Technical Report

# NetApp In-Place Analytics Module

## Best Practices

Karthikeyan Nagalingam, NetApp
June 2018 | TR-4382


## Abstract

This document introduces the NetApp® In-Place Analytics Module for Apache Hadoop. This module enables open-source analytics frameworks such as Hadoop to run on NFS storage natively. The topics covered in this report include the configuration, underlying architecture, primary use cases, integration with Hadoop, and benefits of using Hadoop with NetApp ONTAP® data management software. The NetApp In-Place Analytics Module allows analytics to run NetApp FAS and AFF with ONTAP software. It is easy to install and works with Apache Hadoop, Apache Spark, Tachyon, Apache HBase, and major Hadoop distributors, enabling data on NFSv3 to be analyzed.

TABLE OF CONTENTS



## LIST OF TABLES



## LIST OF FIGURES



# 1 Introduction

Big data is defined broadly as large datasets with a mix of structured, unstructured, and semistructured data, which cannot be processed by traditional database systems. Businesses have turned to big data frameworks that support analytical tools such as Apache Hadoop to help store and analyze these datasets. Apache Hadoop software is a framework that enables the distributed processing of large and varied datasets, across clusters of computers, by using programming models. The Hadoop Distributed File System (HDFS) provides high throughput of application data. Hadoop provides integration to enhance specific workloads, storage efficiency, and data management.

Historically, Hadoop has been used primarily on incoming external data. However, there's been a need to use Hadoop on existing internal data, which is typically stored in network-attached storage (NAS). This use case requires setting up another storage silo to host the HDFS and then running the Hadoop analytics on that storage. This process results in additional data management, reduced efficiency, and increased costs for moving the data between NAS and HDFS.

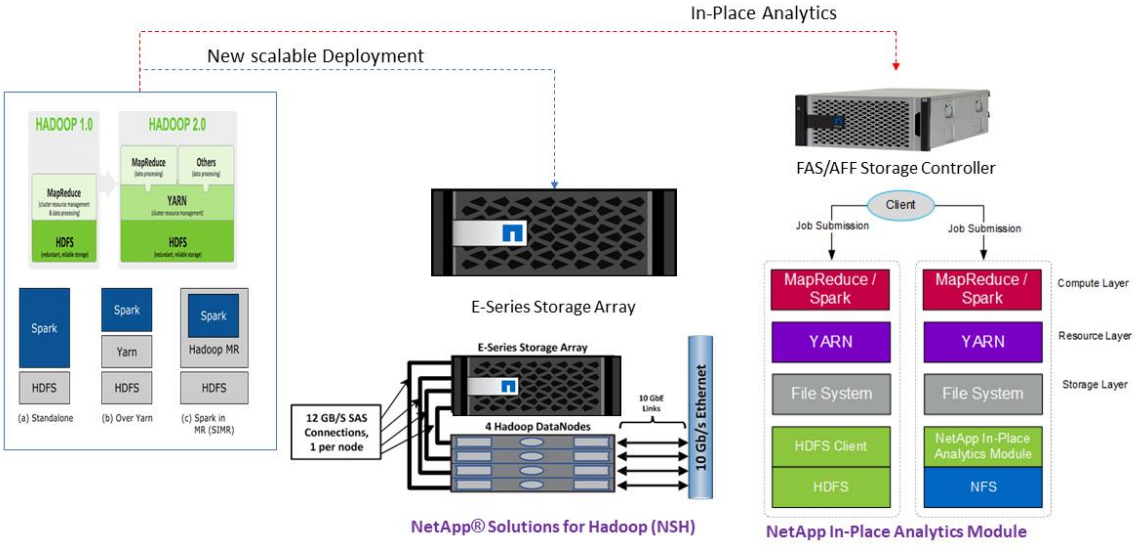
In this document, NetApp introduces the NetApp In-Place Analytics Module. This module enables analytics software such as Apache Hadoop and Apache Spark to access and analyze data by using NetApp ONTAP data management software and the NFS protocol. By using ONTAP software, the NetApp In-Place Analytics Module decouples analytics from storage, leveraging the benefits of NAS to share data.

This report also covers installation, the underlying architecture, use cases, integration with Hadoop, sizing information, benefits of using analytics with ONTAP software, Hortonworks certification, MetroCluster™ validation, and solutions for error messages.

## 1.1 Overview

NetApp provides the NetApp E-Series and FAS arrays for scalable Hadoop deployments. Both are complementary and targeted at different use cases, as highlighted in Figure 1. This document focuses on the NetApp In-Place Analytics Module. For detailed information about the E-Series solution, see TR-3969: NetApp Open Solution for Hadoop Solutions.

**Figure 1) NetApp In-Place Analytics Module.**




As mentioned, the NetApp In-Place Analytics Module enables analytics software to run on the data stored in ONTAP. It works with Apache Hadoop and Apache Spark by using a simple configuration file change

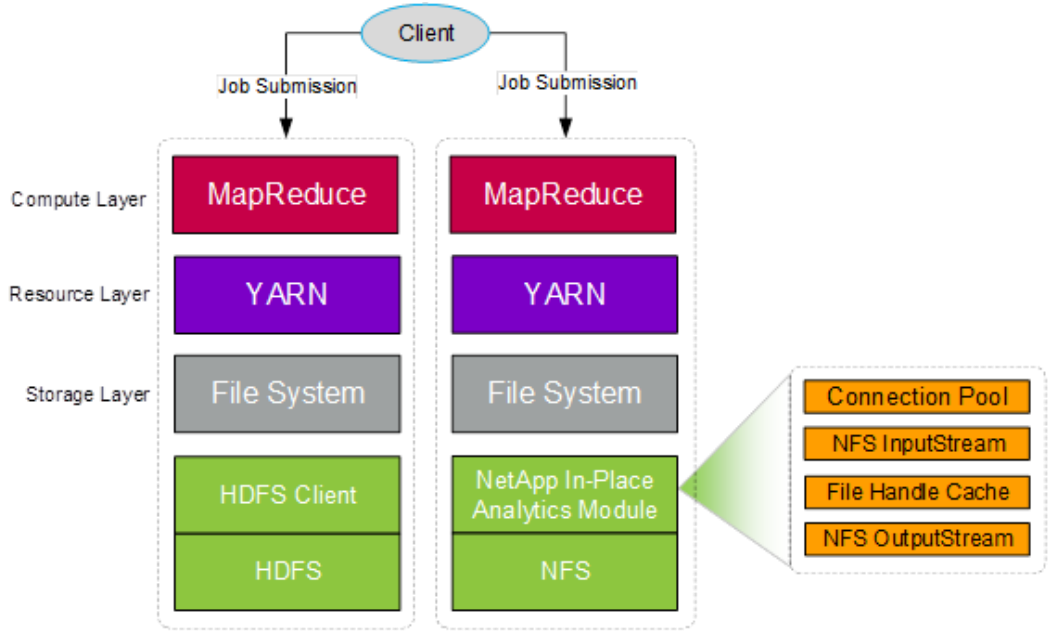
 

that enables data on NFSv3 storage to be analyzed. By using ONTAP, the NetApp In-Place Analytics Modules decouples analytics from storage, leveraging the benefits of NAS. ONTAP data management software enables high availability and storage efficiency. For an even better performance, the NetApp In-Place Analytics Module can be combined with Tachyon to build a scale-out caching tier that is backed by ONTAP software.

The NetApp In-Place Analytics Module is an implementation of the file system interface defined by Apache Hadoop, and it's based on NFSv3. Currently it supports `AUTH_NONE` and `AUTH_SYS` security policies, which are UNIX authentication mechanisms.

Figure 2 shows how the NetApp In-Place Analytics Module plugs into Apache Hadoop. This design has two deployment options, which are covered in section Deployment Options2.2, "Deployment Options."

**Figure 2) NetApp In-Place Analytics Module plugs into Apache Hadoop.**




## 1.2 NetApp In-Place Analytics Module 3.0 Features

The NetApp In-Place Analytics Module version 3.0 has the following features:

- **Performance enhancements.** Network load distribution across multiple network interfaces.
- **Hortonworks certification.** Hortonworks certification for the HDP platform.
- **Apache Ambari Module for Management.** Manage installation and configuration from the Ambari user interface.
- **Simplicity enhancements:**
  - User and access management
  - Configuration management
- Azure integration with HDInsight application. Run Azure HDInsight cluster on NFS data.

# 2 Hadoop in the Enterprise

The use of big data analytics is becoming more popular, but running analytics in an enterprise environment still faces many challenges. Large enterprises have existing hardware and data, and

traditional analytics platforms such as Hadoop cannot be easily installed in those environments. Running analytics in an enterprise environment faces four main challenges:

- **Enterprises have a storage and compute imbalance.** Some environments store large amounts of data but might not analyze all of it continuously; other environments might have smaller amounts of data and analyze it continuously. In a traditional architecture, design, compute, and storage are tightly linked. With a decoupled design, both the compute and storage tiers can be scaled independently.
- **Enterprises have existing hardware.** Many enterprise environments have a shared infrastructure that is used for home directories, databases, and many other services. In the traditional design, the enterprise must dedicate new hardware and storage for data analytics. In addition, the data must be moved from enterprise storage into siloed analytics storage before it can be used: a costly proposition. In contrast, a decoupled design uses existing storage hardware and runs analytics on data in place.
  - A decoupled design is a framework for complex work that allows components to remain completely autonomous and scale independently of each other.
  - Sqoop-like tools are required to extract data from databases such as a relational database management system (RDBMS) and store it in a format that can be processed in the Hadoop framework.
- **Analytics storage JBOD is not efficient.** The storage used for analytics (for example, HDFS) typically utilizes three copies of data for reliability and for performance. This method is not storage efficient. An enterprise storage system provides reliability by using efficient approaches such as NetApp RAID DP® technology.
- **Analytics storage JBOD lacks data management.** File systems used by analytics frameworks lack enterprise features such as deduplication, high availability, and disaster recovery. The NetApp In-Place Analytics Module leverages ONTAP nondisruptive operations and provides the following benefits:
  - Adds storage on demand that is independent of the data nodes
  - Facilitates the scalable distributed processing of datasets stored in existing enterprise storage systems
  - Provides access to the analytics file system in the current analytics framework, while simultaneously enabling data analytics on NFS storage
  - Leverages NetApp data management features such as Snapshot™ technology, FlexClone® technology, and data protection
  - Simplifies job workflows and negates the need to copy data across different storage silos

## 2.1 Benefits and Use Cases

The NetApp In-Place Analytics Module allows computations to analyze data stored on enterprise storage systems. The decoupled design of the NetApp In-Place Analytics Module provides high functional value in the following scenarios:

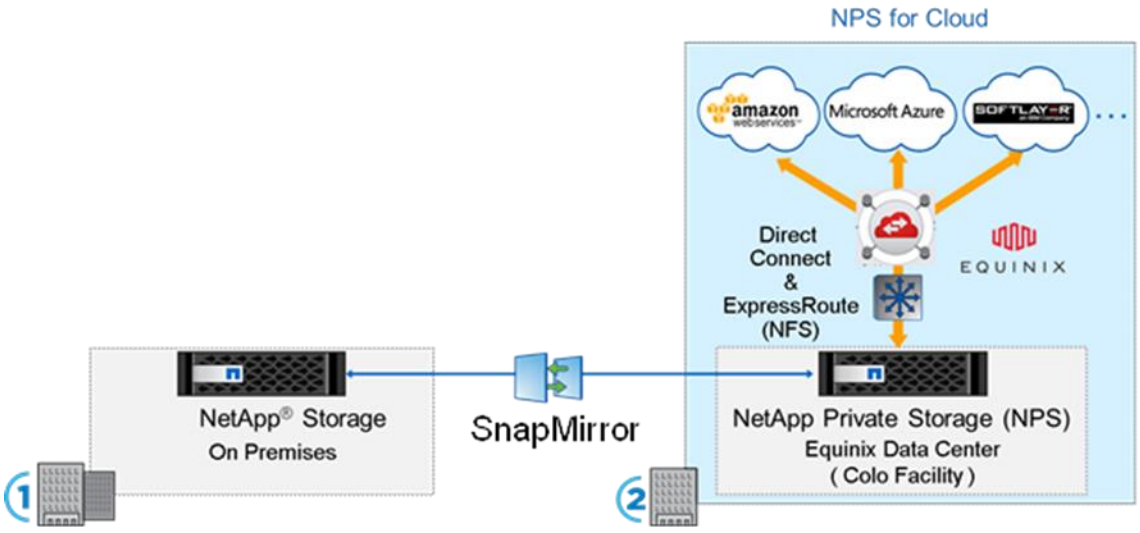
- **Analyzing data on enterprise storage.** Companies can leverage their existing investment in enterprise storage and enable analytics incrementally. Many file-based data sources exist, such as source-code repositories, emails, and log files. These files are generated by traditional applications but currently require a workflow to ingest the data into a separate analytics file system. The NetApp In-Place Analytics Module allows a single storage back end to manage and service data for both enterprise and analytics workloads. Data analytics that use the same file system namespace can analyze enterprise data with no additional ingest workflows.
- **Cross data center deployments.** The decoupled design of the NetApp In-Place Analytics Module also allows independent scaling of compute and storage layers. As shown in Figure 3, this feature gives the flexibility of placing the analytics compute tier on cloud infrastructures, such as Amazon EC2, while keeping the data on the premises. In this scenario, up-front hardware purchases are replaced by the pay-as-you-go cloud model. Cross–data center deployments benefit from products

such as Amazon Web Services (AWS) Direct Connect and NetApp Private Storage (NPS) that enable high-bandwidth connections between private data centers and public clouds.

- NPS enables enterprise customers to leverage the performance, availability, security, and compliance of NetApp storage with the economics, elasticity, and time-to-market benefits of the public cloud.

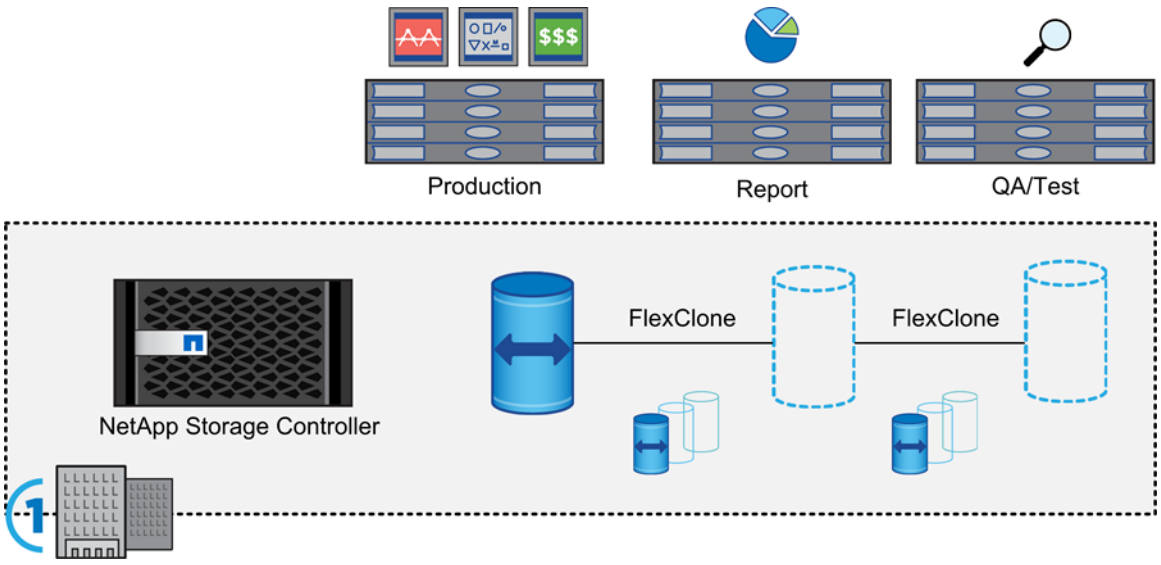
**Figure 3) Cross–data center deployments.**




The NetApp In-Place Analytics Module is optimal for the following use cases:

- **Analyze data on existing NFS storage.** The NetApp In-Place Analytics Module enables analytics on existing workflows and applications that write files to NFS, code repositories on NFS, and data in NetApp SnapLock® volumes.
- **Build testing and QA environments by using clones of existing data.** As shown in Figure 4, the NetApp In-Place Analytics Module enables developers to run variations of analytics code on shared datasets. If the production data is on a NetApp FlexVol® volume, then the volume can simply be cloned and used for testing.

**Figure 4) Duplicate Hadoop cluster using FlexClone technology.**



- **Leverage storage-level caching for iterative machine learning algorithms.** The iterative machine learning algorithms are cache friendly and compute intensive. The NetApp In-Place Analytics Module can leverage storage caches such as NetApp Flash Cache™ caches for acceleration.
- **Use a backup site for analytics.** When data is stored near the cloud (NPS), the NetApp In-Place Analytics Module allows the use of cloud resources such as Amazon EC2 or Microsoft Azure for analytics, while managing data with ONTAP software.

**Additional Information**

For a detailed list of data protection use cases, see TR-4657: NetApp Hybrid Data Protection Solutions for Hadoop and Spark: Customer Use Case-Based Solutions.

## 2.2 Deployment Options

The NetApp In-Place Analytics Module allows users to run one of two different deployment options:

- Run the HDFS as the primary file system and use the NetApp In-Place Analytics Module to analyze data on the NFS storage systems.
- Deploy NFS as the primary storage or default file system.

The appropriate deployment option is based on the use cases and applications used in Apache Hadoop and Apache Spark.

**Deploy NFS as Primary Storage or Default System**

Even though some customers are running this option from the CLI, NetApp does not currently support this option in the NetApp In-Place Analytics Module 3.0 release. However, NetApp is actively working with Hadoop distributors to support this option in future versions.

The NetApp In-Place Analytics Module allows analytics to use existing technologies such as Snapshot, RAID DP, NetApp SnapMirror® data replication, storage efficiency, and FlexClone.

## 2.3 Ease of Deployment

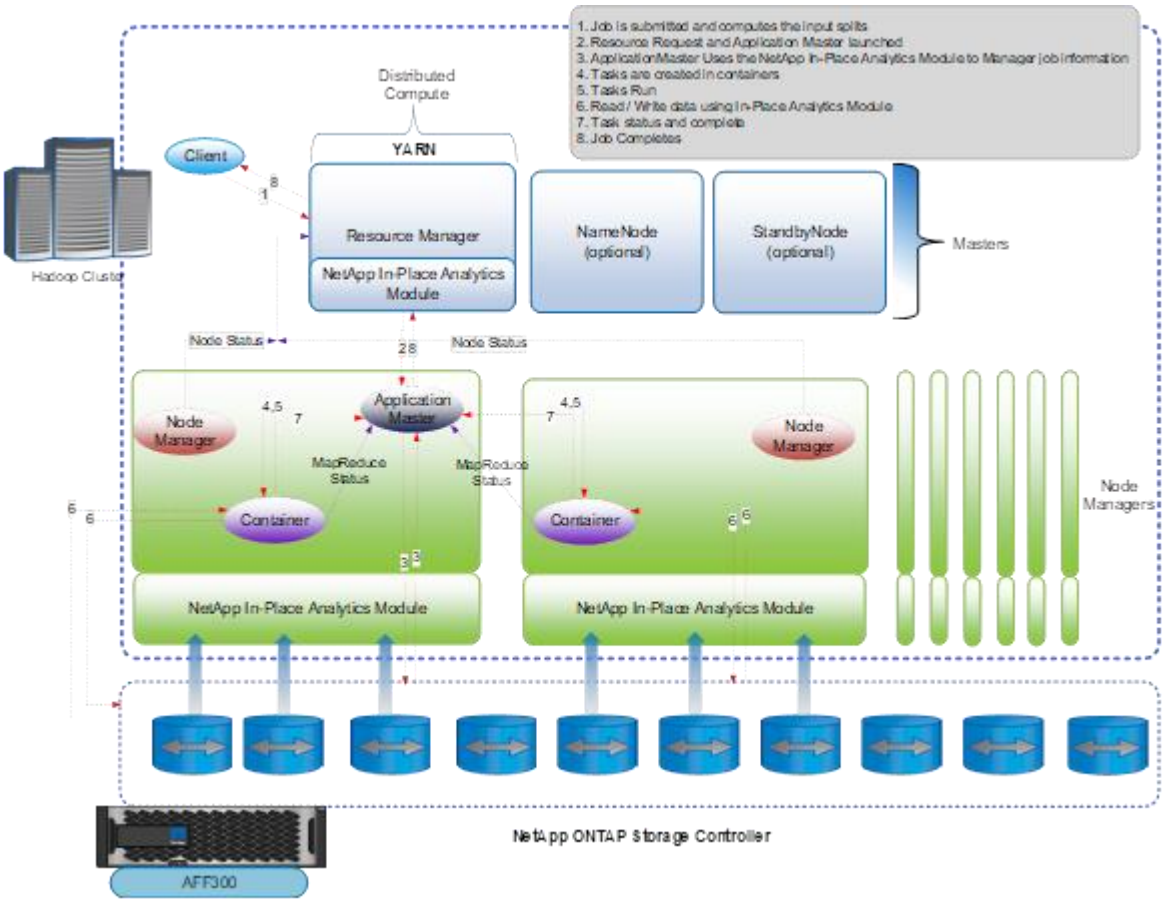
Installing the NetApp In-Place Analytics Module is simple. For Apache Hadoop, install the NetApp In-Place Analytics Module Java Archive (JAR) file and modify the `core-site.xml` file. A similar change is needed for Apache HBase: the `hbase-site.xml` file must be changed. After this modification, applications that use HDFS as their storage system can simply use NFS.

# 3 NetApp In-Place Analytics Module: Architecture and Design

## 3.1 High-Level Architecture

Figure 5 shows a high-level architecture for the NetApp In-Place Analytics Module and the application execution sequence.

**Figure 5) High-level architecture of NetApp In-Place Analytics Module with application execution sequence.**




The high-level architecture of the NetApp In-Place Analytics Module can be explained through a client's application execution sequence:

1. The client program submits the application (called a MapReduce job). This application includes the necessary specifications to launch the application-specific ApplicationMaster. The program first computes the input splits. Then the ApplicationMaster coordinates and manages the lifetime of the job execution.
2. The ResourceManager assumes the responsibility to negotiate a container in which to start the ApplicationMaster and then launches the ApplicationMaster.
3. The ApplicationMaster uses the NetApp In-Place Analytics Module to manage job information, such as status and logs. The ApplicationMaster requests containers for either its map or reduce tasks.
4. For each input split, the ApplicationMaster requests a container for the task analyzing the split and initiates the task in the newly created container.
5. The task, either map or reduce, runs in the container.
6. By using the NetApp In-Place Analytics Module, the task reads and/or writes data stored on NFS. As the task executes, its progress and status are updated periodically to the ApplicationMaster.
7. After the task is complete, it updates its completion status with the ApplicationMaster and exits. The container used by the task is given back to the ResourceManager.
8. After all tasks are complete, the ApplicationMaster updates various statistics and finishes the job. The client is notified of job completion.

## 3.2 Technical Advantages

The NetApp In-Place Analytics Module has the following technical advantages:

- It works with Apache Hadoop, Spark, HBase, Hive, Impala, and Mahout. It also works with Tachyon.
- No changes are needed to existing applications.
- No changes are needed to existing deployments; only the configuration files (`core-site.xml`, `hbase-site.xml`, and so on) are modified.
- Data storage can be modified and upgraded nondestructively by using ONTAP software.
- It supports the latest networks (10/40GbE) and multiple NFS connections.
- The connector enables high availability and nondisruptive operations by using ONTAP software.

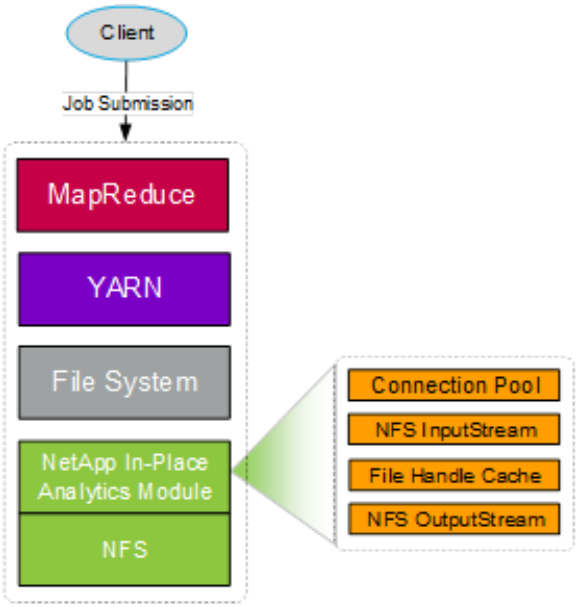
## 3.3 Design

Figure 6 shows the four main components of the NetApp In-Place Analytics Module:

- Connection pool
- NFS InputStream
- File handle cache
- NFS OutputStream

The other minor component is authentication.

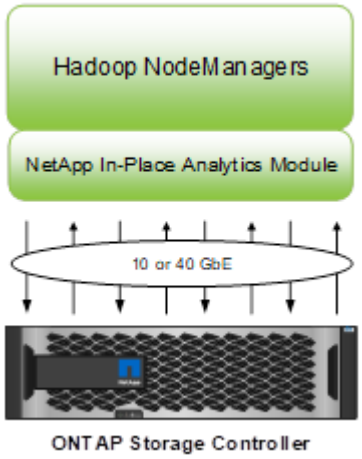
**Figure 6) Components of the NetApp In-Place Analytics Module.**




### Connection Pool

When the NetApp In-Place Analytics Module is loaded by Apache Hadoop, it creates a connection pool consisting of several connections to the NFS server. The size of the pool is dynamic. Later, all NFS traffic uses one of the connections available from the pool and multiplexes among them. This approach allows the NetApp In-Place Analytics Module to take advantage of high-speed (10/40GbE) networking and utilize aggregate bandwidth.

**Figure 7) Connection pool.**



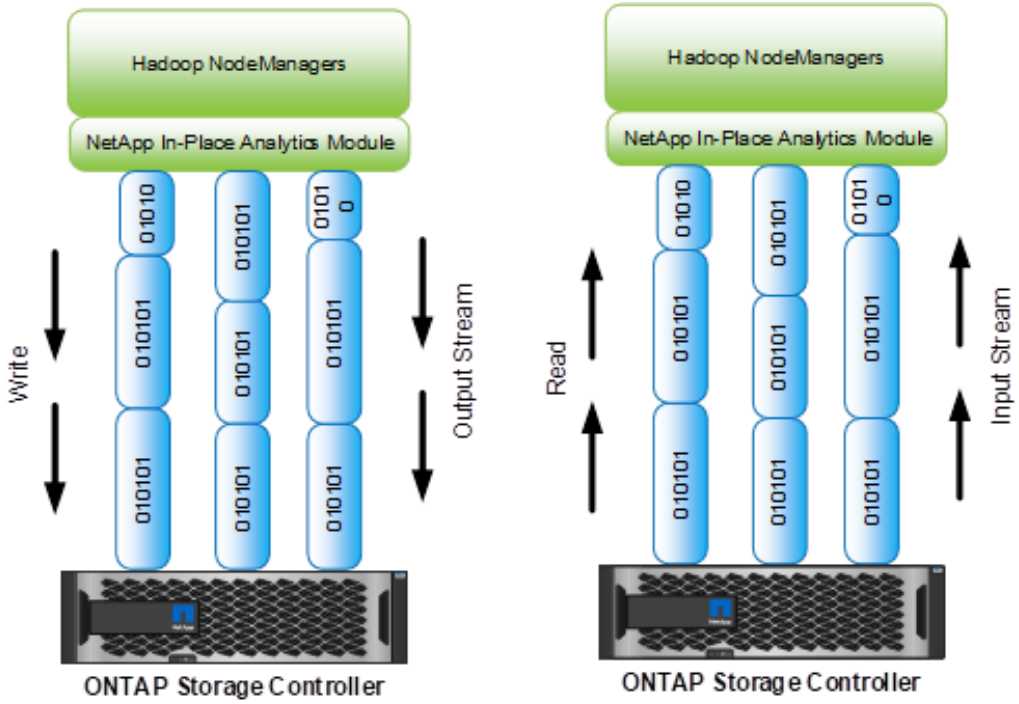

## NFS InputStream

InputStream is the method used by Apache Hadoop to read data from files. The NetApp In-Place Analytics Module is optimized to take full advantage of Hadoop and the underlying network and file system in the following ways:

- **Large sequential reads.** Applications use InputStream to read data. They read it in bytes required by the application, ranging from a single byte to several kilobytes of data. However, this method is not optimal for NFS. The connector modifies the I/O size issued to NFS to optimize for the underlying network. The default read value is 1MB, but it is configurable by setting the `nfsReadSizeBits` option. If the block size is larger than the maximum read request that the NFS server can support, then the NetApp In-Place Analytics Module automatically switches to using the smaller of the two.
- **Multiple outstanding I/Os.** The NetApp In-Place Analytics Module uses a temporary cache and issues multiple I/O requests in parallel. This method allows the amortization of the I/O time and enables the system to prefetch aggressively.
- **Prefetching.** Prefetching is used to improve the performance of streaming reads. When an on-demand read request is received, prefetching for the next 128 blocks is issued. To avoid unnecessary prefetching for Hadoop jobs, a heuristic is implemented. When a seek request is received, it sets the last block to prefetch, based on the offset of the seek request and the split size. The connector stops prefetching when it reaches that block. It never prefetches beyond the boundary of a file split. In the other case, in which the last block to prefetch is not set, the NetApp In-Place Analytics Module simply continues prefetching. The split size is configurable with the `nfsSplitSizeBits` option.

Figure 8) NFS input and output streams.



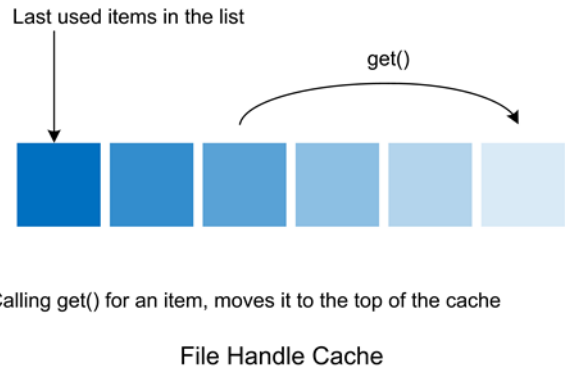

## File Handle Cache

A least recently used (LRU) cache is implemented to cache recently used file handles. This approach lowers the need to issue frequent lookup operations. It works as follows:

1. To get a file handle for a path, the file handle cache is checked.
2. If a handle is found in the cache, a lookup request is sent to the NFS server to check whether the file handle is valid or stale.
3. The handle is returned if it is valid. Otherwise, the same procedure is called to get a valid handle for the parent directory.
   - This process is recursive, and it stops either when a valid handle is found for one of the ancestor directories or when the mount root directory is reached.
4. A lookup request for the file or directory in that parent directory is called.

The process repeats until it reaches the desired path.

Figure 9) LRU cache.



### NFS OutputStream

Similar to InputStream, Apache Hadoop uses OutputStream to write data to files. You can use similar optimizations such as batching writes for large I/Os and taking advantage of Hadoop's consistency semantics:

- A write buffer is maintained in the NetApp In-Place Analytics Module to store write requests. When the buffer becomes full, requests are sent to the NFS server. The size of the write buffer is configurable with the `nfsWriteSizeBits` option.
- The default mode used in write requests sent to the NFS server is nonoptimal because it requires the NFS server to make each write request durable in disk. Instead, the NetApp In-Place Analytics Module sends each write request as nondurable to the NFS server. A commit request is sent to the NFS server to flush all write requests only when the output stream is closed. This approach is one of the optimizations introduced specifically for running Hadoop jobs. There is no need to flush data to disk unless a task succeeds. Failed tasks are automatically restarted by Hadoop.

### Authentication

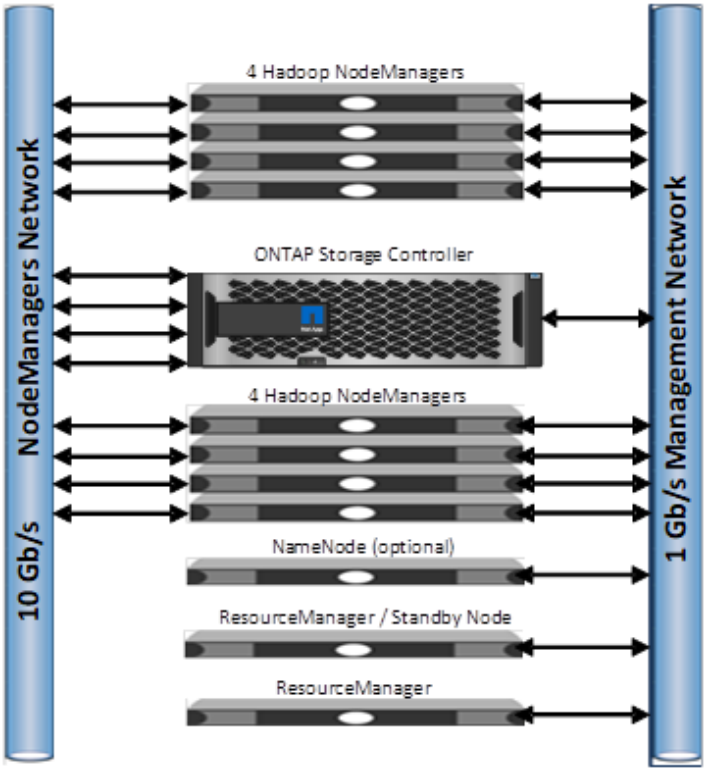
Currently, NetApp supports two types of authentication: none and UNIX. Authentication is configurable with the `nfsAuthScheme` option. NetApp is in the process of adding tighter integration with other authentication schemes such as NIS, Kerberos, and Ranger.

# 4 Solution Architecture

## 4.1 Network Architecture

Figure 10 shows the connectivity between the servers and the FAS/AFF storage controller.

**Figure 10) Network connectivity.**



### Recommended Network Topology

Apache Hadoop with the NetApp In-Place Analytics Module has the following network topology:

- NodeManagers (two or more 10/40GbE ports):
  - Primary purpose: data ingest and movement in a cluster
  - Private interconnection between all NodeManagers
  - Dedicated nonroutable VLAN
- Management network (GbE):
  - Purpose: system administration network
  - Publicly routable subnet
  - NodeManagers and ResourceManager (in Hadoop 2.0) also use GbE
  - GbE interface for administration and data transfer purposes on all nodes
  - Two GbE ports required by FAS/AFF storage systems for management purposes only

## 4.2 Storage Architecture

NetApp FlexVol flexible volume technology decouples the physical connection between data containers and their associated physical disks. The result is a significant increase in flexibility and storage utilization. This technology can shrink or grow data containers based on immediate needs. Adding disks or changing the size of data containers can be done as needed, without disrupting the system and associated applications.

- On an ONTAP system, NAS clients access flexible volumes through a storage virtual machine (SVM). SVMs abstract storage services from their underlying hardware.
- Flexible volumes containing NAS data are junctioned into the owning SVM in a hierarchy. This hierarchy presents NAS clients with a unified view of the storage, regardless of the physical location of the flexible volumes inside the cluster.
- When a flexible volume is created in the SVM, the administrator specifies the junction path for the flexible volume. The junction path is a directory location under the root of the SVM where the flexible volume can be accessed. A flexible volume's name and junction path do not need to be the same.
- Junction paths allow each flexible volume to be browsable: for example, a directory or folder. NFS clients can access multiple flexible volumes using a single mount point. CIFS clients can access multiple flexible volumes using a single CIFS share.
- A namespace consists of a group of volumes connected using junction paths. It is the hierarchy of flexible volumes in a single SVM as presented to NAS clients.
- The storage architecture can consist of one or more FAS/AFF storage controllers. Figure 11 shows a single NetApp FAS/AFF controller, with a volume mounted under its default or own namespace.

**Figure 11) NFS volume from a single NetApp FAS/AFF storage controller.**

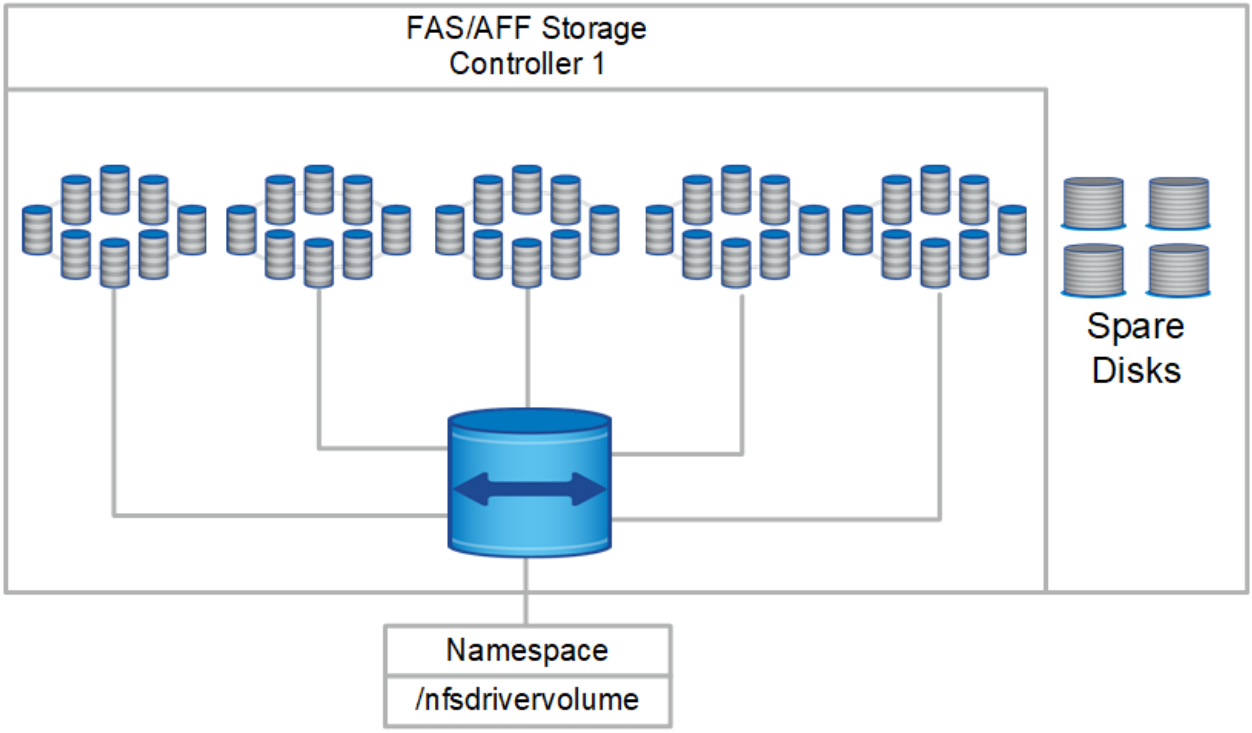
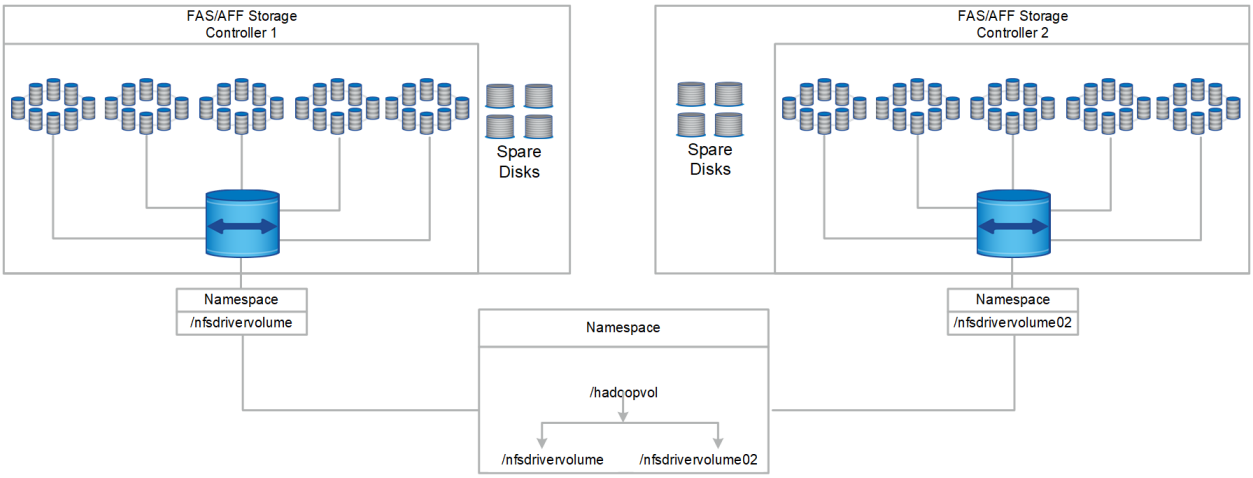



Figure 12 shows a FAS/AFF controller, in which each volume from each controller is mounted under one common namespace. The common namespace is a dummy volume that can be from any one of the controllers, and it's configured as `nfsExportPath` in `core-site.xml` in the Hadoop cluster.

| Best Practice |
| --- |
| NetApp recommends creating a volume with RAID DP, adding more disks for better performance, and keeping 2 disks as global hot spares for up to 100 disk drives of the same type. |

**Figure 12) NFS volumes from two FAS/AFF storage controllers.**



In a dual-controller setup, you can distribute the load among the controllers during job execution. For example, you can store TeraGen map job results and perform a TeraSort map operation on one volume, followed by a reduce operation, based on TeraGen results, on another volume. You can also perform a TeraGen operation on one volume while a TeraSort operation is running on another volume.

| Best Practice |
| --- |
| NetApp recommends having an application cache (`usercache`) in a NetApp storage volume and creating one volume per NodeManager. |

## 4.3 Key Components of Validated Configuration

Table 1 lists the products used for the validation of the NetApp In-Place Analytics Module. You can configure the storage based on the NetApp sizer recommendations for your bandwidth (in Mbps), IOPS, and capacity requirements.

**Table 1) Components used for the NetApp In-Place Analytics Module tests.**

| Component | Product or Solution | Details |
| --- | --- | --- |
| Storage | NetApp A300 storage array with ONTAP 9.3 | • 2 controllers (HA pair) with 24 x 900GB, solid-state drives (SSDs)<br>• 1 hot spare per disk shelf<br>• 1 data aggregate (23 drives, shared to both controllers) per controller<br>• NetApp ONTAP FlexGroup and some test cases with 2 aggregates and 2 volumes<br>• 2 x 4 x 10GbE<br>• RAID DP |
| Servers | PRIMERGY RX2540 M1<br>Intel Xeon CPU E5-2670 v3 at 2.30GHz | • 10 servers, and each has two 2.4GHz (6-core) or 2.3GHz (8-core) processors (40 cores)<br>• 12 DIMMs (up to 192GB), up to 1600MHz (512GB/node)<br>• 1 x 1Gbps Ethernet port, 2 x 10GbE network ports |
| Networking | • 10GbE nonblocking network switch<br>• 1GbE network switch | A Cisco Nexus 5000 switch was used for testing. Any compatible 10GbE network switch can also be used. |
| Server operating system | Red Hat Enterprise Linux Server 7.4 (x86_64) or later | Hadoop typically requires a Linux distribution. |
| Hadoop distribution used in the testing | Cloudera Distribution for Hadoop | Cloudera Manager 5.14 |
| | Hortonworks Data Platform 2.6.5 (tested) | Apache Ambari 2.6 |

# 5 Installation and Configuration

The NetApp In-Place Analytics Module installation is simple and consists of three parts:

1. Configure the FAS/AFF storage controller.

2. Configure the Hadoop cluster.
3. Create the JavaScript Object Notation (JSON) configuration file.

The first step is to download the NetApp In-Place Analytics Module software and installation guide from the NetApp website.

## 5.1 Configure FAS/AFF Storage Controller

Complete the following steps to configure the FAS/AFF storage controller:

1. Create an SVM with NFS access and then disable both `nfs-rootonly` and `mount-rootonly`.
2. In the SVM (also known as Vserver), change the `guid` from 1 to 0 by using the `unix-user modify` command.
3. In the SVM, check that the `export-policy` rule has access (`ip range`) for the Hadoop worker nodes and that the `superuser security` type is set to `sys`.
4. Create a volume and one or more logical network interfaces for the data role.
5. Change the NFS read and write size to 1MB by using `tcp-max-xfer-size` or `v3-tcp-max-write/read-size`.

## 5.2 Configure Hadoop Cluster

Complete the following steps to configure the Hadoop cluster:

1. For Apache Hadoop and Spark-based distributions such as Hortonworks, Cloudera, and MapR, copy the `hadoop-nfs-connector-3.x.x.jar` file to `hadoop classpath` and replace the `hadoop-nfs-<version>.jar` file with the `hadoop-nfs-2.7.1.jar` file provided by NetApp.
2. For Hortonworks, NetApp recommends using the NetApp Ambari UI Service plug-in.

## 5.3 Create JSON Configuration File

The NetApp In-Place Analytics Module requires the creation of a configuration file, which is then distributed to all the nodes on your Hadoop cluster. The location of the file can be anywhere on your Hadoop node as long as it is accessible to all the Hadoop users in the system. See the Installation Guide for a detailed explanation of the JSON file and its parameters.

## 5.4 Modify Hadoop Configuration

Modify the `core-site.xml`, `hbase-site.xml`, and `hive-site.xml` files for Hadoop, Spark, Hive, and Impala. For Cloudera, update the parameters in Table 2 by using `snippet(safety value)` and either use the `custom xml` option or override the existing parameters for Hortonworks.

**Table 2) Parameters for Hadoop configuration.**

| Parameter | Value | Description |
|---|---|---|
| `fs.nfs.prefetch` | `True` | Enables prefetch for the InputStream. |
| `fs.defaultFS` | `nfs://192.168.120.207:2049` | Name of the default file system specified as a URL (this configuration works for some of the Hadoop ecosystems). Some Hadoop distributor management frameworks such as Ambari and Cloudera Manager are designed for HDFS. The current version of the NetApp In-Place Analytics Module does not support this option. |

| Parameter | Value | Description |
|---|---|---|
| `fs.AbstractFileSystem.nfs.impl` | `org.apache.hadoop.netapp.fs.nfs.NFSv3AbstractFilesystem` | Allows versions of Hadoop 2.0 and later to find the connectivity for NFS. |
| `fs.nfs.impl` | `org.apache.hadoop.netapp.fs.nfs.NFSv3FileSystem` | Allows Hadoop to find the NetApp NAS NFS connector. |
| `fs.nfs.configuration` | `/etc/NetApp/conf/nfs-mapping.json` | Defines the cluster architecture for the NetApp In-Place Analytics Module. Make sure the JSON file exists before restarting the required services in Hadoop. |

| Best Practices |
|---|
| • Create separate networks for your Hadoop jobs between the NetApp NFS volumes and NodeManager for improved network bandwidth.<br>• Spread the NetApp NFS volumes across the controllers equally to distribute the load or create FlexGroup volumes. |

## Example Configuration File (Version 3.x)

A configuration file for the `nfs-mapping.json` is shown in the following example:

```
[root@node1 ~]# cat /etc/NetAppNFSConnector/conf/nfs-mapping.json
{
    "spaces": [
        {
            "endpoints": [
               {
                    "exportpath": "/iam_volume2",
                    "hosts": [
                        "nfs://10.63.150.213:2049/",
                        "nfs://10.63.150.127:2049/",
                        "nfs://10.63.150.128:2049/",
                        "nfs://10.63.150.211:2049/"
                    ],
                    "path": "/iam_volume2"
                }
            ],
            "name": "iam_validation",
            "options": {
                "nfsAuthScheme": "AUTH_SYS",
                "nfsExportPath": "/iam_volume2",
                "nfsMountPort": -1,
                "nfsPort": 2049,
                "nfsReadSizeBits": 20,
                "nfsRpcbindPort": 111,
                "nfsSplitSizeBits": 30,
                "nfsWriteSizeBits": 20
            },
            "uri": "nfs://10.63.150.213:2049/"
        }
    ]
}
```

## 5.5 Verification

To verify the installation and configuration of the NetApp In-Place Analytics Module, complete the following steps:

1. Use `copyFromLocal` to test the NetApp In-Place Analytics Module.

```
[root@node1 ~]#hadoop fs -ls  nfs://10.63.150.213:2049/
Found 4 items
drwxrwxrwx   - root root       4096 2018-05-11 04:31 nfs://10.63.150.213:2049/.snapshot
drwxrwxrwx   - root root       4096 2018-05-11 06:28 nfs://10.63.150.213:2049/tg
drwxrwxrwx   - root root       4096 2018-05-11 06:29 nfs://10.63.150.213:2049/ts
drwxrwxrwx   - root root       4096 2018-05-11 06:30 nfs://10.63.150.213:2049/tv
[root@node1 ~]# hadoop fs -copyFromLocal /usr/share/doc/util-linux-ng-2.17.2/README
nfs://10.63.150.213:2049/testfile
```

2. Validate that the `copyFromLocal` operation was successful in the Hadoop file system.

```
[root@node1 ~]#hadoop fs -ls  nfs://10.63.150.213:2049/
Found 5 items
drwxrwxrwx   - root root       4096 2018-05-11 04:31 nfs://10.63.150.213:2049/.snapshot
-rw-r--r--   1 root root       2865 2018-05-15 13:31 nfs://10.63.150.213:2049/testfile
drwxrwxrwx   - root root       4096 2018-05-11 06:28 nfs://10.63.150.213:2049/tg
drwxrwxrwx   - root root       4096 2018-05-11 06:29 nfs://10.63.150.213:2049/ts
drwxrwxrwx   - root root       4096 2018-05-11 06:30 nfs://10.63.150.213:2049/tv
[root@node1 ~]#
```

# 6 Product Validation

To validate the NetApp In-Place Analytics Module for real-world deployments, NetApp used the TeraGen tool to generate a Hadoop dataset. NetApp then used the TeraSort tool to conduct a MapReduce/YARN process to verify that the configuration worked as expected.

The following subsections present the details of each test NetApp conducted to validate the NetApp In-Place Analytics Module with Hadoop 2.6.

## 6.1 Basic Hadoop Functionality Validation

### Setup Procedure

To set up basic validation testing of the Hadoop functionality, NetApp completed the following steps:

1. Remove any previous NFS artifacts before each run.
2. Verify that all components are restored to a nonfaulted condition.

### Run Procedure

To run the validation testing, NetApp completed the following step:

1. Use the included Apache TeraGen and TeraSort utilities. Start from the ResourceManager node.

   **Note:** Use the same TeraGen and TeraSort parameters for all iterations.

Table 3 summarizes the test details.

**Table 3) Test details.**

| Test Information | Details |
|---|---|
| Test type | Initial tuning and full function |
| Execution type | Automated |

| Test Information | Details |
|---|---|
| Configuration | • Memory configured in the command prompt<br>`mapreduce.map.memory.mb = 32768`<br>• TeraGen options<br>`-Dmapreduce.job.maps=128`<br>• TeraSort options<br>`-Dmapreduce.job.reduces=360`<br>• TeraValidate<br>`-Dmapreduce.job.maps=360` |
| Duration | Multiple runs, one day total |
| Description | This test runs a TeraGen job with duration of greater than 10 minutes to generate a substantial dataset. It then runs a TeraSort job on the dataset created by TeraGen. |
| Prerequisites | The NodeManager components have been started. |
| Test results | • Proper output results are received from the TeraSort reduce stage.<br>• No tasks on individual task nodes (NodeManagers) fail.<br>• The file system (NFS) maintains integrity and is not corrupted.<br>• All test environment components are still running. |
| Notes | • Use integrated web and UI tools to monitor the Hadoop tasks and file system.<br>• Use Ganglia to monitor the Linux server in general. |

# 7 Hadoop TeraGen and TeraSort Validation

## 7.1 Hardware Configuration

Table 4 summarizes the configuration details for the hardware.

**Table 4) Hardware configuration details.**

| Component | Product or Solution | Details |
|---|---|---|
| Storage | NetApp A300 storage array with ONTAP 9.3 | • 2 controllers (HA pair) with 24 x 900GB, SSDs<br>• 1 hot spare per disk shelf<br>• 1 data aggregate (23 drives, shared to both controllers) per controller<br>• FlexGroup volumes<br>• 8 x 10GbE |
| Servers | PRIMERGY RX2540 M1<br>Intel Xeon CPU E5-2670 v3 at 2.30GHz | • 10 servers, and each has two 2.4GHz (6-core) or 2.3GHz (8-core) processors (40 cores)<br>• 12 DIMMs (up to 192GB), up to 1600MHz (512GB/node)<br>• 1 x 1Gbps Ethernet port, 2 x 10GbE network ports |

| Component | Product or Solution | Details |
|---|---|---|
| Networking | • 10GbE nonblocking network switch<br>• 1GbE network switch | NetApp used a Cisco Nexus 5000 switch for testing. Any compatible 10GbE network switch can also be used. |
| Server operating system | Red Hat Enterprise Linux Server 7.4 (x86_64) or later | Hadoop typically requires a Linux distribution. |
| Hadoop distribution used in the testing | Cloudera 5.14 (also works with Hortonworks distribution) | Cloudera Manager 5.14 |

## 7.2 JSON File Used for Testing

NetApp used the following JSON file for testing.

```
[root@stlrx2540m1-36 ~]# cat /etc/NetAppNFSConnector/conf/nfs-mapping.json
{
"spaces": [{
"name": "santanderpoc1",
"uri": "nfs://10.63.150.65:2049/",
"options": {
"nfsAuthScheme": "AUTH_SYS",
"nfsExportPath": "/pocfg",
"nfsMountPort": -1,
"nfsPort": 2049,
"nfsRpcbindPort": 111,
"nfsReadSizeBits": 22,
"nfsWriteSizeBits": 22,
"nfsReadMinPoolThreads":64,
"nfsReadMaxPoolThreads":256,
"nfsWriteMinPoolThreads":64,
"nfsWriteMaxPoolThreads":256,
"nfsSplitSizeBits": 30
      },

      "endpoints": [
      {
        "hosts": [
                        "nfs://10.63.150.58:2049/","nfs://10.63.150.65:2049/"
                ],
        "path": "/pocfg",
        "exportPath": "/pocfg"
      }
      ]
   }]
}

[root@stlrx2540m1-36 ~]#
```

## 7.3 TeraGen and TeraSort Validation with A300 Storage Controller

In addition to the basic functionality and fault injection testing described in section 6.1, “Basic Hadoop Functionality Validation,” NetApp used the TeraGen and TeraSort tools to measure how well the Hadoop configuration performed when generating and processing considerably larger datasets. These tests consisted of using TeraGen to create datasets that ranged in size from 100GB and 500GB to 1TB and then using TeraSort to conduct a MapReduce function on each dataset, using 10 nodes in the Hadoop cluster. NetApp recorded the elapsed time required to complete the process. NetApp observed that the duration (in minutes) of TeraGen and TeraSort responses was directly proportional to the size of the datasets.

**Note:** For these tests, NetApp did not attempt to maximize the performance of TeraGen and TeraSort. NetApp believes the performance can be improved with additional tuning.

Figure 13 shows the elapsed time to create the different datasets by using TeraGen. Creating a 1TB dataset took over 4 minutes, and no issues were logged during the TeraGen operations. Also, the time required to generate the datasets increased proportionally with the size of the dataset, indicating that the cluster maintained its data ingest rates over time.

**Note:** NetApp used one FlexGroup volume from the A300 storage controllers for this testing with TeraGen, TeraSort, and TeraValidate.

See the details for the performance validation in Table 3.

**Figure 13) TeraGen, TeraSort, and TeraValidate report.**

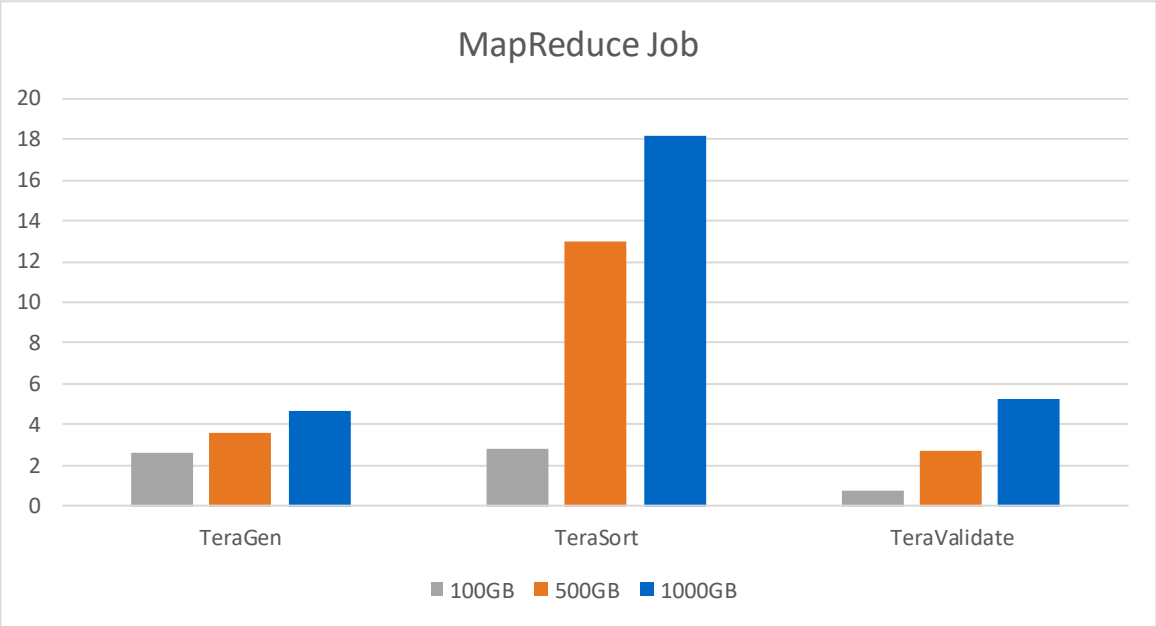



Figure 13 shows the elapsed time required to complete a TeraSort job on each of the increasingly larger datasets described in the preceding paragraphs. The 1TB dataset required 18 minutes to complete the process, and no issues were logged during the TeraSort operations. These results demonstrate that the Hadoop cluster maintained comparable processing rates as the size of the dataset increased. They also demonstrate the stability of the overall Hadoop cluster.

The tests are based on four Hadoop NodeManagers and one A300 HA pair with two storage controllers. During the test, NetApp observed that the storage controllers and disk utilization were less than 30%. Essentially, there was a lot of headroom in the storage system to perform additional operations.

Some Hadoop and Spark distributions must have the NetApp In-Place Analytics Module JAR files in the SQL Engines library path, and their XML files must be updated for the NetApp In-Place Analytics Module. See the following step:

1. Configure and check `core-site.xml` in Hadoop and `hbase-site.xml` in HBase.
   - In Cloudera, you can use Cloudera Manager to update the configuration `[ hdfs->configuration->cluster-wide advanced configuration snippet (safety value) for core-site.xml]` and execute the `restart stale service snad re-deploy client configuration`.

**Additional Information for Cloudera**

`hbase-site.xml` and `hive-site.xml` have to be updated in their clusterwide advanced configuration snippet (safety value) for HBase, Hive, and Impala to access the data from NFS by using the NetApp In-Place Analytics Module.

- In Hortonworks, you can use Ambari to update the configuration: HDFS > configuration > custom core-site.xml > add sign.

The following configuration specifies that the Hadoop cluster is using NFS in parallel with the default (primary) file system:

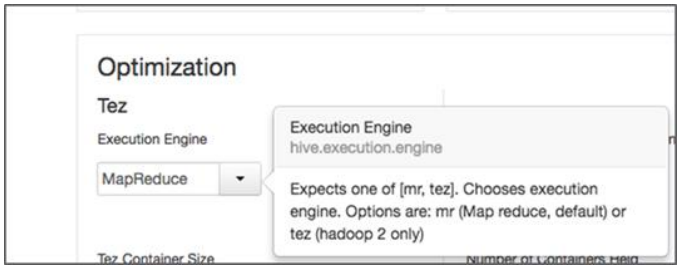
```
<configuration>
<property>
    <name>fs.AbstractFileSystem.nfs.impl</name>
    <value>org.apache.hadoop.netapp.fs.nfs.NFSv3AbstractFilesystem</value>
</property>
<property>
    <name>fs.nfs.impl</name>
    <value>org.apache.hadoop.netapp.fs.nfs.NFSv3FileSystem</value> </property>
<property>
    <name>fs.nfs.configuration</name>
    <value>/<path-to-nfs-mapping-file>/nfs-mapping.json</value> </property>
</configuration>
<property>
    <name>fs.nfs.prefetch</name>
    <value>false</value>
</property>
```

# 8 Hive Validation

Hive is similar to a SQL query, and it is often used for Apache Hadoop data warehouses. This section provides details about the NetApp In-Place Analytics Module with Hive using MapReduce and the Tez execution engine.

## 8.1 Hive with MapReduce

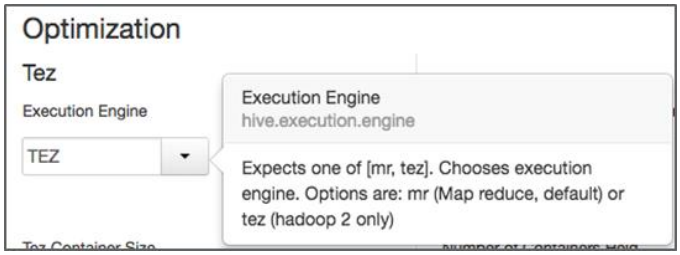
1. Select MapReduce from the Execution Engine drop-down list. By default, it points to Tez.




## 8.2 Hive with Tez

1. Select Tez from the Execution Engine drop-down list.

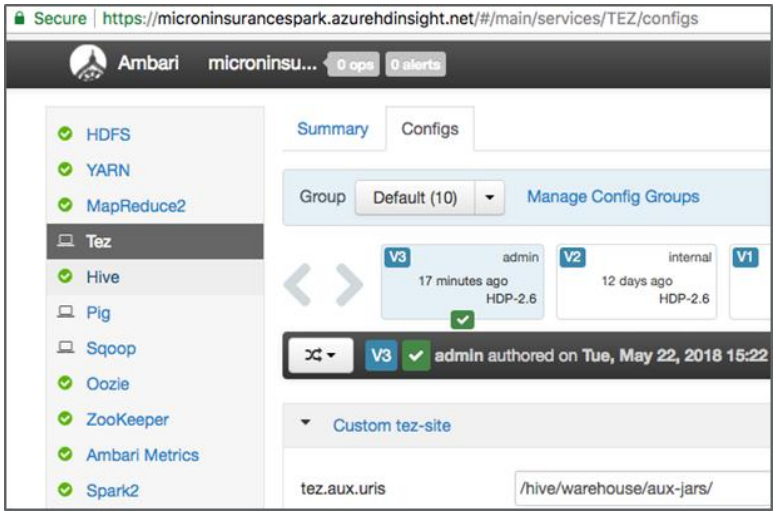



2. Check that `nfs export (nfs volume)` has Hive user and group permissions. Update if needed.
3. Copy the NetApp In-Place Analytics Module JAR files to the `defaultFS` as configured in Ambari. For example, if the `fs.defaultFS` points to HDFS, then copy the JAR files to the HDFS location. In HDInsight, if the `fs.defaultFS` points to WASB, then copy the JAR files to the WASB location.

In the following example, the JAR files were copied to `/hive/warehouse`.

```
hdfs dfs -mkdir /hive/warehouse/aux-jars
hdfs dfs -copyFromLocal /<parentfolder>/hadoop-nfs-2.7.1.jar /hive/warehouse/aux-jars
hdfs dfs -copyFromLocal /<parent folder>/hadoop-nfs-connector-3.0.0.jar /hive/warehouse/aux-jars
```

4. Add the new auxiliary JAR location to the `tez-site.conf` file.
   - This screenshot shows an Ambari example.




   - The following example shows a `Tez-site.conf` manual edit:

```
<tez.aux.uris>
/{path to desired location}/aux-jars
</tez.aux.uris>
```

5. Restart the Tez, Hive, Oozie, and so on services.
   - The following example shows the `nfs-mapping.json` file used for the Hive test.

```
root@hn0-micron:~# cat /netappnfs/nfs-mapping.json
{
    "spaces": [
        {
            "name": "nfsserver",
            "uri": "nfs://172.18.10.68:2049/",
            "options": {
                "nfsExportPath": "/iamntapcloudvolume",
                "nfsReadSizeBits": 20,
                "nfsWriteSizeBits": 20,
```

```
                "nfsSplitSizeBits": 30,
                "nfsAuthScheme": "AUTH_SYS",
                "nfsUserConfigFile": "/netappnfs/users.json",
                "nfsGroupConfigFile": "/netappnfs/groups.json",
                "nfsUsername": "root",
                "nfsGroupname": "root",
                "nfsUid": 0,
                "nfsGid": 0,
                "nfsPort": 2049,
                "nfsMountPort": -1,
                "nfsRpcbindPort": 111
            },
            "endpoints": [
                {
                    "path": "/iamntapcloudvolume",
                    "exportPath": "/iamntapcloudvolume",
                    "hosts": [
                        "nfs://172.18.10.68:2049/"
                    ]
                },
                {
                    "path": "/cloudvolumetesting",
                    "exportPath": "/cloudvolumetesting",
                    "hosts": [
                        "nfs://172.18.10.68:2049/"
                    ]
                }
            ]
        }
    ]
}
root@hn0-micron:~#
```

See the detailed example in the appendix for more information.

# 9 Spark Validation

Apache Spark is a fast, general-purpose cluster computing system. It provides high-level APIs in Java, Scala, Python, and R and an optimized engine that supports general execution graphs. It also supports a rich set of higher-level tools, including Spark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming.

For Hortonworks distribution and Apache Spark, append the `SPARK_CLASSPATH` or `SPARK_DIST_CLASSPATH` with the NetApp In-Place Analytics Module JAR files location in `spark-env` or `spark2-env`. See the following example:

```
export SPARK_CLASSPATH=/usr/hdp/2.5.6.0-40/spark/lib/*:/usr/hdp/2.5.6.0-
40/oozie/share/lib/spark/*:/usr/hdp/2.5.6.0-40/hadoop/*:/usr/hdp/2.5.6.0-40/hadoop/client/*

export SPARK_DIST_CLASSPATH=$SPARK_DIST_CLASSPATH:/usr/hdp/current/spark2-
client/jars/*:/usr/lib/hdinsight-
datalake/*:/usr/hdp/current/spark_llap/*:/usr/hdp/current/spark2-client/conf:/usr/hdp/2.6.3.2-
13/spark/lib/*:/usr/hdp/2.6.3.2-13/oozie/share/lib/spark/*:/usr/hdp/2.6.3.2-
13/hadoop/*:/usr/hdp/2.6.3.2-13/hadoop/client/*
```

NetApp tested Spark with the NetApp In-Place Analytics Module. See the following sample:

```
sshuser@hn0-micron:~$ sudo su -
root@hn0-micron:~# spark-shell
scala> val file =
sc.textFile("nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/MOCK_DATA.csv")
scala> val counts = file.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(_ +
_)
scala> counts.saveAsTextFile("nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/result")
scala> counts.count()
scala> :quit
root@hn0-micron:~# hadoop fs -ls nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest
```

```
Found 4 items
-rwxrwxrwx   1 root root         56 2018-05-11 21:56
nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/000000_0
-rw-r--r--   1 root root      61981 2018-05-11 21:49
nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/MOCK_DATA.csv
drwxrwxrwx   - root root       4096 2018-05-11 22:52
nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/out
drwxrwxrwx   - root root       4096 2018-05-11 22:57
nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/result
root@hn0-micron:~# hadoop fs -ls nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/result
Found 3 items
-rw-r--r--   1 root root          0 2018-05-11 22:57
nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/result/_SUCCESS
-rw-r--r--   1 root root      33125 2018-05-11 22:57
nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/result/part-00000
-rw-r--r--   1 root root      32917 2018-05-11 22:57
nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/result/part-00001
root@hn0-micron:~#
```

See TR-4570: NetApp Storage Solutions for Apache Spark for detailed Spark validation and performance results.

# 10 HBase Validation

HBase is an open-source, nonrelational, distributed database modeled after Google's Bigtable and is written in Java. It is developed as part of Apache Software Foundation's Apache Hadoop project and runs on top of HDFS and non-HDFS such as NFS and s3, providing Bigtable-like capabilities for Hadoop. That is, it provides a fault-tolerant way of storing large quantities of sparse data (small amounts of information caught in a large collection of empty or unimportant data), such as finding the 50 largest items in a group of 2 billion records, or finding the nonzero items representing less than 0.1% of a huge collection.

This section covers the following two validations:

- Generate the HBase Hfiles with the test table and run the HBase load process.
- Evaluate the HBase performance using HBase performance evaluation.

NetApp completed the following steps:

1. Copy the `nfsconnector jar` files to the HBase `classpath` location if required.
2. Change or add a new parameter, `hbase.rootdir`, with an NFS export in an HBase clusterwide advanced configuration snippet (safety value). For example:

```
<property>
<name>hbase.rootdir</name>
<value>nfs://10.63.150.65:2049/projectA/hbase</value>
</property>
```

   – The `nfs-mapping.json` file was used for the HBase testing:

```
[root@stlrx2540m1-36 ~]# cat /etc/NetAppNFSConnector/conf/nfs-mapping.json
{
"spaces": [{
"name": "santanderpoc1",
"uri": "nfs://10.63.150.65:2049/",
"options": {
"nfsAuthScheme": "AUTH_SYS",
"nfsExportPath": "/pocfg",
"nfsMountPort": -1,
"nfsPort": 2049,
"nfsRpcbindPort": 111,
"nfsReadSizeBits": 20,
"nfsWriteSizeBits": 20,
"nfsReadMinPoolThreads":64,
"nfsReadMaxPoolThreads":128,
"nfsWriteMinPoolThreads":64,
"nfsWriteMaxPoolThreads":128,
```

```
"nfsSplitSizeBits": 30
      },

      "endpoints": [
      {
        "hosts": [

        "nfs://10.63.150.58:2049/","nfs://10.63.150.59:2049/","nfs://10.63.150.60:2049/","nfs://1
0.63.150.61:2049/",

        "nfs://10.63.150.62:2049/","nfs://10.63.150.63:2049/","nfs://10.63.150.64:2049/","nfs://1
0.63.150.65:2049/"
               ],
        "path": "/projectA",
        "exportPath": "/pocfg/projectA"
      }
      ]
   }]
}
```

3. Generate Hfiles with the test table and run the HBase load process.
4. Create a test table called wordcount.

```
[root@stlrx2540m1-36 src]# hbase shell
18/03/30 18:09:07 INFO Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use
io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.14.0, rUnknown, Sat Jan  6 13:47:53 PST 2018

hbase(main):001:0> list
TABLE
emp
1 row(s) in 0.1780 seconds

=> ["emp"]
hbase(main):002:0> create 'wordcount', {NAME => 'f'}, {SPLITS => ['g','m','r','w']}
0 row(s) in 2.3540 seconds

=> Hbase::Table - wordcount
hbase(main):003:0> list
TABLE
emp
wordcount
2 row(s) in 0.0060 seconds

=> ["emp", "wordcount"]
hbase(main):004:0> quit
[root@stlrx2540m1-36 src]#
```

5. Import the csv file using `hbase importTsv` without designating an output location.

```
[root@stlrx2540m1-36 src]# hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -
Dimporttsv.separator=, -Dimporttsv.columns=HBASE_ROW_KEY,f:count wordcount
nfs://10.63.150.65:2049/projectA/word_count.csv
18/03/30 18:48:10 INFO zookeeper.RecoverableZooKeeper: Process identifier=hconnection-0x2261adb1
connecting to ZooKeeper ensemble=stlrx2540m1-36:2181,stlrx2540m1-35:2181,stlrx2540m1-34:2181
18/03/30 18:48:10 INFO zookeeper.ZooKeeper: Client environment:zookeeper.version=3.4.5-cdh5.14.0-
-1, built on 01/06/2018 21:31 GMT
18/03/30 18:48:10 INFO zookeeper.ZooKeeper: Client environment:host.name=stlrx2540m1-36
18/03/30 18:48:10 INFO zookeeper.ZooKeeper: Client environment:java.version=1.7.0_67
…
[some of the content removed to reduce the number of pages]
…

                ImportTsv
                        Bad Lines=0
                File Input Format Counters
                        Bytes Read=0
                File Output Format Counters
                        Bytes Written=0
```

```
[root@stlrx2540m1-36 src]#
```

6. Check the `hbase rootdir` location for the wordcount.

```
[root@stlrx2540m1-36 src]# hadoop fs -ls nfs://10.63.150.65:2049/projectA/hbase/data/default
Found 2 items
drwxrwxrwx   - hbase hbase       4096 2018-03-27 10:57
nfs://10.63.150.65:2049/projectA/hbase/data/default/emp
drwxrwxrwx   - hbase hbase       4096 2018-03-30 13:29
nfs://10.63.150.65:2049/projectA/hbase/data/default/wordcount
[root@stlrx2540m1-36 src]# hadoop fs -ls
nfs://10.63.150.65:2049/projectA/hbase/data/default/wordcount
Found 7 items
drwxrwxrwx   - hbase hbase       4096 2018-03-30 13:29
nfs://10.63.150.65:2049/projectA/hbase/data/default/wordcount/.tabledesc
drwxrwxrwx   - hbase hbase       4096 2018-03-30 13:29
nfs://10.63.150.65:2049/projectA/hbase/data/default/wordcount/.tmp
drwxrwxrwx   - hbase hbase       4096 2018-03-30 13:29
nfs://10.63.150.65:2049/projectA/hbase/data/default/wordcount/3371924e375f0f77a78d683d3f1fc122
drwxrwxrwx   - hbase hbase       4096 2018-03-30 13:29
nfs://10.63.150.65:2049/projectA/hbase/data/default/wordcount/3b26d51f7255c54e8345098aca63712c
drwxrwxrwx   - hbase hbase       4096 2018-03-30 13:29
nfs://10.63.150.65:2049/projectA/hbase/data/default/wordcount/3e3c8b9155b34fadd09b6e08aabf39d2
drwxrwxrwx   - hbase hbase       4096 2018-03-30 13:29
nfs://10.63.150.65:2049/projectA/hbase/data/default/wordcount/71e286208c191127830bee9e58f57fb3
drwxrwxrwx   - hbase hbase       4096 2018-03-30 13:29
nfs://10.63.150.65:2049/projectA/hbase/data/default/wordcount/beb1b080221bde6d5a5595da59eb8f6e
[root@stlrx2540m1-36 src]#
```

7. Import the csv file using `hbase importTsv` and designate an output location.

```
 [root@stlrx2540m1-36 src]# hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -
Dimporttsv.separator=, -Dimporttsv.bulk.output=nfs://10.63.150.65:2049/projectA/output -
Dimporttsv.columns=HBASE_ROW_KEY,f:count wordcount
nfs://10.63.150.65:2049/projectA/word_count.csv
18/03/30 18:59:49 INFO zookeeper.RecoverableZooKeeper: Process identifier=hconnection-0x5eb83c00
connecting to ZooKeeper ensemble=stlrx2540m1-36:2181,stlrx2540m1-35:2181,stlrx2540m1-34:2181
18/03/30 18:59:49 INFO zookeeper.ZooKeeper: Client environment:zookeeper.version=3.4.5-cdh5.14.0-
-1, built on 01/06/2018 21:31 GMT

..
[some of the content removed to reduce the number of pages]
..
                ImportTsv
                                Bad Lines=0
                Shuffle Errors
                                BAD_ID=0
                                CONNECTION=0
                                IO_ERROR=0
                                WRONG_LENGTH=0
                                WRONG_MAP=0
                                WRONG_REDUCE=0
                File Input Format Counters
                                Bytes Read=0
                File Output Format Counters
                                Bytes Written=32057
[root@stlrx2540m1-36 src]#
```

8. Check the `hbase rootdir` location for the wordcount.

```
[root@stlrx2540m1-36 ~]# hadoop fs -ls nfs://10.63.150.65:2049/projectA/output
Found 2 items
-rw-r--r--   1 root root          0 2018-03-30 14:18
nfs://10.63.150.65:2049/projectA/output/_SUCCESS
drwxrwxrwx   - root root       4096 2018-03-30 14:18 nfs://10.63.150.65:2049/projectA/output/f
[root@stlrx2540m1-36 ~]# hadoop fs -ls nfs://10.63.150.65:2049/projectA/output/f
Found 5 items
-rw-rw-rw-   1 root root       5562 2018-03-30 14:18
nfs://10.63.150.65:2049/projectA/output/f/46e8755b67004680aebb149a0dff897c
```

```
-rw-rw-rw-   1 root root      10234 2018-03-30 14:18
nfs://10.63.150.65:2049/projectA/output/f/d6d82276af95476483d1dcc8ed9534df
-rw-rw-rw-   1 root root       2416 2018-03-30 14:18
nfs://10.63.150.65:2049/projectA/output/f/dead558d1dc343c4a6cfee932683b871
-rw-rw-rw-   1 root root       6344 2018-03-30 14:18
nfs://10.63.150.65:2049/projectA/output/f/e734c5bb5dfe4941876b061db6d2c4d7
-rw-rw-rw-   1 root root       7501 2018-03-30 14:18
nfs://10.63.150.65:2049/projectA/output/f/ff3b8a72fa604b74bfd51424c4bd33f8
[root@stlrx2540m1-36 ~]#

root@stlrx2540m1-36 ~]# hadoop fs -chown -R hbase:hbase nfs://10.63.150.65:2049/projectA/output
```

9. Check the wordcount table.

```
[root@stlrx2540m1-36 ~]# hbase shell
18/03/30 18:54:42 INFO Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use
io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.14.0, rUnknown, Sat Jan  6 13:47:53 PST 2018

hbase(main):001:0> list
TABLE
emp
wordcount
2 row(s) in 0.1620 seconds

=> ["emp", "wordcount"]
hbase(main):004:0> scan 'wordcount'
ROW                              COLUMN+CELL
a                               column=f:count, timestamp=1522450090123, value=53
able                            column=f:count, timestamp=1522450090123, value=1
about                           column=f:count, timestamp=1522450090123, value=1
..
[some of the content removed to reduce the number of pages]
..
youre                           column=f:count, timestamp=1522450090123, value=2
yourself                        column=f:count, timestamp=1522450090123, value=1
722 row(s) in 0.8830 seconds
```

To evaluate the HBase performance with the NetApp In-Place Analytics Module, NetApp completed the following steps:

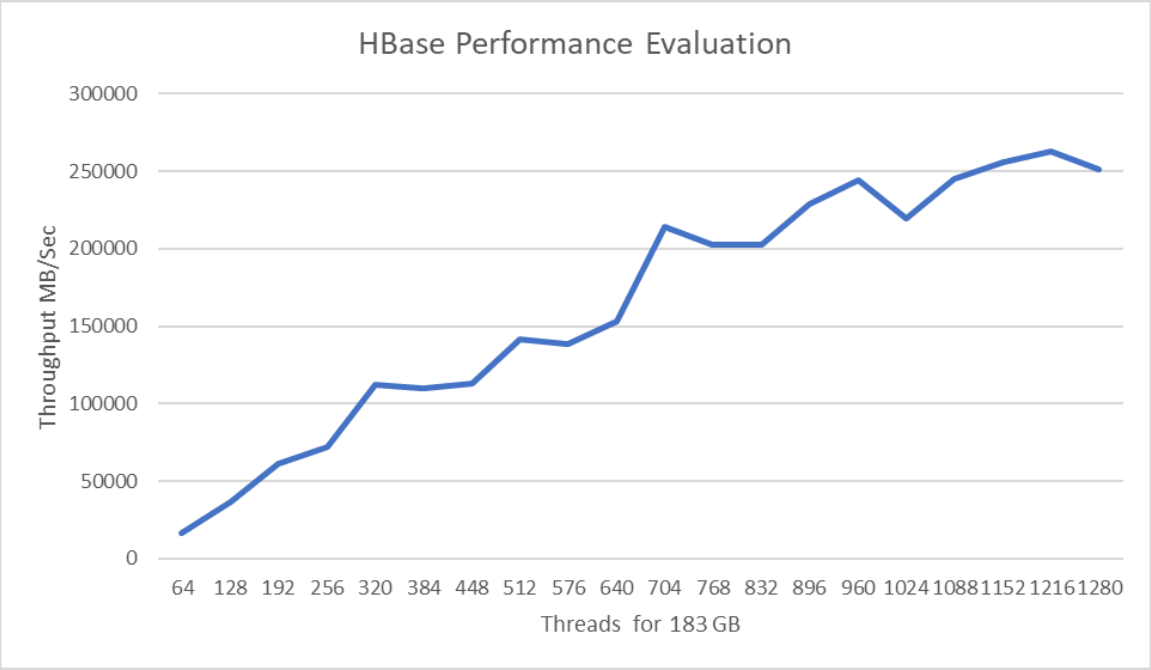
1. Use the HBase Performance Evaluation utility to run several preconfigured tests on your cluster measure performance. For example:

```
#!/bin/bash
for i in $(seq 64 64 1280 )
do
time hbase org.apache.hadoop.hbase.PerformanceEvaluation -Dmapreduce.map.memory.mb=32768 -
table=TestTable –writeToWAL=False –autoFlush=False –presplit=9 –size=180 randomWrite 1280
sync;
done
```

Figure 14 shows the test results.

**Figure 14) Test results.**



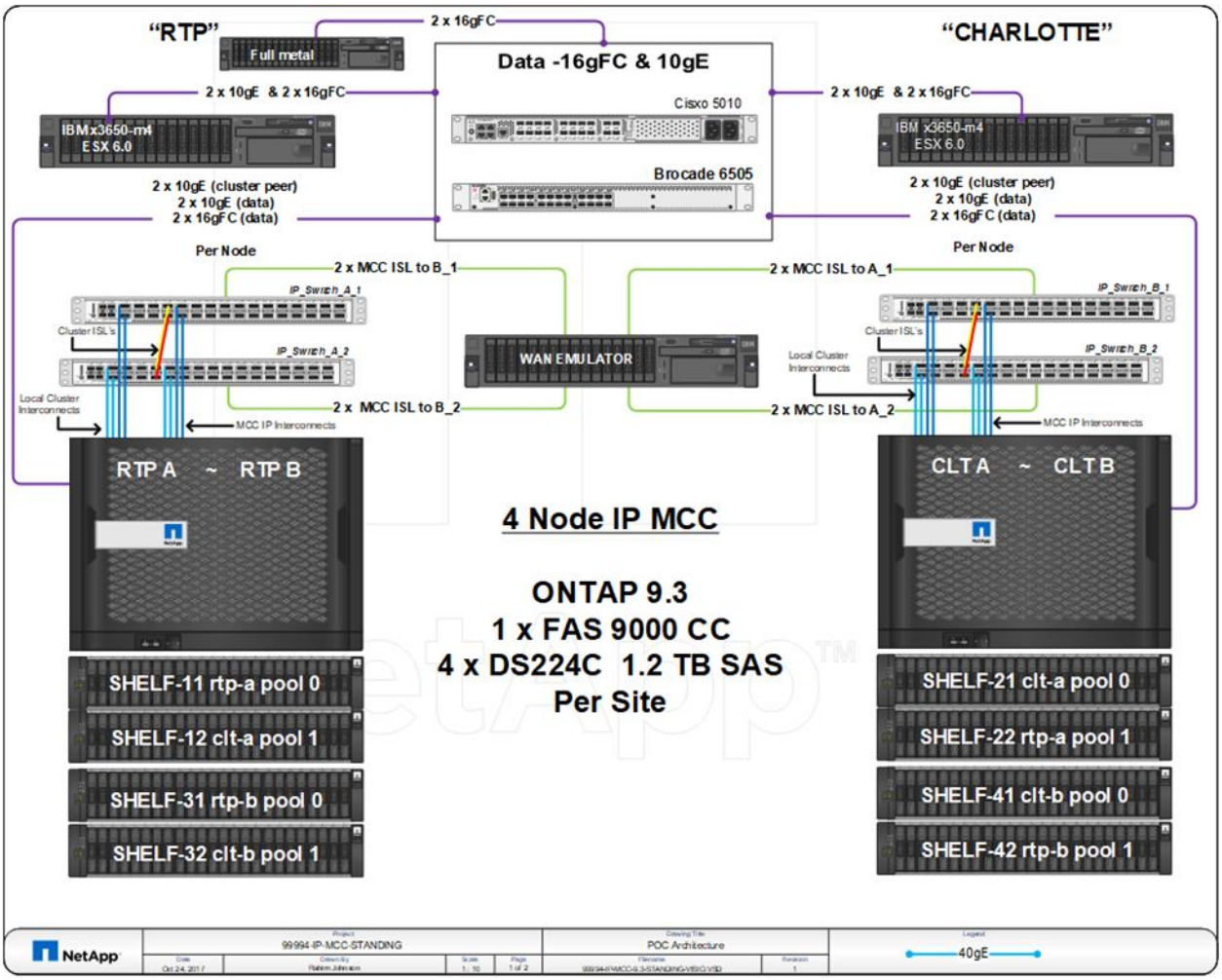

# 11 MetroCluster Validation

Based on requests from customers for active-active data protection across a site for the Hadoop cluster, NetApp tested NetApp MetroCluster (MCC-IP) in our lab. This section covers the testing details.

The architecture shown in Figure 15 was used for the validation.

Figure 15) Tested architecture.




## 11.1 Test Scenarios

NetApp completed the following test scenarios for the MetroCluster validation:

- Controller failure in one site: recover from local storage controller failure
- Total storage failure on one site (MCC unplanned site switchover): recover from MCC site switchover
- Disk failure (fail a disk on one site)
- Hadoop cluster failure
- Total site failure: compute and storage (full disaster recovery)

### Scenario 1: Controller Failure in One Site

NetApp completed the following steps for the first test scenario:

1. Start the Hadoop job, which does the read and write operations in the NFS volume on one of the storage controllers in the HA pair.
2. Create a nondisruptive failure in one of the controllers in the HA pair.
   - This failure could be either planned for system maintenance or an unplanned failure.

- A disruptive failure was simulated by powering off one controller, which provided the read and write operations for the Hadoop job.

Based on the testing, a site failover did not occur at the storage controller. Therefore, the Hadoop job was not affected.

3. Recover from the failure by turning the power controller back on.

## Scenario 2: Total Storage Failure on One Site (MCC Unplanned Site Switchover)

NetApp completed the following steps for the second test scenario:

1. Start a Hadoop job.
2. Create a total storage failure on one site by simulating the unplanned failure of an entire storage array (both controllers in an HA pair) on one site.
3. Use the PDU to power off both controllers.
   - In our test, the RTP site was powered off.
4. Use the MCC tiebreaker node to monitor both sites and make the decision to trigger the MCC-IP site switchover if one of the sites is not reachable.
   - In our test, the switchover was to the Charlotte site.
   - The Hadoop nodes on the failing site (RTP) could access their data on the second site (Charlotte) after the switchover completed.
5. Recover from the failure by powering the storage controllers back on at the RTP site.
6. Issue the following commands to the MCC switchback on the Charlotte site:

```
metrocluster heal –phase aggregates
metrocluster heal –phase root_aggregates
aggregate show-resync-status
```

7. Enter the following after the resync completes:

```
metrocluster switchback –override-vetoes true
```

## Scenario 3: Disk Failure

NetApp completed the following steps for the third test scenario:

1. Simulate a disk failure by doing a software disk fail on one controller.
   - This test is nondisruptive because the data continues to be served to the Hadoop job.
2. Issue the following commands to fail the disk on the controller that serves the volume for the Hadoop job and check the status of the disk rebuild:

```
clt::*> disk fail –immediate 1.21.7
clt::*> aggr show-resync-status -aggregate clt_A_data
```

## Scenarios 4 and 5: Hadoop Cluster and Total Site Failure (Full Disaster Recovery)

NetApp completed the following steps for the fourth and fifth test scenarios:

1. Run the NetApp In-Place Analytics Module on both Hadoop clusters for the RTP and Charlotte sites with identical JSON files.
2. Run the Hadoop job on site A (the RTP site) and power off the entire site.

   **Note:** The tiebreaker node switched the RTP site over to the Charlotte site. The Hadoop cluster in the Charlotte site was able to run the job by using the same volume used by the RTP site.

| Additional Information |
| --- |
| To view a detailed video about the MCC-IP validation, contact the author of this TR. |

# 12 Hortonworks Certification

NetApp certified the NetApp In-Place Analytics Module with Hortonworks for its ecosystem components such as ZooKeeper, YARN, MapReduce, Hive, HBase, Pig, Spark, Mahout, Sqoop, HiveServer2 Concur, HiveServer2, Kafka, Accumulo, Phoenix, Phoenix Query Server, Oozie, Storm, Flume, Hive on Spark, and Falcon. See the Hortonworks website for the certification details. The old product name, NetApp FAS NFS Connector for Hadoop, was used during the certification process. Contact the author of this TR for more information about the ecosystem components and their validation.

**Note:** NIPAM is not supported or certified with MapR.

# 13 Impala Validation

NetApp completed the following steps to validate the Impala configuration:

1. Update the configuration in `hive-site.xml(safety)` from the Hive configuration tab.

```
<property>
<name>fs.AbstractFileSystem.nfs.impl</name>
<value>org.apache.hadoop.netapp.fs.nfs.NFSv3AbstractFilesystem</value>
</property>
<property>
<name>fs.nfs.impl</name>
<value>org.apache.hadoop.netapp.fs.nfs.NFSv3FileSystem</value>
</property>
<property>
<name>fs.nfs.configuration</name>
<value>/etc/NetAppNFSConnector/conf/nfs-mapping.json</value>
</property>
<property>
<name>fs.nfs.prefetch</name>
<value>false</value>
</property>
```

2. Copy the NetApp In-Place Analytics Module JAR files to the `/var/lib/impala/` location.
   - `/opt/cloudera/parcels/CDH-x.cdhx_x/lib/hadoop/hadoop-nfs-x.x.x-cdhx.x.x.jar` is overridden by `hadoop-nfs-2.7.1.jar`, which is bundled with the NetApp In-Place Analytics Module.

```
root@stlrx2540m1-36 Downloads]# pscp.pssh -h /root/hosts /opt/cloudera/parcels/CDH-5.14.0-
1.cdh5.14.0.p0.24/lib/hadoop/hadoop-nfs-2.6.0-cdh5.14.0.jar /var/lib/impala/
Bad line: "127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4".
Format should be [user@]host[:port] [user]
Bad line: "::1         localhost localhost.localdomain localhost6 localhost6.localdomain6".
Format should be [user@]host[:port] [user]
[1] 18:45:21 [SUCCESS] 10.63.150.86
[2] 18:45:21 [SUCCESS] 10.63.150.87
[3] 18:45:21 [SUCCESS] 10.63.150.89
[4] 18:45:21 [SUCCESS] 10.63.150.88
[5] 18:45:21 [SUCCESS] 10.63.150.94
[6] 18:45:21 [SUCCESS] 10.63.150.93
[7] 18:45:21 [SUCCESS] 10.63.150.90
[8] 18:45:21 [SUCCESS] 10.63.150.91
[9] 18:45:21 [SUCCESS] 10.63.150.92
[10] 18:45:21 [SUCCESS] 10.63.150.85
You have new mail in /var/spool/mail/root
[root@stlrx2540m1-36 Downloads]# pscp.pssh -h /root/hosts /opt/cloudera/parcels/CDH-5.14.0-
1.cdh5.14.0.p0.24/lib/hadoop/hadoop-nfs-connector-3.1.6.jar /var/lib/impala/
```

```
Bad line: "127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4".
Format should be [user@]host[:port] [user]
Bad line: "::1         localhost localhost.localdomain localhost6 localhost6.localdomain6".
Format should be [user@]host[:port] [user]
[1] 18:45:35 [SUCCESS] 10.63.150.87
[2] 18:45:35 [SUCCESS] 10.63.150.86
[3] 18:45:35 [SUCCESS] 10.63.150.89
[4] 18:45:35 [SUCCESS] 10.63.150.88
[5] 18:45:35 [SUCCESS] 10.63.150.93
[6] 18:45:35 [SUCCESS] 10.63.150.94
[7] 18:45:35 [SUCCESS] 10.63.150.90
[8] 18:45:35 [SUCCESS] 10.63.150.92
[9] 18:45:35 [SUCCESS] 10.63.150.91
[10] 18:45:35 [SUCCESS] 10.63.150.85
[root@stlrx2540m1-36 Downloads]#
```

3. Create a table in Impala by using the Apache Parquet format based on the csv files.

```
 [root@hdp1 apache-drill-1.13.0]# ls -ltrh /usr/src/Air_Traffic_Passenger_Statistics.csv
-rw-r--r-- 1 root root 1.8M Apr  3 17:28 /usr/src/Air_Traffic_Passenger_Statistics.csv
[root@hdp1 apache-drill-1.13.0]# pwd
/usr/src/apache-drill-1.13.0
[root@hdp1 apache-drill-1.13.0]# ./bin/sqlline -u jdbc:drill:zk=local
May 23, 2018 5:25:22 PM org.glassfish.jersey.server.ApplicationHandler initialize
INFO: Initiating Jersey application, version Jersey: 2.8 2014-04-29 01:25:26...
apache drill 1.13.0
"a drill in the hand is better than two in the bush"
0: jdbc:drill:zk=local> select * from dfs.`/usr/src/Air_Traffic_Passenger_Statistics.csv` limit
5;
+----------+
| columns |
+----------+
| ["Activity Period","Operating Airline","Operating Airline IATA Code","Published
Airline","Published Airline IATA Code","GEO Summary","GEO Region","Activity Type Code","Price
Category Code","Terminal","Boarding Area","Passenger Count"] |
| ["200507","ATA Airlines","TZ","ATA Airlines","TZ","Domestic","US","Deplaned","Low
Fare","Terminal 1","B","27271"] |
| ["200507","ATA Airlines","TZ","ATA Airlines","TZ","Domestic","US","Enplaned","Low
Fare","Terminal 1","B","29131"] |
| ["200507","ATA Airlines","TZ","ATA Airlines","TZ","Domestic","US","Thru / Transit","Low
Fare","Terminal 1","B","5415"] |
| ["200507","Air Canada","AC","Air
Canada","AC","International","Canada","Deplaned","Other","Terminal 1","B","35156"] |
+----------+
5 rows selected (1.67 seconds)
0: jdbc:drill:zk=local> alter system set `store.format`='parquet';
+-------+------------------------+
|  ok   |        summary         |
+-------+------------------------+
| true  | store.format updated.  |
+-------+------------------------+
1 row selected (0.045 seconds)
0: jdbc:drill:zk=local> CREATE TABLE dfs.tmp.`/tmp/airport_data_new` AS
. . . . . . . . . . . > SELECT
. . . . . . . . . . . > columns[0] as `YEAR`,
. . . . . . . . . . . > columns[1] as `AIRLINE`,
. . . . . . . . . . . > columns[2] as `IATA_CODE`,
. . . . . . . . . . . > columns[3] as `AIRLINE_2`,
. . . . . . . . . . . > columns[4] as `IATA_CODE_2`,
. . . . . . . . . . . > columns[5] as `GEO_SUMMARY`,
. . . . . . . . . . . > columns[6] as `GEO_REGION`,
. . . . . . . . . . . > columns[7] as `ACTIVITY_CODE`,
. . . . . . . . . . . > columns[8] as `PRICE_CODE`,
. . . . . . . . . . . > columns[9] as `TERMINAL`,
. . . . . . . . . . . > columns[10] as `BOARDING_AREA`,
. . . . . . . . . . . > columns[11] as `PASSENGER_COUNT`
. . . . . . . . . . . > FROM dfs.`/usr/src/Air_Traffic_Passenger_Statistics.csv`;
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
+-----------+----------------------------+
```

```
| Fragment  | Number of records written  |
+-----------+----------------------------+
| 0_0       | 17960                      |
+-----------+----------------------------+
1 row selected (0.794 seconds)
0: jdbc:drill:zk=local> Closing: org.apache.drill.jdbc.impl.DrillConnectionImpl
[root@hdp1 apache-drill-1.13.0]# ls -ltrh /tmp/tmp/airport_data_new
total 2.4M
-rw-rw-r-- 1 root root 2.4M May 23 17:27 0_0_0.parquet
[root@hdp1 apache-drill-1.13.0]#
```

4. Copy the Parquet file to the Impala server.

```
[root@hdp1 apache-drill-1.13.0]# scp -r /tmp/tmp/airport_data_new root@10.63.150.62:/tmp
root@10.63.150.62's password:
0_0_0.parquet
100% 2451KB  61.9MB/s   00:00
.0_0_0.parquet.crc
100%   19KB   3.0MB/s   00:00
[root@hdp1 apache-drill-1.13.0]#
```

5. Check the file in the Impala server.

```
[root@stlrx2540m1-36 ~]# ls -ltrh /tmp/airport_data_new/
total 2.4M
-rw-r--r-- 1 root root 2.4M May 23 17:31 0_0_0.parquet
[root@stlrx2540m1-36 ~]# hadoop fs -mkdir nfs://10.63.150.65:2049/projectA/impala/airport_data
[root@stlrx2540m1-36 ~]# hadoop fs -copyFromLocal /tmp/airport_data_new/0_0_0.parquet
nfs://10.63.150.65:2049/projectA/impala/airport_data
[root@stlrx2540m1-36 ~]# hadoop fs -ls nfs://10.63.150.65:2049/projectA/impala/airport_dataFound
1 items
-rw-r--r--   1 root root    2510250 2018-05-23 13:19
nfs://10.63.150.65:2049/projectA/impala/airport_data/0_0_0.parquet
[root@stlrx2540m1-36 ~]#
```

6. Create a table and check it by using the Select query in the Impala server.

```
[root@stlrx2540m1-36 ~]# impala-shell -i 10.63.150.94
Starting Impala Shell without Kerberos authentication
Connected to 10.63.150.94:21000
Server version: impalad version 2.11.0-cdh5.14.0 RELEASE (build
d68206561bce6b26762d62c01a78e6cd27aa7690)
********************************************************************************
Welcome to the Impala shell.
(Impala Shell v2.11.0-cdh5.14.0 (d682065) built on Sat Jan  6 13:27:16 PST 2018)

The SET command shows the current value of all shell and query options.
********************************************************************************
[10.63.150.94:21000] > show databases;
Query: show databases
+------------------+-----------------------------------------------+
| name             | comment                                       |
+------------------+-----------------------------------------------+
| _impala_builtins | System database for Impala builtin functions |
| default          | Default Hive database                         |
| nfs_db           |                                               |
| nfs_db1          |                                               |
| nfs_db2          |                                               |
| nfs_db3          |                                               |
| nfs_db4          |                                               |
| nfs_db5          |                                               |
+------------------+-----------------------------------------------+
Fetched 8 row(s) in 0.01s
[10.63.150.94:21000] > use nfs_db1;
Query: use nfs_db1
[10.63.150.94:21000] > show tables;
Query: show tables
Fetched 0 row(s) in 0.00s
[10.63.150.94:21000] > create table sample_data_nfs
                     > (YEAR int, AIRLINE STRING, IATA_CODE STRING, AIRLINE_2 STRING, IATA_CODE_2
STRING, GEO_SUMMARY STRING, CEO_REGION STRING, ACTIVITY_CODE STRING, PRICE_CODE STRING, TERMINAL
STRING, BOARDING_AREA STRING, PASSENGER_COUNT STRING)
```

```
                    > stored as parquet location
'nfs://10.63.150.65:2049/projectA/impala/airport_data';
Query: create table sample_data_nfs
(YEAR int, AIRLINE STRING, IATA_CODE STRING, AIRLINE_2 STRING, IATA_CODE_2 STRING, GEO_SUMMARY
STRING, CEO_REGION STRING, ACTIVITY_CODE STRING, PRICE_CODE STRING, TERMINAL STRING,
BOARDING_AREA STRING, PASSENGER_COUNT STRING)
stored as parquet location 'nfs://10.63.150.65:2049/projectA/impala/airport_data'
Fetched 0 row(s) in 0.64s
[10.63.150.94:21000] > show tables;
Query: show tables
+-----------------+
| name            |
+-----------------+
| sample_data_nfs |
+-----------------+
Fetched 1 row(s) in 0.00s
[10.63.150.94:21000] > select count(*) from sample_data_nfs;
Query: select count(*) from sample_data_nfs
Query submitted at: 2018-05-23 17:40:43 (Coordinator: http://stlrx2540m1-31:25000)
Query progress can be monitored at: http://stlrx2540m1-
31:25000/query_plan?query_id=fe4ed80b705385d2:f82e935500000000
+----------+
| count(*) |
+----------+
| 17960    |
+----------+
Fetched 1 row(s) in 5.43s
[10.63.150.94:21000] >

[root@hdp1 apache-drill-1.13.0]#
```

# 14 Solutions for Error Messages

This section provides solutions for error messages that NetApp encountered during the validation process.

## 14.1 Error Messages

### AUTH_ERROR

See this example of the `AUTH_ERROR` message:

```
Error: 18/01/25 12:48:19 ERROR rpc.RpcClient: RPC: xid=8000001 RpcReply request denied:
xid:8000001,messageType:RPC_REPLYverifier_flavor:AUTH_NONErejectState:AUTH_ERROR
18/01/25 12:48:19 ERROR mount.MountClient: Mount MNT operation failed with RpcException RPC:
xid=8000001 RpcReply request denied:
xid:8000001,messageType:RPC_REPLYverifier_flavor:AUTH_NONErejectState:AUTH_ERROR
```

**Solution**

Disable `mount-rootonly` and `nfs-rootonly` on the NetApp SVM. See the following example:

```
Cluster::> vserver nfs modify -vserver Hadoop_SVM -nfs-rootonly disabled
Cluster::> vserver nfs modify -vserver Hadoop_SVM -mount-rootonly disabled
Hadoop-AFF8080::> vserver nfs show -vserver Hadoop_SVM -fields nfs-rootonly
vserver    nfs-rootonly
---------- ------------
Hadoop_SVM disabled
Hadoop-AFF8080::> vserver nfs show -vserver Hadoop_SVM -fields mount-rootonly
vserver    mount-rootonly
---------- --------------
Hadoop_SVM disabled
```

## Could Not Parse File

See this example of the `could not parse file` message:

```
Could not parse config file /etc/NetAppNFSConnector/conf/nfs-mapping.json
```

**Solution**

Check https://jsonlint.com/ to see if the JSON file is valid, because it might have a syntax error.

## Unsupported Verifier AUTH_SYS

See this example of the `Unsupported verifier AUTH_SYS` message:

```
ERROR rpc.RpcClientHandler: NfsConnectorV3.0.0 RPC: Got an exception
java.lang.UnsupportedOperationException: Unsupported verifier flavorAUTH_SYS
```

**Solution**

Modify `superuser security types` to `sys` in the NetApp SVM. See the following example:

```
stlaff300-1and2::*> export-policy rule show -vserver hadoopsvm -policyname default -instance

                                Vserver: hadoopsvm
                            Policy Name: default
                             Rule Index: 1
                        Access Protocol: cifs, nfs, flexcache
List of Client Match Hostnames, IP Addresses, Netgroups, or Domains: 0.0.0.0/0
                          RO Access Rule: any
                          RW Access Rule: any
User ID To Which Anonymous Users Are Mapped: 65534
                  Superuser Security Types: any
              Honor SetUID Bits in SETATTR: true
                 Allow Creation of Devices: true
              NTFS Unix Security Options: fail
       Vserver NTFS Unix Security Options: use_export_policy
                   Change Ownership Mode: restricted
            Vserver Change Ownership Mode: use_export_policy

stlaff300-1and2::*>

stlaff300-1and2::*> export-policy rule modify -vserver hadoopsvm -policyname default -superuser
sys  -ruleindex *
1 entry was modified.

stlaff300-1and2::*> export-policy rule show -vserver hadoopsvm -policyname default -fields
superuser
vserver   policyname ruleindex superuser
--------- ---------- --------- ---------
hadoopsvm default    1         sys

stlaff300-1and2::*>
```

## Split Metadata Size Exceeds 10,000,000

If the split metadata size exceeds 10,000,000, then complete these tasks:

- Change `mapreduce.job.split.metainfo.maxsize=-1` in XML (or) change it in the CLI by using `-Dmapreduce.job.split.metainfo.maxsize=-1`.

## java.lang.OutOfMemoryError: Java Heap Space

See this example of the `java.lang.OutOfMemoryError` message:

```
java.lang.OutOfMemoryError: Java heap space
18/04/24 16:53:38 ERROR rpc.RpcClientHandler: NfsConnectorV3.0.0 RPC: Got an exception
```

```
java.lang.OutOfMemoryError: GC overhead limit exceeded
Apr 24, 2018 4:53:38 PM org.jboss.netty.channel.DefaultChannelPipeline
WARNING: An exception was thrown by a user handler while handling an exception event ([id:
0x01029bf6, /192.168.30.175:54688 => /192.168.30.111:2049] EXCEPTION: java.lang.OutOfMemoryError:
Java heap space)
java.lang.OutOfMemoryError: GC overhead limit exceeded

Apr 24, 2018 4:53:38 PM org.jboss.netty.channel.socket.nio.AbstractNioSelector
WARNING: Unexpected exception in the selector loop.
java.lang.OutOfMemoryError: GC overhead limit exceeded
```

**Solution**

`JAVA_OPTIONS` is the standard environment variable that some servers and other Java apps append to the call that executes the `JAVA` command. For example, allocate 240GB for `heapsize` with the `JAVA` command and export the `_JAVA_OPTIONS` with the following settings:

```
[root@hdp1 ~]# export _JAVA_OPTIONS=-Xmx240000M
[root@hdp1 ~]# echo $_JAVA_OPTIONS
-Xmx240000M
```

## Temp Space

If the `temp space` in the NodeManager is not equal to or greater than the dataset size, then complete this task:

- Run the `yarn.nodemanager.localizer.cache.target-size-mb` command to control the space.

## Root User

See this example of the root user error:

```
permission denied
```

**Solution**

Change the root user in the SVM from 1 to 0 by running the `unix-user modify` command. See the following example:

```
Hadoop_SVM::> unix-user show -instance -user root

       User Name: root
         User ID: 0
Primary Group ID: 1
User's Full Name:
Hadoop_SVM::> unix-user modify -user root -id 0 -primary-gid 0

Hadoop_SVM::> unix-user show -instance -user root

       User Name: root
         User ID: 0
Primary Group ID: 0
User's Full Name:

Hadoop_SVM::>
```

## Class org.apache.hadoop.netapp.fs.nfs.NFSv3FileSystem Not Found

This error message sometimes displays in Hive:

```
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask.
MetaException(message:java.lang.RuntimeException:java.lang.ClassNotFoundException: Class
org.apache.hadoop.fs.nfs.NFSv3FileSystem not found
```

**Solution**

If Hive cannot find the NetApp In-Place Analytics Module JAR file in the Hive prompt, then add the JAR files to Hive by using the `add jar` command.

```
hive> add jar /usr/hdp/2.5.3.51-3/hadoop/hadoop-nfs-2.7.3.2.5.3.51-3.jar;
Added [/usr/hdp/2.5.3.51-3/hadoop/hadoop-nfs-2.7.3.2.5.3.51-3.jar] to class path
Added resources: [/usr/hdp/2.5.3.51-3/hadoop/hadoop-nfs-2.7.3.2.5.3.51-3.jar]
Added [/usr/hdp/2.5.3.51-3/hadoop/hadoop-nfs-connector-2.0.0.jar] to class path
Added resources: [/usr/hdp/2.5.3.51-3/hadoop/hadoop-nfs-connector-2.0.0.jar]
```

### Could Not Get Root File Handle for Endpoint

See this example of a `could not get root file handle for endpoint` message:

```
ERROR nfs.NFSv3FileSystemStore: NfsConnectorV3.0.0 Could not get root file
handle for endpoint ep=Endpoint: hosts=[nfs://10.63.150.118:2049/]
export=/faiz_vol path=/
ls: Could not establish channel to any interface in vserver. Check network
configuration
```

**Solution**

- Check the `nfs server export policy` rules in the NetApp SVM, which the Hadoop servers access.
- The new SVMs normally don't have the rules in the default export policy; therefore, create a rule with `nfsv3 access, unix` permissions.
- Check that the `2049 port` is listening for the `nfs server IP/LIF`.
- Check an online website such as jsonlint.com to verify that the JSON syntax is correct.

### No Such File or Directory

See this example of a `no such file or directory` error:

```
mount.nfs: mounting 10.63.150.118:/faiz_vol1 failed, reason given by server: No such file or
directory
```

**Solution**

In the NetApp SVM (formerly known as Vserver) `namespace` configuration, make sure that the `path` and `storage object` are the same.

# 15 Conclusion

The NetApp In-Place Analytics Module is easy to deploy and runs analytics natively on existing NAS storage (NFS or CIFS), with high availability and only one copy of Hadoop data. Its performance scales in proportion to the dataset size. It supports key Hadoop ecosystem projects such as Apache Hadoop, Apache Spark, Apache HBase, and Tachyon.

# Appendix A: Hive Example

The following example works for both the Tez and MapReduce execution engines.

1. Check the folders from NFS.

```
root@hn0-micron:/tmp# ls -l /tmp/On_Time_On_Time_Performance_2015_12.csv
-rw-r--r-- 1 sshuser sshuser 191083027 May 11 19:13 /tmp/On_Time_On_Time_Performance_2015_12.csv
```

```
root@hn0-micron:/tmp# hadoop fs -copyFromLocal /tmp/On_Time_On_Time_Performance_2015_12.csv
nfs://172.18.10.68:2049/iamntapcloudvolume/
root@hn0-micron:/tmp# hadoop fs -ls nfs://172.18.10.68:2049/iamntapcloudvolume/
Found 7 items
drwxrwxrwx   - root root       4096 2018-05-11 16:57
nfs://172.18.10.68:2049/iamntapcloudvolume/.snapshot
-rw-r--r--   1 root root  191083027 2018-05-11 19:32
nfs://172.18.10.68:2049/iamntapcloudvolume/On_Time_On_Time_Performance_2015_12.csv
drwxrwxrwx   - root root       4096 2018-05-11 19:16
nfs://172.18.10.68:2049/iamntapcloudvolume/tg
drwxrwxrwx   - root root       4096 2018-05-11 19:18
nfs://172.18.10.68:2049/iamntapcloudvolume/ts
drwxrwxrwx   - root root       4096 2018-05-11 19:27
nfs://172.18.10.68:2049/iamntapcloudvolume/ts1
drwxrwxrwx   - root root       4096 2018-05-11 19:30
nfs://172.18.10.68:2049/iamntapcloudvolume/ts2
drwxrwxrwx   - root root       4096 2018-05-11 19:31
nfs://172.18.10.68:2049/iamntapcloudvolume/tv2
root@hn0-micron:/tmp# hadoop fs -ls -h nfs://172.18.10.68:2049/iamntapcloudvolume/
Found 7 items
drwxrwxrwx   - root root        4 K 2018-05-11 16:57
nfs://172.18.10.68:2049/iamntapcloudvolume/.snapshot
-rw-r--r--   1 root root    182.2 M 2018-05-11 19:32
nfs://172.18.10.68:2049/iamntapcloudvolume/On_Time_On_Time_Performance_2015_12.csv
drwxrwxrwx   - root root        4 K 2018-05-11 19:16
nfs://172.18.10.68:2049/iamntapcloudvolume/tg
drwxrwxrwx   - root root        4 K 2018-05-11 19:18
nfs://172.18.10.68:2049/iamntapcloudvolume/ts
drwxrwxrwx   - root root        4 K 2018-05-11 19:27
nfs://172.18.10.68:2049/iamntapcloudvolume/ts1
drwxrwxrwx   - root root        4 K 2018-05-11 19:30
nfs://172.18.10.68:2049/iamntapcloudvolume/ts2
drwxrwxrwx   - root root        4 K 2018-05-11 19:31
nfs://172.18.10.68:2049/iamntapcloudvolume/tv2
root@hn0-micron:/tmp#
```

2. Use any sample csv file. Our Hive test used the sample csv file `MOCK.csv`.

```
root@hn0-micron:/tmp# hadoop fs -ls nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/
Found 1 items
-rw-r--r--   1 root root      61981 2018-05-11 21:49
nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/MOCK_DATA.csv
root@hn0-micron:/tmp# hadoop fs -ls -d nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/
drwxrwxrwx   - root root       4096 2018-05-11 21:52
nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest
root@hn0-micron:/tmp# hadoop fs -ls -d nfs://172.18.10.68:2049/iamntapcloudvolume/
drwxrwx---   - root root       4096 2018-05-11 21:52 nfs://172.18.10.68:2049/iamntapcloudvolume
root@hn0-micron:/tmp# hadoop fs -chmod 777 nfs://172.18.10.68:2049/iamntapcloudvolume/
root@hn0-micron:/tmp# hadoop fs -ls -d nfs://172.18.10.68:2049/iamntapcloudvolume/
drwxrwxrwx   - root root       4096 2018-05-11 21:52 nfs://172.18.10.68:2049/iamntapcloudvolume
root@hn0-micron:/tmp# hadoop fs -ls -d nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/
drwxrwxrwx   - root root       4096 2018-05-11 21:52
nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest
root@hn0-micron:/tmp#
```

3. Test the csv file with a few values.

```
root@hn0-micron:~# hadoop fs -cat
nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/MOCK_DATA.csv
id,first_name,last_name,email,gender,ip_address
1,Hetti,Blayney,hblayney0@symantec.com,Female,35.217.186.248
2,Harrison,Pretti,hpretti1@mayoclinic.com,Male,29.66.223.175
3,Gino,Keepence,gkeepence2@globo.com,Male,228.224.184.130
4,Reyna,Zanotti,rzanotti3@twitpic.com,Female,145.230.133.233
5,Graham,Shelly,gshelly4@ehow.com,Male,56.102.55.112
…
[due to space constraint, we removed some contents]
…
996,Pembroke,Proudman,pproudmanrn@networkadvertising.org,Male,17.221.214.71
997,Byran,Benthall,bbenthallro@ucoz.com,Male,171.8.222.142
998,Victor,MacNess,vmacnessrp@printfriendly.com,Male,82.137.224.230
```

```
999,Chrissy,Swatman,cswatmanrq@statcounter.com,Female,194.60.63.233
1000,Carmen,Simkins,csimkinsrr@typepad.com,Female,28.93.173.86
```

4. Create a Hive table, select the table, and insert the table operations.

```
root@hn0-micron:/tmp# /usr/bin/hive

Logging initialized using configuration in file:/etc/hive/2.6.3.2-13/0/hive-log4j.properties
hive> create external table sample
    > (id INT, FirstName STRING, LastName STRING, Email STRING, Sex STRING, IpAddress STRING)
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
    > LOCATION 'nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/';
OK
Time taken: 2.933 seconds
hive> show tables;
OK
airline_data
hivesampletable
sample
Time taken: 0.157 seconds, Fetched: 3 row(s)
hive> select count(*) from sample;
Query ID = root_20180511215555_7fa3ae2e-7ad4-4e34-bcaf-c1368924bcc9
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1525982525446_0033, Tracking URL = http://hn1-
micron.mqp4j4qmfi1evfdl3pror4yaba.dx.internal.cloudapp.net:8088/proxy/application_1525982525446_0
033/
Kill Command = /usr/hdp/2.6.3.2-13/hadoop/bin/hadoop job  -kill job_1525982525446_0033
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-05-11 21:56:08,765 Stage-1 map = 0%,  reduce = 0%
2018-05-11 21:56:15,718 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.5 sec
2018-05-11 21:56:23,243 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 7.98 sec
MapReduce Total cumulative CPU time: 7 seconds 980 msec
Ended Job = job_1525982525446_0033
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 7.98 sec   HDFS Read: 7762 HDFS Write: 5
SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 980 msec
OK
1001
Time taken: 30.67 seconds, Fetched: 1 row(s)
hive> insert into table sample values (10001,
'Lavina','Khilnani','lavina@redhat.com','Female','10.0.0.1');
Query ID = root_20180511215632_d94cb4c4-d21e-4788-85cf-849f90116543
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1525982525446_0034, Tracking URL = http://hn1-
micron.mqp4j4qmfi1evfdl3pror4yaba.dx.internal.cloudapp.net:8088/proxy/application_1525982525446_0
034/
Kill Command = /usr/hdp/2.6.3.2-13/hadoop/bin/hadoop job  -kill job_1525982525446_0034
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2018-05-11 21:56:43,570 Stage-1 map = 0%,  reduce = 0%
2018-05-11 21:56:51,347 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.47 sec
MapReduce Total cumulative CPU time: 4 seconds 470 msec
Ended Job = job_1525982525446_0034
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to directory nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/.hive-
staging_hive_2018-05-11_21-56-32_379_6024610194214902321-1/-ext-10000
Loading data to table default.sample
Table default.sample stats: [numFiles=2, totalSize=62037]
MapReduce Jobs Launched:
```

```
Stage-Stage-1: Map: 1   Cumulative CPU: 4.47 sec   HDFS Read: 4861 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 470 msec
OK
Time taken: 23.315 seconds
hive> select * from sample where id=1001;
OK
Time taken: 0.511 seconds
hive> select * from sample where id=10001;
OK
10001   Lavina  Khilnani        lavina@redhat.com       Female  10.0.0.1
Time taken: 0.479 seconds, Fetched: 1 row(s)
hive> select * from sample where FirstName='lavina';
OK
Time taken: 0.491 seconds
hive> select * from sample where FirstName='Lavina';
OK
10001   Lavina  Khilnani        lavina@redhat.com       Female  10.0.0.1
Time taken: 0.836 seconds, Fetched: 1 row(s)
hive> desc sample;
OK
id                      int
firstname               string
lastname                string
email                   string
sex                     string
ipaddress               string
Time taken: 0.191 seconds, Fetched: 6 row(s)
hive>
root@hn0-micron:~#
```

You can try other operations that are supported for the external table.

5. Check the folders after the Hive jobs.

```
root@hn0-micron:~# hadoop fs -ls nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest
Found 4 items
-rwxrwxrwx   1 root root         56 2018-05-11 21:56
nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/000000_0
-rw-r--r--   1 root root      61981 2018-05-11 21:49
nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/MOCK_DATA.csv
drwxrwxrwx   - root root       4096 2018-05-11 22:52
nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/out
drwxrwxrwx   - root root       4096 2018-05-11 22:57
nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/result
root@hn0-micron:~# hadoop fs -ls nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/result
Found 3 items
-rw-r--r--   1 root root          0 2018-05-11 22:57
nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/result/_SUCCESS
-rw-r--r--   1 root root      33125 2018-05-11 22:57
nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/result/part-00000
-rw-r--r--   1 root root      32917 2018-05-11 22:57
nfs://172.18.10.68:2049/iamntapcloudvolume/hivetest/result/part-00001
root@hn0-micron:~#
```

## Where to Find Additional Information

To learn more about the information described in this document, refer to the following documents and/or websites:

- NetApp Solutions for Hadoop Reference Architecture
  www.netapp.com/us/media/wp-7196.pdf
- NetApp Solutions for Hadoop Reference Architecture: Cloudera
  http://www.netapp.com/us/media/wp-7217.pdf
- MixApart: Decoupled Analytics for Shared Storage Systems
  https://www.usenix.org/system/files/conference/fast13/fast13-final58.pdf

- Apache Hadoop YARN—Concepts and Applications
  http://hortonworks.com/blog/apache-hadoop-yarn-concepts-and-applications/
- Apache HBase
  https://en.wikipedia.org/wiki/Apache_HBase
- HBase Performance Evaluation
  https://www.cloudera.com/documentation/enterprise/5-9-x/topics/cdh_ig_hbase_tools.html

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.



TR-4382-0618