As the cluster manager of the recovered site now serves as the standby manager, ensure it is included in the future configuration updates in preparation for the site failure on the other site.

When the FlashBlade comes back online in the recovered site, the pending changes from either site are replicated. The FlashBlade in the recovered site replicates the pending changes up until the time the site failed. The FlashBlade on the surviving site replicates the pending changes starting from the time of the failure until the current time. The time required to resync the FlashBlade systems depends on factors such as the duration of the failure and the replication throughput.

Suggested fix

To eliminate the duplicate data generated after the failed site is back online and the FlashBlade systems replicate the pending changes, run the **dedup** command from one of the indexers.

For more information on how to remove the duplicate data, see Appendix A.

Best Practices to Configure Splunk Multisite SmartStore with FlashBlade

We recommend the following best practices for configuring Splunk Multisite SmartStore with FlashBlade systems.

1. Ensure the Purity//FB version is at least **3.3.3** on both the FlashBlade systems as that is the minimum Purity version that supports Splunk Multisite SmartStore.

2. Ensure a Global Server Load Balancing (GSLB) or a third-party Virtual IP (VIP) is used to route the traffic from the indexers on both sites to the FlashBlade in their respective site. In case of a FlashBlade failure, the GSLB should have the failover mechanism to automatically route the traffic to the FlashBlade on the other site.

   The **`remote.s3.endpoint`** in the remote volume definition of **`indexes.conf`** should be configured to point to the URI that is supported by the GSLB and not hard coded with the IP address. Using a hard coded IP address impacts the automatic failover if a FlashBlade fails.

3. Set the **`remote_storage_upload_timeout`** setting in the **`server.conf`** of all the indexers to a time higher than the maximum replication lag between the two FlashBlade systems. We recommend 600 seconds (10 minutes), but it should be updated to a higher value if you experience higher replication lag.

   If this setting is left at the default 60 seconds (1 minute), you might notice duplicate warm bucket copies across both FlashBlade systems.

   See Appendix A for information on removing the duplicate data.

4. Ensure the **`remote.s3.supports_versioning`** option in the remote volume definition of the **`indexes.conf`** is set to False. If it is set to True (which is the default), the delete command issued by the Splunk to delete the S3 objects on the FlashBlade is not propagated to the other FlashBlade. This might result in inconsistent object counts between the FlashBlade systems.

5. Ensure that the lifecycle policy for the S3 bucket that hosts the Splunk data on both the FlashBlade systems is enabled to physically remove the deleted data and to reclaim the space. Not enabling the lifecycle policy results in holding the deleted data, which in turn impacts the space availability.

   To enable the lifecycle policy, set the “Keep Previous Version For” to a number that represents the number of days after which the deleted objects are physically removed. The lowest granularity is 1 day.

See the Best Practices for Splunk on Pure Storage article on Pure Storage support website for additional information.

Conclusion

This document described the possible failure scenarios in Splunk Multisite SmartStore setup, when using Pure Storage FlashBlade systems as the object stores. As seen from the tested scenarios, the key functions of ingest and search remain operational in most cases with some exceptions.

The following table summarizes the failure scenarios with the expected behavior.

Scenarios	Data Ingest		Searches		Data Availability	Behavior
	Site1	Site2	Site1	Site2		
Storage replication link failure	Y	Y	Y	Y	Both FlashBlade systems	Data on FlashBlade systems are not in sync for `remote_storage_upload_timeout` period. After link resumption, duplicate bucket data might exist.
Inter-site network connectivity failure	Y	Y	Y	Y	Both FlashBlade systems	Data on FlashBlade systems is not in sync during the failure. This results in each site holding local data until the network resumes. Searches work for data available before the failure. Searches for data after the failure is limited to the local data.
Object store failure on one site	Y	Y	Y	Y	One FlashBlade	All functions are operational except that the data is available only on one FlashBlade until the other one recovers and resynchronizes.
Object store failure on both sites	Y	Y	Y	Y	None	Ingest continues until the local tier runs out of space. Ingested data is not available in the FlashBlade systems during the failure. Searches for the data in cache works, but historical searches fail.
Site failure	N	Y	N	Y	One FlashBlade	`site_replication_factor` and `site_search_factor` are not met. Data is available only on one FlashBlade. For high availability, the failed site should be online again and the FlashBlade on that site should be synchronized with the other FlashBlade.

Appendices

Appendix A: Removing Duplicate Data from the Object Store

To eliminate the duplicate bucket copies uploaded to the object store by multiple indexer nodes in failure scenarios, Splunk offers a **dedup** command that can be run from one of the indexers.

We recommend that you run the dedup command with the `--dry-run` option to see whether any duplicate buckets exist before removing them. You can run this command either at the volume level or at the individual index level.

```
# splunk cmd splunkd rfs -- --dry-run dedup --min-age 1 volume:<remote_store>

# splunk cmd splunkd rfs -- --dry-run dedup --min-age <seconds> index:<index_name>
```

To remove the duplicate data, run the above command without the `--dry-run` option. The `min-age` parameter helps to specify the age (in seconds) of the duplicates that you want to check.

Example

```
[root@west-ix01 ~]# splunk cmd splunkd rfs -- --dry-run dedup --min-age 60 index:apache

========= Processing index = index:apache =========
***Entering the duplicates detection phase ...
#for full paths run: splunkd rfs -- ls --starts-with volume:remote_store/apache/

***Entering the dedup phase in dry-run mode...
Duplicates found: bucket=apache/db/1e/c9/60~218E1E55-39FD-49FB-8C09-821F55AEC9BB has one receipt
 and 2 uploaded bucket copies
    Deduping bucket=apache/db/1e/c9/60~218E1E55-39FD-49FB-8C09-821F55AEC9BB with one receipt and 2
 uploaded bucket copies
Duplicates found: bucket=apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978 has one receipt
 and 2 uploaded bucket copies
    Deduping bucket=apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978 with one receipt and 2
 uploaded bucket copies

***Total number of buckets:                     35
***Total number of buckets with duplicates:     2
***Total number of buckets deduped successfully: 0
***Percentage of buckets with duplicates:       5.714286 %
```

We recommend that you run the **dedup** command with the `--dry-run` option periodically (monthly or quarterly) to see whether there are any duplicate buckets and take necessary actions.

NOTE: Running the dedup command removes the duplicate bucket data from the object store. We recommend that you review the data before using this command, especially in your production environment. If you have questions, check with the Splunk support team before running the command.

Appendix B: Site Failover Process

When there is a site failure, perform the following steps on the second site to keep the Splunk Multisite SmartStore environment operational:

NOTE: Ideally, you must synchronize the configuration bundles of primary cluster manager and stand-by manager when you set up the multisite SmartStore environment. Make sure to follow the process defined by Splunk to keep the configuration bundles synchronized.

1. Bring up the stand-by manager on the second site which is operational. Follow the directions provided by Splunk.

2. If the replacement manager uses the same IP address and management port as the failed primary manager, employ DNS-based failover, a load balancer or other similar options. If the stand-by manager does not use the same IP address or manager port as the failed primary manager, then update the `manager_uri` setting on the indexers and search heads on the second site to point to the new manager.

```
## Update indexer nodes to point to the standby manager
[root@west-ix01]# splunk edit cluster-config -mode slave -manager_uri
https://<standby-mgr>:8089 -secret <ClusterSecret> -auth admin:<password>

## Update search heads to point to the standby manager
[root@west-sh01]# splunk edit cluster-config -mode searchhead -manager_uri
https://<standby-mgr>:8089 -secret <ClusterSecret> -auth admin:<password>
```

3. Restart indexing in the multisite cluster by running the following command on the stand-by manager to unblock indexing when replication factor number of peers are not available.

   See Splunk documentation for additional details.

```
# splunk set indexing-ready -auth admin:<password>
```

4. Forwarders configured for load balancing across the cluster automatically redirect to peers in the remaining site. If not, update the forwarders to point to the indexers in the remaining site.

Appendix C: Object Key Nomenclature

Here is an uploaded Splunk warm bucket on the object storage. We used the rfs command to obtain this information.

```
# splunk cmd splunkd rfs -- ls --starts-with bucket:apache~61~4B72571E-D509-4020-8EFA-ED88B7216978
size,name
10,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-4B72571E-D509-4020-8EFA-ED88B7216978/.rawSize
9,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-4B72571E-D509-4020-8EFA-ED88B7216978/.sizeManifest4.1
208119989,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-4B72571E-D509-4020-8EFA-ED88B7216978/1672919382-1672906
215-3406021761429336798.tsidx
231,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-4B72571E-D509-4020-8EFA-ED88B7216978/Hosts.data
122,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-4B72571E-D509-4020-8EFA-ED88B7216978/SourceTypes.data
321,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-4B72571E-D509-4020-8EFA-ED88B7216978/Sources.data
285,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-4B72571E-D509-4020-8EFA-ED88B7216978/Strings.data
134632,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-4B72571E-D509-4020-8EFA-ED88B7216978/bloomfilter
75,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-4B72571E-D509-4020-8EFA-ED88B7216978/bucket_info.csv
183331862,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-4B72571E-D509-4020-8EFA-ED88B7216978/rawdata/journal.gz
26128,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-4B72571E-D509-4020-8EFA-ED88B7216978/rawdata/slicemin.dat
256695,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-4B72571E-D509-4020-8EFA-ED88B7216978/rawdata/slicesv2.dat
98,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-4B72571E-D509-4020-8EFA-ED88B7216978/splunk-autogen-params.dat
1752,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/receipt.json
```

Here is the structure of the S3 object, which corresponds to each file from the warm bucket.

```
{index-name}/db/{2 letter hash}/{2 letter hash}/{bucket_id_number~origin_guid}/{“guidSplunk”
-uploader_guid}/{bucket content}
```

The `bucket_id_number` from the above is 61 and it corresponds to the warm bucket from the linux-idx index. The origin indexer is 4B72571E-D509-4020-8EFA-ED88B7216978. In this case, it was the same indexer that uploaded the bucket contents to the S3 object store as the `uploader_guid` is also 4B72571E-D509-4020-8EFA-ED88B7216978.

Here is the same bucket on the object storage when uploaded by a target peer.

NOTE: For clarity and brevity, we filtered the result to show the objects that were uploaded by the target peer.

```
10,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-688E14B4-5AE3-482C-9244-88BD8656DD3F/.rawSize
9,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-688E14B4-5AE3-482C-9244-88BD8656DD3F/.sizeManifest4.1
215996746,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-688E14B4-5AE3-482C-9244-88BD8656DD3F/1672919382-1672906
215-5977550622213747351.tsidx
231,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-688E14B4-5AE3-482C-9244-88BD8656DD3F/Hosts.data
122,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-688E14B4-5AE3-482C-9244-88BD8656DD3F/SourceTypes.data
321,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-688E14B4-5AE3-482C-9244-88BD8656DD3F/Sources.data
285,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-688E14B4-5AE3-482C-9244-88BD8656DD3F/Strings.data
134885,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-688E14B4-5AE3-482C-9244-88BD8656DD3F/bloomfilter
183331862,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-688E14B4-5AE3-482C-9244-88BD8656DD3F/rawdata/journal.gz
26128,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-688E14B4-5AE3-482C-9244-88BD8656DD3F/rawdata/slicemin.dat
256695,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-688E14B4-5AE3-482C-9244-88BD8656DD3F/rawdata/slicesv2.dat
98,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/guidSplunk-688E14B4-5AE3-482C-9244-88BD8656DD3F/splunk-autogen-params.dat
1671,apache/db/fb/17/61~4B72571E-D509-4020-8EFA-ED88B7216978/receipt.json
```

While the `bucket_id_number~origin_guid` remains the same, the `uploader_guid` is different. This means that it was uploaded by a target peer and not the source indexer.

The `receipt.json` is the manifest file that contains the details of the warm bucket such as the size, name, uploader information and the checksum. It does not have the `{“guidSplunk”-uploader_guid}` in the object key. Hence there is always only one receipt.json for a warm bucket. However, it can be overwritten if the same bucket is uploaded by a target peer due to the failure scenarios discussed earlier in this document.

Appendix D: Test Environment Details

FlashBlade configuration on each site

Component	Description
FlashBlade	15 × 17 TB blades
Capacity	240 TB raw 162.46 TB usable (with no data reduction)
Connectivity	4 × 40 Gbps Ethernet (data) 2 × 1 Gbps redundant Ethernet (Management port)
Physical	4U
Software	Purity//FB 3.3.3

VMware ESX host details

Component	Description
ESXi Host	Dell R6515 with AMD EPYC 7713P 64 cores, 256 GB RAM - 6 Hosts per site
ESXi Version	VMware ESXi 6.7.0

Splunk multisite indexer cluster details

Component	Quantity per Site	Cores	Memory	Splunk Version	OS Version	Description
Cluster Manager	1	4	32	8.2.2	CentOS 7.5	Primary on Site1, Standby on Site2
Indexer	4	12	64	8.2.2	CentOS 7.5	Cache - 500 GB (Local NVMe storage)
Search Head	1	8	64	8.2.2	CentOS 7.5	Single search head per site
Forwarders	4	8	32	8.2.2	CentOS 7.5	Failover configured to the other site

GSLB details

Component	Description
Platform	BIG-IP, Virtual Edition
Software Version	BIG-IP v15.0.1
Load Balancing Method	Topology - DNS name resolution using proximity-based load balancing

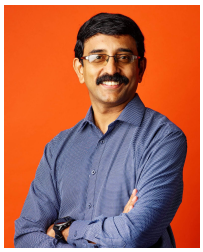
Additional Resources

Supporting Information

- Multisite Indexer Clusters with SmartStore
  https://docs.splunk.com/Documentation/Splunk/9.0.3/Indexer/MultisiteSmartStore
- Best Practices for Splunk on Pure Storage
  https://support.purestorage.com/Solutions/Splunk/Splunk_Reference/Best_Practices_for_Splunk_on_Pure_Storage_products
- How to Set Up Splunk Multisite SmartStore with FlashBlade
  https://support.purestorage.com/Solutions/Splunk/Splunk_Reference/How_to_setup_Splunk_Multisite_SmartStore_with_FlashBlade
- Load Balancing your Applications
  https://www.f5.com/solutions/load-balancing-your-applications
- BIG-IP Global Server Load Balancing
  https://techdocs.f5.com/kb/en-us/products/big-ip_gtm/manuals/product/gtm-concepts-11-5-0/1.html

About the Author

Somu Rajarathinam is a Technical Director in the Solutions engineering team at Pure Storage. His responsibilities include defining application solutions for Pure Storage products, performing benchmarks, and developing reference architectures for applications running on Pure Storage products. Somu has over 25 years of experience in the database and analytics area. He was a member of the Systems Performance and Oracle Applications Performance Groups at Oracle Corporation. He also worked with Logitech, Inspirage, and Autodesk, where his assignments include delivering database and performance solutions, managing infrastructure, and providing support to database and analytical applications both on-prem and on the Cloud.



Contributor

Christopher Roberts is a Senior Architect in the Professional Services engineering team at Pure Storage. He mainly focuses on data analytics and observability. He is responsible for ensuring that all PS engagements are delivered in a consistent manner, no matter the geo-region. Christopher has over 8 years of experience deploying, using, and supporting data analytics and observability platforms both on-prem and on the Cloud. Christopher is a Splunk Core Certified Consultant, Splunk Certified Advanced Power User, and holds various other Splunk accreditations.