Technical Report and Best Practices Guide

NetApp SolidFire and Datos IO RecoverX with Cassandra

Stephen Carl, NetApp
July 2017 | TR-4612

Abstract

This technical report offers guidelines with detailed information to help make sure of a stable, high-performance NetApp® SolidFire® SAN using Datos IO RecoverX software and Cassandra NoSQL database. SolidFire storage offers a compelling advantage for a wide range of database application use cases. This paper gives the Cassandra database administrator information about SolidFire storage and the Datos IO RecoverX application. It describes important system design paradigms to consider when using SolidFire storage for database applications such as Cassandra. From these design points, the reader learns about application profiles that are best suited to SolidFire storage and how to identify those types of applications. Best practices for configuring SolidFire and Datos IO RecoverX with Cassandra environments are explained, and the document concludes with descriptions of the test environment and use cases.

NetApp®

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

Architecting your next-generation data center (NGDC) requires different tools than those that most businesses use today in their current data centers. NGDCs require more flexibility to meet customer and business needs, have different uptime requirements, and have data protection expectations that require tools to match the changing environment.

SolidFire provides the flexibility for an NGDC to meet unplanned demands. Cassandra and NoSQL databases with the scale, availability, and elasticity demands of applications running in Datos IO provide an application-aware data management solution to protect data without affecting application performance.

NetApp SolidFire

NetApp SolidFire delivers an unstoppable force for data center transformation. Born out of some of the largest cloud infrastructures in the world, SolidFire is built to serve next-generation data center needs such as scaling with multitenancy, set-and-forget management, and guaranteed performance. The SolidFire quality of service (QoS) feature means that all tenants have guaranteed configured performance, which can be dynamically changed as needed. Enterprises are adopting SolidFire because they demand greater predictability from their shared storage infrastructure. Database administrators benefit from the ability to deploy applications more quickly, guarantee application performance at the volume level, and reduce operational and capital expenses.

Cassandra

NoSQL databases, including wide column store types such as Apache Cassandra, are providing companies with capabilities for faster data insights in real time, enabling quick response to customer data-related needs and requirements. The scale-out architecture of Apache Cassandra allows customers to easily scale their data size depending on application growth. Organizations are choosing the power and flexibility offered by Cassandra's key-value datastores, and the adoption rate is increasing steadily. Whether you choose dedicated hardware or shared infrastructure for your database deployment, platforms backed by SolidFire offer high-performing block storage for use in the persistent data layer of any application.

Datos IO

Datos IO is the application-centric data management company for the multicloud world. The flagship Datos IO RecoverX delivers a radically novel approach to data management, helping organizations embrace the cloud with confidence by delivering solutions that protect, mobilize, and monetize their data at scale. Datos IO was recently awarded Product of the Year by *Storage Magazine* and was recognized by Gartner in the 2016 Hype Cycle for Storage Technologies.

The combination of NetApp SolidFire and Datos IO RecoverX enables enterprises to deploy and scale their Cassandra databases with a confidence in the recoverability of their data. Datos IO RecoverX offers fully orchestrated, any point-in-time recovery of Cassandra tables directly back into the same Cassandra database or to a different Cassandra database instance with a different topology. The combination of NetApp SolidFire storage and Datos IO RecoverX offers increased storage efficiency and solutions for traditional and new applications.

2 NetApp SolidFire

2.1 Storage Efficiencies

Thin Provisioning

SolidFire uses 4k granular thin provisioning that does not require any reserve space, increasing effective capacity by immediately consuming less space. This feature increases efficiency and reduces overhead by using the smallest allocation possible while maintaining alignment with the native 4KB allocation format used by modern operating systems.

SolidFire volumes don't use any reserve space. Therefore, it is practical to deploy a volume capacity for the estimated maximum size of the database and to purchase only enough physical hardware to support the actual space consumed by the database. As database space consumption approaches the physical limits of the cluster, additional nodes can be dynamically added to the cluster to increase its physical capacity. This process is completely transparent to applications and imposes no need for downtime or reconfiguration of the operating system or the database.

Furthermore, SolidFire Double Helix replication automatically redistributes existing data over the added nodes to create ideal load balancing of both existing and new data. With this deployment paradigm, logical storage capacity can be configured once for the lifetime of the supported databases rather than using incremental updates to accommodate the needs of the database.

Compression and Deduplication

Each SolidFire node includes a PCIe NVRAM card that serves as a write cache. When a host sends writes, they are divided into 4KB data blocks, which are immediately hashed, compressed, and stored in the NVRAM of the storage nodes before an acknowledgement is returned. The resulting value serves as a block ID that determines block placement, which is randomly distributed across all nodes to create an even load distribution.

The SolidFire deduplication block service identifies blocks that have previously been written based on the block ID. If a block already exists, metadata is updated, and the duplicate is discarded. The process is inline and global to the storage cluster.

The combination of inline compression and deduplication has the following advantages:

- Reduce repetitive writes to media, increasing the life of the drives
- Increase system performance by minimizing system resources
- Evenly distribute capacity and performance loads across the system, eliminating hot spots

One key to SolidFire technology is its ability to minimize writes by doing the compression in memory before the actual writes to disk. The writes from the host are divided into 4KB data blocks, which are immediately hashed and compressed into the node's NVRAM write cache. Each compressed block is synchronously replicated to one or more additional storage nodes for data protection. Any future writes coming to the nodes are compared against the hash value and are discarded if they are already present.

2.2 Multitenancy

Quality-of-Service (QoS) Control

NetApp SolidFire all-flash arrays present performance and capacity as dynamic independent pools. This feature enables administrators to set the performance requirements for all the databases or tenants hosted on the same cluster. The minimum, maximum, and burst control settings in the QoS policy guarantee the required performance and can be dynamically changed any time. If SolidFire hardware resources are pushed close to their physical limits, IT staff can dynamically and seamlessly add more

SolidFire nodes to the existing cluster. The Double Helix data distribution automatically redistributes data for optimal load balancing over all hardware resources. This process is transparent to upstream applications.

VLAN Tagging

Workload consolidation is another major effort going on in many data centers. This means that multiple applications and potentially different customers or divisions operate on the same equipment within the data center. VLAN tagging is a feature that enhances the security in the solution, separates network traffic, and provides consistency in how a network is configured. VLAN tagging is available in all SolidFire all-flash arrays.

2.3 Scale

In the modern data center, a constant struggle exists between applications and storage. Applications are very sensitive and require adequate storage performance and capacity to operate. Any imbalance can negatively affect an application's ability to perform, causing a ripple effect to the business. Flash solves many of the performance problems, but due to the significant performance available in an all-flash array, most systems run out of capacity before performance.

As a new node is introduced to a cluster, its performance and capacity (block and metadata) are added to the collective pool of the SolidFire cluster. When block drives are added into the system, SolidFire evenly distributes the existing block data to the newly added block capacity. This results in the newly added node having an equal share of block data compared to each other node in the system.

In a database environment, you are not likely to need to add an exactly equal amount of performance or capacity, and it's never a guarantee that you need to add the same level of capacity and performance that you have been purchasing. Therefore, it is essential to choose an architecture that allows you to pick a ratio of performance and capacity that best fits your needs. SolidFire is designed to provide what you need when you need it by delivering the following:

- **Nondisruptive scale-out/scale-in.** Add or remove nodes to a SolidFire cluster without disrupting service or compromising volume-level QoS settings. Data is automatically redistributed in the background across all nodes in the cluster, maintaining perfect balance as the system grows.
- **Instant resource availability.** Newly added storage capacity and performance resources are instantly available to each volume within the system, eliminating the need to reallocate volumes over new drives.
- **Simplified capacity planning.** Initial implementations begin as small as a 4-node/4U cluster configuration and scale out easily in 1U node increments, allowing performance and capacity resources to be added as needs dictate. Eliminate multiyear capacity and performance projections and scale on demand.
- **Seamless generational upgrades.** New nodes with more capacity and performance are simply added to the established cluster, while old nodes are removed, retired, or repurposed. No rebalancing, restriping, or volume reallocation is required. And all QoS settings remain enforced.

3 Application Considerations

3.1 Database Consolidation

SolidFire is an optimal storage system well suited for database consolidation. SolidFire per-volume QoS controls mean that individual databases get the I/O throughput they need without being affected by other databases running in parallel on the same storage system. With QoS and data reduction efficiencies, you can get significantly higher database density from a shared storage infrastructure. The use case of deploying hundreds of individual databases is an excellent fit for SolidFire.

With SolidFire, a single LUN can be used to provision all the data files and achieve the required performance, instead of spreading data files across multiple volumes, controllers, and arrays. SolidFire QoS guarantees performance for each database and can eliminate the need to implement complex logical volume management configurations to consolidate multiple LUNs or volumes to meet the performance needs of your business.

3.2 Data Protection and Disaster Recovery

SolidFire Double Helix data protection is a shared-nothing, distributed replication algorithm that spreads two redundant copies of data across all drives within the entire cluster. The architecture means no single point of failure across the solution and, when combined with storage efficiency and QoS, provides a compelling disaster recovery (DR) solution. This solution allows the same storage resources to be shared by DR and dev/test without any risk of a performance penalty.

3.3 Dev/Test

Storage Snapshot™ copies provide a point-in-time view of the contents of an active file system or storage volume. Snapshot copies are used for rapid recovery of datasets that might be damaged or corrupted, as well as to create space-efficient copies of datasets for development, test, and related uses through the deduplication feature. The cloning process can be coupled with SolidFire QoS control so that database clones can coexist with production copies without performance effects on upstream applications.

The copy volume feature of NetApp SolidFire allows admins to refresh an existing clone copy of a database without any file system remount operations. In this use case, you frequently refresh a copy of the database by only taking changes from the production copy.

4 Datos IO RecoverX Overview

Datos IO RecoverX is elastic, scale-out software for multicloud environments, delivering efficient data protection and data management services for traditional and new applications. RecoverX is built on top of a seminal data management architecture called consistent orchestrated distributed recovery (CODR), which is not dependent on media servers and transfers data in parallel to and from file-based and object-based secondary storage. The architecture is fully distributed in nature, which provides high availability in failure scenarios and uses elastic compute resources for scalable performance. CODR delivers application-consistent data protection and management that allows massive storage efficiency, native formats, and subtable-level granular recovery/mobility solutions at scale for traditional and next-generation applications.

Benefits

Unlike legacy solutions, RecoverX is built specifically to address the requirements of highly scalable cloud-first applications, driving massive benefits for enterprises:

- Software-only elastic architecture allows on-premises and public cloud deployment.
- Fully orchestrated and granular recovery reduces application downtime by about 4x to 6x.
- Industry-first semantic deduplication results in 80% to 90% storage cost savings.
- Native failure handling provides operational resiliency when there are node or database failures.
- Application awareness enables automated refresh of test/dev environments and results in an increase of about 8x in operational efficiency.
- Application awareness enables universal data mobility across private and public clouds.
- Globally distributed metadata catalog allows you to back up anywhere, recover anywhere, and migrate anywhere.

4.1 Product Features and Use Cases

Datos IO RecoverX offers data protection, single-click granular recovery, industry-first semantic deduplication, and cross-cloud data mobility for innovative nonrelational databases (for example, Cassandra, MongoDB), big data file systems (for example, Apache HDFS and commercial distributions from Cloudera), and relational databases (for example, Microsoft SQL Server). RecoverX uses semantic deduplication to achieve massive storage efficiency, thereby reducing capital costs at scale. The rich policy management capabilities give database administrators and storage and backup administrators operational leverage to create workflows in line with their application and workload requirements. Cross-cloud data mobility allows universal visibility and portability of data and results in massive operational efficiency. The key use cases that RecoverX addresses are (1) point-in-time backup and recovery, (2) test/dev refresh, (3) cross-cloud data mobility, and (4) archival and long-term retention. For more information about Datos IO RecoverX, see the company website.

4.2 Granular Data Protection for Cassandra

Apache Cassandra is a popular key-value database that is used by enterprises for various applications such as product catalogs, recommendation/personalization engines, Internet of Things, customer 360 overviews, and fraud detection. Datos IO RecoverX supports column family level data protection and data mobility of Cassandra databases.

Scale-Out Software

RecoverX is built to scale out horizontally to make sure of high availability of data protection infrastructure as well as increase in performance (RPOs) to meet growing application needs:

- **High availability.** Like any other enterprise software, there can be internal system process failures or external infrastructure failures, especially when commodity hardware is used. A single-node deployment creates a single point of failure. Deploying a 3-node RecoverX cluster makes sure that all tasks handled by the failed RecoverX node are redistributed to the remainder nodes in the Datos IO cluster.
- **Higher throughput performance.** The scale-out architecture of Apache Cassandra allows customers to easily scale their data size depending on application growth. The scale-out architecture of RecoverX brings a high degree of parallelism to achieve higher backup and recovery throughput, leading to lower RPO.

Scalable Versioning (Backup)

The versioning operation is highly parallel in nature, whereby RecoverX only acts as a control plane that orchestrates data movement from the data source cluster to version (secondary) storage. This allows Datos IO to handle large clusters and workloads. Databases protected and versioned by RecoverX are stored in the databases' native format. Therefore, the data recovery process is made easier, and vendor lock-in is avoided. RecoverX also brings operational ease of use by allowing administrators to generate versions of their databases at any user-specified time interval and at any granularity (table level or entire database). Finally, given that most distributed databases run on commodity infrastructure, failures (network, storage, node, database) should be planned for and mitigated against. RecoverX makes sure that backup operations are resilient to such failures.

Fully Orchestrated and Granular Recovery

At the time of recovery, the user can either choose a specific version to restore or restore at any point in time from which data can be restored. RecoverX creates the schema in the target cluster and then proceeds to stream data into the cluster directly from the backup storage. No processing is performed on the data at the time of recovery. This is one of the key reasons for the performance advantages of RecoverX compared to traditional solutions. One of the key tenets of RecoverX solution is that, at the end of the restore, the target cluster has a copy of the data that is fully consistent, and hence no additional

database repairs are required. Additionally, Datos IO RecoverX supports advanced recovery features such as any point-in-time recovery, incremental recovery, and query-based recovery.

Industry-First Semantic Deduplication

Semantic deduplication is an industry-first capability that Datos IO has developed specifically to reduce the cost of storing backup data over its retention period. Apache Cassandra natively maintains multiple copies of the primary data, also called replicas. As part of the versioning process, RecoverX removes the redundant datasets to make sure that the backup has no replicas of a primary data, thus providing deduplication of source data across all replicas. This groundbreaking semantic deduplication feature results in up to 70% reduction in secondary storage.

5 NetApp SolidFire Configuration

A 4-node SolidFire all-flash storage system was configured as the storage for the Cassandra clusters and the Datos RecoverX system cluster nodes. Ideally, a separate 4-node cluster for the RecoverX back end would be more desirable for additional high availability and redundant solution design. However, because of resource constraints, we chose to use a single SolidFire 4-node cluster. In this test, SolidFire SF2405 nodes were used. See Table 4 for more information.

SolidFire has features for thin provisioning, always-on deduplication, compression, scalability, database consolidation, and QoS control, allowing flexibility for solutions, including Datos IO RecoverX. One of the most beneficial capabilities of NetApp SolidFire for a Cassandra database deployment is the ability to control IOPS per database volume for each Cassandra cluster node. This allows adjustment of the IOPS to a greater throughput if necessary, avoiding bottlenecks of performance and cluster compaction. SolidFire can also provision additional storage capacity without disruptions. Coupled with the Datos IO RecoverX point-in-time capabilities, the combined features make a compelling solution.

6 Cassandra Configuration

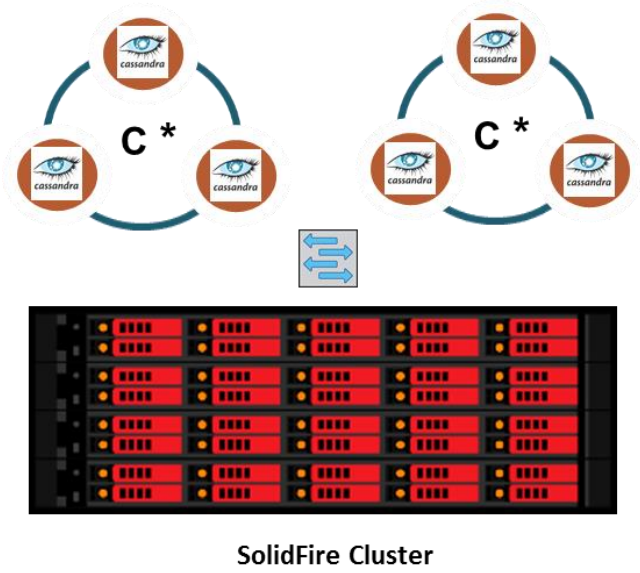
6.1 Installing DataStax Cassandra Enterprise

A CentOS distribution was used for the Cassandra nodes. DataStax Cassandra version 4.8.12 and Datos IO RecoverX version 1.1 were installed per product guide recommendations. Before you begin your installation process, check the DataStax documentation release notes.

For the testing in this document, two Cassandra sharded clusters were deployed. The clusters were configured with a replication factor of 3, which is a common setup for Cassandra cluster deployments.

Each Cassandra server was a VMware virtual machine with the NetApp SolidFire volumes created for the database storage in the 4-node SolidFire cluster. For more information about the Cassandra configuration, see Figure 1. For information about the Cassandra node server's specification, see Table 2.

Figure 1) Cassandra cluster configuration.



In the tested configuration, two Cassandra clusters used a 3-node topology, as shown in Figure 1. Cassandra deployment replicates multiple copies in the cluster using a network topology strategy during keyspace creation. For more information about replication in DataStax Cassandra, see this document.

7 Datos IO RecoverX Configuration

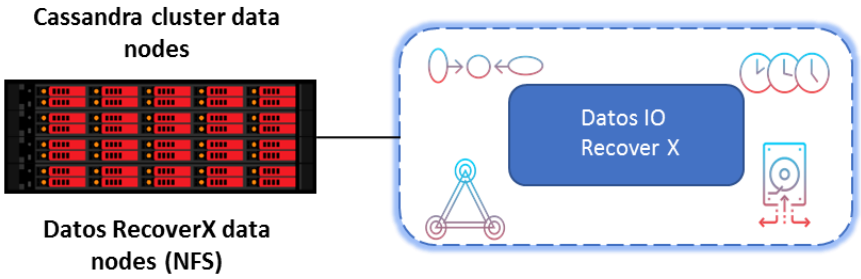
RecoverX can be deployed on physical servers, virtual machines, or any compute instance in the cloud as a single node or as a clustered 3 to 5-node configuration. The infrastructure requirements are minimal, and a wide variety of backup storage—NFS or object—is supported. The clustered configuration provides high availability and high performance to handle massive scale environments (100TB+). RecoverX performance also scales elastically with the underlying infrastructure (compute and memory) that is provided. This makes it possible to dynamically change the infrastructure footprint in response to the changing application workload to get the requisite data protection performance from RecoverX.

Figure 2 shows the RecoverX configuration with SolidFire storage for a high-availability solution emphasizing separation of Cassandra cluster volumes and RecoverX volumes.

Figure 2) Datos IO RecoverX and two SolidFire clusters.

Figure 3 shows a RecoverX configuration with a single SolidFire storage cluster for both the Cassandra cluster and Datos IO RecoverX cluster volumes. This configuration was used for this document. For RecoverX server information, see Table 1.

Figure 3) Datos IO RecoverX and a single SolidFire cluster.



8 Test Environment

The configuration in Figure 4 shows the logical design of the solution deployment. Cassandra communicates through a Secure Shell (SSH) connection that forms a control plane to orchestrate data movement. The data can be backed up to a secondary NetApp SolidFire cluster system. In addition to CLIs and RESTful APIs, customers can use the RecoverX consumer-grade UI to manage their data protection environment.

This Cassandra cluster uses SolidFire volumes for the database storage and commit logs (commit logs). The commit log is a special capped collection that keeps a rolling record of all operations that modify the data stored in the databases. Cassandra applies database operations to the clustered nodes and records them to the commit logs on the nodes and then to a per–column family structure called a memtable. When a memtable is full, it is written to disk as an SSTable to the SolidFire volumes. The commit log is synced per a timed parameter in milliseconds for data durability. This enables all the Cassandra nodes to have a consistent view of all data in the cluster.

Figure 4) Logical test configuration.

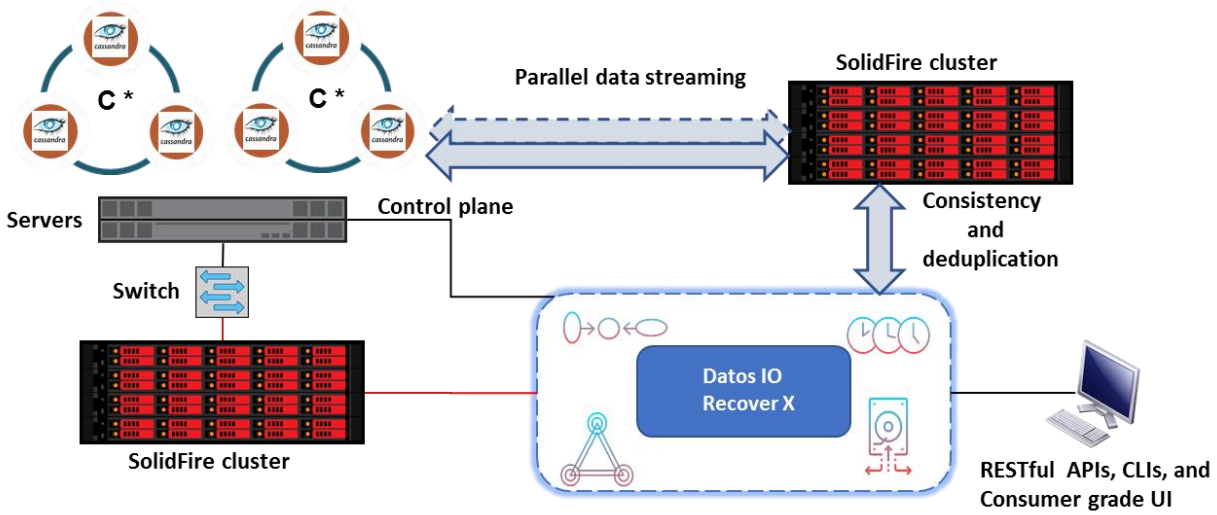


Table 1) Datos IO RecoverX node.

Qty	RecoverX version	OS	Cores/ CPUs	RAM	OS Disk (SSD)	NFS Storage Volume	Python
3	1.1	CentOS 6.7	8 vCPUs on VMware	32GB	285GB SolidFire iSCSI volume	SolidFire 1TB volume type XFS	2.6

Table 2) Cassandra nodes for three clusters.

Cassandra Cluster	Qty	DataStax Cassandra Version	Type	Cores/ CPUs	RAM	OS
Cluster 1	3	4.8.12	VMware	8	32GB	CentOS 6.7
Cluster 2	3	4.8.12	VMware	8	32GB	CentOS 6.7

All Datos IO RecoverX nodes and Cassandra data nodes were configured on a SolidFire SF2405 4-node cluster running NetApp SolidFire Element® OS version 9.0.0.1549.

An additional server as shown in Table 3 was used in the test configuration for a dedicated NFS server to provide a SolidFire volume to the RecoverX cluster nodes and the Cassandra cluster nodes.

Table 3) NFS server with SolidFire volume.

NFS Server	Qty	Type	Cores/ CPUs	RAM	OS
NFS version 3	1	VMware	8	32GB	CentOS 6.7

Table 4) SolidFire SF2405 hardware specifications.

Hardware	SolidFire SF2405 Specifications
Server	Dell PowerEdge 620
Memory	64GB
Drives	Samsung 240GB PM853T
SATA DIMM	100GB
CPU	Intel Xeon E5-2620 v2 @ 2.10GHz
NVRAM	1 x Radian (RMS-200 rev-B04)
Controller	1 x Serial Attached SCSI Controller: LSI Logic/Symbios Logic SAS2008 PCI-Express Fusion---MPT SAS---2 [Falcon] [rev 03)
Ethernet	Broadcom NetXtreme II BCM57800 10 Gigabit Ethernet (rev 10)
Mpt2SAS ver.	FW-13.00.57.00 Chip Rev. (0x03), Bios ver. 07.25.00.00

9 Use Cases

This section describes various test cases to showcase various recovery modes for Datos IO RecoverX. The SolidFire cluster provides the storage required for the Cassandra cluster and the Datos IO RecoverX cluster. Cassandra keyspaces and tables can be recovered directly back into the same Cassandra database cluster (operational recovery). They can also be recovered to a different Cassandra database cluster (testing and development refresh) with a different topology (the number of nodes on the destination cluster differs from the node count of the source cluster). The recovery process deals only with the logical data, making it 3x faster than traditional approaches. The data is directly transferred from secondary storage into target databases.

Use Case 1

Restore a database table from a historic point in time to a different topology Cassandra cluster.

Use Case 2

Restore a database table from the latest point in time back to the same database as a new table.

Use Case 3

Restore a database table from a historic point in time to the same database as a new table.

Use Case 4

Restore a database table while a RecoverX node has failed and the remaining RecoverX cluster nodes are still available.

The use cases were run on the configuration shown in Figure 4 containing two Cassandra clusters, as specified in Table 2. Each cluster has three nodes.

Figure 5) RecoverX dashboard.

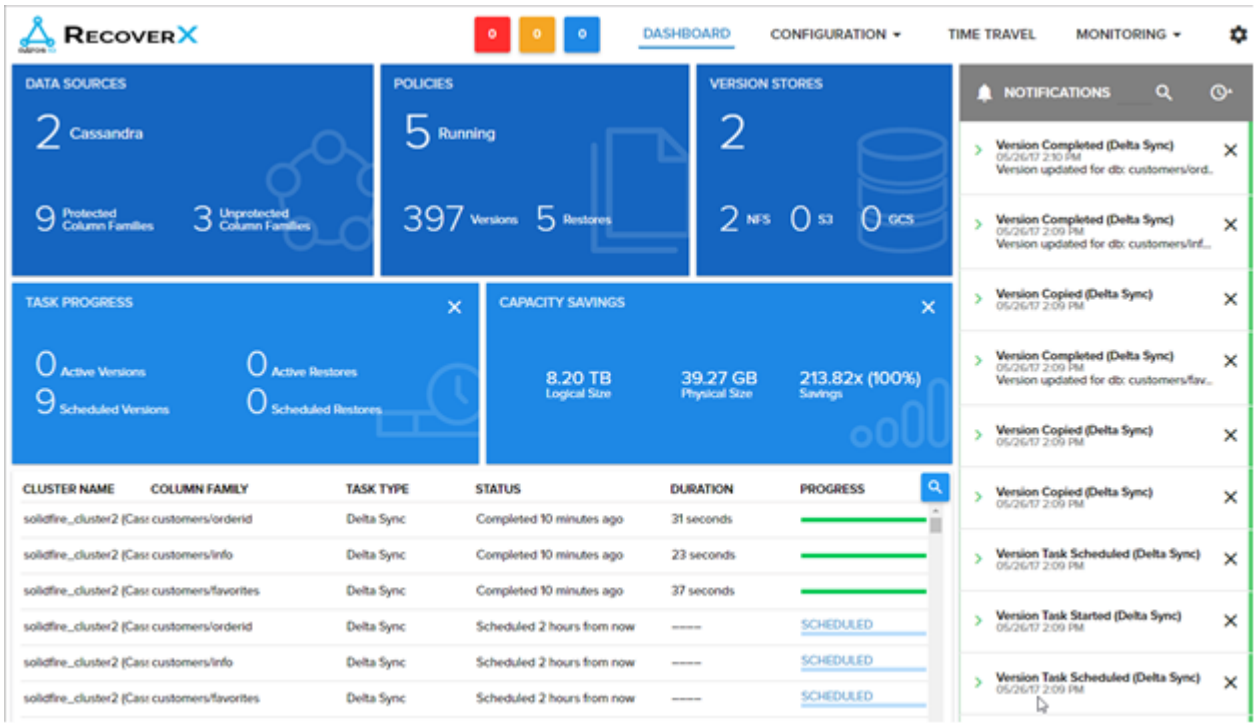


Figure 6) RecoverX data sources view of Cassandra clusters.

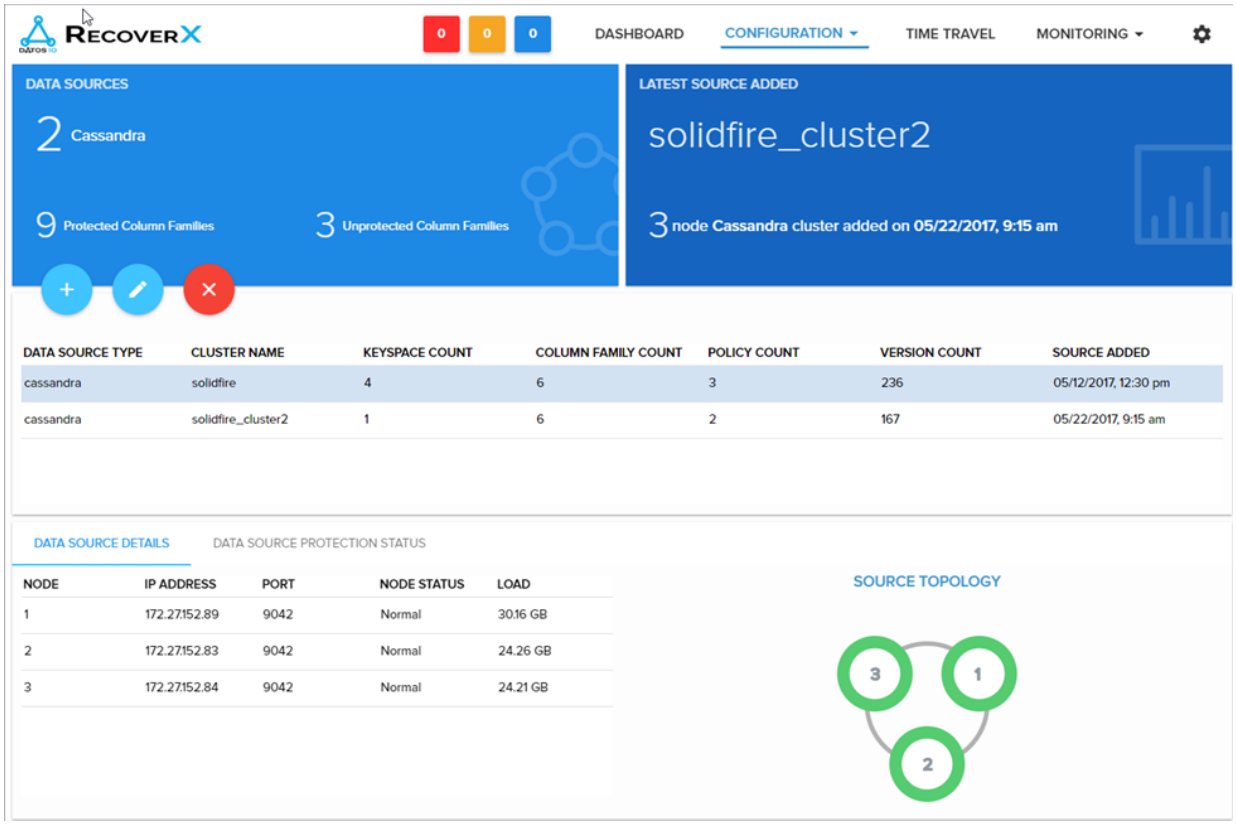
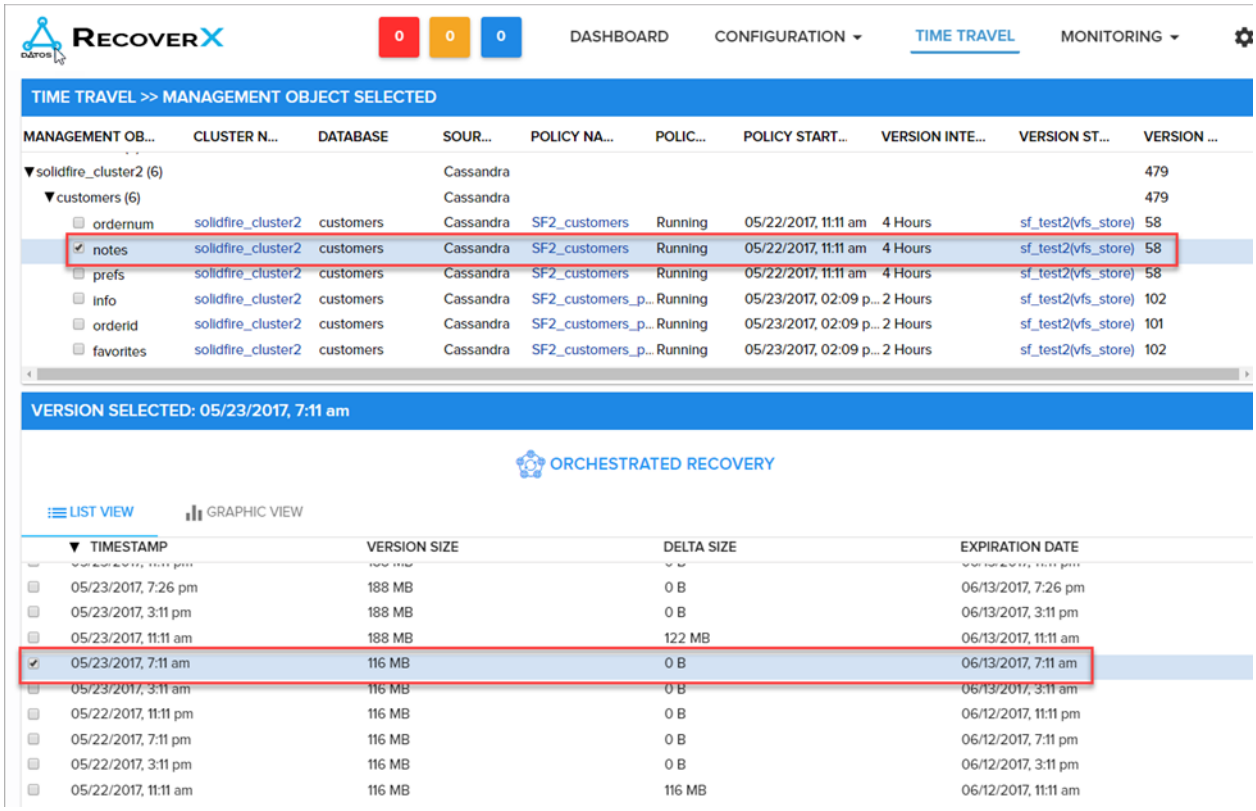


Figure 5 shows the Datos IO RecoverX UI dashboard with the two Cassandra data sources and running policies. Figure 6 shows the configuration screen with the data source details of the two Cassandra clusters and their nodes. A 3-node Cassandra cluster is shown in the details view.

9.1 Use Case 1 Test

Figures 7 through 16 show use case 1 restoration of a collection from a historic point in time to a different topology Cassandra cluster.

Figure 7) Orchestrated point-in-time recovery for use case 1.



In Figure 7, the source Cassandra cluster keyspace “customers” and table “notes” are chosen for recovery. This table has been backed up by Datos IO RecoverX, and users can restore from any available point-in-time version. In Figure 8, the data source details of the available keyspaces and tables in the cluster are displayed. In this test, we recovered the cluster “solidfire2” keyspace “customers” and table “notes” to another cluster named “solidfire.”

In Figure 7, a historical point-in-time version for the table “notes” is selected for restore to a different Cassandra cluster.

In Figure 8, the RecoverX GUI shows the cluster name as “solidfire_cluster2” in the top half of the screen, and the lower window shows the databases and tables in that cluster. The other information is for the data protection status of all the databases and tables in that cluster.

Figure 8) Data source protection status for Cassandra databases.

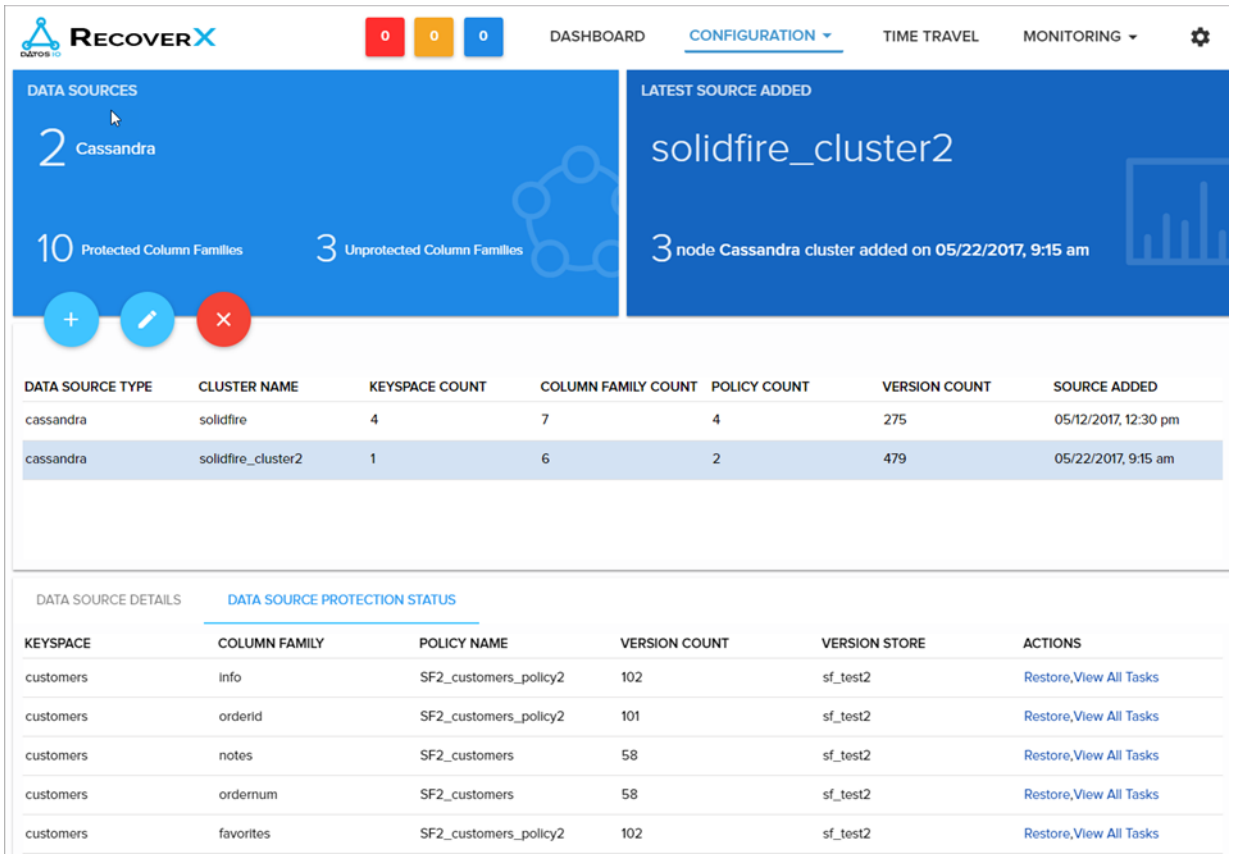


Figure 9 shows the original target database keyspaces from the “solidfire” cluster. The keyspace “customers” from the source cluster database is used for the restore, which creates a new keyspace and table to the target database.

Figure 9) Cassandra cluster for target database.

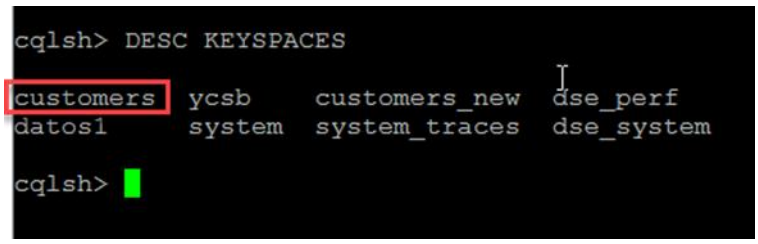


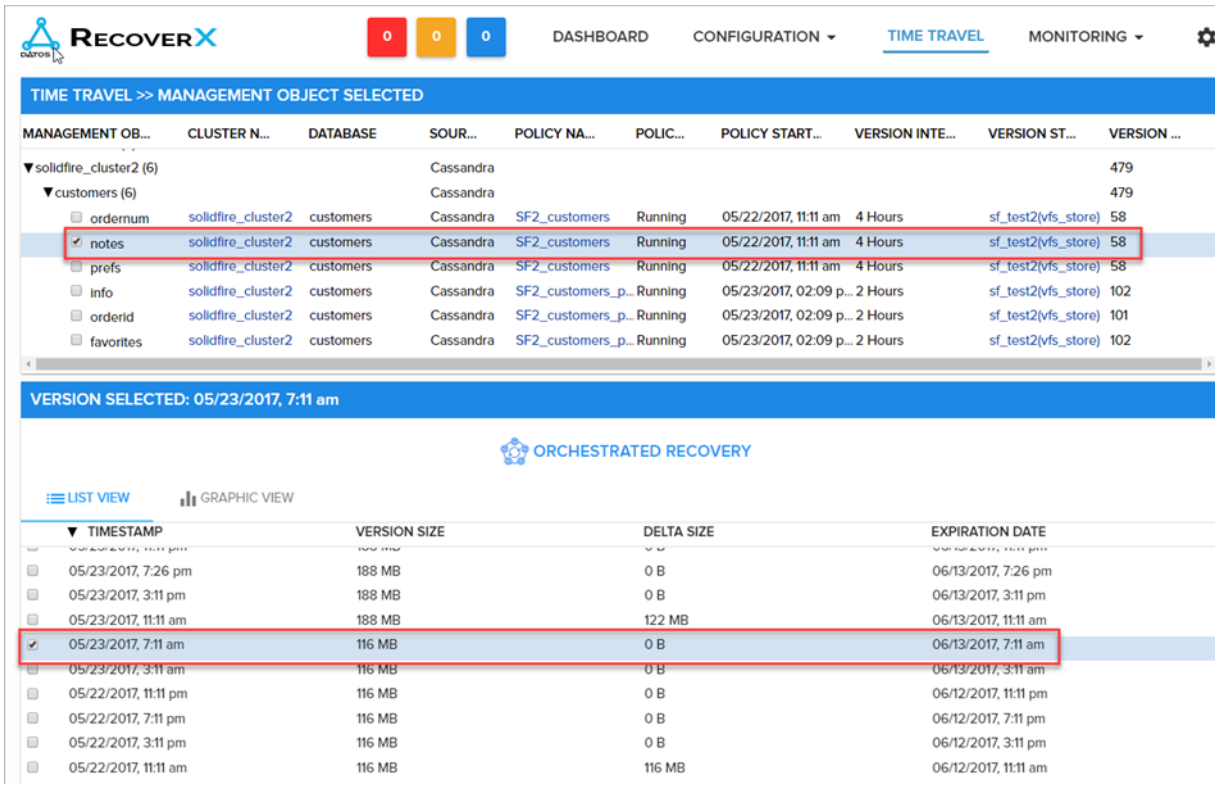
Figure 10 shows that we have access to Cassandra source cluster “solidfire_cluster2.” This is the list of current keyspaces. In the recovery test, we restore a version of the “customers” keyspace and “notes” table from the “solidfire_cluster2” cluster to the “solidfire” cluster. Datos IO uses its proprietary versioning algorithms to create a single cluster-wide consistent version of a Cassandra database, keyspace, or table in a cluster. This logical backup allows the restore of the database to a different topology cluster. Datos IO RecoverX allows full orchestration of selected keyspace and tables for the recovery with no user intervention on the Cassandra cluster nodes.

Figure 10) Cassandra cluster for source database.

```
cqlsh> DESC keyspaces

system_traces  dse_perf  customers  dse_system  system

cqlsh> DESC table customers.notes

CREATE TABLE customers.notes (
    ceh_uuid uuid,
    evt_ts timestamp,
    chn_id int,
    evt_payld text,
    evt_uuid uuid,
    src_sys text,
    PRIMARY KEY (ceh_uuid, evt_ts, chn_id)
) WITH CLUSTERING ORDER BY (evt_ts DESC, chn_id ASC)
```

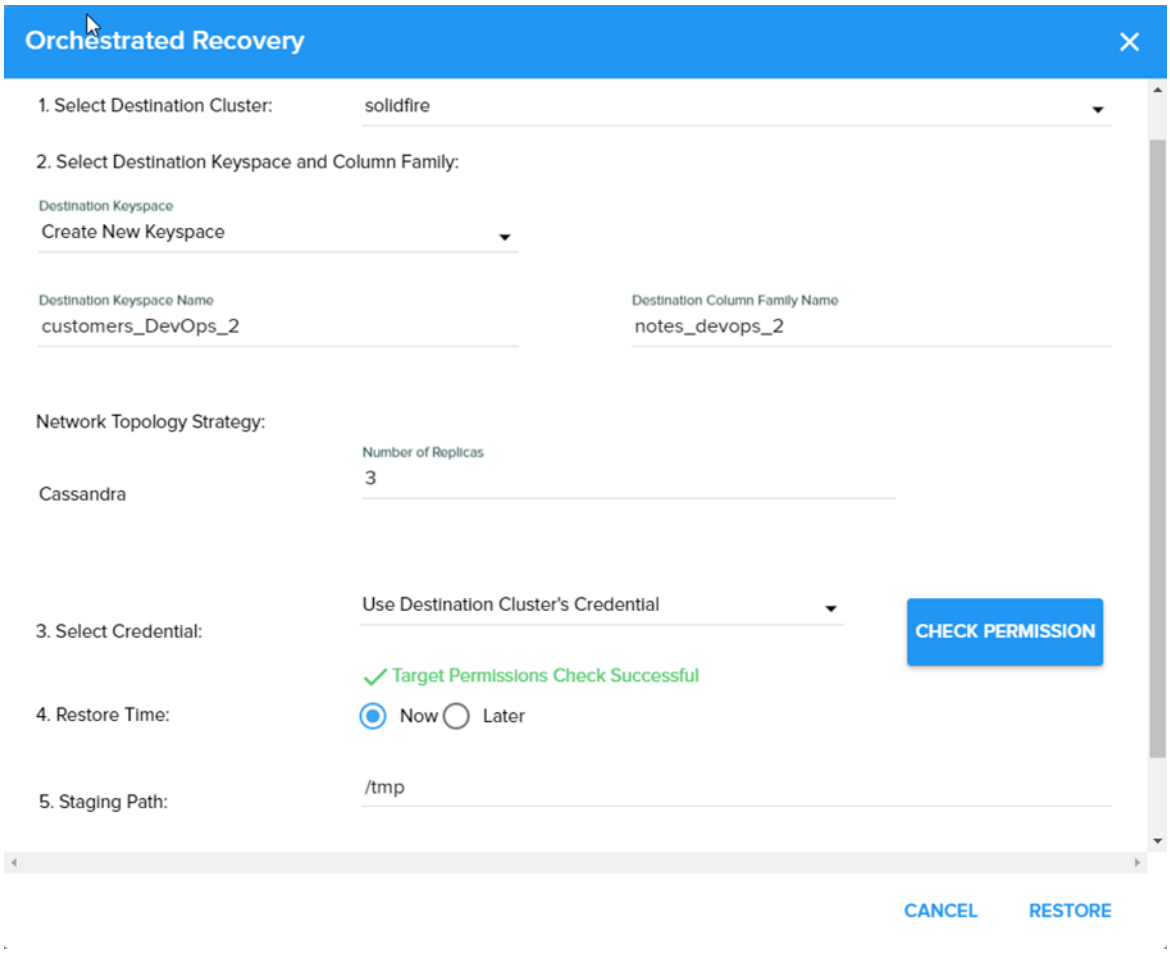
In Figure 11, the source cluster keyspace and table “customers.notes” are selected for the orchestrated restore.

Figure 11) Source cluster keyspace and table restore selection.



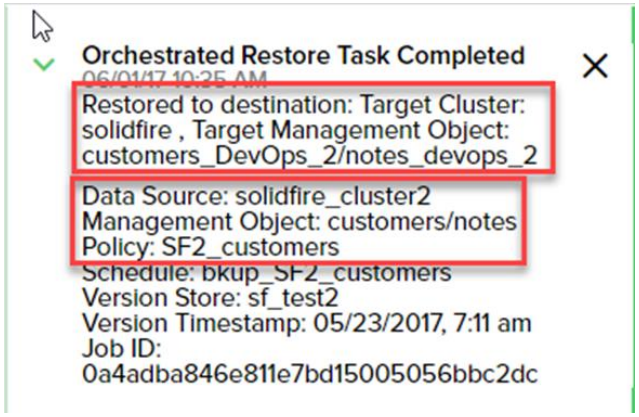
In Figure 12, we specify the destination cluster and the keyspace and table to be automatically created by RecoverX. In this example, we create the keyspace “customers_DevOps_2” and the table “notes_devops_2.” The RecoverX nodes perform checks for permissions and available disk space in the staging path.

Figure 12) Use case 1 orchestrated recovery creation.



In Figure 13, the dashboard notification view shows the completed orchestrated restore.

Figure 13) Use case 1 orchestrated recovery notifications view.



When the recovery is completed, the new database keyspace and table are viewable at the destination cluster. Figure 14 shows the Cassandra cluster node view containing the new database.

Figure 14) Use case 1 target Cassandra cluster postrestore view.

```
[datos_db_user@cassandra-node1 root]$ nodetool cfstats customers_DevOps_2.notes_devops_2
Keyspace: customers_DevOps_2
        Read Count: 0
        Read Latency: NaN ms.
        Write Count: 0
        Write Latency: NaN ms.
        Pending Flushes: 0
                Table: notes_devops_2
                SSTable count: 3
                Space used (live): 124946404
                Space used (total): 124946404
                Space used by snapshots (total): 0
                Off heap memory used (total): 290788
                SSTable Compression Ratio: 0.05114382229345397
                Number of keys (estimate): 21
```

In Figure 15, the RecoverX dashboard view shows the completed restore with a duration of 2 minutes into a different topology Cassandra cluster.

Figure 15) Use case 1 dashboard view of successful restore.

CLUSTER NAME	COLUMN FAMILY	TASK TYPE	STATUS	DURATION	PROC
solidfire (Cassandra)	customers_DevOps_2/notes_devops_2	Init Sync	Completed 21 minutes ago	1 minute	
solidfire (Cassandra)	customers_DevOps_2/notes_devops_2	Delta Sync	Scheduled 4 hours from now	-----	SCHE
solidfire (Cassandra)	customers_DevOps_2/notes_devops_2	Rebase	Scheduled 14 days from now	-----	SCHE
solidfire (Cassandra)	customers_DevOps_2/notes_devops_2	Add Policy	Completed 23 minutes ago	9 seconds	
solidfire_cluster2 (Cassandı	customers/notes	Orchestrated Restore	Completed 34 minutes ago	2 minutes	

The Datos IO RecoverX GUI, Figure 16, shows the create policy task that users can add after a table is recovered. After the policy is created, an initial full backup of keyspace “customers_DevOps_2” and table “notes_devops_2” is created.

Figure 16) Use case 1 RecoverX successful restore create policy details.

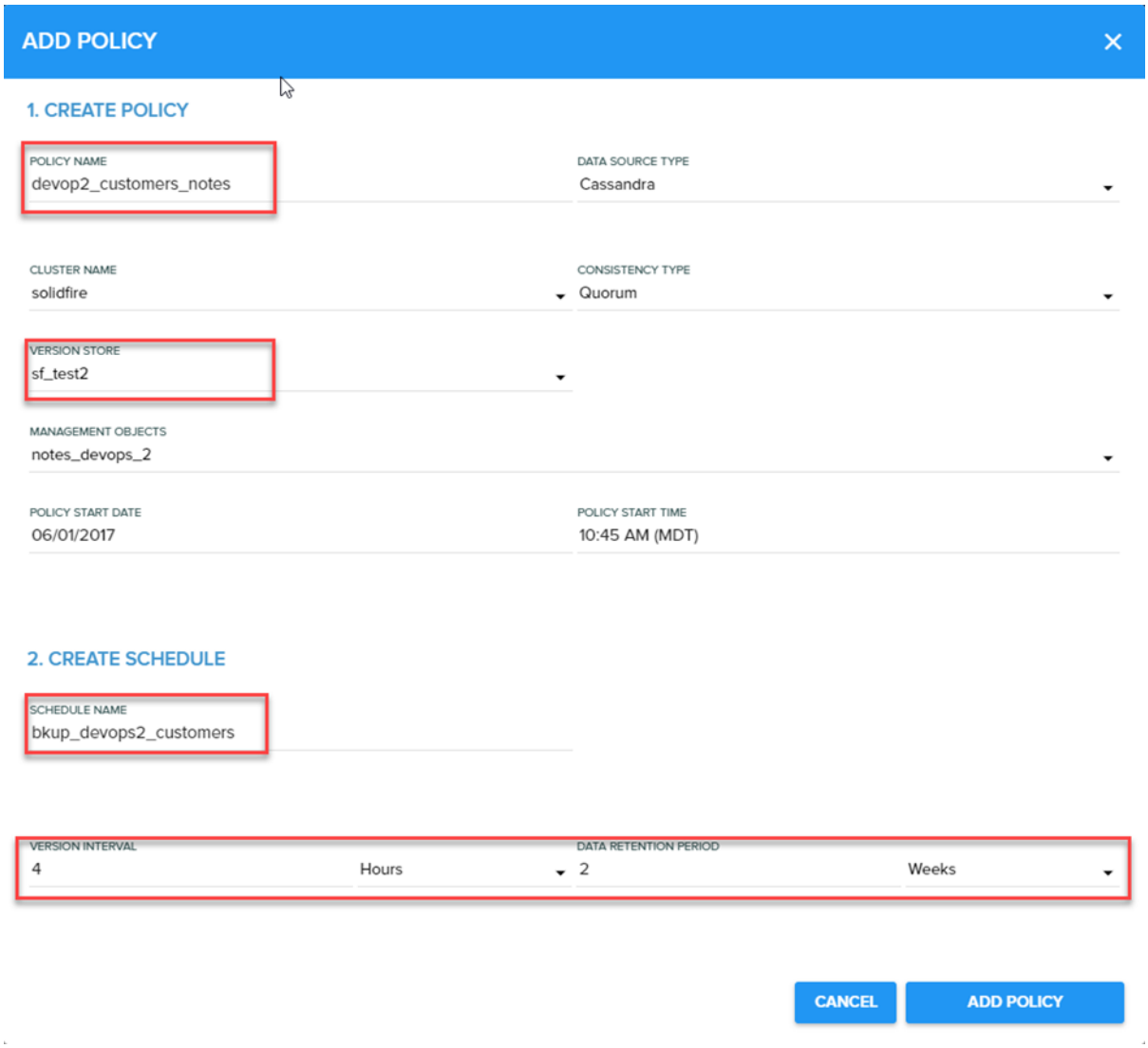
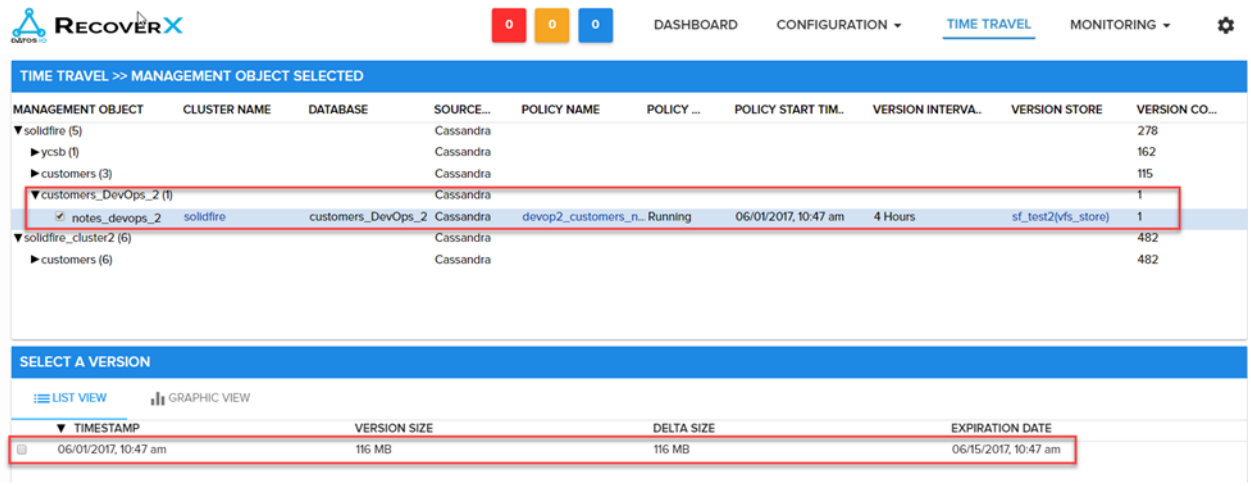


Figure 17 shows the first initial version is completed. The version interval specifies that RecoverX creates a new version using commit log changes of “customers_DevOps.notes_devops_2” every 4 hours.

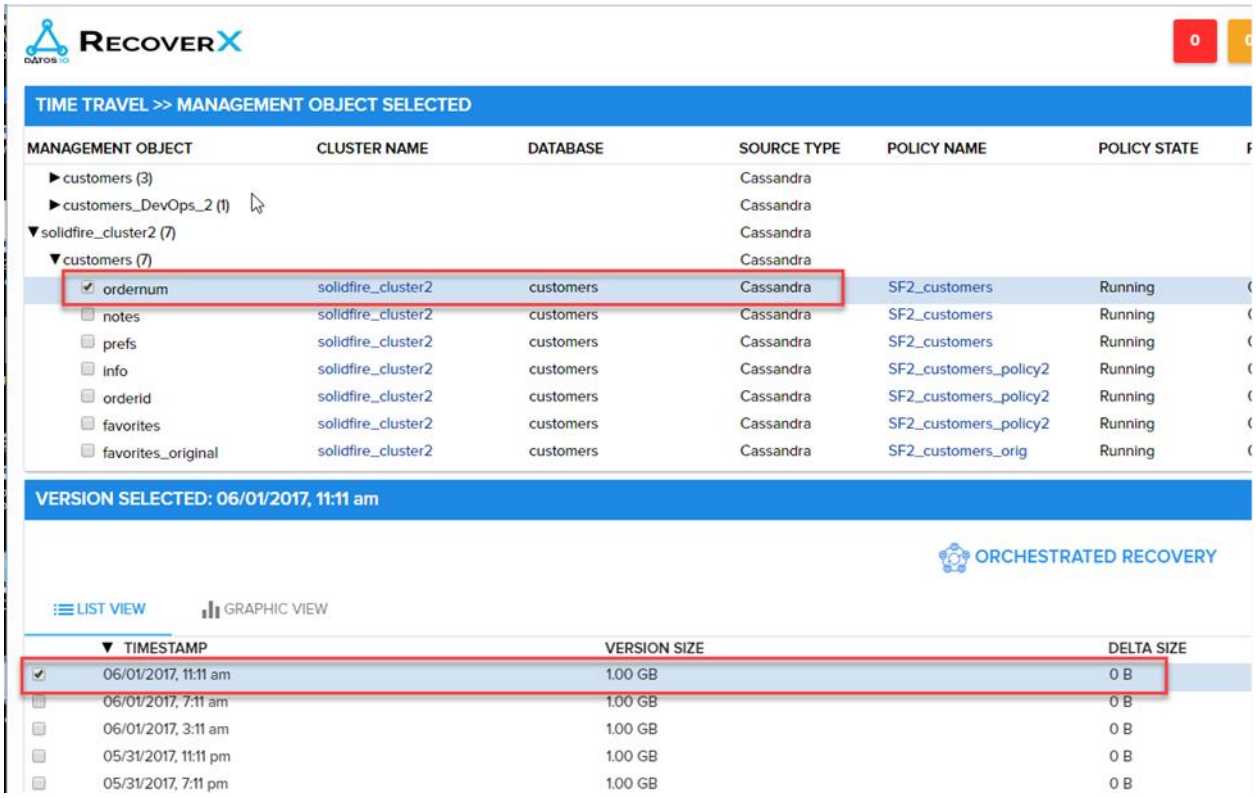
Figure 17) Use case 1 target cluster completed backup version.



9.2 Use Case 2 Test

Use case 2 restores a Cassandra table from the latest point-in-time version back to the same database cluster under a new keyspace and table. Following the same orchestrated recovery steps from the use case 1 example, Figure 18 shows the latest point-in-time version for “customers.ordernum.” A new keyspace and table can be used for a variety of purposes, reducing the operational burden of refreshing test/dev clusters for continuous development dev/ops environments. A simple click of Orchestrated Recovery and we can begin the process of creating a new table.

Figure 18) Use case 2 RecoverX view of latest point-in-time version available for a restore.



In Figure 19, the new keyspace and table “customers.ordernum_latest_pit” are created for usage while leaving the original table as is. The destination cluster “solidfire_cluster2” and the destination keyspace “customers” are the same because we only create a new table. This is a typical example of the low-touch restore process for Cassandra administrators and users, because the creation of the new database is completed by RecoverX and not any direct interaction with Cassandra nodes.

Figure 19) Use case 2 orchestrated recovery view.

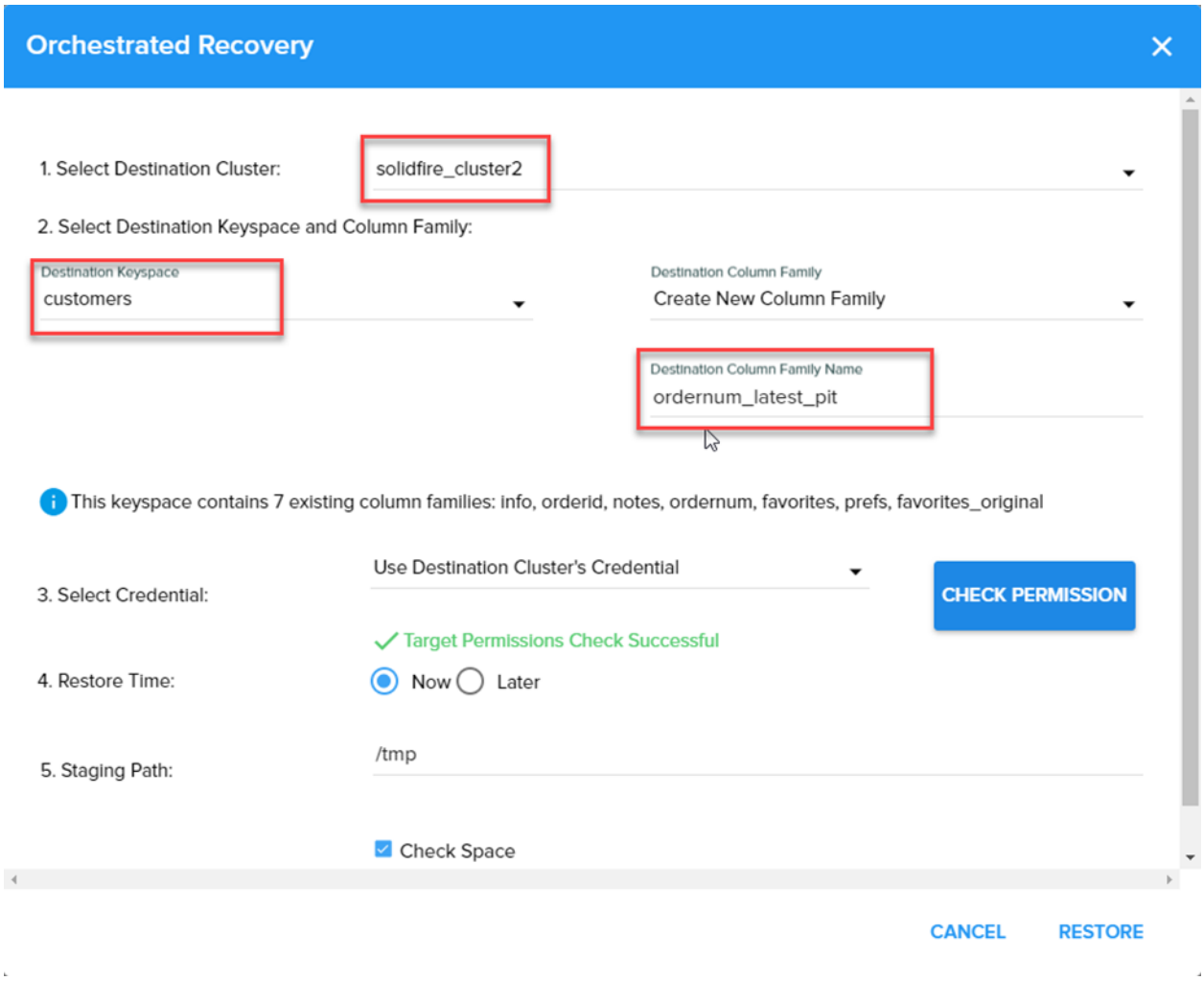
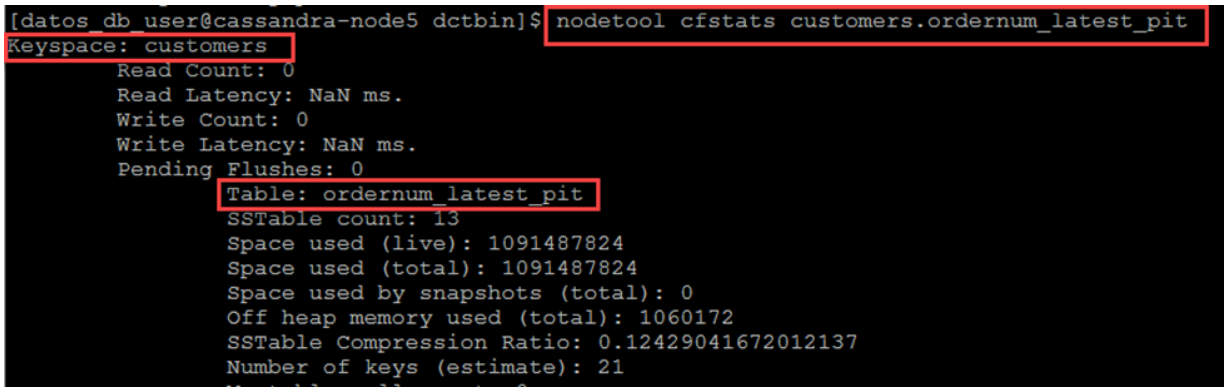


Figure 20 shows the restored table on the Cassandra database cluster.

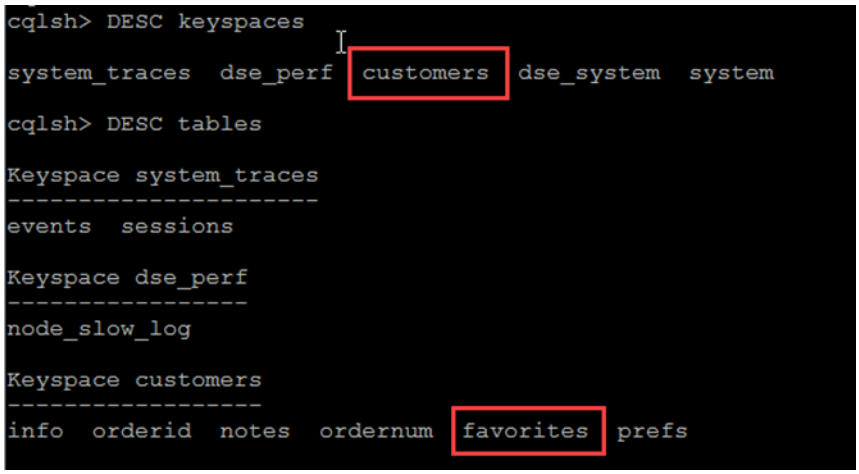
Figure 20) Use case 2 Cassandra view of completed point-in-time recovery.



9.3 Use Case 3 Test

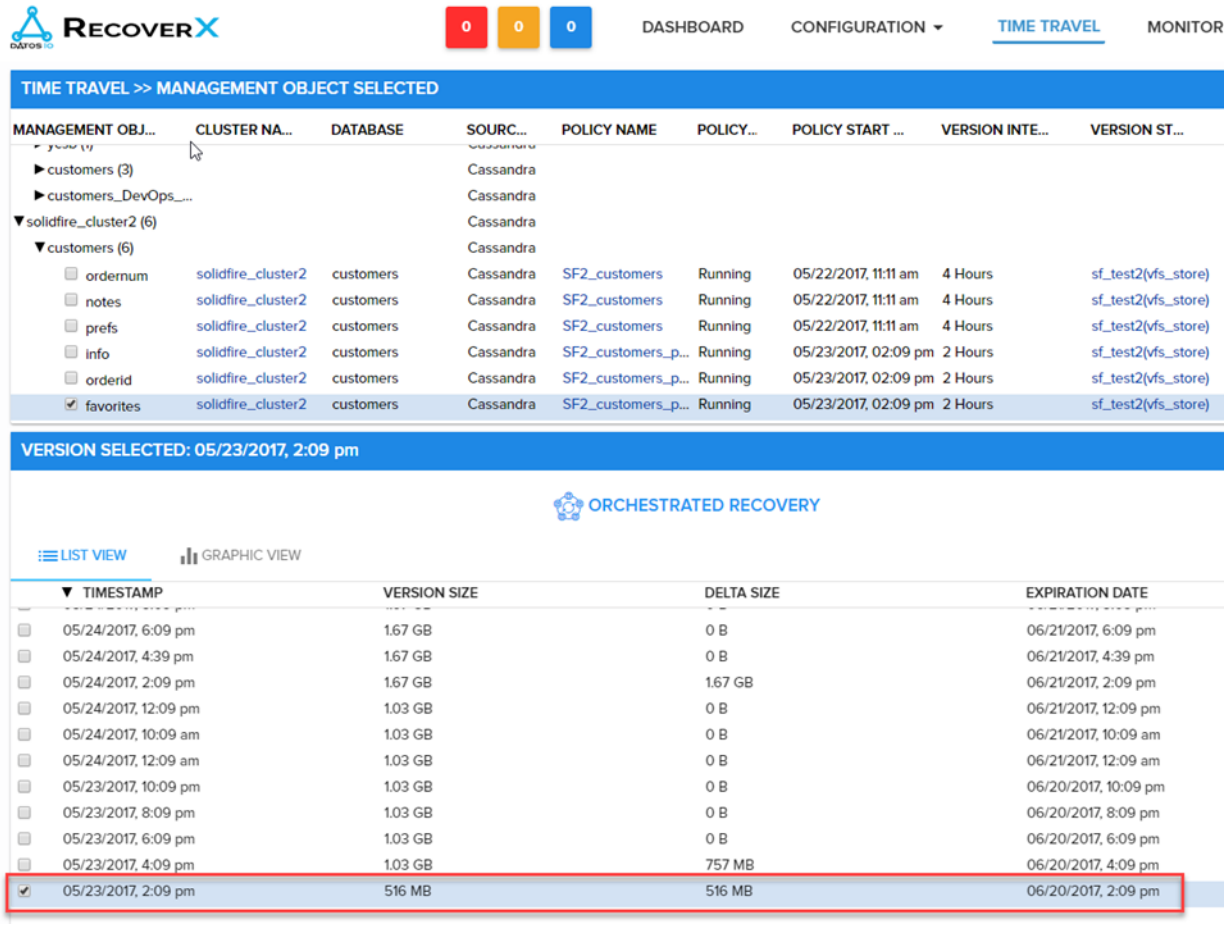
This use case shows a restore of a Cassandra table from a historic point in time to the same database cluster and keyspace as a new table. The same steps and procedure as described in use case 1 and 2 are used. In the Cassandra cluster shown in Figure 21, the keyspace and table “customers.favorites” are recovered to a new keyspace and table named “customers.favorites_original.” We used a historical point-in-time backup to reconcile the database to a previous older state. This is a convenient method to restore a database to a prior working state from several versions in past time. As with the other use case restores, the SolidFire data node volumes have the databases available for the RecoverX cluster nodes to create the new database. No manual intervention is needed in the Cassandra cluster nodes for this orchestrated restore.

Figure 21) Use case 3 Cassandra cluster server view of source keyspace and table.



In Figure 22, an older historical point-in-time version is selected for an orchestrated recovery.

Figure 22) Use case 3 RecoverX historical point-in-time version.



The recovery process uses the logical data, which is directly transferred from SolidFire secondary storage through the NFS server shared directory into target databases. An additional benefit of the SolidFire, Cassandra, and RecoverX solution is the storage savings. In Figure 23, the RecoverX dashboard shows the capacity savings of the actual physical size of the two Cassandra database clusters, 37.14GB, and the logical size, 11.87TB. RecoverX accomplishes these data savings by taking incremental forever backups, semantic deduplication, and advanced support for compacted SStables.

Figure 23) RecoverX capacity savings view.

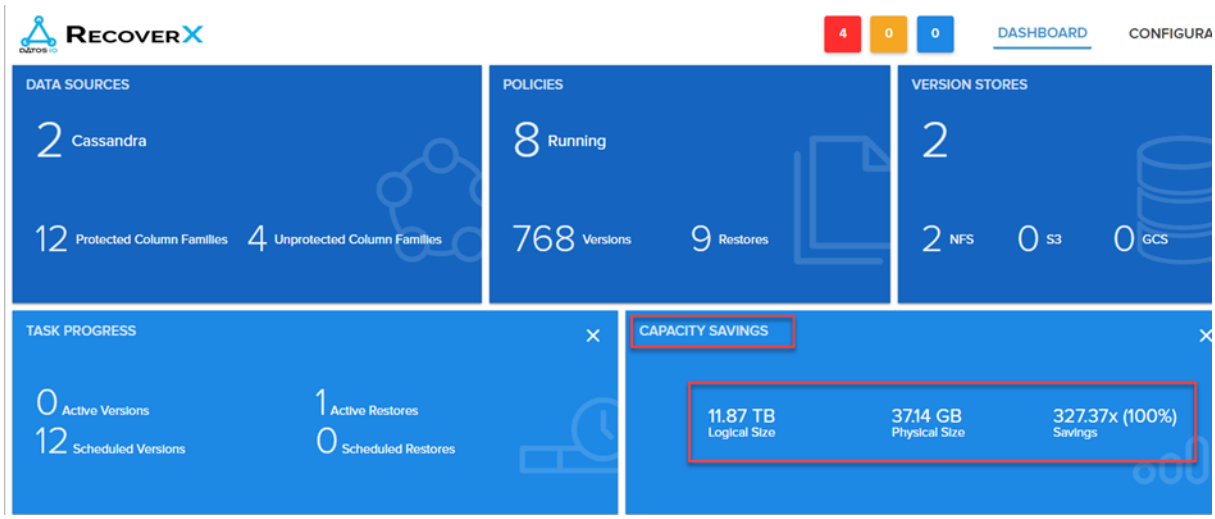
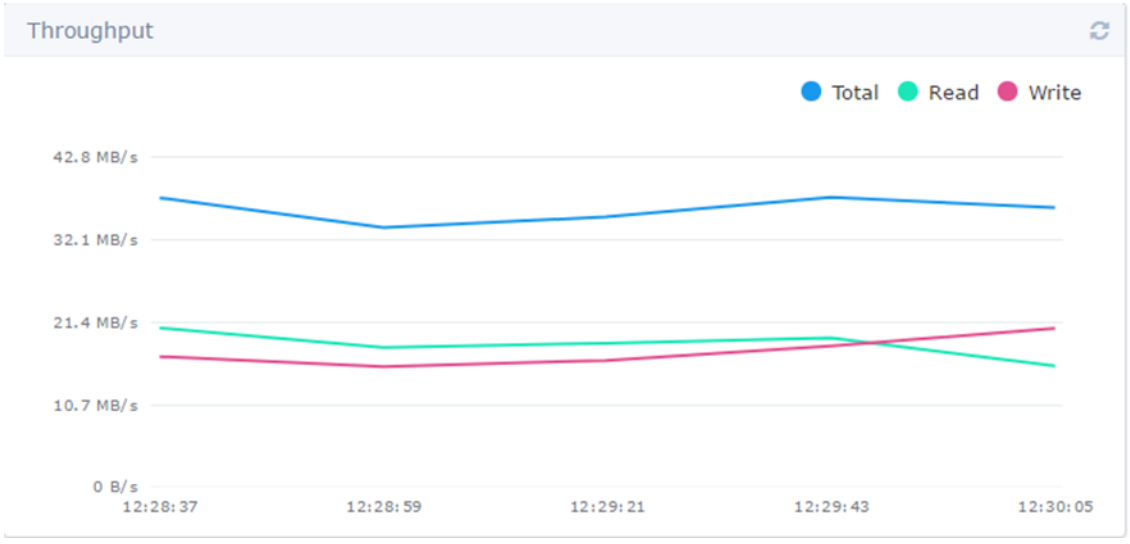


Figure 24, a performance graph in the SolidFire GUI, shows a typical recovery in throughput for a database restore. In the testing for this test and document, small-sized databases were loaded in the Cassandra clusters. Normally Cassandra can have databases in the several TB range. In this test, orchestrated recoveries took minutes to complete.

Figure 24) SolidFire performance graph.



9.4 Use Case 4 Test

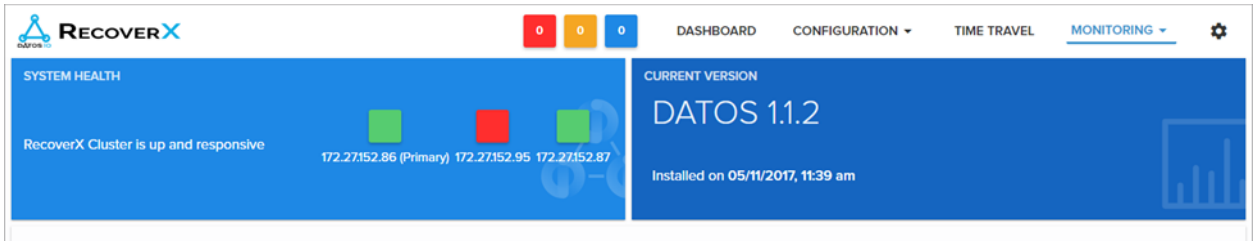
This use case shows a restore of a collection from a historic point-in-time backup from a Cassandra cluster to another cluster while a RecoverX cluster node is down and not available. In Figure 25, the RecoverX cluster CLI shows that only two of the three servers are available, but the cluster is still up and responsive. High availability is a key feature of RecoverX, so users may back up and recover their data even when there are failures at the infrastructure layer (VM, servers).

Figure 25) RecoverX downed node status CLI view.

```
[datos_db_user@datos-rx-2 datosinstall]$ ./datos_status

**************************************************
Datos Status: RecoverX Cluster is up and responsive
Datos Primary Node: 172.27.152.86
**************************************************
Node 172.27.152.86: All processes are up
Node 172.27.152.95: No processes detected
Node 172.27.152.87: All processes are up
**************************************************
```

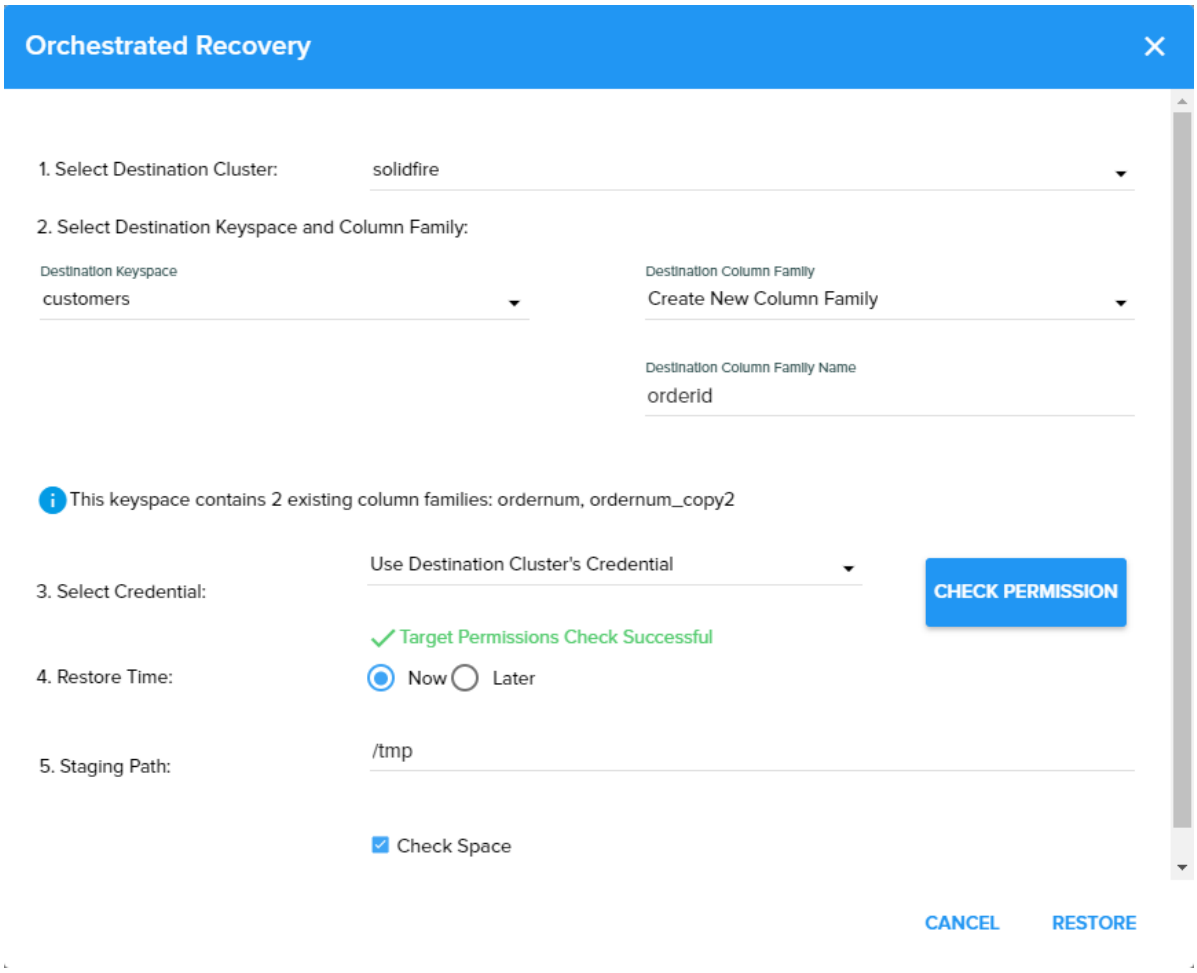
In Figure 26, the RecoverX monitoring view displays the cluster status is up and responsive, while one node is marked red for down.

Figure 26) RecoverX downed node GUI view.



For use case 4 we use the same orchestrated recovery steps as in the previous use cases to create a new keyspace and table for recovery. In this test, the table is recovered to a different target Cassandra cluster. The “customers.orderid” table is restored to the “solidfire” Cassandra cluster. In Figure 27, the orchestrated recovery information is entered.

Figure 27) RecoverX downed node orchestrated recovery creation.



After the orchestrated recovery is successful, in Figure 28 the view from the Cassandra cluster node member of the Cassandra cluster “solidfire” displays the new “customers.orderid” database, which matches the same database from the source database. This orchestrated recovery is done without issues with one of the RecoverX cluster nodes in a down state.

Figure 28) RecoverX downed node successful orchestrated recovery.

```
[root@cassandra-node2 datos_db_user]# nodetool cfstats customers.orderid
Keyspace: customers
        Read Count: 0
        Read Latency: NaN ms.
        Write Count: 0
        Write Latency: NaN ms.
        Pending Flushes: 0
                Table: orderid
                SSTable count: 6
                Space used (live): 795546906
                Space used (total): 795546906
                Space used by snapshots (total): 0
                Off heap memory used (total): 771856
                SSTable Compression Ratio: 0.1243162344454062
                Number of keys (estimate): 21
```

Appendix: Test Configuration Information

RecoverX Cluster Node Configuration

Figure 29 shows the disk configuration of the Datos IO RecoverX node. The “/home” directory is a separate partition from the “/” or root partition. Per Datos RecoverX node specification, in a Cassandra deployment the “/home “directory should have 140GB of space. In this test, a nonproduction test environment of 145GB is available. The SolidFire volume used for the RecoverX cluster nodes is an NFS volume from a dedicated VM server for NFS only and mounted and located in the “/datos/rb” directory.

Figure 29) RecoverX node disk configuration.

```
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/vg_mongodatosr1-lv_root
                       44G  6.2G   38G  15% /
tmpfs                  32G  132K   32G   1% /dev/shm
/dev/sda1             477M   80M  373M  18% /boot
/dev/mapper/lv_home-lvd_home
                      145G   14G  124G  10% /home
datos-nfs-1:/datos/rb
                      931G   43G  889G   5% /mnt/versions
```

Figure 30 shows the NFS volume on SolidFire in which the Datos RecoverX cluster nodes directory stores the backups.

Figure 30) RecoverX node directory details.

```
/mnt/versions
drwxrwxr-x  3 datos_db_user cassandra 4096 May 30 13:27 sf_test1
drwxrwxr-x  4 datos_db_user cassandra 4096 Jun 15 19:14 sf_test2
```

Figure 31 shows the /etc/fstab mount information for the SolidFire volume for the RecoverX node.

Figure 31) RecoverX disk mount details (/etc/fstab file).

```
datos-nfs-1:/datos/rb /mnt/versions    nfs
auto,hard,nfsvers=3,actimeo=0,lookupcache=none,noac      0       0
```

Figure 32 shows the Datos RecoverX SolidFire volume, NFS mounted to a Cassandra cluster node. Each Cassandra cluster data node has this directory mounted for parallel processing and point-in-time recovery data.

Figure 32) NFS VM server SolidFire volume.

```
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/vg_mongodatosr1-lv_root
                       44G  7.9G   36G  19% /
tmpfs                  32G   72K   32G   1% /dev/shm
/dev/sda1             477M   80M  373M  18% /boot
datos-nfs-1:/datos/rb
                      931G   43G  889G   5% /mnt/versions
/dev/mapper/mpathcp1  466G  6.5G  459G   2% /datos/db
```

Figure 33) NFS VM server SolidFire volume details.

```
From /etc/fstab
```

```
UUID="5df552c6-f2d2-4404-abc9-021161b3e7a0"     /datos/rb       xfs
noatime,discard,nobarrier,_netdev    0       0

From exports file
[root@datos-nfs-1 ~]# cat /etc/exports
#Share access to all networks
/datos/rb       *(rw,sync)
```

In Figure 33, the "/datos/rb" mounted directory is a single SolidFire volume configured by NFS on each Datos RecoverX cluster node. The RecoverX node uses the commit logs from the Cassandra cluster nodes to keep the backed-up database versions up to date per the policy schedules.

Figure 34) Cassandra node SolidFire volume mounted file systems.

```
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/vg_mongodatosr1-lv_root
                       44G  7.9G   36G  19% /
tmpfs                  32G   72K   32G   1% /dev/shm
/dev/sda1             477M   80M  373M  18% /boot
datos-nfs-1:/datos/rb
                      931G   43G  889G   5% /mnt/versions
/dev/mapper/mpathcp1  466G  6.5G  459G   2% /datos/db
```

In Figure 34, the "/datos/rb" mounted directory is a single SolidFire volume configured by NFS on each Cassandra cluster node to contain the database and the commit logs. The RecoverX node uses these commit logs to keep the backed-up database versions up to date per the policy schedules.

Figure 35) Cassandra node NFS mount fstab details.

```
datos-nfs-1:/datos/rb /mnt/versions    nfs
auto,hard,actimeo=0,lookupcache=none,noac,nfsvers=3      0       0
```

Figure 35 shows the /etc/fstab entry for the SolidFire NFS mounted volume on each Cassandra node. The NFS version 3 parameter is used for compatibility with Datos RecoverX version 1.1, used in this test.

RecoverX and Cassandra Node Checklists

Checklist for RecoverX Nodes

- RecoverX nodes meet hardware and software requirements that are listed.
- Datos IO user on RecoverX nodes has been set up with the same GID as the Datos IO database user.
- Datos IO user home directory is on a nonroot volume with at least 140GB.
- Datos IO user has sudo privileges to required commands.
- Requisite network ports that are specified in the installation guide are open.
- Maximum user processes are set to unlimited.
- The nofile limit is set to 64,000.
- The short name and FQDN of your RecoverX server and source/target nodes are referenced in file /etc/hosts.
- Check for Python version 2.6.
- Check for GNU AWK.
- /tmp directory on each node has at least 2GB empty space.
- Check that passwordless SSH is configured for all RecoverX nodes.

Checklist for Cassandra Data Nodes

- Datos IO database user on data source nodes has been set up with the same GID as the Cassandra/Mongo user.
- Datos IO database user on every data source node has write permission to its home directory.
- Datos IO database user on every Cassandra node has read and execute (r+x) permissions to the $CASSANDRA data directory and its parent directory on every Cassandra node.
- Max SSH sessions is 500 on every data source node.
- Auto_bootstrap=True on every Cassandra node.
- System time is synced up properly among data source nodes.
- Requisite network ports that are specified in the installation guide are open.
- Check the Cassandra version. Apache Cassandra v2.1.0 is not supported.
- Java is installed on all nodes and in $PATH of Datos IO database user.
- Either Murmur3Partitioner or RandomPartitioner is configured for Cassandra nodes.

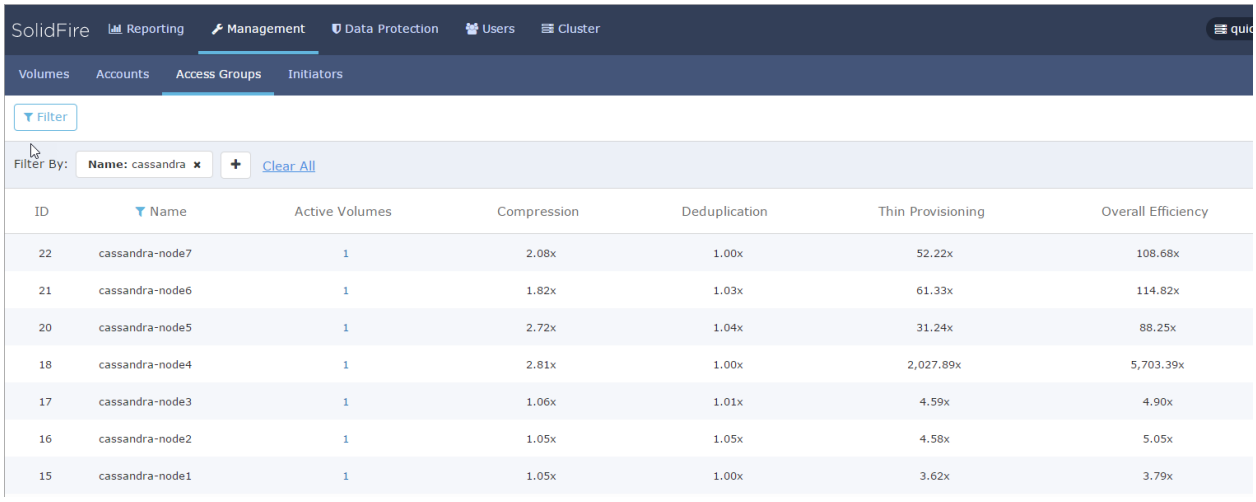
Checklist for NFS Version Store

- NFS store is hard-mounted on all Cassandra and RecoverX nodes.
- NFS ports mentioned in installation guide are open.
- All Cassandra and RecoverX nodes have read/write access to the NFS store.
- NFS protocol in use is v3.
- NFS store is mounted without caching option (actimeo=0,lookupcache=none,noac) on all Cassandra and RecoverX nodes.

SolidFire Configuration

Figure 36, the SolidFire Element GUI screen, shows the access groups configured for the Cassandra data nodes and the Datos IO RecoverX node. Each access group has one initiator to a host and one active volume.

Figure 36) SolidFire access groups for Cassandra nodes.



ID	Name	Active Volumes	Compression	Deduplication	Thin Provisioning	Overall Efficiency
22	cassandra-node7	1	2.08x	1.00x	52.22x	108.68x
21	cassandra-node6	1	1.82x	1.03x	61.33x	114.82x
20	cassandra-node5	1	2.72x	1.04x	31.24x	88.25x
18	cassandra-node4	1	2.81x	1.00x	2,027.89x	5,703.39x
17	cassandra-node3	1	1.06x	1.01x	4.59x	4.90x
16	cassandra-node2	1	1.05x	1.05x	4.58x	5.05x
15	cassandra-node1	1	1.05x	1.00x	3.62x	3.79x

Figure 37 shows the NFS server SolidFire volume. This volume is an NFS share and is accessible for all Cassandra data nodes and RecoverX cluster nodes.

Figure 37) RecoverX and Cassandra SolidFire volume.

ID	Name	Account	Access Groups	Access	Used	Size	Snapshots	Min IOPS	Max IOPS	Burst IOPS	Attributes	512e
596	datos-nfs-1	datos-mon...	datos-nfs-1	Read / Write	4.62%	1.0 TB	0	50	100,000	100,000	-	Yes

Figure 38 shows the SolidFire volumes for a Cassandra cluster. All Cassandra nodes in this test were configured on SolidFire with the same paramters.

Figure 38) Cassandra node SolidFire volume.

ID	Name	Account	Access Groups	Access	Used	Size	Snapshots	Min IOPS	Max IOPS	Burst IOPS	Attributes	512e
599	cassandra-node7	datos-mongo	cassandra-node7	Read / Write	1.56%	500.0 GB	0	50	100,000	100,000	-	Yes
598	cassandra-node6	datos-mongo	cassandra-node6	Read / Write	1.63%	500.0 GB	0	50	100,000	100,000	-	Yes
597	cassandra-node5	datos-mongo	cassandra-node5	Read / Write	1.43%	500.0 GB	0	50	100,000	100,000	-	Yes

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.