



Technical Report

Apache Hadoop on Data Fabric Enabled by NetApp

Hadoop Across Data Centers with NFS Connector for Hadoop and NetApp Private Storage

Karthikeyan Nagalingam and Pradeep Nayak, NetApp
July 2016 | TR-4529

TABLE OF CONTENTS

1	Introduction	4
2	Solution Overview	4
2.1	Data Fabric Enabled by NetApp	4
2.2	NPS for AWS	4
2.3	ONTAP Overview	6
2.4	NetApp FAS NFS Connector for Hadoop	6
2.5	Solution Benefits and Use Cases	8
2.6	Key Hadoop Components	10
3	Solution Architecture	11
3.1	Network Architecture	11
3.2	Storage Architecture	12
4	MapReduce Functionality and Performance Validation	14
4.1	TeraGen, TeraSort, and TeraValidate	14
4.2	DFSIO	15
5	Data Analysis Validation	16
5.1	Impala	17
5.2	Hive	19
6	Apache Pig	20
7	Mrjob	21
8	Apache Spark	23
	Summary	25
	Appendixes	25
	Impala (Create Table and Select Query)	25
	Impala Qualification with NetApp FAS NFS Connector for Hadoop for Join Query	29
	Hive	46
	Pig	50
	References	52
	Acknowledgements	53

LIST OF TABLES

Table 1)	Key components of the validated configuration	13
----------	---	----

LIST OF FIGURES

Figure 1) NPS for AWS architecture.....	5
Figure 2) Components of the NetApp FAS NFS Connector for Hadoop.....	7
Figure 3) NetApp FAS NFS Connector for Hadoop.....	8
Figure 4) Cross-data-center deployments.	9
Figure 5) Smart applications with Hadoop solution.	10
Figure 6) Network architecture.	12
Figure 7) Storage architecture.....	13
Figure 8) TeraGen, TeraSort, and TeraValidate results.	15
Figure 9) DFSIO write throughput per AWS Hadoop instance (MBps).	16
Figure 10) DFSIO read throughput per AWS Hadoop instance (MBps).	16
Figure 11) Hive with NetApp NFS FAS Connector for Hadoop.	19
Figure 12) Pig deployment in AWS through NetApp NFS FAS Connector and NPS.....	21
Figure 13) Hadoop streaming job with NFS FAS Connector for Hadoop.....	22
Figure 14) Spark deployment modes.	24
Figure 15) Apache Spark performance validation in AWS with single 10GbE Direct Connect.	25

1 Introduction

This document describes Apache Hadoop solutions with NetApp® FAS NFS Connector in the Data Fabric enabled by NetApp. This Data Fabric uses compute resources from Amazon Web Services (AWS) and storage from NetApp Private Storage (NPS).

This document describes the installation, underlying architecture, use cases, and integration with Hadoop in a hybrid cloud environment. Additionally, it discusses the validation of a number of important Hadoop ecosystem components, such as Impala, Hive, mrjob, Pig, and Apache Spark, with the NetApp FAS NFS Connector. In addition, this report showcases the benefits of performing cloud-based analytics with the Data Fabric enabled by NetApp.

This Data Fabric helps IT organizations to leverage the agility that the cloud promises, such as:

- **Simplified data management and control.** The NetApp universal data platform provides simplified data management and control. The platform offers customers multiple deployment and procurement options to adjust easily and efficiently to align with changing business needs.
- **Dynamic data portability across clouds.** NetApp SnapMirror® and the NetApp OnCommand® Cloud Manager portal provide the ability to move data and applications dynamically among cloud resources. Their universal data platform and dynamic data portability remove the barriers to private and public clouds to create a reliable hybrid cloud environment while maintaining your choice of resources.
- **Extensive choices for technologies, applications, and public cloud service providers.** The NetApp cloud strategy offerings comprise an evolving portfolio of solutions that span private cloud, cloud service providers, and hyperscale cloud providers. Some of the cloud offerings are:
 - NPS for Cloud: Combines the power of NetApp ONTAP® management software with multiple cloud providers such as AWS, Microsoft Azure, and IBM SoftLayer for on-demand performance computing, efficient disaster recovery, and optional multitier backup and recovery.
 - ONTAP Cloud: Leverages the same benefits of ONTAP in the cloud with on-demand, pay-as-you-go options.
 - NetApp StorageGRID® Webscale: Provides reliable object-based storage that connects private clouds with public cloud resources, including Amazon S3.
 - NetApp AltaVault™ cloud-integrated storage: Opens new pathways to cloud-based backup and archive solutions while preserving existing storage investments.

2 Solution Overview

2.1 Data Fabric Enabled by NetApp

The Data Fabric is NetApp's vision for the future of data management. A data fabric seamlessly connects different data management environments across disparate clouds into a cohesive, integrated whole. The Data Fabric enabled by NetApp helps organizations to maintain control and choice in how they manage, secure, protect, and access their data across the hybrid cloud, no matter where it is.

Although a data fabric evolves constantly, organizations can start leveraging it by using NetApp technologies that enable data management and seamless data movement across the hybrid cloud. For more information about the Data Fabric powered by NetApp, see [WP-7218: NetApp Data Fabric Architecture Fundamentals: Building a Data Fabric Today](#).

2.2 NPS for AWS

NetApp offers a family of storage solutions to help customers optimize their data use through the AWS cloud. By combining the world's number-one public cloud and the world's number-one open networked branded storage operating system (source: IDC Worldwide Quarterly Enterprise Storage Systems Tracker 2015 Q4, March 2016 [Open Networked Enterprise Storage Systems revenue]), you can confidently

manage and maintain control of your data while gaining the flexibility and efficiency benefits of the AWS cloud. You can rent, lease, or buy data storage in or near the cloud to best match your IT resources to your business requirements. Figure 1 illustrates the NPS for AWS architecture.

Figure 1) NPS for AWS architecture.

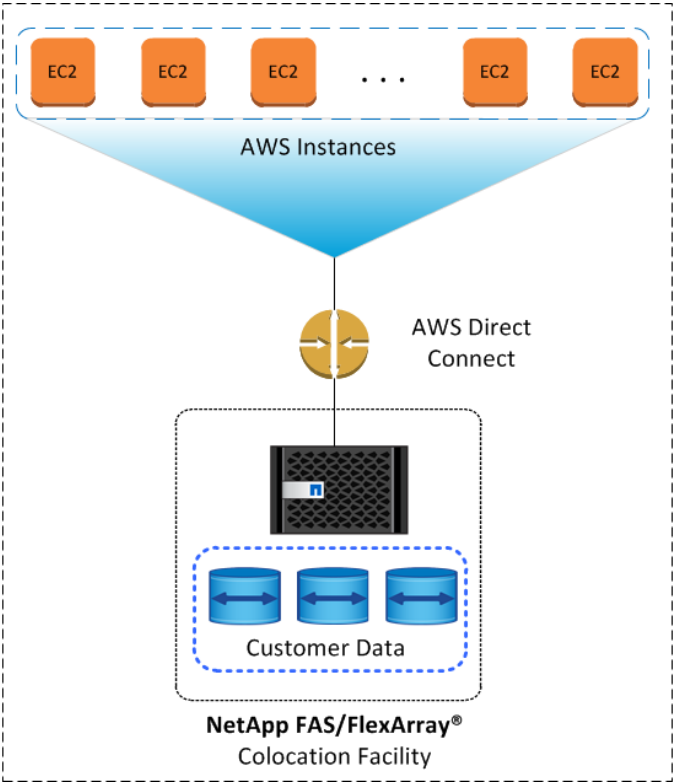


Table 1 lists the NPS for AWS use cases.

Table 1) NPS for AWS use cases.

Use Case
Use Amazon EC2 for large, variable, and enterprise workloads.
Provide disaster recovery at a lower cost.
Leverage temporary cloud resources for data center migration and consolidation.
Adjust your capex-opex spending throughout your storage-as-a-service lifecycle.

NetApp provides the following flexibility and control with NPS for AWS:

- Gain public cloud benefits and proven enterprise storage with a hybrid cloud solution.
- Get secure connectivity to the AWS cloud with AWS Direct Connect.
- Meet compliance and sovereignty requirements.
- Obtain world-class physical security for your data.

NetApp storage solutions with AWS provides the following benefits:

- Obtain AWS cloud benefits with the power of ONTAP.
- Change your capex-opex mix to meet your business needs.

- Accelerate innovation and time to market, maintaining control of your data.

2.3 ONTAP Overview

ONTAP helps achieve results and get products to market faster by providing the throughput and scalability needed to meet the demanding requirements of high-performance computing and digital media content applications. ONTAP also facilitates high levels of performance, manageability, and reliability for large Linux, UNIX, and Microsoft Windows clusters.

NetApp ONTAP features include:

- Scale-up, scale-out, and scale-down are possible with numerous nodes using a global namespace.
- Storage virtualization with storage virtual machines (SVMs) eliminates the physical boundaries of a single controller (memory, CPU, ports, disks, and so on).
- Nondisruptive operations are available when you redistribute load or rebalance capacity combined with network load balancing options within the cluster for upgrading or expanding its nodes.
- NetApp storage efficiency features such as NetApp Snapshot® copies, thin provisioning, space-efficient cloning, deduplication, data compression, and NetApp RAID DP® technology are also available.

The main ONTAP components include:

- **SVM:** An SVM is a logical file system namespace capable of spanning beyond the boundaries of physical nodes in a cluster.
 - Clients can access virtual servers from any node in the cluster, but only through the associated LIFs.
 - Each SVM has a root volume under which additional volumes are mounted, extending the namespace.
 - Each SVM can span several physical nodes.
 - Each SVM is associated with one or more logical interfaces; clients access the data on the virtual server through the logical interfaces that can live on any node in the cluster.
- **LIF:** A LIF is essentially an IP address with associated characteristics, such as a home port, a list of ports for failover, a firewall policy, a routing group, and so on.
 - Client network data access is through LIFs dedicated to the SVM.
 - An SVM can have more than one LIF. You can have many clients mounting one LIF or one client mounting several LIFs. This means that IP addresses are no longer tied to a single physical interface.
- **Aggregates:** An aggregate is a RAID-level collection of disks; it can contain more than one RAID group.
 - Aggregates serve as resources for SVMs and are shared by all SVMs.
- **Flexible volumes:** A volume is a logical unit of storage. The disk space that a volume occupies is provided by an aggregate.
 - Each volume is associated with one individual aggregate, and therefore with one physical node.
 - In ONTAP, data volumes are owned by an SVM.

2.4 NetApp FAS NFS Connector for Hadoop

The NetApp FAS NFS Connector for Hadoop allows analytics software such as Apache Hadoop and Spark to use ONTAP to access data using the NFS protocol and a simple configuration file change. By using ONTAP, the connector decouples analytics from storage, leveraging the benefits of NAS to share data. ONTAP enables high availability (HA) and storage efficiency. For even higher performance, the NetApp FAS NFS Connector for Hadoop can be combined with Tachyon to build a scale-out caching tier that is backed by ONTAP.

The NetApp FAS NFS Connector for Hadoop design has two deployment options enabling NFS to be leveraged as a non-Hadoop Distributed File System (non-HDFS) or in parallel with HDFS. The deployment options are based on the use cases and the applications used in Apache Hadoop and Apache Spark. Regardless of the file system selected, the user applications don't have to be modified.

Figure 2) Components of the NetApp FAS NFS Connector for Hadoop.

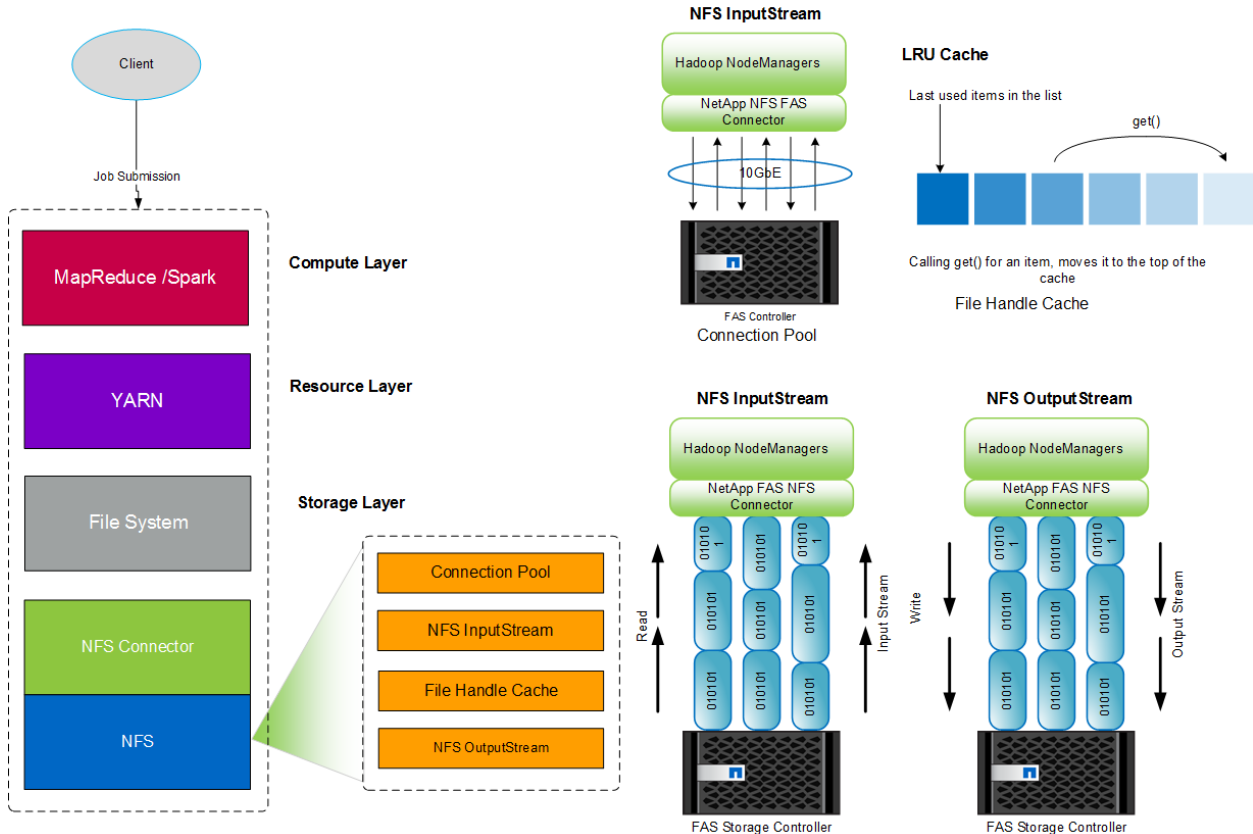


Figure 2 shows the four main components of the NetApp FAS NFS Connector for Hadoop: connection pool, NFS input stream, file handle cache, and NFS output stream. There are two other minor components: authentication and user and group name.

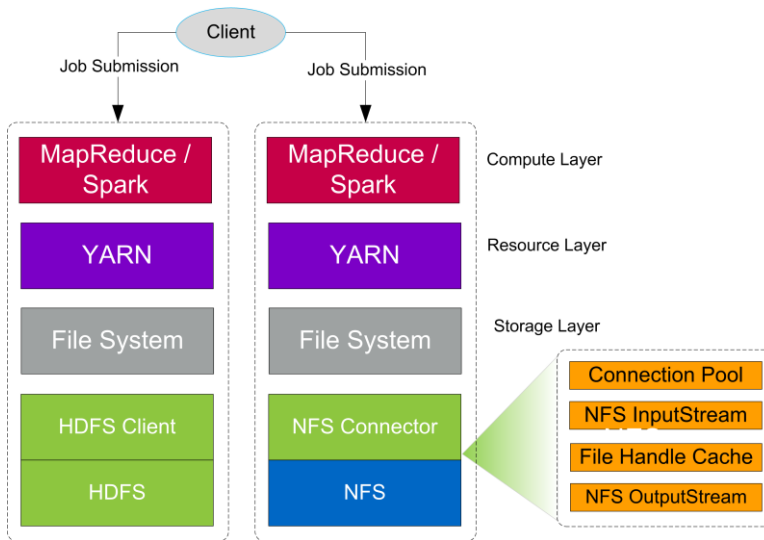
- **Connection pool:** Makes multiple connections to the storage server; allows the full use of 10GbE.
- **NFS input stream:** Issues large sequential reads (1MB by default) and uses Hadoop semantics for aggressive and intelligent prefetching.
- **File handle cache:** Reduces NFS lookup calls and uses an LRU cache.
- **NFS output stream:** Batches write operations into large inputs/outputs (I/Os) (1MB by default, 64K with ONTAP) and use Hadoop semantics to write in background and commit at task end.

For details about FAS NFS Connector for Hadoop components, see [TR-4382: NetApp FAS NFS Connector for Hadoop](#).

NetApp FAS NFS Connector for Hadoop Design

Figure 3 shows how the NetApp FAS NFS Connector for Hadoop plugs into a typical Hadoop environment.

Figure 3) NetApp FAS NFS Connector for Hadoop.



The NetApp FAS NFS Connector for Hadoop design allows users to run one of two different deployment options:

- Run HDFS as the primary file system and use the connector to analyze data on NFS storage systems (non-HDFS) such as ONTAP. With this approach, users can immediately analyze data on existing NFS-based storage such as NetApp FAS storage arrays. Existing analytics hardware can be used to analyze NFS data, as well.
- Deploy NFS as the primary file system replacing HDFS for the entire Hadoop cluster.

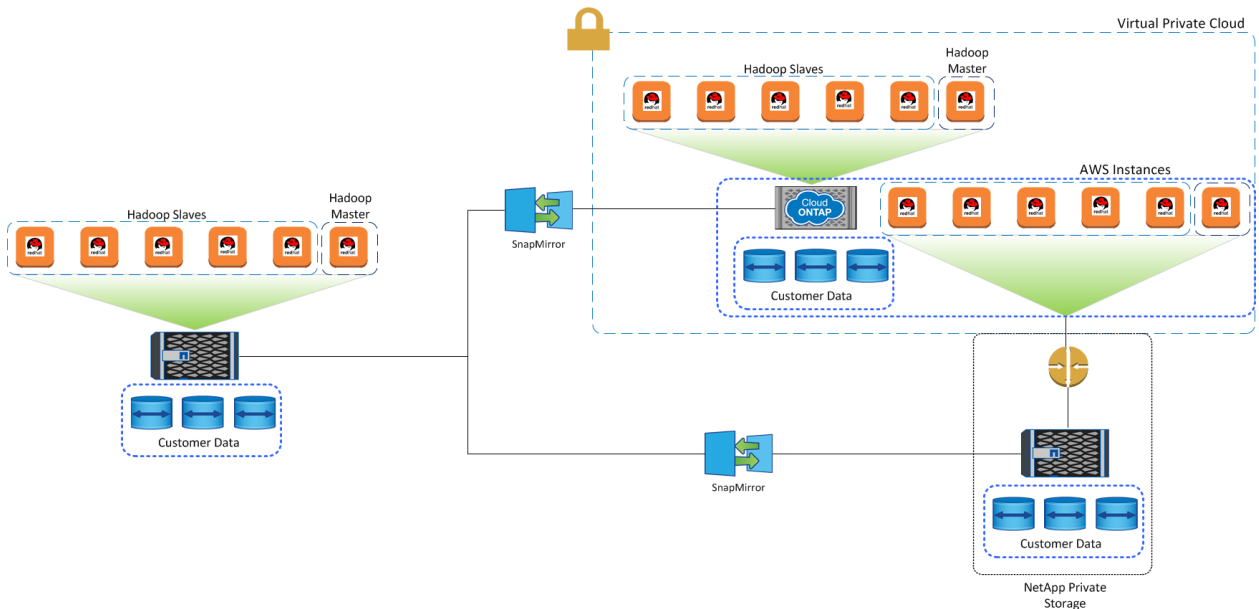
Regardless of the approach taken, the NetApp FAS NFS Connector for Hadoop allows analytics to use existing NetApp technologies such as Snapshot, RAID DP, SnapMirror data replication, and NetApp FlexClone® volumes to provide a rich set of data management features for use with Hadoop.

2.5 Solution Benefits and Use Cases

Cross-Data-Center Deployments

The decoupled design of NetApp FAS NFS Connector for Hadoop allows independent scaling of compute and storage layers. This capability provides the flexibility to place your analytics compute tier on cloud architectures such as Amazon EC2 while keeping the data safely on storage you control on the premises. In these scenarios, up-front hardware purchases are replaced by the pay-as-you-go cloud model specifically on compute. Cross-data-center deployments benefit from products, such as AWS Direct Connect and NPS, that enable high-bandwidth connections between private data centers and public clouds. Figure 4 shows an example of a cross-data-center deployment with Hadoop enabled by technologies such as ONTAP Cloud, NPS, and SnapMirror.

Figure 4) Cross-data-center deployments.



Note: 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.

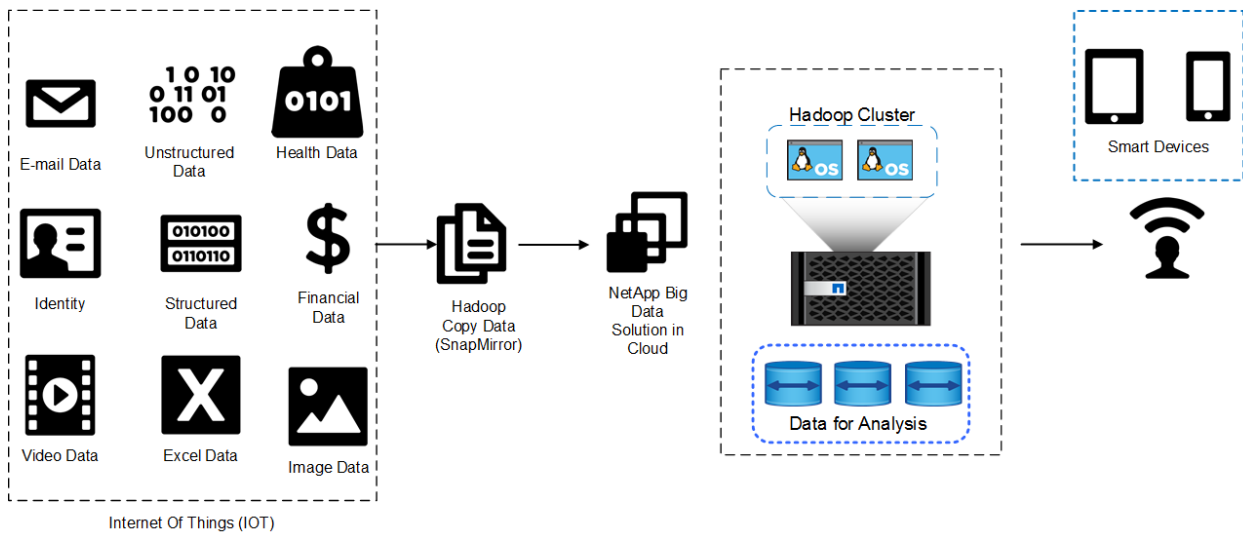
On-Demand Big Data Analytics

You can create analytics applications in public clouds, such as in AWS, for your enterprise by building out compute resources ranging from a few to thousands of servers in minutes. Then you can direct them to operate on the specific data in different locations, and then power them down when finished. This process enables you to perform analytics in less time and at a lower cost compared to that of maintaining physical server resources for analytics. NetApp products such as SnapMirror and NPS for AWS enable this use case by providing users with the flexibility to put data where it can be easily accessed by flexible analytics compute resources in the cloud.

Smart Applications

You can capture and store terabytes of data per hour from various sources such as website clickstreams, financial transactions, social media feeds, IT logs, and location tracking events. This data can be ingested to centralized NetApp storage controllers or transferred to a colocation facility using SnapMirror. The data can then be processed using cloud compute power such as AWS and machine learning to create real time predictions. Figure 5 shows how the typical Internet of Things sources are delivered to a cloud-based analytics environment using NetApp SnapMirror. There the data is analyzed using cloud-based compute resources with results provided in smart applications such as smartphone devices.

Figure 5) Smart applications with Hadoop solution.



Data Warehousing

You can optimize query performance and reduce costs by deploying a data warehousing architecture in a cloud such as AWS. The Apache Hadoop framework can perform an extract, transform, and load (ETL) process and load the processed data into NPS for big data analytics and business intelligence applications.

2.6 Key Hadoop Components

This section describes the key Hadoop components that were tested and validated with NetApp FAS NFS Connector for Hadoop in NPS/AWS to validate a Hadoop environment using the following use cases:

- MapReduce functionality and performance were tested using the following applications:
 - TeraGen: Generates the input data for TeraSort
 - TeraSort: Sorts the data generated
 - TeraValidate: Validates the sorted data output
 - DFSIO: Distributed I/O benchmark for read and write tests on the HDFS and non-HDFS file systems. Tests were conducted in the following areas:
 - Write performance: Benchmark for measuring the write throughput capabilities of the Hadoop cluster
 - Read performance: Benchmark for measuring the read throughput capabilities of the Hadoop cluster
- **Note:** For more validation information, see the section “MapReduce Functionality and Performance Validation.”
- Data analysis:
 - Apache Impala: A SQL query engine on Hadoop through which you can query data in HDFS, non-HDFS, or Apache Hbase.
 - Apache Hive: A data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis through SQL.
 - Apache Pig: A high-level platform for creating MapReduce programs used with Hadoop. The platform uses Pig Latin, which abstracts the programming from the Java MapReduce idiom into a MapReduce programming high level such as SQL for RDBMS.

Note: For more validation information, see the section “Data Analysis Validation.”

- Streaming:
 - mrjobs: A Python 2.6+/3.3+ package that helps you write and run Hadoop streaming jobs.

Note: For more validation information, see the section “Mrjob.”

3 Solution Architecture

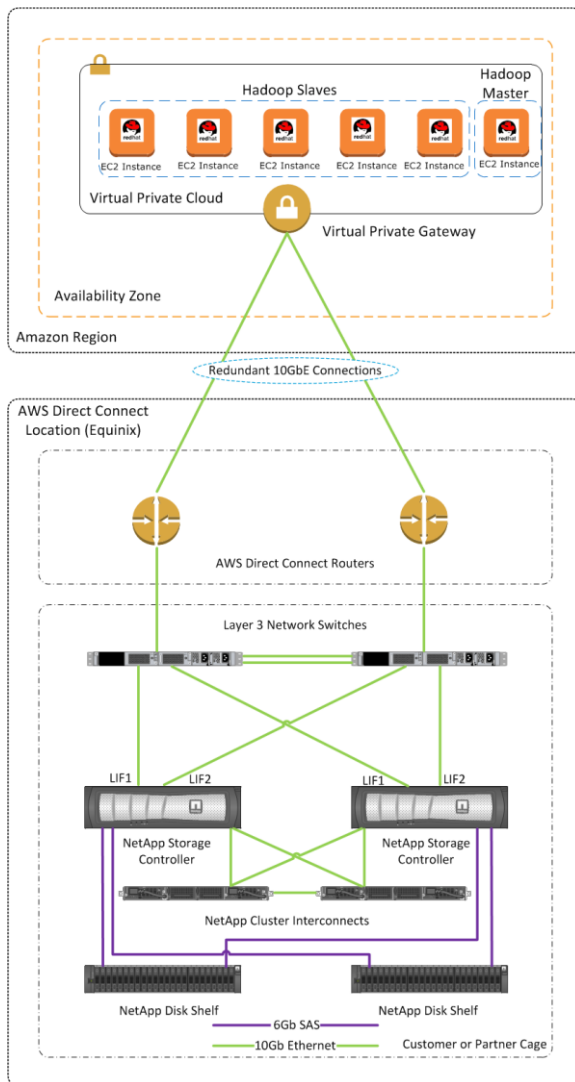
3.1 Network Architecture

The NetApp test setup consisted of a Hadoop cluster with five slaves and one master, which are m4.10xlarge AWS instances. Based on the workload and performance requirements for your specific environment, you can adjust the number and type of AWS instances. The Hadoop nodes running as AWS instances access the NetApp FAS storage located in an NPS colocation facility through Direct Connect routers using two 10GbE redundant connections. One network connection is active and the other is passive. This enables users to achieve a network throughput of approximately 1GBps between the AWS compute resources and the NetApp storage hosted in the NPS colocation facility. It is possible to increase the available bandwidth between the AWS Hadoop nodes and the storage by configuring multiple 10GbE connections.

Two FAS8060 HA pairs of NetApp storage controllers are connected to layer 3 switches in the Equinix data center. The storage controller provides four volumes for the AWS Hadoop instances through the NetApp FAS NFS Connector for Hadoop using 10GbE (TCP/IP) connection.

Figure 6 is an illustration of the network architecture.

Figure 6) Network architecture.



A total of eight LIFs were created for data access, two LIFs for each FAS storage controller. The disk shelves are connected to the FAS controller using 6Gb SAS cables. In this setup, the AWS instances can access the Internet through an AWS public gateway. In Figure 6, each storage controller has 2 x 10Gb Ethernet connections to Layer 3 network switches, which are connected to AWS Direct Connect routers.

3.2 Storage Architecture

Table 1 provides details about the hardware and software components and the configurations used in the NetApp test setup.

Note: An AWS infrastructure with two aggregates was used in the test setup. Each aggregate contained seven volumes. Two volumes from each aggregate were used for Hadoop testing.

Figure 7 is an illustration of the storage architecture.

Figure 7) Storage architecture.

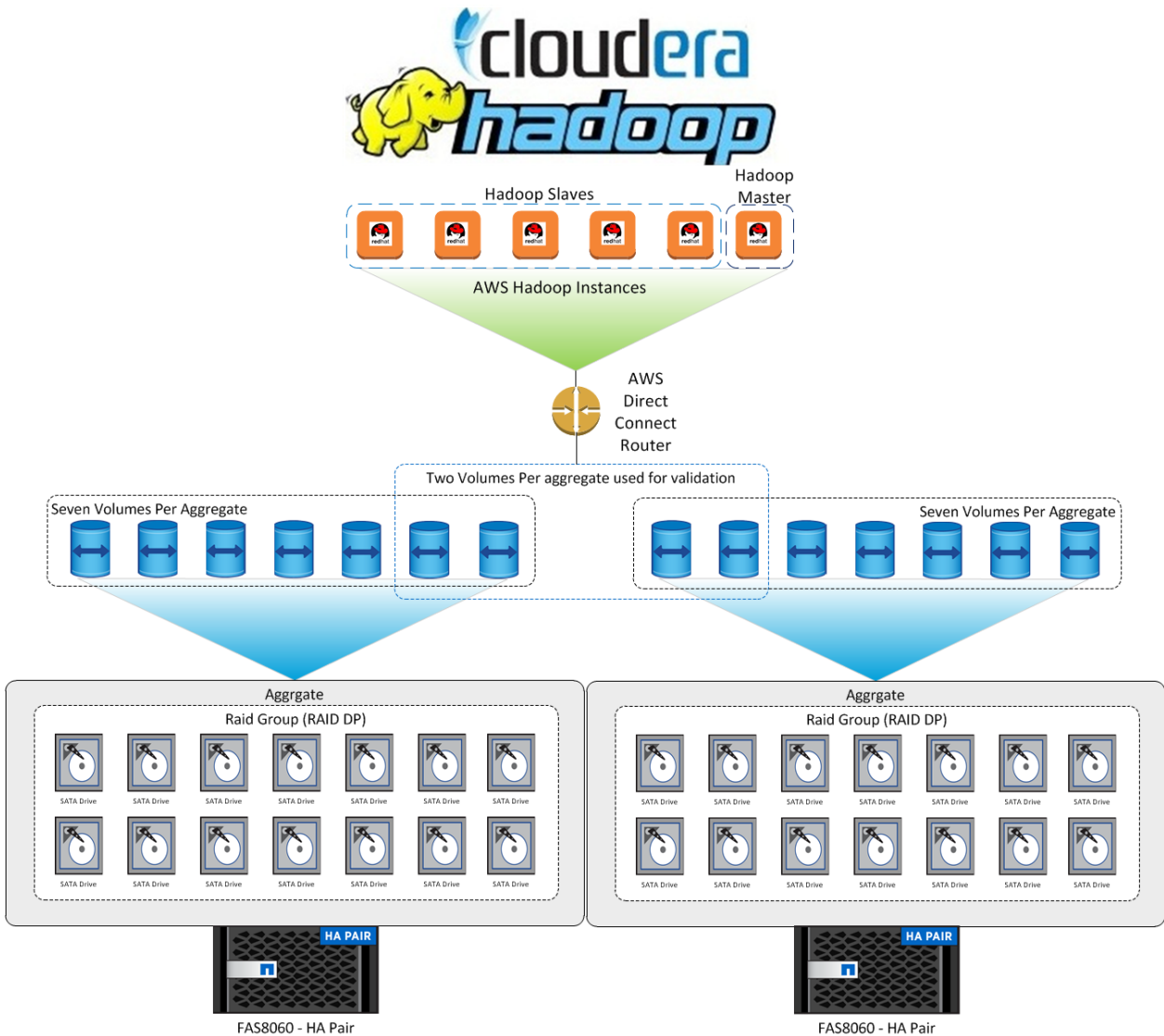


Table 1) Key components of the validated configuration.

Component	Products	Details
Storage	NetApp FAS8060 with clustered Data ONTAP 8.3 or later	<ul style="list-style-type: none">• Four controllers• Two aggregates of size 14 x 3.64TB each• Each aggregate has seven volumes• SATA disk type• 7200 RPM• RAID DP
AWS instances	m4.10xlarge	40 vCPU 160GB RAM
Networking	AWS Direct Connect routers	10Gb
OS	Red Hat Enterprise Linux (RHEL)	7.2 or later works in most of the Linux

Component	Products	Details
	server	distributions
Hadoop distribution	Cloudera distribution	5.5.2

4 MapReduce Functionality and Performance Validation

To validate the performance and stability of this solution, NetApp leveraged a number of standard Hadoop workload tools including TeraGen, TeraSort, TeraValidate, and DFSIO. The results of these tests are described in the following sections.

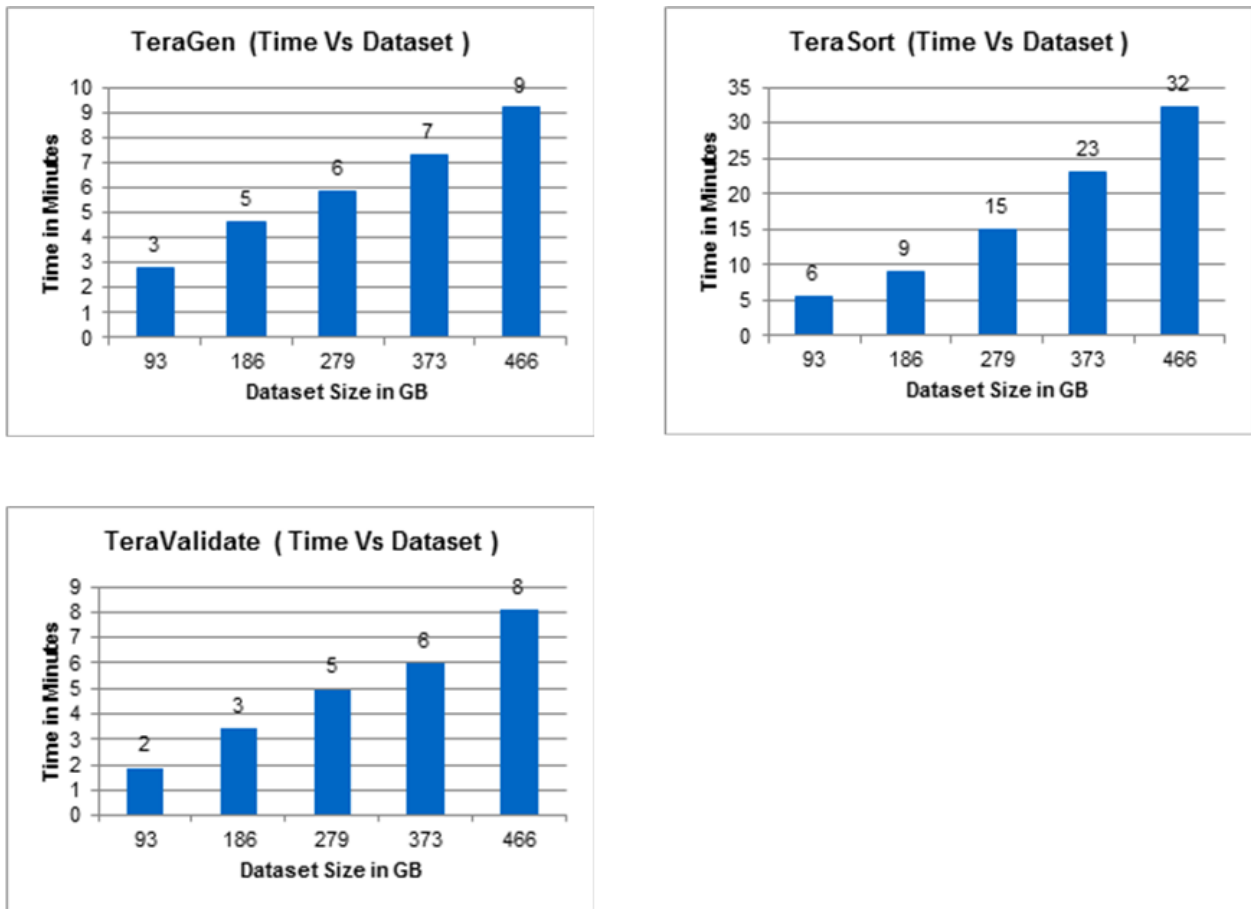
4.1 TeraGen, TeraSort, and TeraValidate

TeraSort is one of the mostly widely used Hadoop benchmarks. Most of the Hadoop distributions bundle it with their distributions. TeraGen generates random data as the input to the TeraSort tool. TeraGen runs on multiple nodes with multiple tasks by specifying the number of maps, which is based on the number of rows generation and the number of cluster nodes. TeraSort sorts the data generated by TeraGen. TeraValidate makes sure that the output produced by TeraSort is globally sorted.

The TeraGen, TeraSort, and TeraValidate operations were performed on the AWS instances whose data was accessed from NPS using the NetApp FAS NFS Connector for Hadoop and the NFS protocol. For these tests, NetApp created and processed a series of ever larger data sets ranging in size from 93GB–to 466GB. NetApp also measured the elapsed time for each of the tools to complete their processing. Generally, the time required to complete the test depends on the size of the dataset, as shown in Figure 8.

Figure 8 graphs the TeraGen, TeraSort, and TeraValidate results.

Figure 8) TeraGen, TeraSort, and TeraValidate results.



In the tested configuration, there was one 10GbE direct connection between the AWS instance and storage; NetApp discovered that this was the primary factor limiting the performance during these tests. As mentioned above, it is possible to increase the number of active connections between the Hadoop nodes and the NPS-based storage if additional performance capacity is required. In any case, NetApp believes that these tests validate the overall stability of the configuration.

Because of the limited network usage as well as the NFS Connector caching feature, the write (TeraGen) and read (TeraValidate) operations were faster than the read/write (TeraSort) operation. For better performance with NFS Connector, make sure to provide multiple direct (active) connections between the AWS instance and NPS.

4.2 DFSIO

During testing, NetApp used the HiBench/DFSIO-E tool to measure the read and write performance of MapReduce jobs behavior between the AWS compute resources and NPS storage through the NetApp FAS NFS Connector for Hadoop. NetApp configured 136 maps per NodeManager, as recommended by the Cloudera tuning YARN. Figure 9 and Figure 10 illustrate the write and read throughput per AWS Hadoop instance.

Note: AWS does not support jumbo frames for Direct Connect. The MTU to be used is 1,500.

Figure 9) DFSIO write throughput per AWS Hadoop instance (MBps).

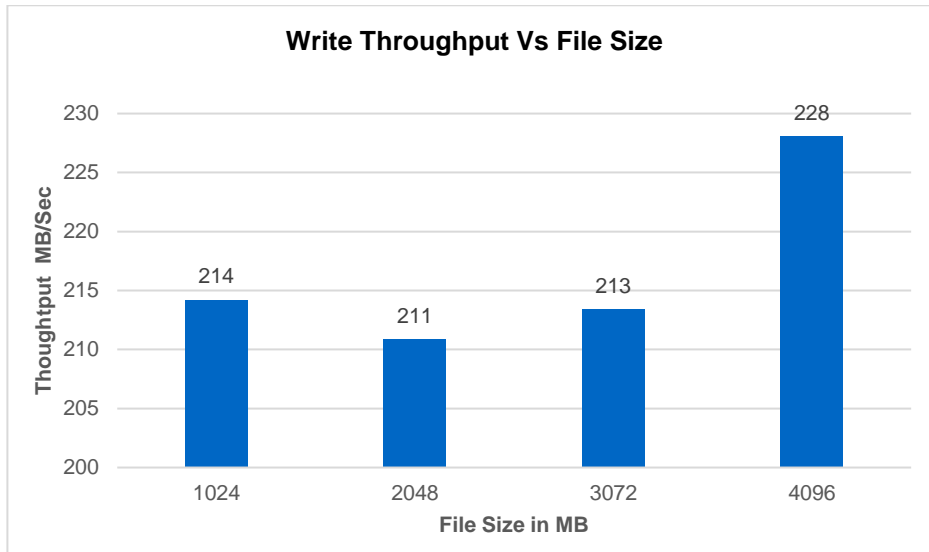
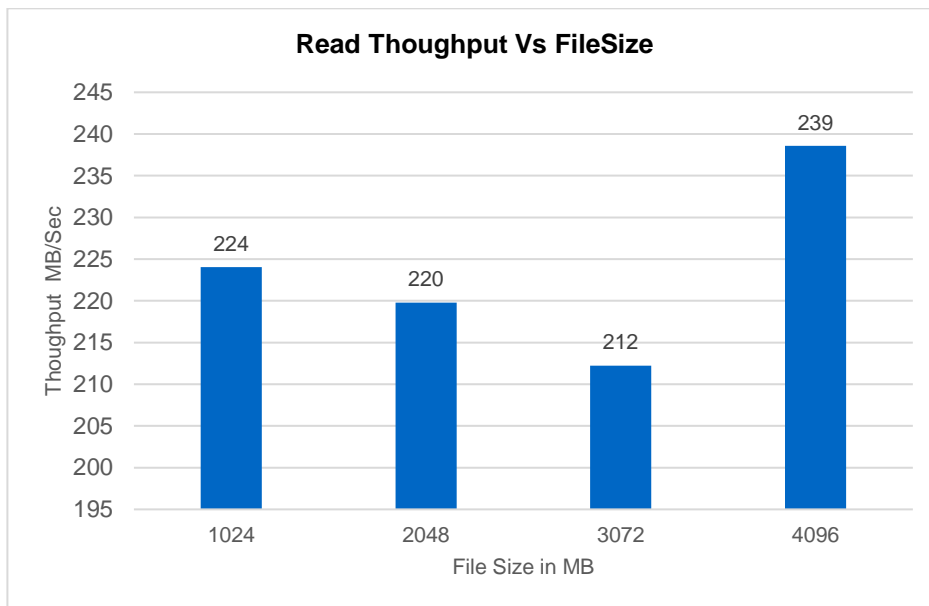


Figure 10) DFSIO read throughput per AWS Hadoop instance (MBps).



Based on the read and write performance results, the AWS Hadoop instance provides better throughput with 4GB file size through NetApp FAS NFS Connector for Hadoop. Additionally, the maximum throughput of 1GBps between AWS instances and NPS was achieved when the throughput of all the AWS instances was combined and was the limiting factor in these tests. Both the FAS8060 storage and AWS-based Hadoop nodes were capable of delivering better performance if additional 10GbE links were applied between AWS and the NPS-based storage systems.

5 Data Analysis Validation

This section discusses additional validation testing that NetApp performed using Hadoop Ecosystem Applications with the NetApp FAS NFS Connector for Hadoop. For this test, NetApp used Impala, Hive, and Pig. The goal of these tests was not to measure performance of the applications but to simply

validate that their basic functionality worked correctly when used with NFS Connector with the NFS protocol.

5.1 Impala

For validation of the NFS Connector with Impala, NetApp used Cloudera Impala 2.6.0, an open-source massively parallel processing SQL query engine for data stored in a computer cluster running Apache Hadoop.

For the NFS Connector to work with Impala, refer to the following procedure:
<http://gerrit.cloudera.org:8080/#/c/1121/5>.

Note

The IMPALA-1850 fix allows the non-HDFS file system to be set as `fs.defaultFS`. Refer to the following link for the procedure and code review: <http://gerrit.cloudera.org:8080/#/c/1121/>.

Basic Validation of Impala with NFS Connector

For the basic validation, NetApp used the following methodology:

1. Create an external table from the CSV file containing all of the records currently available to the Impala application. This file resides in an NFS file system and can be accessed using the NFS Connector.
2. Run a select query to validate the structure of the external table records.

```
ip-172-30-1-57.ec2.internal:21000] > create external table tab3_nfs_new
(
  id INT,
  poilcynumber INT,
  addedrundatecyclenumber INT,
  col_4 BOOLEAN,
  col_5 DOUBLE,
  policydate TIMESTAMP,
  state string
)
row format delimited fields terminated by ','
location 'nfs://172.30.254.29:2049/Hadoop_Vol_1/tab3_nfs';
Query: create external table tab3_nfs_new
(
  id INT,
  poilcynumber INT,
  addedrundatecyclenumber INT,
  col_4 BOOLEAN,
  col_5 DOUBLE,
  policydate TIMESTAMP,
  state string
)
row format delimited fields terminated by ','
location 'nfs://172.30.254.29:2049/Hadoop_Vol_1/tab3_nfs'
```

Fetches 0 row(s) in 0.51s

```
[ip-172-30-1-57.ec2.internal:21000] > select * from tab3_nfs_new;
Query: select * from tab3_nfs_new
```

	id	poilcynumber	addedrundatecyclenumber	col_4	col_5	policydate	state
1	58337846	77964269	false	1.1	2016-04-06 13:23:01	AL	
2	14570261	36802977	false	2.2	2016-04-06 13:23:01	AL	

3	26057863	19326062	false	3.3	2016-04-06 13:23:02
AL					
4	44560941	58568095	false	4.4	2016-04-06 13:23:03
AL					
5	46495575	31203909	false	5.5	2016-04-06 13:23:04
AL					
6	86886735	79571645	false	6.6	2016-04-06 13:23:05
AL					
7	12648737	88157410	false	7.7	2016-04-06 13:23:06
AL					
8	45563151	13580238	false	8.8000000000000001	2016-04-06 13:23:07
AL					
9	89106722	46517872	false	9.9	2016-04-06 13:23:08
AL					
10	74724602	10586628	false	11	2016-04-06 13:23:08
AL					

For more complicated Impala queries that join multiple tables such as select and joins, perform the following procedure to configure the NFS data through the NFS Connector for Hadoop to access the data from Cloudera Impala:

1. Retain the default file system (HDFS) and restart the Hadoop cluster.
2. Verify that the default file system is HDFS by running the `hadoop fs -df -h` command.
3. Create and verify (select query) an external table based on the CSV file, which is located in the non-HDFS file system through the NFS Connector.
4. Verify the `desc` and `select` queries for the NFS tables. A message is displayed indicating that it failed for the NFS tables.
5. Restart only the Impala instances from the Impala page in Cloudera Manager.
6. Run the `select` and `desc` query works for table `tab1` and `tab2`; their location is `nfs://172.30.254.29:2049/user/impala/sample_data/tab1` and `nfs://172.30.254.29:2049/user/impala/sample_data/tab2`.

Note: For testing purposes, NetApp updated the `/usr/lib64/cmf/service/impala/impala.sh` file in one of the Impala servers as well as the metadata and catalog server.

```
export HADOOP_CONF_CLASSPATH=$(/bin/hadoop classpath)

JDBC_JARS="$CLOUDERA_MYSQL_CONNECTOR_JAR:$CLOUDERA_POSTGRESQ_L_JDBC_JAR:$CLOUDERA_ORACLE_CONNECTOR_JAR:$HADOOP_CONF_CLASSPATH"

if [ "impalad" = "$1" ]; then
    exec "$IMPALA_HOME/../../bin/impalad" --flagfile=${FLAG_FILE} --abort_on_config_error=false ${IMPALA_SERVER_ARGS}
elif [ "statestore" = "$1" ]; then
    exec "$IMPALA_HOME/../../bin/statestored" --flagfile=${FLAG_FILE} --abort_on_config_error=false ${IMPALA_SERVER_ARGS}
elif [ "catalogd" = "$1" ]; then
    exec "$IMPALA_HOME/../../bin/catalogd" --flagfile=${FLAG_FILE} --abort_on_config_error=false ${IMPALA_SERVER_ARGS}
fi
```

7. Update the `fs.defaultfs` in `core-site.xml` using Snippet (Safety Value) to change the default file system from HDFS to NFS.
8. After the update, do not restart the HDFS instances. Wait until the cluster perspective of the primary file systems is HDFS before restarting the service.
9. Restart the `stale config` from the HDFS panel and then redeploy the configuration.

Note: During testing, NetApp changed the configuration of only one Impala server. You can try testing the configuration for multiple servers.

10. The default file system changes from HDFS to NFS.

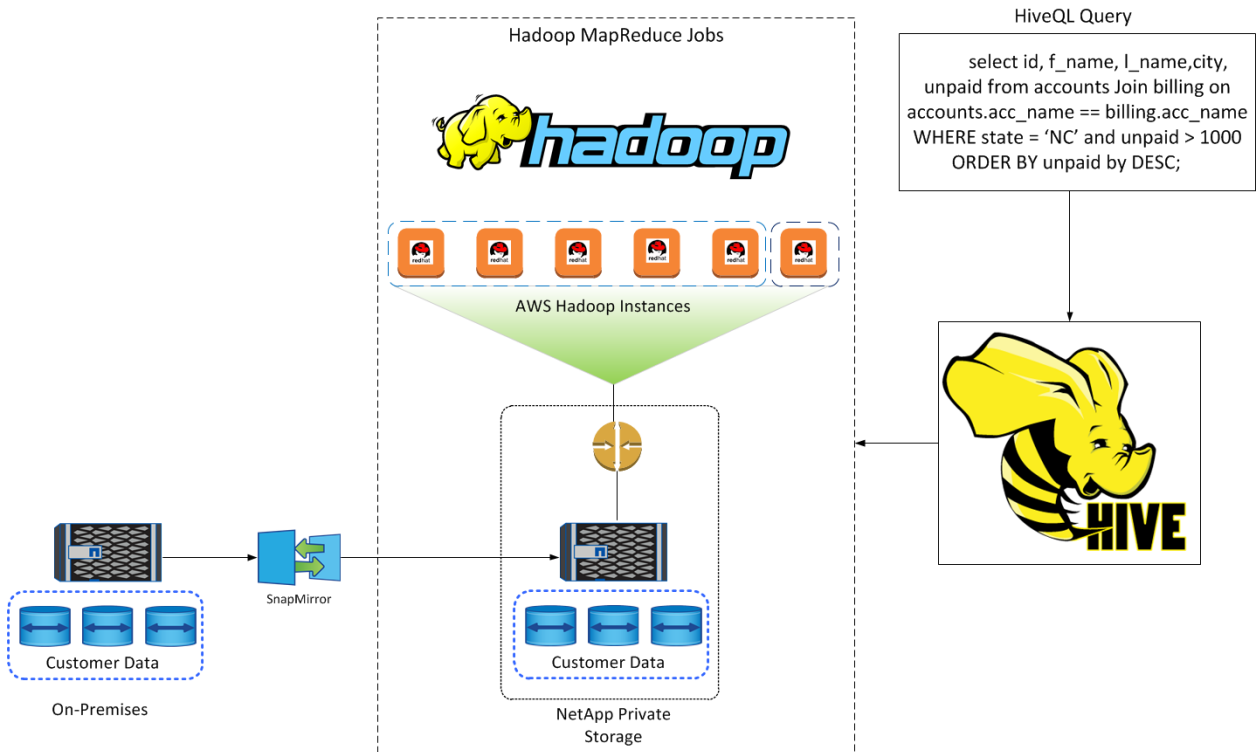
11. Verify the join query.

Note: For more information, see the section “Impala Qualification with NetApp FAS NFS Connector for Hadoop for Hadoop for Join Query” in the appendix.

5.2 Hive

To validate the NetApp FAS NFS Connector with Hive, NetApp used Cloudera Hive to run SQL queries on custom-built warehouse data using the CSV file that contains the records accessible to the database for `select` and `join` operations in the AWS Hadoop instance and NPS storage through NetApp FAS NFS Connector. In this validation, NetApp kept the CSV file data in an NPS location that was accessed using the NFS protocol through the NetApp FAS NFS Connector for Hadoop. Figure 11 shows the HiveQL query execution with Hadoop instances running in AWS.

Figure 11) Hive with NetApp NFS FAS Connector for Hadoop.



During testing, NetApp had to validate the Hive functionality through SQL queries through NetApp FAS NFS Connector for Hadoop. NetApp created two tables with several million records and successfully performed the `select`, `create`, `desc`, and `join` queries. The join queries used MapReduce programs to run on the CSV files that contain the records accessible to the database. These records reside in the NFS volume through NetApp FAS NFS Connector for Hadoop.

NetApp tested the following queries in Hive through NFS Connector; you can run other queries based on workload.

1. Based on the testing, set `hive.auto.convert.join` to false before running the queries.

```
set hive.auto.convert.join=false;
```

2. Change `mapred.reduce.tasks` from 1 to 20 or the desired value in the Hive configuration.

3. Verify the database by running the `show` command.

```
show databases
```

4. Change into the NFS-specific database by running the `use` command.

```
use nfs_db;
```

5. Verify the tables by running the `show` command.

```
show tables
```

6. Verify the number of records in the first table by running the `select` query.

```
select count(*) from tab3_nfs_new;
```

7. Verify the number of records in the second table by running the `select` query.

```
select count(*) from tab4_nfs_new;
```

8. Run the join query by running the `select` query on both tables.

```
select count(*) from tab3_nfs_new join tab4_nfs_new where tab3_nfs_new.id = tab4.nfs_new.id;
```

9. You can run all of the SQL queries; for example, to limit the number of records, run the `limit` query.

```
select count(*) from tab3_nfs_new join tab4_nfs_new where tab3_nfs_new.id = tab4.nfs_new.id limit 100;
```

Note: For detailed results, see the section titled “Hive” in the appendix.

Based on the testing validation, NetApp was able to run the join query against two tables as well as basic SQL queries.

6 Apache Pig

Apache Pig is a client-side application for data analysis and processing on Hadoop. The application executes as MapReduce jobs and uses a textual language known as Pig Latin for processing data.

Figure 12) Pig deployment in AWS through NetApp NFS FAS Connector and NPS.

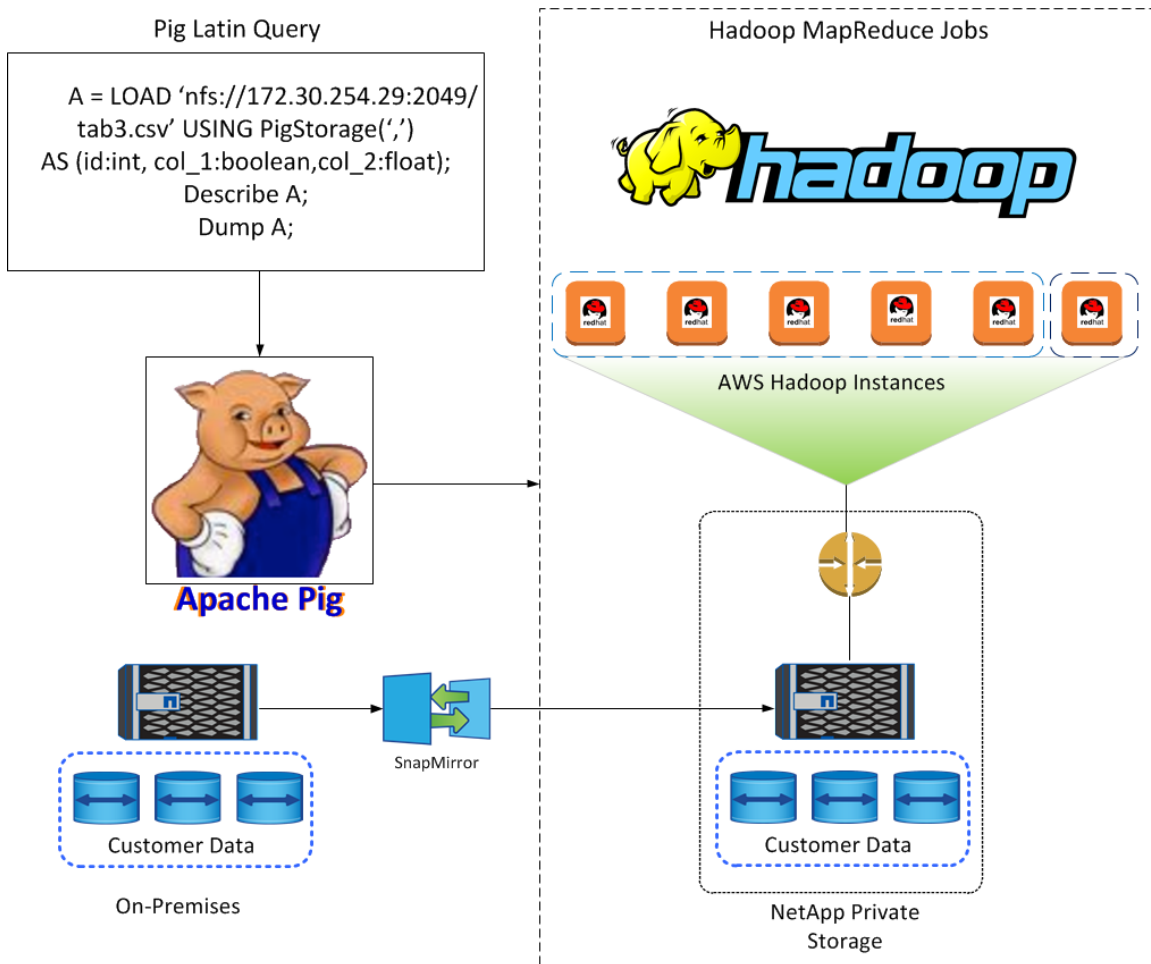


Figure 12 shows the Pig deployment in NPS. For the basic testing validation, NetApp used the following methodology with Pig Latin:

1. Read the data from the CSV file that contains the records accessible to the database by using a LOAD statement.
2. Used transformation statements to process the data.
3. Generated the output of the results by using the DUMP statement.

Note: For the DUMP query to generate data from the NFS data tables, see the section “Pig” in the appendix.

Based on the Pig validation through NFS Connector, NetApp was able to execute and process the Pig Latin queries in the Hadoop AWS instance on data that is in NPS.

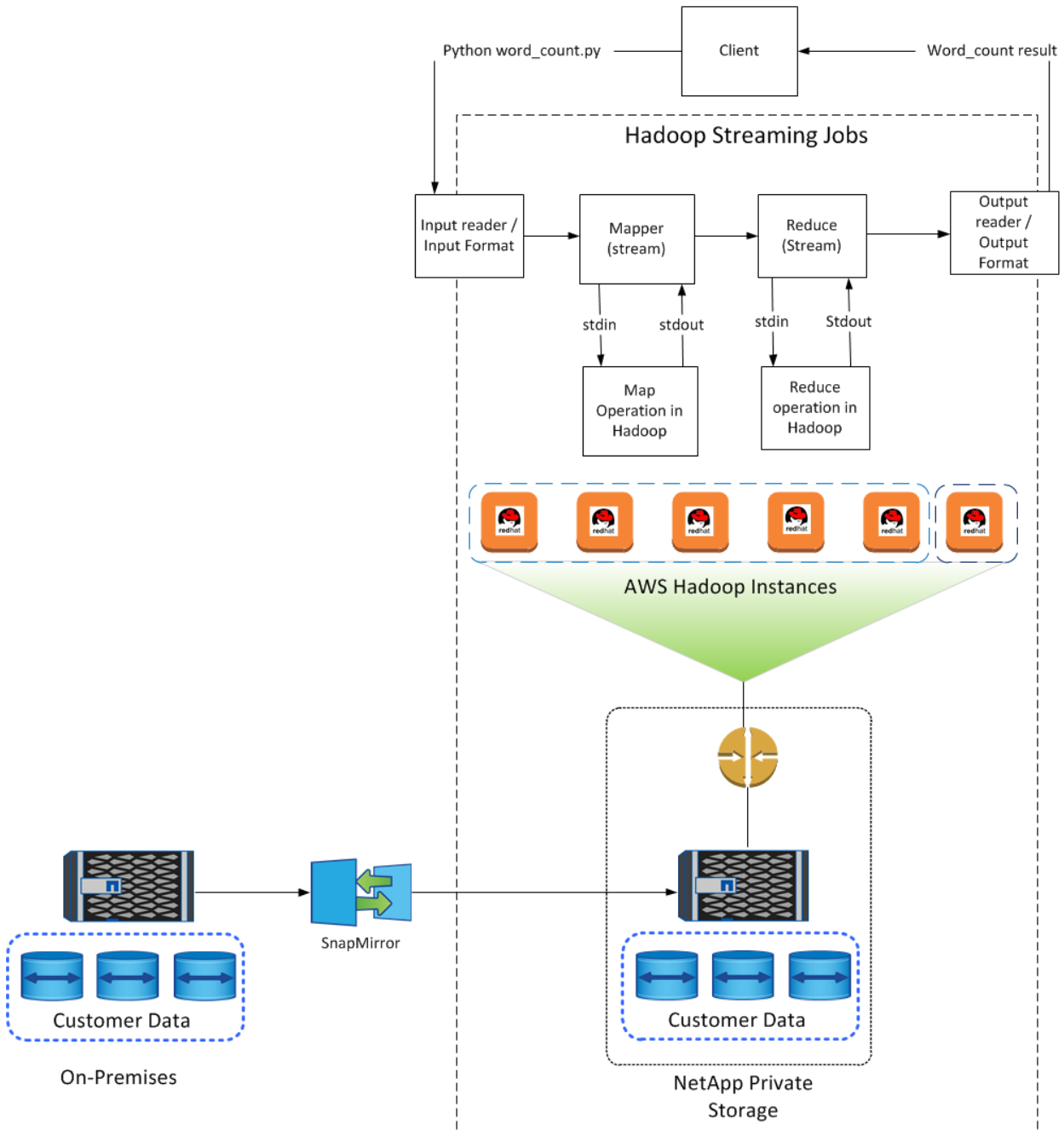
Note: NetApp will provide the Pig performance information in the performance validation update of this paper.

7 Mrjob

The mrjob can be used to execute the Hadoop streaming jobs using non-Java programs such as Python. The following example is an mrjob query on NFS data. The data is kept in the NFS file shares and accessed by using NFS Connector.

For our streaming job validation of the NFS Connector, NetApp used the open-source WordCount python program to count the number of characters, lines, and words from the CSV file that is accessed from the NFS volume from NPS storage through the NFS protocol. Figure 13 shows the workflow of streaming jobs with mapper and reducers jobs running in stream for map and reduce operations.

Figure 13) Hadoop streaming job with NFS FAS Connector for Hadoop.



NetApp tested the following queries in streaming through NFS Connector; however, you can run other queries based on your workload:

1. Log in as a Hadoop user such as `yarn` user.

2. Copy the CSV file from the OS to the Hadoop non-HDFS file system. In this example, the NFS file system through NPS storage through the NFS Connector.
3. Verify the CSV file by using the Hadoop CLI command.
4. Run the Python `word_count` program against the CSV file and verify the results. Verify the following command results:

```
[root@ip-172-30-1-117 ~]# su - yarn
Last login: Tue Apr 12 11:15:35 EDT 2016 on pts/0
[yarn@ip-172-30-1-117 ~]$ hadoop fs -ls -h /
Found 2 items
drwxrwxrwt   - hdfs supergroup          0 2016-02-25 16:52 /tmp
drwxr-xr-x   - hdfs supergroup          0 2016-04-04 14:04 /user
[yarn@ip-172-30-1-117 ~]$ hadoop dfs -put /tmp/tab_allstate10_n1.csv /tmp
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

[yarn@ip-172-30-1-117 ~]$
[yarn@ip-172-30-1-117 ~]$ hadoop fs -ls -h /tmp
Found 4 items
drwxrwxrwx   - hdfs   supergroup          0 2016-04-12 11:37
/tmp/.cloudera_health_monitoring_canary_files
drwx-wx-wx   - hive   supergroup          0 2016-04-04 20:18 /tmp/hive
drwxrwxrwt   - mapred hadoop              0 2016-04-04 14:07 /tmp/logs
-rw-r--r--   1 yarn   supergroup    220.3 M 2016-04-12 11:38 /tmp/tab_allstate10_n1.csv
[yarn@ip-172-30-1-117 ~]$

yarn@ip-172-30-1-117 nfsconnectorjars_from_121]$ python mr_word_count.py
/tmp/tab_allstate10_n1.csv
No configs found; falling back on auto-configuration
Creating temp directory /tmp/mr_word_count.yarn.20160412.153925.700419
Running step 1 of 1...
Streaming final output from /tmp/mr_word_count.yarn.20160412.153925.700419/output...
"chars"      227246458
"lines"      3716616
"words"      7433232
Removing temp directory /tmp/mr_word_count.yarn.20160412.153925.700419...
[yarn@ip-172-30-1-117 nfsconnectorjars_from_121]$
```

8 Apache Spark

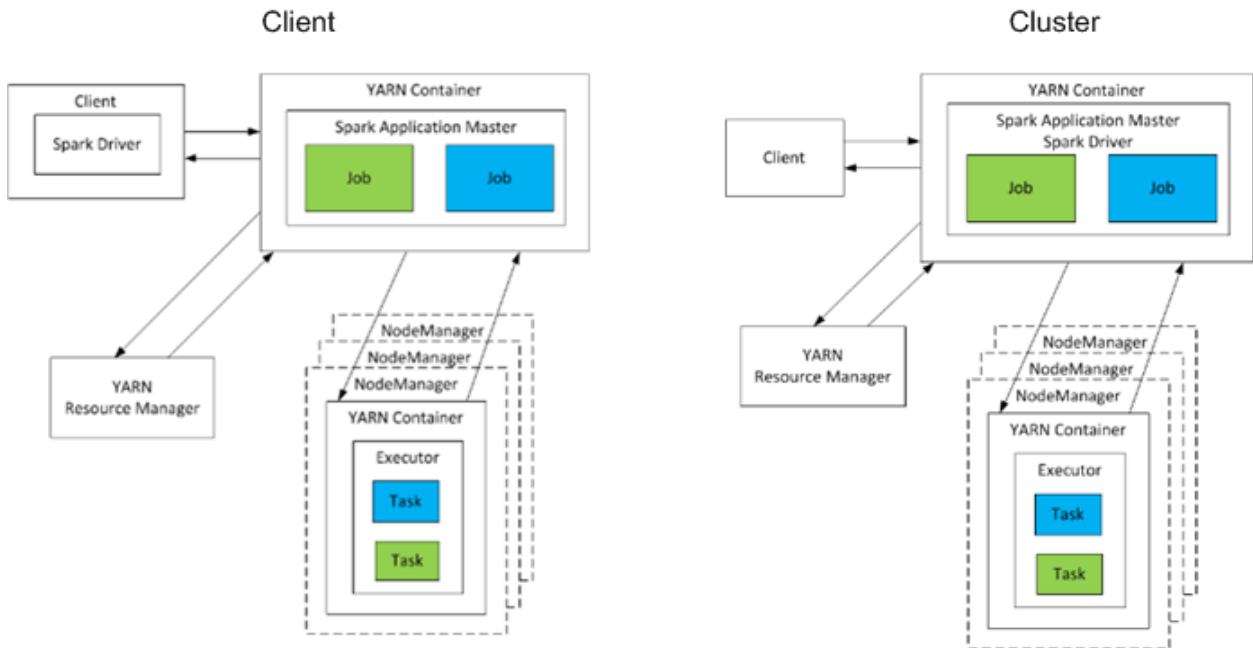
Apache Spark is a fast, general engine for large-scale data processing on a cluster (including Map and Reduce) as well as for nonbatch processing models. The engine works near-real-time processing and allows configurable in-memory data caching for efficient iteration. It processes applications in a distributed manner across worker nodes to provide distributed storage that is horizontally scalable and fault tolerant. Spark applications are written in Python, Scala, or Java, and consist of one or more tasks. The Spark applications have a Spark driver that runs in YARN client mode (the driver runs on the client) mode or YARN cluster mode (the driver runs on the cluster on the application master, which continue to run even if the client disconnects).

Spark applications include one or more Spark jobs. Jobs run tasks in the executors and the executors run in YARN containers. Each executor runs in a single container. Executors exist throughout the life of an application. The executor is fixed once the application starts and YARN does not resize the already allocated container. An executor can run tasks concurrently on in-memory data.

Apache service has two versions: Spark on YARN, which requires HDFS and YARN and is supported in the latest version of the popular Hadoop distributions; and Spark standalone. Both versions come with the Spark History server.

Figure 14 shows the Spark deployment modes. In the testing validation, NetApp wanted to make sure that Spark applications would run with NetApp NFS Connector in NPS and provide some guidelines based on basic performance results with the WordCount program in Spark through the client deployment mode.

Figure 14) Spark deployment modes.



To validate whether the Apache Spark functionality worked with the NFS Connector, NetApp completed the following procedure:

1. In the Hadoop cluster, we modified the primary file system (HDFS) to non-HDFS (in this case, NFS) by changing the `fs.defaultFS` parameter and NFS Connector-specific dependency parameters in `core-site.xml`.
2. We updated the `users.json` and `groups.json` with the Spark user and group details, which are part of NFS Connector configurations.
3. We copied the NFS Connector to the Spark library locations.
4. We updated the connector with execute permissions by running the `chmod` command (if you do not have execute permission).
5. We verified the basic WordCount functionality by running the cluster deploy mode.

The following command validates JavaWordCount:

- deploy-mode client:

```
spark-submit --class org.apache.spark.examples.JavaWordCount --master yarn --deploy-mode client
--num-executors 300 --executor-memory 2G --conf "spark.kryoserializer.buffer.max=512m" --conf
"spark.rdd.compress=true" --conf "spark.storage.memoryFraction=1" --
conf="spark.core.connection.ack.wait.timeout=600" --conf="spark.akka.frameSize=320" --conf
"spark.default.parallelism=320" --conf="spark.executor.cores=2"
/opt/cloudera/parcels/CDH/jars/spark-examples-1.5.0-cdh5.5.2-hadoop2.6.0-cdh5.5.2.jar
/Hadoop_Vol_1/tab_allstate600g.csv /Hadoop_Vol_1/resultJavaWordCount_600g
```

- deploy-mode cluster:

```
spark-submit --class org.apache.spark.examples.JavaWordCount --master yarn --deploy-mode cluster
--num-executors 300 --executor-memory 2G --conf "spark.kryoserializer.buffer.max=512m" --conf
"spark.rdd.compress=true" --conf "spark.storage.memoryFraction=1" --
conf="spark.core.connection.ack.wait.timeout=600" --conf="spark.akka.frameSize=320" --conf
"spark.default.parallelism=320" --conf="spark.executor.cores=2"
/opt/cloudera/parcels/CDH/jars/spark-examples-1.5.0-cdh5.5.2-hadoop2.6.0-cdh5.5.2.jar
/Hadoop_Vol_1/tab_allstate500g.csv /Hadoop_Vol_1/resultJavaWordCount_500g
```


Figure 15) Apache Spark performance validation in AWS with single 10GbE Direct Connect.

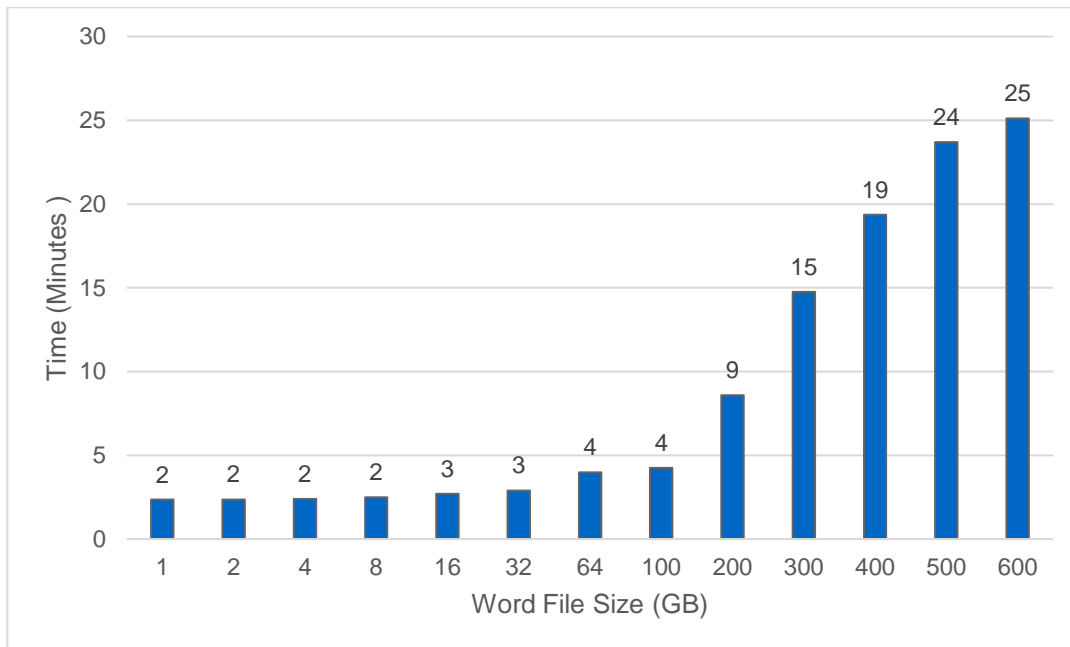


Figure 15 shows the test results with an m4.10xlarge instance. The results indicate that for smaller file sizes of up to 100GB, the WordCount operation completes in approximately the same time as the file fits in AWS Instance memory. In addition, the operation doesn't require access to the disk. A file size greater than 200GB requires storage access with completion time increasing with the word file size.

Summary

This report provides detailed information about NetApp FAS NFS Connector deployment in the Data Fabric enabled by NetApp Data Fabric, which uses compute resources from AWS and storage from NPS through the underlying architecture and use cases. This document demonstrates NFS Connector's basic Hadoop MapReduce functionality validation. It also demonstrates Hadoop Ecosystem validation for Hive, Pig, and mrjobs in AWS with NPS and basic performance results for some of them. In addition, it shows that the NFS Connector works with Apache Spark, which is demonstrated through a sample program. As described in this document, NFS Connector supports key Hadoop ecosystem projects such as Apache Hadoop, Apache Spark, Hive, Pig, Impala, and mrjobs in AWS with NPS.

Appendixes

Impala (Create Table and Select Query)

For testing purposes, NetApp created a table and executed the select query through Impala.

1. Create a table with a comma-separated list of rows.
2. Copy the rows into Hadoop using the `hadoop fs -put` command through NFS Connector.

```
root@ip-172-30-1-117 nfsconnectorjars_from_121]# hadoop fs -put tab_allstate10.csv
nfs://172.30.254.29:2049/Hadoop_Vol_1/
Picked up _JAVA_OPTIONS: -Xmx14000M
16/04/25 13:41:45 INFO nfs.NFSv3FileSystem: User config file:
/etc/NetAppNFSConnector/conf/users.json
16/04/25 13:41:45 INFO nfs.NFSv3FileSystem: Group config file:
/etc/NetAppNFSConnector/conf/groups.json
```

```

Store with ep Endpoint: host=nfs://172.30.254.29:2049/ export=/ path=/ has fsId 2147484980
Store with ep Endpoint: host=nfs://172.30.254.37:2049/ export=null path=/Hadoop_Vol_1/ has fsId
2147484980
16/04/25 13:42:06 WARN stream.NFSBufferedOutputStream: Flushing a closed stream. Check your code.
16/04/25 13:42:06 INFO stream.NFSBufferedOutputStream: STREAMSTATSstreamStatistics:
STREAMSTATS name: class
org.apache.hadoop.fs.nfs.stream.NFSBufferedInputStream/Hadoop_Vol_1/tab_allstate10.csv._COPYING_
STREAMSTATS streamID: 1
STREAMSTATS =====OutputStream Statistics=====
STREAMSTATS BytesWritten: 904546018
STREAMSTATS writeOps: 13803
STREAMSTATS timeWritten: 20.498 s
STREAMSTATS =====NFS Write Statistics=====
STREAMSTATS BytesNFSWritten: 904546018
STREAMSTATS NFSWriteOps: 13803
STREAMSTATS timeNFSWritten: 20.324 s
STREAMSTATS =====Bandwidth=====
STREAMSTATS Write: 42.084 MB/s
STREAMSTATS NFSWrite: 42.445 MB/s
STREAMSTATS =====Average Latency=====
STREAMSTATS Write: 1.485 ms
STREAMSTATS NFSWrite: 1.472 ms

16/04/25 13:42:06 INFO stream.NFSBufferedOutputStream: OutputStream shutdown took 7 ms
16/04/25 13:42:06 WARN stream.NFSBufferedOutputStream: Closing an already closed output stream
16/04/25 13:42:06 INFO stream.NFSBufferedOutputStream: STREAMSTATSstreamStatistics:
STREAMSTATS name: class
org.apache.hadoop.fs.nfs.stream.NFSBufferedInputStream/Hadoop_Vol_1/tab_allstate10.csv._COPYING_
STREAMSTATS streamID: 1
STREAMSTATS =====OutputStream Statistics=====
STREAMSTATS BytesWritten: 904546018
STREAMSTATS writeOps: 13803
STREAMSTATS timeWritten: 20.498 s
STREAMSTATS =====NFS Write Statistics=====
STREAMSTATS BytesNFSWritten: 904546018
STREAMSTATS NFSWriteOps: 13803
STREAMSTATS timeNFSWritten: 20.324 s
STREAMSTATS =====Bandwidth=====
STREAMSTATS Write: 42.084 MB/s
STREAMSTATS NFSWrite: 42.445 MB/s
STREAMSTATS =====Average Latency=====
STREAMSTATS Write: 1.485 ms
STREAMSTATS NFSWrite: 1.472 ms

16/04/25 13:42:06 INFO stream.NFSBufferedOutputStream: OutputStream shutdown took 0 ms
16/04/25 13:42:06 INFO nfs.NFSv3FileSystemStore: RPC RENAME took 1 ms

```

3. Verify the files in the Hadoop NFS file system.

```

[root@ip-172-30-1-117 nfsconnectorjars_from_121]# hadoop fs -ls /
Picked up _JAVA_OPTIONS: -Xmx14000M
16/04/25 13:42:14 INFO nfs.NFSv3FileSystem: User config file:
/etc/NetAppNFSConnector/conf/users.json
16/04/25 13:42:14 INFO nfs.NFSv3FileSystem: Group config file:
/etc/NetAppNFSConnector/conf/groups.json
Store with ep Endpoint: host=nfs://172.30.254.29:2049/ export=/ path=/ has fsId 2147484980
Found 10 items
drwxrwxrwx - root root 4096 2016-04-25 13:05 /.snapshot
drwxrwxrwx - root root 4096 2016-04-25 13:41 /Hadoop_Vol_1
drwxrwxrwx - root root 4096 2016-04-25 13:10 /Hadoop_Vol_2
drwxrwxrwx - root root 4096 2016-04-25 13:10 /Hadoop_Vol_3
drwxrwxrwx - root root 4096 2016-04-23 22:27 /Hadoop_Vol_4
drwxrwxrwx - root root 4096 2016-03-14 16:53 /benchmarks
-rw-r--r-- 1 root root 61326 2016-03-08 14:21 /terasort-1.0-SNAPSHOT-2.jar
drwxrwxrwx - root root 4096 2016-04-18 19:28 /tmp
drwxrwxrwx - root root 4096 2016-03-14 10:04 /tttt
drwxrwxrwx - root root 4096 2016-04-04 17:01 /user
^[A[root@ip-172-30-1-117 nfsconnectorjars_from_121]# hadoop fs -ls /Hadoop_Vol_1
Picked up _JAVA_OPTIONS: -Xmx14000M
16/04/25 13:42:17 INFO nfs.NFSv3FileSystem: User config file:
/etc/NetAppNFSConnector/conf/users.json

```

```

16/04/25 13:42:17 INFO nfs.NFSv3FileSystem: Group config file:
/etc/NetAppNFSConnector/conf/groups.json
Store with ep Endpoint: host=nfs://172.30.254.29:2049/ export=/ path=/ has fsId 2147484980
Store with ep Endpoint: host=nfs://172.30.254.37:2049/ export=null path=/Hadoop_Vol_1/ has fsId
2147484980
Found 4 items
-rw-r--r-- 1 root root 4296 2016-04-18 16:58 /Hadoop_Vol_1/README.rst
-rw-r--r-- 1 root root 904546018 2016-04-25 13:41 /Hadoop_Vol_1/tab_allstate10.csv
-rw-r--r-- 1 root 1 0 2016-03-07 12:25 /Hadoop_Vol_1/test
drwxr-xr-x - root root 4096 2016-03-14 10:08 /Hadoop_Vol_1/tttttttt
[root@ip-172-30-1-117 nfsconnectorjars_from_121]#

[root@ip-172-30-1-117 ~]# hadoop fs -ls -h /Hadoop_Vol_1
Picked up _JAVA_OPTIONS: -Xmx140000M
16/04/26 11:59:01 INFO nfs.NFSv3FileSystem: User config file:
/etc/NetAppNFSConnector/conf/users.json
16/04/26 11:59:01 INFO nfs.NFSv3FileSystem: Group config file:
/etc/NetAppNFSConnector/conf/groups.json
Store with ep Endpoint: host=nfs://172.30.254.29:2049/ export=/ path=/ has fsId 2147484980
Store with ep Endpoint: host=nfs://172.30.254.37:2049/ export=null path=/Hadoop_Vol_1/ has fsId
2147484980
Found 4 items
-rw-r--r-- 1 root root 4.2 K 2016-04-18 16:58 /Hadoop_Vol_1/README.rst
drwxrwxrwx - impala impala 4 K 2016-04-25 17:36 /Hadoop_Vol_1/tab3_nfs
-rw-r--r-- 1 root 1 0 2016-03-07 12:25 /Hadoop_Vol_1/test
drwxr-xr-x - root root 4 K 2016-03-14 10:08 /Hadoop_Vol_1/tttttttt
[root@ip-172-30-1-117 ~]# hadoop fs -ls -h /Hadoop_Vol_1/tab3_nfs
Picked up _JAVA_OPTIONS: -Xmx140000M
16/04/26 11:59:15 INFO nfs.NFSv3FileSystem: User config file:
/etc/NetAppNFSConnector/conf/users.json
16/04/26 11:59:15 INFO nfs.NFSv3FileSystem: Group config file:
/etc/NetAppNFSConnector/conf/groups.json
Store with ep Endpoint: host=nfs://172.30.254.37:2049/ export=null path=/Hadoop_Vol_1/ has fsId
2147484980
Store with ep Endpoint: host=nfs://172.30.254.29:2049/ export=/ path=/ has fsId 2147484980
Found 1 items
-rw-r--r-- 1 root root 862.6 M 2016-04-25 13:41 /Hadoop_Vol_1/tab3_nfs/tab_allstate10.csv
[root@ip-172-30-1-117 ~]#
[root@ip-172-30-1-117 ~]#

```

4. Create a folder and move the CSV file into the new folder.

```

[impala@ip-172-30-1-117 ~]$ hadoop dfs -mkdir nfs://172.30.254.29:2049/Hadoop_Vol_1/tab3_nfs
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Picked up _JAVA_OPTIONS: -Xmx140000M
16/04/25 17:36:54 INFO nfs.NFSv3FileSystem: User config file:
/etc/NetAppNFSConnector/conf/users.json
16/04/25 17:36:54 INFO nfs.NFSv3FileSystem: Group config file:
/etc/NetAppNFSConnector/conf/groups.json
Store with ep Endpoint: host=nfs://172.30.254.37:2049/ export=null path=/Hadoop_Vol_1/ has fsId
2147484980
Store with ep Endpoint: host=nfs://172.30.254.29:2049/ export=/ path=/ has fsId 2147484980
[impala@ip-172-30-1-117 ~]$ hadoop dfs -mv
nfs://172.30.254.29:2049/Hadoop_Vol_1/tab_allstate10.csv
nfs://172.30.254.29:2049/Hadoop_Vol_1/tab3_nfs
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Picked up _JAVA_OPTIONS: -Xmx140000M
16/04/25 17:37:23 INFO nfs.NFSv3FileSystem: User config file:
/etc/NetAppNFSConnector/conf/users.json
16/04/25 17:37:23 INFO nfs.NFSv3FileSystem: Group config file:
/etc/NetAppNFSConnector/conf/groups.json
Store with ep Endpoint: host=nfs://172.30.254.37:2049/ export=null path=/Hadoop_Vol_1/ has fsId
2147484980
Store with ep Endpoint: host=nfs://172.30.254.29:2049/ export=/ path=/ has fsId 2147484980
16/04/25 17:37:23 INFO nfs.NFSv3FileSystemStore: RPC RENAME took 11 ms
[impala@ip-172-30-1-117 ~]$

```

```

[impala@ip-172-30-1-117 ~]$ hadoop dfs -ls nfs://172.30.254.29:2049/Hadoop_Vol_1/tab3_nfs
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Picked up JAVA OPTIONS: -Xmx140000M
16/04/25 17:37:48 INFO nfs.NFSv3FileSystem: User config file:
/etc/NetAppNFSConnector/conf/users.json
16/04/25 17:37:48 INFO nfs.NFSv3FileSystem: Group config file:
/etc/NetAppNFSConnector/conf/groups.json
Store with ep Endpoint: host=nfs://172.30.254.37:2049/ export=null path=/Hadoop_Vol_1/ has fsId
2147484980
Store with ep Endpoint: host=nfs://172.30.254.29:2049/ export=/ path=/ has fsId 2147484980
Found 1 items
-rw-r--r-- 1 root root 904546018 2016-04-25 13:41
nfs://172.30.254.29:2049/Hadoop_Vol_1/tab3_nfs/tab_allstate10.csv
[impala@ip-172-30-1-117 ~]$

```

5. Create an external table based on the CSV file.

```

ip-172-30-1-57.ec2.internal:21000] > create external table tab3_nfs_new
(
  id INT,
  poilcynumber INT,
  addedrundatecyclenumber INT,
  col_4 BOOLEAN,
  col_5 DOUBLE,
  policydate TIMESTAMP,
  state string
)
row format delimited fields terminated by ','
location 'nfs://172.30.254.29:2049/Hadoop_Vol_1/tab3_nfs';
Query: create external table tab3_nfs_new
(
  id INT,
  poilcynumber INT,
  addedrundatecyclenumber INT,
  col_4 BOOLEAN,
  col_5 DOUBLE,
  policydate TIMESTAMP,
  state string
)
row format delimited fields terminated by ','
location 'nfs://172.30.254.29:2049/Hadoop_Vol_1/tab3_nfs'

Fetched 0 row(s) in 0.51s
[ip-172-30-1-57.ec2.internal:21000] > select * from tab3_nfs_new;
Query: select * from tab3_nfs_new
+-----+-----+-----+-----+-----+-----+-----+
+-----+
| id | poilcynumber | addedrundatecyclenumber | col_4 | col_5 | policydate | state |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | 58337846 | 77964269 | false | 1.1 | 2016-04-06 13:23:01 | AL |
| 2 | 14570261 | 36802977 | false | 2.2 | 2016-04-06 13:23:01 | AL |
| 3 | 26057863 | 19326062 | false | 3.3 | 2016-04-06 13:23:02 | AL |
| 4 | 44560941 | 58568095 | false | 4.4 | 2016-04-06 13:23:03 | AL |
| 5 | 46495575 | 31203909 | false | 5.5 | 2016-04-06 13:23:04 | AL |
| 6 | 86886735 | 79571645 | false | 6.6 | 2016-04-06 13:23:05 | AL |
| 7 | 12648737 | 88157410 | false | 7.7 | 2016-04-06 13:23:06 | AL |
| 8 | 45563151 | 13580238 | false | 8.8000000000000001 | 2016-04-06 13:23:07 | AL |

```

9	89106722	46517872	false	9.9	2016-04-06 13:23:08
AL					
10	74724602	10586628	false	11	2016-04-06 13:23:08
AL					
1	52498005	45952204	false	2.1	2016-04-06 13:23:10
AK					
2	43044976	74548676	false	3.2	2016-04-06 13:23:10
AK					
3	69391811	43795243	false	4.3	2016-04-06 13:23:11
AK					
4	59557262	23853787	false	5.4	2016-04-06 13:23:12
AK					
5	69819844	53112629	false	6.5	2016-04-06 13:23:13
AK					
6	88569990	86323772	false	7.6	2016-04-06 13:23:14
AK					
7	22493732	76583367	false	8.699999999999999	2016-04-06 13:23:15
AK					
8	54821563	26099181	false	9.800000000000001	2016-04-06 13:23:16
AK					
9	22973786	49945936	false	10.9	2016-04-06 13:23:17
AK					
10	16222913	59371557	false	12	2016-04-06 13:23:17
AK					
1	59004284	26292983	false	3.1	2016-04-06 13:23:18
AZ					
2	94103241	16760969	false	4.2	2016-04-06 13:23:19
AZ					
3	78031154	11901485	false	5.3	2016-04-06 13:23:20
AZ					
4	31930265	15689716	false	6.4	2016-04-06 13:23:21
AZ					
5	19839623	36914067	false	7.5	2016-04-06 13:23:22
AZ					
6	41754633	39857554	false	8.6	2016-04-06 13:23:23
AZ					

Impala Qualification with NetApp FAS NFS Connector for Hadoop for Join Query

This section details the Impala qualification for the join query between two external tables that are created from the CSV files.

1. HDFS is the default file system. We don't have NFS in core-site.xml Snippet (Safety Value). Update the `core-site.xml` file with the new NFS Connector configuration to use NFS instead of HDFS.

cloudera manager Clusters Hosts Diagnostics Audits Charts Backup Administration Search (Hotkey: /) Support admin

HDFS (Cluster 1) April 7, 2016, 2:28 PM EDT

Status Instances Configuration Commands Audits File Browser Charts Library Cache Statistics NameNode Web UI Quick Links Actions

Configuration Selected Filters: x core_site_safety_valve Switch to the classic layout Role Groups History and Rollback

Filters Clear Reason for change... Save Changes

SEARCH
core_site_safety_valve

STATUS
Error 0
Warning 0
Edited 0
Non-default 1
Has Overrides 0

SCOPE
HDFS (Service-Wide) 2
Balancer 0
DataNode 0
Gateway 0
HttpFS 0
JournalNode 0
NFS Gateway 0
NameNode 0

Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml

```
<property>
  <name>fs.AbstractFileSystem.nfs.impl</name>
  <value>org.apache.hadoop.fs.nfs.NFSv3AbstractFileSystem</value>
</property>
<property>
  <name>fs.nfs.impl</name>
  <value>org.apache.hadoop.fs.nfs.NFSv3FileSystem</value>
</property>
<property>
  <name>fs.nfs.configuration</name>
  <value>/etc/NetAppNFSConnector/conf/nfs-mapping.json</value>
</property>
```

Suppress Parameter Validation: Cluster-wide Advanced Configuration HDFS (Service-Wide)

ip-172-30-1-57.ec2.internal:7180/cm/services/14/commands alve) for

- Restart the cluster.
- Verify the status of the cluster.

cloudera manager Clusters Hosts Diagnostics Audits Charts Backup Administration Search (Hotkey: /) Su

Cluster 1 (CDH 5.5.2, Parcels) 93.2 seconds preceding April 7, 201

Status Configuration

Status

Component	Status
Hosts	6
HDFS	
Hive	
Hue	
Impala	
Oozie	
YARN (MR2 Inclu...	1
ZooKeeper	1

Charts

Cluster CPU

Cluster Disk IO

Cluster Network IO

HDFS IO

- The details of the cluster host are displayed.

cloudera manager Clusters Hosts Diagnostics Audits Charts Backup Administration Search (Hotkey: /) Support admin

Hosts

April 7, 2016, 2:33 PM EDT

Status Configuration Roles Templates Disks Overview Parcels

Status Add New Hosts to Cluster Inspect All Hosts Re-run Upgrade Wizard

Filters

SEARCH

STATUS

Good Health 6

CLUSTERS

CORES

COMMISSION STATE

LAST HEARTBEAT

LOAD (1 MINUTE)

LOAD (5 MINUTES)

LOAD (15 MINUTES)

MAINTENANCE MODE

RACK

SERVICES

HEALTH TESTS

Actions for Selected Columns: 9 Selected

Status	Name	IP	Roles	Last Heartbeat	Load Average	Disk Usage	Physical Memory	Swap Space
Good Health	ip-172-30-1-117.ec2.internal	172.30.1.117	13 Role(s)	11.98s ago	1.13 1.03 1.00	16.1 GiB / 25 GiB	8.5 GiB / 157 GiB	
Good Health	ip-172-30-1-134.ec2.internal	172.30.1.134	6 Role(s)	11.81s ago	0.00 0.01 0.05	7.7 GiB / 1.2 TiB	4.2 GiB / 157 GiB	
Good Health	ip-172-30-1-144.ec2.internal	172.30.1.144	6 Role(s)	12.05s ago	1.58 0.74 0.82	11.6 GiB / 1.2 TiB	4.7 GiB / 157 GiB	
Good Health	ip-172-30-1-154.ec2.internal	172.30.1.154	5 Role(s)	11.89s ago	0.00 0.01 0.05	7 GiB / 1.2 TiB	5.4 GiB / 157 GiB	
Good Health	ip-172-30-1-212.ec2.internal	172.30.1.212	5 Role(s)	11.83s ago	0.00 0.01 0.05	6.9 GiB / 1.2 TiB	4.2 GiB / 157 GiB	
Good Health	ip-172-30-1-57.ec2.internal	172.30.1.57	5 Role(s)	11.86s ago	0.00 0.03 0.05	8.3 GiB / 1.2 TiB	6.5 GiB / 157 GiB	

5. Select the Status tab and verify that the health status of the Impala server is good (green).

cloudera manager Clusters Hosts Diagnostics Audits

Impala (Cluster 1)

Status Instances Configuration Commands Audits Queries Charts

Health Tests

Create Trigger

Show 3 Good

Assignment Locality Suppress...

100.00% of assignments operating on local data over the past 15 minute(s). 0 local assignments. 0 total assignments. At least 10 assignments required for this test to return a value.

Status Summary

Impala Catalog Server	1 Good Health
Impala Daemon	5 Good Health
Impala StateStore	1 Good Health
Hosts	6 Good Health

6. Select the Instance tab to display the details of the Impala instance.

cloudera manager Clusters Hosts Diagnostics Audits Charts Backup Administration Search (Hotkey: /) Support admin

Impala (Cluster 1) April 7, 2016, 2:35 PM EDT

Status Instances Configuration Commands Audits Queries Charts Library Best Practices Web UI Quick Links Actions

Role Instances Add Role Instances Role Groups

Filters

SEARCH

STATUS

Good Health 7

COMMISSION STATE

MAINTENANCE MODE

RACK

ROLE GROUP

ROLE TYPE

STATE

Actions for Selected

Role Type	State	Host	Commission State	Role Group
Impala Catalog Server	Started	ip-172-30-1-117.ec2.internal	Commissioned	Impala Catalog Server Default Group
Impala Daemon	Started	ip-172-30-1-212.ec2.internal	Commissioned	Impala Daemon Default Group
Impala Daemon	Started	ip-172-30-1-57.ec2.internal	Commissioned	Impala Daemon Default Group
Impala Daemon	Started	ip-172-30-1-154.ec2.internal	Commissioned	Impala Daemon Default Group
Impala Daemon	Started	ip-172-30-1-144.ec2.internal	Commissioned	Impala Daemon Default Group
Impala Daemon	Started	ip-172-30-1-134.ec2.internal	Commissioned	Impala Daemon Default Group
Impala StateStore	Started	ip-172-30-1-117.ec2.internal	Commissioned	Impala StateStore Default Group

7. From the CLI, verify that the default file system is HDFS.

```
[impala@ip-172-30-1-117 ~]$ hostname
ip-172-30-1-117.ec2.internal
[impala@ip-172-30-1-117 ~]$ id
uid=979(impala) gid=975(impala) groups=975(impala),980(hive)
[impala@ip-172-30-1-117 ~]$ hadoop dfs -df -h
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Filesystem                                Size      Used Available  Use%
hdfs://ip-172-30-1-117.ec2.internal:8020  112.4 G   1.4 G     85.2 G     1%
[impala@ip-172-30-1-117 ~]$
[impala@ip-172-30-1-117 ~]$
```

8. Check the table's CSV files in the NFS location, which can be accessed by the NFS Connector using the /etc/NetAppNFSConnector/conf/nfs-mapping.json configuration through the secondary file system in Hadoop. The current primary file system until now has been HDFS.

```
[impala@ip-172-30-1-117 ~]$ hadoop dfs -ls nfs://172.30.254.29:2049/user/impala/sample_data/tab*
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

16/04/07 14:38:09 INFO nfs.NFSv3FileSystem: User config file:
/etc/NetAppNFSConnector/conf/users.json
16/04/07 14:38:09 INFO nfs.NFSv3FileSystem: Group config file:
/etc/NetAppNFSConnector/conf/groups.json
Store with ep Endpoint: host=nfs://172.30.254.29:2049/ export=/ path=/ has fsId 2147484980
Found 1 items
-rw-r--r--  1 root root      193 2016-03-31 15:53
nfs://172.30.254.29:2049/user/impala/sample_data/tab1/tab1.csv
Found 1 items
-rw-r--r--  1 root root      158 2016-03-31 15:53
nfs://172.30.254.29:2049/user/impala/sample_data/tab2/tab2.csv
```

9. Check the databases and tables that were created in NFS.

```
[impala@ip-172-30-1-117 ~]$ impala-shell -i ip-172-30-1-57.ec2.internal -c
Starting Impala Shell without Kerberos authentication
Connected to ip-172-30-1-57.ec2.internal:21000
Server version: impalad version 2.3.0-cdh5.5.2 RELEASE (build
cc1125f10419a7269366f7f950f57b24b07acd64)
*****
Welcome to the Impala shell. Copyright (c) 2015 Cloudera, Inc. All rights reserved.
(Impala Shell v2.3.0-cdh5.5.2 (cc1125f) built on Mon Jan 25 16:03:27 PST 2016)

When pretty-printing is disabled, you can use the '--output_delimiter' flag to set
```



```

the delimiter for fields in the same row. The default is ','.
*****
[ip-172-30-1-57.ec2.internal:21000] > show databases;
Query: show databases
+-----+
| name          |
+-----+
| _impala_builtins |
| default       |
| nfs_db        |
+-----+
Fetched 3 row(s) in 0.12s
[ip-172-30-1-57.ec2.internal:21000] > use nfs_db;
Query: use nfs_db
[ip-172-30-1-57.ec2.internal:21000] > show tables;
Query: show tables
+-----+
| name          |
+-----+
| tab1           |
| tab2           |
| test1_nfs      |
| test2_nfs      |
| test_nfs       |
+-----+
Fetched 5 row(s) in 0.01s

```

10. The desc and select queries fail for NFS tables.

```

[ip-172-30-1-57.ec2.internal:21000] > desc formatted tab1;
Query: describe formatted tab1
ERROR: AnalysisException: java.lang.NoClassDefFoundError: Could not initialize class
com.cloudera.impala.util.HdfsCachingUtil
CAUSED BY: ExecutionException: java.lang.NoClassDefFoundError: Could not initialize class
com.cloudera.impala.util.HdfsCachingUtil
CAUSED BY: NoClassDefFoundError: Could not initialize class
com.cloudera.impala.util.HdfsCachingUtil
CAUSED BY: TableLoadingException: java.lang.NoClassDefFoundError: Could not initialize class
com.cloudera.impala.util.HdfsCachingUtil
CAUSED BY: ExecutionException: java.lang.NoClassDefFoundError: Could not initialize class
com.cloudera.impala.util.HdfsCachingUtil
CAUSED BY: NoClassDefFoundError: Could not initialize class
com.cloudera.impala.util.HdfsCachingUtil

[ip-172-30-1-57.ec2.internal:21000] > desc formatted tab2;
Query: describe formatted tab2
ERROR: AnalysisException: java.lang.NoClassDefFoundError: Could not initialize class
com.cloudera.impala.util.HdfsCachingUtil
CAUSED BY: ExecutionException: java.lang.NoClassDefFoundError: Could not initialize class
com.cloudera.impala.util.HdfsCachingUtil
CAUSED BY: NoClassDefFoundError: Could not initialize class
com.cloudera.impala.util.HdfsCachingUtil
CAUSED BY: TableLoadingException: java.lang.NoClassDefFoundError: Could not initialize class
com.cloudera.impala.util.HdfsCachingUtil
CAUSED BY: ExecutionException: java.lang.NoClassDefFoundError: Could not initialize class
com.cloudera.impala.util.HdfsCachingUtil
CAUSED BY: NoClassDefFoundError: Could not initialize class
com.cloudera.impala.util.HdfsCachingUtil

[ip-172-30-1-57.ec2.internal:21000] > select * from tab1;
Query: select * from tab1
ERROR: AnalysisException: Failed to load metadata for table: 'tab1'
CAUSED BY: TableLoadingException: java.lang.NoClassDefFoundError: Could not initialize class
com.cloudera.impala.util.HdfsCachingUtil
CAUSED BY: ExecutionException: java.lang.NoClassDefFoundError: Could not initialize class
com.cloudera.impala.util.HdfsCachingUtil
CAUSED BY: NoClassDefFoundError: Could not initialize class
com.cloudera.impala.util.HdfsCachingUtil

[ip-172-30-1-57.ec2.internal:21000] > select * from tab2;

```

```
Query: select * from tab2
ERROR: AnalysisException: Failed to load metadata for table: 'tab2'
CAUSED BY: TableLoadingException: java.lang.NoClassDefFoundError: Could not initialize class
com.cloudera.impala.util.HdfsCachingUtil
CAUSED BY: ExecutionException: java.lang.NoClassDefFoundError: Could not initialize class
com.cloudera.impala.util.HdfsCachingUtil
CAUSED BY: NoClassDefFoundError: Could not initialize class
com.cloudera.impala.util.HdfsCachingUtil

[ip-172-30-1-57.ec2.internal:21000] >
```

11. Restart only the Impala instances from the Impala page.

cloudera manager

Clusters ▾
Hosts
Diagnostics ▾
Audits
Charts ▾
Backup ▾
Administration ▾

Search (Hotkey: /)

Support ▾ admin ▾

Impala (Cluster 1)

April 7, 2016, 2:45 PM EDT

Status

Instances

Configuration

Commands

Audits

Queries

Charts Library

Best Practices

Web UI ▾

Quick Links ▾

Actions ▾

Role Instances

Filters

SEARCH

STATUS

Good Health

7

COMMISSION STATE

MAINTENANCE MODE

RACK

ROLE GROUP

ROLE TYPE

STATE

HEALTH TESTS

Actions for Selected ▾

	Role Type	State	Host	Commission State
<input type="checkbox"/>	Impala Catalog Server	Started	ip-172-30-1-117.ec2.internal	Commissioned
<input type="checkbox"/>	Impala Daemon	Started	ip-172-30-1-212.ec2.internal	Commissioned
<input type="checkbox"/>	Impala Daemon	Started	ip-172-30-1-57.ec2.internal	Commissioned
<input type="checkbox"/>	Impala Daemon	Started	ip-172-30-1-154.ec2.internal	Commissioned
<input type="checkbox"/>	Impala Daemon	Started	ip-172-30-1-144.ec2.internal	Commissioned
<input type="checkbox"/>	Impala Daemon	Started	ip-172-30-1-134.ec2.internal	Commissioned
<input type="checkbox"/>	Impala StateStore	Started	ip-172-30-1-117.ec2.internal	Commissioned

Start

Stop

Restart

Add Role Instances

Rename

Enter Maintenance Mode

Create Impala User Directory

Enable Integrated Resource Management

Impala Daemon Default Group

Impala StateStore Default Group

Restart Command

Successfully stopped service.

Start	Impala	Apr 7, 2:46:00 PM	22.37s
Successfully started service.			
Starting 7 roles on service		Apr 7, 2:46:00 PM	22.36s
Successfully started 7 roles on service.			
Start this Impala Daemon	Impala Daemon (ip-172-30-1-144)	Apr 7, 2:46:00 PM	22.27s
Successfully started process.			
Start this Impala StateStore	Impala StateStore (ip-172-30-1-117)	Apr 7, 2:46:00 PM	22.26s
Successfully started process.			
Start this Impala Catalog Server	Impala Catalog Server (ip-172-30-1-117)	Apr 7, 2:46:00 PM	22.26s
Successfully started process.			
Start this Impala Daemon	Impala Daemon (ip-172-30-1-212)	Apr 7, 2:46:00 PM	22.3s
Successfully started process.			
Start this Impala Daemon	Impala Daemon (ip-172-30-1-154)	Apr 7, 2:46:00 PM	22.22s
Successfully started process.			
Start this Impala Daemon	Impala Daemon (ip-172-30-1-57)	Apr 7, 2:46:00 PM	22.24s
Successfully started process.			
Start this Impala Daemon	Impala Daemon (ip-172-30-1-134)	Apr 7, 2:46:00 PM	22.2s
Successfully started process.			

Close

12. The select and desc query should now work for tables tab1 and tab2; their locations are
nfs:// 172.30.254.29:2049/user/impala/sample_data/tab1 and
nfs://172.30.254.29:2049/user/impala/sample_data/tab2.

```
[impala@ip-172-30-1-117 ~]$ impala-shell -i ip-172-30-1-57.ec2.internal -c
Starting Impala Shell without Kerberos authentication
Connected to ip-172-30-1-57.ec2.internal:21000
Server version: impalad version 2.3.0-cdh5.5.2 RELEASE (build
cc1125f10419a7269366f7f950f57b24b07acd64)
*****
Welcome to the Impala shell. Copyright (c) 2015 Cloudera, Inc. All rights reserved.
(Impala Shell v2.3.0-cdh5.5.2 (cc1125f) built on Mon Jan 25 16:03:27 PST 2016)
```

To see a summary of a query's progress that updates in real-time, run 'set
LIVE_PROGRESS=1;'.

```
[ip-172-30-1-57.ec2.internal:21000] > use nfs_db;
Query: use nfs_db
[ip-172-30-1-57.ec2.internal:21000] > select * from tab2;
Query: select * from tab2
```

id	col_1	col_2
1	true	12789.123
2	false	1243.5
3	false	24453.325
4	false	2423.3254
5	true	243.325
60	false	243565423.325
70	true	243.325
80	false	243423.325
90	true	243.325

Fetches 9 row(s) in 0.99s

```
[ip-172-30-1-57.ec2.internal:21000] > select * from tab1;
Query: select * from tab1
```

id	col_1	col_2	col_3
1	true	123.123	2012-10-24 08:55:00
2	false	1243.5	2012-10-25 13:40:00
3	false	24453.325	2008-08-22 09:33:21.123000000
4	false	243423.325	2007-05-12 22:32:21.334540000
5	true	243.325	1953-04-22 09:11:33

Fetches 5 row(s) in 0.29s

```
[ip-172-30-1-57.ec2.internal:21000] > desc formatted tab1;
Query: describe formatted tab1
```

name	comment	type

# col_name		data_type
comment		
		NULL
id		int
col_1		boolean
col_2		double
col_3		timestamp
		NULL
# Detailed Table Information		
		NULL

Database:	nfs_db	
NULL		
Owner:	impala	
NULL		
CreateTime:	Mon Apr 04 19:14:54 EDT 2016	
NULL		
LastAccessTime:	UNKNOWN	
NULL		
Protect Mode:	None	
NULL		
Retention:	0	
NULL		
Location:	nfs://172.30.254.29:2049/user/impala/sample_data/tab1	
NULL		
Table Type:	MANAGED_TABLE	
NULL		
Table Parameters:	NULL	
NULL		
	COLUMN_STATS_ACCURATE	
false		
	numFiles	0
	numRows	-1
	rawDataSize	-1
	totalSize	0
	transient_lastDdlTime	
1459811694		
	NULL	
NULL		
# Storage Information	NULL	
NULL		
SerDe Library:	org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe	
NULL		
InputFormat:	org.apache.hadoop.mapred.TextInputFormat	
NULL		
OutputFormat:	org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat	
NULL		
Compressed:	No	
NULL		
Num Buckets:	0	
NULL		
Bucket Columns:	[]	
NULL		
Sort Columns:	[]	
NULL		
Storage Desc Params:	NULL	
NULL		
	field.delim	,
	serialization.format	,

-----+

-----+

Fetchd 35 row(s) in 0.05s
[ip-172-30-1-57.ec2.internal:21000] > desc formatted tab2;
Query: describe formatted tab2

-----+

name	type	
comment		

-----+

# col_name	data_type	
comment		
	NULL	
NULL		
id	int	
NULL		


```
[ip-172-30-1-57.ec2.internal:21000] >
```

13. For testing purposes, update the `/usr/lib64/cmf/service/impala/impala.sh` file in one of the Impalad servers (172.30.1.57) as well as the metadata and catalogd server (172.30.1.117).

```
export HADOOP_CONF_CLASSPATH=$(/bin/hadoop classpath)

JDBC_JARS="$CLOUDERA_MYSQL_CONNECTOR_JAR:$CLOUDERA_POSTGRES_SQL_JDBC_JAR:$CLOUDERA_ORACLE_CONNECTOR_JAR:$HADOOP_CONF_CLASSPATH"

if [ "impalad" = "$1" ]; then
    exec "$IMPALA_HOME/../../bin/impalad" --flagfile=${FLAG_FILE} --abort_on_config_error=false ${IMPALA_SERVER_ARGS}
elif [ "statestore" = "$1" ]; then
    exec "$IMPALA_HOME/../../bin/statestored" --flagfile=${FLAG_FILE} --abort_on_config_error=false ${IMPALA_SERVER_ARGS}
elif [ "catalogd" = "$1" ]; then
    exec "$IMPALA_HOME/../../bin/catalogd" --flagfile=${FLAG_FILE} --abort_on_config_error=false ${IMPALA_SERVER_ARGS}
fi
```

14. Update `fs.defaultfs` in `core-site.xml` by using Snippet (Safety Value) to change the default file system from HDFS to NFS.

```
<property>
  <name>fs.defaultFS</name>
  <value>nfs://172.30.254.29:2049</value>
</property>
<property>
  <name>fs.AbstractFileSystem.nfs.impl</name>
  <value>org.apache.hadoop.fs.nfs.NFSv3AbstractFilesystem</value>
</property>
<property>
  <name>fs.nfs.impl</name>
  <value>org.apache.hadoop.fs.nfs.NFSv3FileSystem</value>
</property>
<property>
  <name>fs.nfs.configuration</name>
  <value>/etc/NetAppNFSConnector/conf/nfs-mapping.json</value>
</property>
```

HDFS (Cluster 1) April 7, 2016, 2:5

Status Instances **Configuration** Commands Audits File Browser Charts Library Cache Statistics NameNode Web UI Quick Links

Configuration Selected Filters: x core_site_safety_valve Switch to the classic layout Role Groups History and

Filters Clear

SEARCH

core_site_safety_valve

STATUS

Error 0

Warning 0

Edited 1

Non-default 1

Has Overrides 0

SCOPE

HDFS (Service-Wide) 2

Balancer 0

DataNode 0

Gateway 0

HttpFS 0

JournalNode 0

NFS Gateway 0

NameNode 0

Reason for change...

Save Changes 1 Edited Value

Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml

HDFS (Service-Wide)

```
<property>
  <name>fs.defaultFS</name>
  <value>nfs://172.30.254.29:2049</value>
</property>
<property>
  <name>fs.AbstractFileSystem.nfs.impl</name>
  <value>org.apache.hadoop.fs.nfs.NFSv3AbstractFileSystem</value>
</property>
<property>
  <name>fs.nfs.impl</name>
  <value>org.apache.hadoop.fs.nfs.NFSv3FileSystem</value>
</property>
<property>
  <name>fs.nfs.configuration</name>
  <value>/etc/NetAppNFSConnector/conf/nfs-mapping.json</value>
</property>
```

Suppress Parameter HDFS (Service-Wide)

15. After the update, do not restart the HDFS instances.

cloudera manager Clusters Hosts Diagnostics Audits Charts Backup Administration Search (Hotkey: /) Support admin

HDFS (Cluster 1) April 7, 2016, 3:05 PM EDT

Status Instances **Configuration** Commands Audits File Browser Charts Library Cache Statistics NameNode Web UI Quick Links Actions

Configuration Selected Filters: x core_site_safety_valve Switch to the classic layout Role Groups History and Rollback

Filters Clear

SEARCH

core_site_safety_valve

STATUS

Error 0

Warning 0

Edited 0

Non-default 1

Has Overrides 0

SCOPE

HDFS (Service-Wide) 2

Balancer 0

DataNode 0

Gateway 0

HttpFS 0

JournalNode 0

NFS Gateway 0

NameNode 0

Reason for change...

Save Changes

Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml

HDFS (Service-Wide)

```
<property>
  <name>fs.defaultFS</name>
  <value>nfs://172.30.254.29:2049</value>
</property>
<property>
  <name>fs.AbstractFileSystem.nfs.impl</name>
  <value>org.apache.hadoop.fs.nfs.NFSv3AbstractFileSystem</value>
</property>
<property>
  <name>fs.nfs.impl</name>
  <value>org.apache.hadoop.fs.nfs.NFSv3FileSystem</value>
</property>
<property>
  <name>fs.nfs.configuration</name>
  <value>/etc/NetAppNFSConnector/conf/nfs-mapping.json</value>
</property>
```

Suppress Parameter HDFS (Service-Wide)

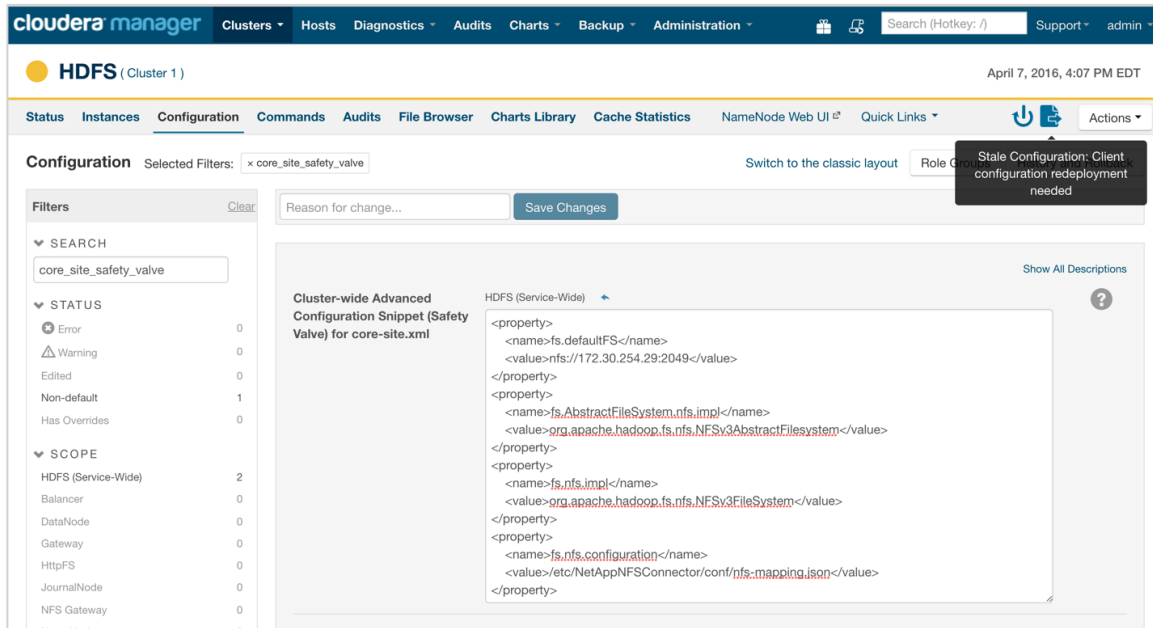
16. The primary file system is still HDFS.

```
[impala@ip-172-30-1-117 ~]$ hadoop dfs -df -h
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

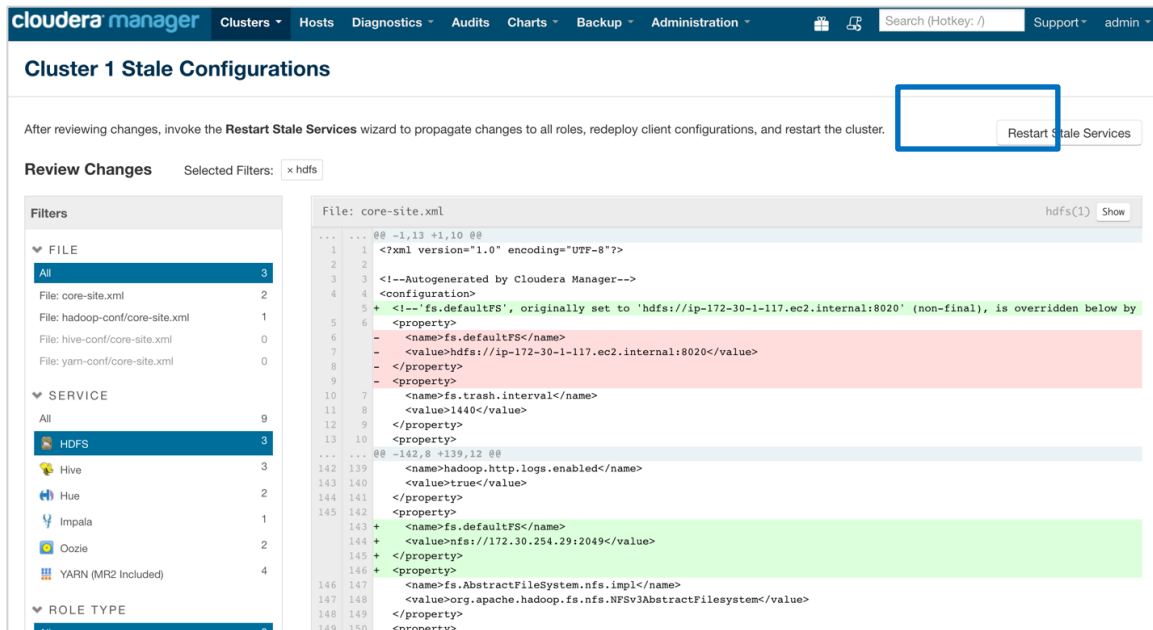
Filesystem                                Size      Used    Available  Use%
hdfs://ip-172-30-1-117.ec2.internal:8020  112.4 G   1.4 G    85.2 G     1%
```

[impala@ip-172-30-1-117 ~]\$

17. Restart stale config from the HDFS panel and redeploy the configuration.



The screenshot shows the Cloudera Manager interface for the HDFS configuration of Cluster 1. The top navigation bar includes Clusters, Hosts, Diagnostics, Audits, Charts, Backup, Administration, and a search bar. The main header indicates 'HDFS (Cluster 1)' and the date 'April 7, 2016, 4:07 PM EDT'. The left sidebar contains tabs for Status, Instances, Configuration (selected), Commands, Audits, File Browser, Charts Library, Cache Statistics, NameNode Web UI, Quick Links, and Actions. The Configuration tab is active, showing a 'Selected Filters: x core_site_safety_valve' and a 'Reason for change...' field. The main content area displays the 'Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml'. It shows a list of properties for HDFS (Service-Wide) configuration, including fs.defaultFS, fs.AbstractFileSystem.nfs.impl, fs.nfs.impl, fs.nfs.configuration, and org.apache.hadoop.fs.nfs.NFSv3AbstractFileSystem. A notification banner at the top right states 'Stale Configuration: Client configuration redeployment needed'.



The screenshot shows the Cloudera Manager interface for the 'Cluster 1 Stale Configurations' page. The top navigation bar is the same as the previous screenshot. The main header indicates 'Cluster 1 Stale Configurations'. Below the header, there is a message: 'After reviewing changes, invoke the Restart Stale Services wizard to propagate changes to all roles, redeploy client configurations, and restart the cluster.' A 'Restart Stale Services' button is highlighted with a blue box. The left sidebar contains tabs for Review Changes (selected), Selected Filters: x hdfs, and a list of filters. The Review Changes tab is active, showing a list of files and services. The main content area displays the 'File: core-site.xml' configuration snippet, showing the fs.defaultFS property being overridden by the cluster configuration. The snippet includes the following XML code:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Autogenerated by Cloudera Manager-->
<configuration>
  <!-- 'fs.defaultFS', originally set to 'hdfs://ip-172-30-1-117.ec2.internal:8020' (non-final), is overridden below by -->
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://ip-172-30-1-117.ec2.internal:8020</value>
  </property>
  <property>
    <name>fs.trash.interval</name>
    <value>1440</value>
  </property>
  <property>
    <name>hadoop.http.logs.enabled</name>
    <value>true</value>
  </property>
  <property>
    <name>fs.defaultFS</name>
    <value>nfs://172.30.254.29:2049</value>
  </property>
  <property>
    <name>fs.AbstractFileSystem.nfs.impl</name>
    <value>org.apache.hadoop.fs.nfs.NFSv3AbstractFileSystem</value>
  </property>
</configuration>
```


Details Completed 3 of 3 step(s).					<input checked="" type="radio"/> All <input type="radio"/> Failed Only <input type="radio"/> Running Only	
Step	Context	Start Time	Duration	Actions		
➤ Stop All services successfully stopped.	Cluster 1	Apr 7, 4:08:41 PM	25.88s			
▼ Start All services successfully started.	Cluster 1	Apr 7, 4:09:07 PM	2.2m			
➤ Start Successfully started HDFS service	HDFS	Apr 7, 4:09:07 PM	22.43s			
➤ Start Successfully started service.	YARN (MR2 Included)	Apr 7, 4:09:29 PM	44.56s			
➤ Start Successfully started service.	Hive	Apr 7, 4:10:14 PM	22.21s			
▼ Execute command Start concurrently on 2 services Successfully completed 2 steps.		Apr 7, 4:10:36 PM	22.47s			
➤ Start Successfully started service.	Oozie	Apr 7, 4:10:36 PM	22.4s			
➤ Start Successfully started service.	Impala	Apr 7, 4:10:36 PM	22.4s			
➤ Start Successfully started service.	Hue	Apr 7, 4:10:59 PM	22.2s			
➤ Deploy Client Configuration Successfully deployed all client configurations.	Cluster 1	Apr 7, 4:11:21 PM	15.39s			
<div> ⏪ Back 1 2 3 4 ⏩ Finish </div>						


18. After a successful restart, verify that the configuration was changed for only one Impala server.



Impala (Cluster 1)

[Status](#)
[Instances](#)
[Configuration](#)
[Commands](#)
[Audits](#)
[Queries](#)
[Chart](#)

Health Tests







Create Trigger

- 
[Impala Daemon Health](#)
[Suppress...](#)

Healthy Impala Daemon: 1. Concerning Impala Daemon: 0. Total Impala Daemon: 5. Percent healthy: 20.00%. Percent healthy or concerning: 20.00%. Critical threshold: 90.00%.
- 
[Show 2 Good](#)
- 
[Assignment Locality](#)
[Suppress...](#)

100.00% of assignments operating on local data over the past 15 minute(s). 0 local assignments. 0 total assignments. At least 10 assignments required for this test to return a value.

Status Summary

Impala Catalog Server	 1 Good Health
Impala Daemon	 4 Down  1 Good Health
Process Status	 4
Impala StateStore	 1 Good Health
Hosts	 6 Good Health

19. Verify that the default file system changed from HDFS to NFS.

```
[impala@ip-172-30-1-117 ~]$ hadoop dfs -df -h
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

16/04/07 16:15:31 INFO nfs.NFSv3FileSystem: User config file:
/etc/NetAppNFSConnector/conf/users.json
16/04/07 16:15:31 INFO nfs.NFSv3FileSystem: Group config file:
/etc/NetAppNFSConnector/conf/groups.json
Store with ep Endpoint: host=nfs://172.30.254.29:2049/ export=/ path=/ has fsId 2147484980
Filesystem              Size  Used  Available  Use%
nfs://172.30.254.29:2049 8.0 E    0      8.0 E    0%
[impala@ip-172-30-1-117 ~]$
```

20. Verify the join query.

```
[impala@ip-172-30-1-117 ~]$ impala-shell -i ip-172-30-1-57.ec2.internal -c
Starting Impala Shell without Kerberos authentication
Connected to ip-172-30-1-57.ec2.internal:21000
Server version: impalad version 2.3.0-cdh5.5.2 RELEASE (build
cc1125f10419a7269366f7f950f57b24b07acd64)
*****
Welcome to the Impala shell. Copyright (c) 2015 Cloudera, Inc. All rights reserved.
(Impala Shell v2.3.0-cdh5.5.2 (cc1125f) built on Mon Jan 25 16:03:27 PST 2016)

The HISTORY command lists all shell commands in chronological order.
*****
```

```
[ip-172-30-1-57.ec2.internal:21000] > show databases;
```

```
Query: show databases
```

```
+-----+
| name           |
+-----+
| _impala_builtins |
| default        |
| nfs_db         |
+-----+
```

```
Fetches 3 row(s) in 0.09s
```

```
[ip-172-30-1-57.ec2.internal:21000] > use nfs_db;
```

```
Query: use nfs_db
```

```
[ip-172-30-1-57.ec2.internal:21000] > show tables;
```

```
Query: show tables
```

```
+-----+
| name           |
+-----+
| tab1            |
| tab2            |
| test1_nfs       |
| test2_nfs       |
| test_nfs        |
+-----+
```

```
Fetches 5 row(s) in 0.01s
```

```
[ip-172-30-1-57.ec2.internal:21000] > desc formatted tab1;
```

```
Query: describe formatted tab1
```

```
+-----+-----+-----+
| name           | type           |
+-----+-----+-----+
| comment        |                |
+-----+-----+-----+
| # col_name     | data_type      |
+-----+-----+-----+
| comment        | NULL           |
| id             | int            |
| col_1          | boolean        |
| col_2          | double         |
| col_3          | timestamp      |
|               | NULL           |
| # Detailed Table Information | NULL           |
| Database:      | nfs_db         |
| Owner:         | impala         |
| CreateTime:    | Mon Apr 04 19:14:54 EDT 2016 |
| LastAccessTime: | UNKNOWN        |
| Protect Mode:  | None           |
| Retention:     | 0              |
| Location:      | nfs://172.30.254.29:2049/user/impala/sample_data/tab1 |
| Table Type:    | MANAGED_TABLE  |
| Table Parameters: | NULL          |
|               | COLUMN_STATS_ACCURATE |
| false         |                |
|               | numFiles       | 0
```

	numRows	-1
	rawDataSize	-1
	totalSize	0
	transient_lastDdlTime	
1459811694		
	NULL	
NULL		
# Storage Information	NULL	
NULL		
SerDe Library:	org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe	
NULL		
InputFormat:	org.apache.hadoop.mapred.TextInputFormat	
NULL		
OutputFormat:	org.apache.hadoop.hive ql.io.HiveIgnoreKeyTextOutputFormat	
NULL		
Compressed:	No	
NULL		
Num Buckets:	0	
NULL		
Bucket Columns:	[]	
NULL		
Sort Columns:	[]	
NULL		
Storage Desc Params:	NULL	
NULL		
	field.delim	,
	serialization.format	,
-----+		
-----+		
Fetched 35 row(s) in 0.06s		
[ip-172-30-1-57.ec2.internal:21000] > desc formatted tab2;		
Query: describe formatted tab2		
-----+		
-----+		
name	type	
comment		
-----+		
-----+		
# col_name	data_type	
comment		
	NULL	
NULL		
id	int	
NULL		
col_1	boolean	
NULL		
col_2	double	
NULL		
	NULL	
NULL		
# Detailed Table Information	NULL	
NULL		
Database:	nfs_db	
NULL		
Owner:	impala	
NULL		
CreateTime:	Mon Apr 04 19:21:10 EDT 2016	
NULL		
LastAccessTime:	UNKNOWN	
NULL		
Protect Mode:	None	
NULL		
Retention:	0	
NULL		
Location:	nfs://172.30.254.29:2049/user/impala/sample_data/tab2	
NULL		

Table Type:	MANAGED_TABLE			
NULL				
Table Parameters:	NULL			
NULL				
	COLUMN_STATS_ACCURATE			
false				
	numFiles	0		
	numRows	-1		
	rawDataSize	-1		
	totalSize	0		
	transient_lastDdlTime			
1459812070				
	NULL			
NULL				
# Storage Information	NULL			
NULL				
SerDe Library:	org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe			
NULL				
InputFormat:	org.apache.hadoop.mapred.TextInputFormat			
NULL				
OutputFormat:	org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat			
NULL				
Compressed:	No			
NULL				
Num Buckets:	0			
NULL				
Bucket Columns:	[]			
NULL				
Sort Columns:	[]			
NULL				
Storage Desc Params:	NULL			
NULL				
	field.delim	,		
	serialization.format	,		
+-----+				
-----+				
Fetched 34 row(s) in 0.01s				
[ip-172-30-1-57.ec2.internal:21000] > select * from tab1 join tab2;				
Query: select * from tab1 join tab2				
+-----+				
id col_1 col_2 col_3 id col_1 col_2				
+-----+				
1 true 123.123 2012-10-24 08:55:00 1 true 12789.123				
1 true 123.123 2012-10-24 08:55:00 2 false 1243.5				
1 true 123.123 2012-10-24 08:55:00 3 false 24453.325				
1 true 123.123 2012-10-24 08:55:00 4 false 2423.3254				
1 true 123.123 2012-10-24 08:55:00 5 true 243.325				
1 true 123.123 2012-10-24 08:55:00 60 false 243565423.325				
1 true 123.123 2012-10-24 08:55:00 70 true 243.325				
1 true 123.123 2012-10-24 08:55:00 80 false 243423.325				
1 true 123.123 2012-10-24 08:55:00 90 true 243.325				
2 false 1243.5 2012-10-25 13:40:00 1 true 12789.123				
2 false 1243.5 2012-10-25 13:40:00 2 false 1243.5				
2 false 1243.5 2012-10-25 13:40:00 3 false 24453.325				
2 false 1243.5 2012-10-25 13:40:00 4 false 2423.3254				
2 false 1243.5 2012-10-25 13:40:00 5 true 243.325				
2 false 1243.5 2012-10-25 13:40:00 60 false 243565423.325				
2 false 1243.5 2012-10-25 13:40:00 70 true 243.325				
2 false 1243.5 2012-10-25 13:40:00 80 false 243423.325				
2 false 1243.5 2012-10-25 13:40:00 90 true 243.325				
3 false 24453.325 2008-08-22 09:33:21.123000000 1 true 12789.123				
3 false 24453.325 2008-08-22 09:33:21.123000000 2 false 1243.5				
3 false 24453.325 2008-08-22 09:33:21.123000000 3 false 24453.325				
3 false 24453.325 2008-08-22 09:33:21.123000000 4 false 2423.3254				
3 false 24453.325 2008-08-22 09:33:21.123000000 5 true 243.325				

3	false	24453.325	2008-08-22 09:33:21.123000000	60	false	243565423.325
3	false	24453.325	2008-08-22 09:33:21.123000000	70	true	243.325
3	false	24453.325	2008-08-22 09:33:21.123000000	80	false	243423.325
3	false	24453.325	2008-08-22 09:33:21.123000000	90	true	243.325
4	false	243423.325	2007-05-12 22:32:21.334540000	1	true	12789.123
4	false	243423.325	2007-05-12 22:32:21.334540000	2	false	1243.5
4	false	243423.325	2007-05-12 22:32:21.334540000	3	false	24453.325
4	false	243423.325	2007-05-12 22:32:21.334540000	4	false	2423.3254
4	false	243423.325	2007-05-12 22:32:21.334540000	5	true	243.325
4	false	243423.325	2007-05-12 22:32:21.334540000	60	false	243565423.325
4	false	243423.325	2007-05-12 22:32:21.334540000	70	true	243.325
4	false	243423.325	2007-05-12 22:32:21.334540000	80	false	243423.325
4	false	243423.325	2007-05-12 22:32:21.334540000	90	true	243.325
5	true	243.325	1953-04-22 09:11:33	1	true	12789.123
5	true	243.325	1953-04-22 09:11:33	2	false	1243.5
5	true	243.325	1953-04-22 09:11:33	3	false	24453.325
5	true	243.325	1953-04-22 09:11:33	4	false	2423.3254
5	true	243.325	1953-04-22 09:11:33	5	true	243.325
5	true	243.325	1953-04-22 09:11:33	60	false	243565423.325
5	true	243.325	1953-04-22 09:11:33	70	true	243.325
5	true	243.325	1953-04-22 09:11:33	80	false	243423.325
5	true	243.325	1953-04-22 09:11:33	90	true	243.325

Fetches 45 row(s) in 1.17s

[ip-172-30-1-57.ec2.internal:21000] > select * from tab1;

Query: select * from tab1

id	col_1	col_2	col_3
1	true	123.123	2012-10-24 08:55:00
2	false	1243.5	2012-10-25 13:40:00
3	false	24453.325	2008-08-22 09:33:21.123000000
4	false	243423.325	2007-05-12 22:32:21.334540000
5	true	243.325	1953-04-22 09:11:33

Fetches 5 row(s) in 0.28s

[ip-172-30-1-57.ec2.internal:21000] > select * from tab2;

Query: select * from tab2

id	col_1	col_2
1	true	12789.123
2	false	1243.5
3	false	24453.325
4	false	2423.3254
5	true	243.325
60	false	243565423.325
70	true	243.325
80	false	243423.325
90	true	243.325

Fetches 9 row(s) in 0.49s

[ip-172-30-1-57.ec2.internal:21000] > select count(*) from tab1 join tab2;

Query: select count(*) from tab1 join tab2

count(*)
45

Fetches 1 row(s) in 0.72s

[ip-172-30-1-57.ec2.internal:21000] >

Hive

1. Find the available databases.

```
[root@ip-172-30-1-117 ~]# hive
Picked up _JAVA_OPTIONS: -Xmx140000M
Picked up _JAVA_OPTIONS: -Xmx140000M
Picked up _JAVA_OPTIONS: -Xmx140000M
```

```
Picked up _JAVA_OPTIONS: -Xmx140000M
```

```
Logging initialized using configuration in jar:file:/opt/cloudera/parcels/CDH-5.5.2-1.cdh5.5.2.p0.4/jars/hive-common-1.1.0-cdh5.5.2.jar!/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> show databases;
OK
default
nfs_db
Time taken: 1.484 seconds, Fetched: 2 row(s)
```

2. Find the tables in the NFS database.

```
hive> use nfs_db;
OK
Time taken: 0.036 seconds
hive> show tables;
OK
allstatetest
tab1
tab2
tab3_nfs
tab3_nfs_new
tab4_nfs
tab4_nfs_new
test1_nfs
test2_nfs
test_nfs
Time taken: 0.029 seconds, Fetched: 10 row(s)
hive>
```

3. Find the number of records in the table; for example, tab3_nfs_new.

```
hive> select count(*) from tab3_nfs_new;
Query ID = root_20160509120909_3f8673b8-3fdd-4615-b265-82808fb2bed3
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_1462586435822_0026, Tracking URL = http://ip-172-30-1-117.ec2.internal:8088/proxy/application_1462586435822_0026/
Kill Command = /opt/cloudera/parcels/CDH-5.5.2-1.cdh5.5.2.p0.4/lib/hadoop/bin/hadoop job -kill job_1462586435822_0026
Hadoop job information for Stage-1: number of mappers: 3; number of reducers: 1
2016-05-09 12:09:42,712 Stage-1 map = 0%, reduce = 0%
2016-05-09 12:09:54,002 Stage-1 map = 67%, reduce = 0%, Cumulative CPU 22.26 sec
2016-05-09 12:09:55,033 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 22.71 sec
2016-05-09 12:10:01,222 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 24.14 sec
MapReduce Total cumulative CPU time: 24 seconds 140 msec
Ended Job = job_1462586435822_0026
MapReduce Jobs Launched:
Stage-Stage-1: Map: 3 Reduce: 1 Cumulative CPU: 24.14 sec HDFS Read: 0 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 24 seconds 140 msec
OK
14216336
Time taken: 26.245 seconds, Fetched: 1 row(s)
```

4. Find the number of records in the table; for example, tab4_nfs_new.

```
hive> select count(*) from tab4_nfs_new;
Query ID = root_20160509121010_56761d96-35a9-4f4b-a4c7-89bf47db4223
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_1462586435822_0027, Tracking URL = http://ip-172-30-1-117.ec2.internal:8088/proxy/application_1462586435822_0027/
Kill Command = /opt/cloudera/parcels/CDH-5.5.2-1.cdh5.5.2.p0.4/lib/hadoop/bin/hadoop job -kill job_1462586435822_0027
Hadoop job information for Stage-1: number of mappers: 3; number of reducers: 1
2016-05-09 12:10:37,180 Stage-1 map = 0%, reduce = 0%
2016-05-09 12:10:46,654 Stage-1 map = 33%, reduce = 0%, Cumulative CPU 7.3 sec
2016-05-09 12:10:47,687 Stage-1 map = 78%, reduce = 0%, Cumulative CPU 22.03 sec
2016-05-09 12:10:49,746 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 23.63 sec
2016-05-09 12:10:56,938 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 25.09 sec
MapReduce Total cumulative CPU time: 25 seconds 90 msec
Ended Job = job_1462586435822_0027
MapReduce Jobs Launched:
Stage-Stage-1: Map: 3 Reduce: 1 Cumulative CPU: 25.09 sec HDFS Read: 0 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 25 seconds 90 msec
OK
14987634
Time taken: 27.456 seconds, Fetched: 1 row(s)

```

5. Run the join query to find the total records in the two tables. This example uses tables tab3_nfs_new and tab4_nfs_new.

```

hive> select count(*) from tab3_nfs_new join tab4_nfs_new where tab3_nfs_new.id =
tab4_nfs_new.id;
Query ID = root_20160509121212_eb6b05ed-51da-4140-af87-ffc33afeaa51
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 28
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_1462586435822_0028, Tracking URL = http://ip-172-30-1-117.ec2.internal:8088/proxy/application_1462586435822_0028/
Kill Command = /opt/cloudera/parcels/CDH-5.5.2-1.cdh5.5.2.p0.4/lib/hadoop/bin/hadoop job -kill job_1462586435822_0028
Hadoop job information for Stage-1: number of mappers: 6; number of reducers: 28
2016-05-09 12:13:04,082 Stage-1 map = 0%, reduce = 0%
2016-05-09 12:13:17,422 Stage-1 map = 11%, reduce = 0%, Cumulative CPU 56.4 sec
2016-05-09 12:13:18,453 Stage-1 map = 33%, reduce = 0%, Cumulative CPU 78.76 sec
2016-05-09 12:13:20,510 Stage-1 map = 44%, reduce = 0%, Cumulative CPU 82.06 sec
2016-05-09 12:13:21,537 Stage-1 map = 61%, reduce = 0%, Cumulative CPU 102.72 sec
2016-05-09 12:13:23,587 Stage-1 map = 78%, reduce = 0%, Cumulative CPU 111.01 sec
2016-05-09 12:13:27,691 Stage-1 map = 89%, reduce = 0%, Cumulative CPU 123.09 sec
2016-05-09 12:13:28,718 Stage-1 map = 94%, reduce = 0%, Cumulative CPU 125.01 sec
2016-05-09 12:13:30,773 Stage-1 map = 96%, reduce = 0%, Cumulative CPU 128.32 sec
2016-05-09 12:13:31,801 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 129.94 sec
2016-05-09 12:13:40,030 Stage-1 map = 100%, reduce = 86%, Cumulative CPU 356.9 sec
2016-05-09 12:13:42,090 Stage-1 map = 100%, reduce = 90%, Cumulative CPU 380.01 sec
2016-05-09 12:13:43,117 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 466.15 sec
MapReduce Total cumulative CPU time: 7 minutes 46 seconds 150 msec
Ended Job = job_1462586435822_0028
Launching Job 2 out of 2
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_1462586435822_0029, Tracking URL = http://ip-172-30-1-117.ec2.internal:8088/proxy/application_1462586435822_0029/
Kill Command = /opt/cloudera/parcels/CDH-5.5.2-1.cdh5.5.2.p0.4/lib/hadoop/bin/hadoop job -kill job_1462586435822_0029
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1

```



```

2016-05-09 12:13:50,977 Stage-2 map = 0%, reduce = 0%
2016-05-09 12:13:57,134 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 2.12 sec
2016-05-09 12:14:03,291 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 3.58 sec
MapReduce Total cumulative CPU time: 3 seconds 580 msec
Ended Job = job_1462586435822_0029
MapReduce Jobs Launched:
Stage-Stage-1: Map: 6 Reduce: 28 Cumulative CPU: 466.15 sec HDFS Read: 0 HDFS Write: 0
SUCCESS
Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 3.58 sec HDFS Read: 0 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 7 minutes 49 seconds 730 msec
OK
799657050
Time taken: 66.984 seconds, Fetched: 1 row(s)
hive>

```

6. Run the join query between two tables. This example uses tables `tab3_nfs_new` and `tab4_nfs_new`.

```

hive> select * from tab3_nfs_new join tab4_nfs_new where tab3_nfs_new.id = tab4_nfs_new.id limit
100;
Query ID = root_20160509121414_47c9f613-b184-4c2f-82a7-d3936dd2e625
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 28
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_1462586435822_0030, Tracking URL = http://ip-172-30-1-
117.ec2.internal:8088/proxy/application_1462586435822_0030/
Kill Command = /opt/cloudera/parcels/CDH-5.5.2-1.cdh5.5.2.p0.4/lib/hadoop/bin/hadoop job -kill
job_1462586435822_0030
Hadoop job information for Stage-1: number of mappers: 6; number of reducers: 28
2016-05-09 12:14:56,684 Stage-1 map = 0%, reduce = 0%
2016-05-09 12:15:19,266 Stage-1 map = 6%, reduce = 0%, Cumulative CPU 124.21 sec
2016-05-09 12:15:20,292 Stage-1 map = 11%, reduce = 0%, Cumulative CPU 139.85 sec
2016-05-09 12:15:23,370 Stage-1 map = 33%, reduce = 0%, Cumulative CPU 172.77 sec
2016-05-09 12:15:25,438 Stage-1 map = 38%, reduce = 0%, Cumulative CPU 182.75 sec
2016-05-09 12:15:26,466 Stage-1 map = 78%, reduce = 0%, Cumulative CPU 194.95 sec
2016-05-09 12:15:31,596 Stage-1 map = 83%, reduce = 0%, Cumulative CPU 212.84 sec
2016-05-09 12:15:34,677 Stage-1 map = 87%, reduce = 0%, Cumulative CPU 219.39 sec
2016-05-09 12:15:35,703 Stage-1 map = 89%, reduce = 0%, Cumulative CPU 220.48 sec
2016-05-09 12:15:37,754 Stage-1 map = 95%, reduce = 0%, Cumulative CPU 223.23 sec
2016-05-09 12:15:40,860 Stage-1 map = 99%, reduce = 0%, Cumulative CPU 226.44 sec
2016-05-09 12:15:41,887 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 227.44 sec
2016-05-09 12:15:43,951 Stage-1 map = 100%, reduce = 39%, Cumulative CPU 265.16 sec
2016-05-09 12:15:44,979 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 332.55 sec
MapReduce Total cumulative CPU time: 5 minutes 32 seconds 550 msec
Ended Job = job_1462586435822_0030
MapReduce Jobs Launched:
Stage-Stage-1: Map: 6 Reduce: 28 Cumulative CPU: 332.55 sec HDFS Read: 0 HDFS Write: 0
SUCCESS
Total MapReduce CPU Time Spent: 5 minutes 32 seconds 550 msec
OK
28      44768680      58853913      true      109.8      2016-04-06
13:44:01      SD      28      44768680      58853913      true      109.8      201
6-04-06 13:44:01      SD      28      64369998      97268880      true      117.8      201
28      44768680      58853913      true      109.8      2016-04-06
13:44:01      SD      28      64369998      97268880      true      117.8      201
6-04-06 13:45:12      VT
28      44768680      58853913      true      109.8      2016-04-06
13:44:01      SD      28      91179928      14954221      true      115.8      201
6-04-06 13:44:54      UT
28      44768680      58853913      true      109.8      2016-04-06
13:44:01      SD      28      16968019      94351966      true      119.8      201
6-04-06 13:45:29      VA
28      44768680      58853913      true      109.8      2016-04-06
13:44:01      SD      28      91321547      57362176      true      103.8      201
6-04-06 13:43:08      PA

```

28	44768680	58853913	true	109.8	2016-04-06		
13:44:01	SD	28	84234570	28024191	true	95.8	2016
-04-06 13:41:57		ND					
...							

Note: The records shown in this example are truncated for brevity.

28	64369998	97268880	true	117.8	2016-04-06		
13:45:12	VT	28	21537495	79750993	true	69.8	2016
-04-06 13:38:06		MA					
28	64369998	97268880	true	117.8	2016-04-06		
13:45:12	VT	28	97616471	82415825	true	85.8	2016
-04-06 13:40:28		NH					
28	64369998	97268880	true	117.8	2016-04-06		
13:45:12	VT	28	78043267	94059754	true	53.8	2016
-04-06 13:35:44		IL					
28	64369998	97268880	true	117.8	2016-04-06		
13:45:12	VT	28	48732944	31236590	true	83.8	2016
-04-06 13:40:10		NV					
28	64369998	97268880	true	117.8	2016-04-06		
13:45:12	VT	28	55202392	46887305	true	33.8	2016
-04-06 13:32:46		AZ					
28	64369998	97268880	true	117.8	2016-04-06		
13:45:12	VT	28	20605925	94316425	true	51.8	2016
-04-06 13:35:26		ID					
28	64369998	97268880	true	117.8	2016-04-06		
13:45:12	VT	28	30168742	27854650	true	29.8	2016
-04-06 13:32:10		AL					
Time taken: 56.183 seconds, Fetched: 100 row(s)							
hive>							

Pig

1. Start the Pig shell.

```
[impala@ip-172-30-1-117 ~]$ pig
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2016-04-07 12:00:05,019 [main] INFO org.apache.pig.Main - Apache Pig version 0.12.0-cdh5.5.2
(rUnversioned directory) compiled Jan 25 2016, 16:24:52
2016-04-07 12:00:05,019 [main] INFO org.apache.pig.Main - Logging error messages to:
/var/lib/impala/pig_1460044805005.log
2016-04-07 12:00:05,038 [main] INFO org.apache.pig.impl.util.Utils - Default bootup file
/var/lib/impala/.pigbootup not found
2016-04-07 12:00:05,483 [main] INFO org.apache.hadoop.conf.Configuration.deprecation -
mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2016-04-07 12:00:05,483 [main] INFO org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
2016-04-07 12:00:05,483 [main] INFO
org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system
at: nfs://172.30.254.29:2049
2016-04-07 12:00:05,852 [main] INFO org.apache.hadoop.fs.nfs.NFSv3FileSystem - User config file:
/etc/NetAppNFSConnector/conf/users.json
2016-04-07 12:00:05,852 [main] INFO org.apache.hadoop.fs.nfs.NFSv3FileSystem - Group config
file: /etc/NetAppNFSConnector/conf/groups.json
2016-04-07 12:00:05,865 [main] INFO org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
2016-04-07 12:00:05,909 [main] INFO org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
2016-04-07 12:00:05,952 [main] INFO org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
2016-04-07 12:00:05,989 [main] INFO org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
2016-04-07 12:00:06,024 [main] INFO org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
2016-04-07 12:00:06,054 [main] INFO org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
2016-04-07 12:00:06,117 [main] INFO org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
```

```
2016-04-07 12:00:06,147 [main] INFO org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
2016-04-07 12:00:06,177 [main] INFO org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
grunt>
```

2. Read the data from the NFS location by using the LOAD statement.

```
grunt> A = LOAD 'nfs://172.30.254.29:2049/user/impala/sample_data/tab2/tab2.csv' USING
PigStorage(',')
>> AS (id:int, col_1:boolean,col_2:float);
Store with ep Endpoint: host=nfs://172.30.254.29:2049/ export=/ path=/ has fsId 2147484980
grunt> describe A;
A: {id: int,col_1: boolean,col_2: float}
grunt>
```

3. Generate the output of the NFS table.

```
grunt> dump A;
2016-04-07 12:23:38,646 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig features
used in the script: UNKNOWN
2016-04-07 12:23:38,678 [main] INFO
org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABLED=[AddForEach,
ColumnMapKeyPrune, DuplicateForEachColumnRewrite, GroupByConstParallelSetter,
ImplicitSplitInserter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach,
NewPartitionFilterOptimizer, PushDownForEachFlatten, PushUpFilter, SplitFilter,
StreamTypeCastInserter], RULES_DISABLED=[FilterLogicExpressionSimplifier,
PartitionFilterOptimizer]}
2016-04-07 12:23:38,773 [main] INFO
org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation
threshold: 100 optimistic? false
2016-04-07 12:23:38,793 [main] INFO
org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size
before optimization: 1
2016-04-07 12:23:38,793 [main] INFO
org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size
after optimization: 1
```

Note: The output from the previous command was truncated for brevity.

```
2016-04-07 12:23:44,675 [main] INFO
org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - detailed
locations: M: A[1,4],A[-1,-1] C: R:
2016-04-07 12:23:44,711 [main] INFO
org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 0% complete
2016-04-07 12:23:54,222 [main] INFO
org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 50% complete
2016-04-07 12:24:00,090 [main] INFO org.apache.hadoop.conf.Configuration.deprecation -
mapred.reduce.tasks is deprecated. Instead, use mapreduce.job.reduces
2016-04-07 12:24:00,136 [main] INFO org.apache.hadoop.fs.nfs.NFSv3FileSystem -
getAndVerifyHandle(): Parent path nfs://172.30.254.29:2049/tmp/temp1489487504/tmp1307303389 could
not be found
2016-04-07 12:24:00,139 [main] INFO org.apache.hadoop.fs.nfs.NFSv3FileSystem -
getAndVerifyHandle(): Parent path nfs://172.30.254.29:2049/tmp/temp1489487504/tmp1307303389 could
not be found
2016-04-07 12:24:00,139 [main] INFO
org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 100% complete
2016-04-07 12:24:00,142 [main] INFO org.apache.pig.tools.pigstats.SimplePigStats - Script
Statistics:
```

HadoopVersion	PigVersion	UserId	StartedAt	FinishedAt	Features
2.6.0-cdh5.5.2	0.12.0-cdh5.5.2	impala	2016-04-07 12:23:39		2016-04-07
12:24:00	UNKNOWN				

Success!

Job Stats (time in seconds):

JobId	Maps	Reduces	MaxMapTime	MinMapTime	AvgMapTime	Med
ianMapTime		MaxReduceTime	MinReduceTime	AvgReduceTime		MedianReducetime
Alias		Feature	Outputs			

```

job_1460041670554_0004      1      0      3      3      3      3      n/a      n/
a      n/a      n/a      A      MAP_ONLY      nfs://172.30.254.29:2049/tmp/temp14894875
04/tmp942386654,

Input(s):
Successfully read 9 records from:
"nfs://172.30.254.29:2049/user/impala/sample_data/tab2/tab2.csv"

Output(s):
Successfully stored 9 records in: "nfs://172.30.254.29:2049/tmp/temp1489487504/tmp942386654"

Counters:
Total records written : 9
Total bytes written : 0
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1460041670554_0004

2016-04-07 12:24:00,255 [main] INFO
org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2016-04-07 12:24:00,257 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation -
fs.default.name is deprecated. Instead, use fs.defaultFS
2016-04-07 12:24:00,258 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key
[pig.schematuple] was not set... will not generate code.
2016-04-07 12:24:00,287 [main] INFO  org.apache.hadoop.fs.nfs.NFSv3FileSystem - END
getFileBlockLocations(): length is <= start so no locations
2016-04-07 12:24:00,290 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat -
Total input paths to process : 1
2016-04-07 12:24:00,291 [main] INFO
org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
2016-04-07 12:24:00,303 [main] INFO  org.apache.hadoop.fs.nfs.stream.NFSBufferedInputStream -
Changing prefetchBlockLimit to 0
(1,true,12789.123)
(2,false,1243.5)
(3,false,24453.324)
(4,false,2423.3254)
(5,true,243.325)
(60,false,2.43565424E8)
(70,true,243.325)
(80,false,243423.33)
(90,true,243.325)
grunt>

```

References

The following references were used in this technical report:

- Cloudera Impala
https://en.wikipedia.org/wiki/Cloudera_Impala
- Apache Hive
https://en.wikipedia.org/wiki/Apache_Hive
- Mrjobs
<https://github.com/Yelp/mrjob>
- TR-4133: NetApp Private Storage for AWS
<https://www.netapp.com/us/media/tr-4133.pdf>
- TR-4067: Clustered Data ONTAP NFS Best Practice and Implementation Guide
<https://www.netapp.com/us/media/tr-4067.pdf>

Acknowledgements

- Pranoop Erasani, Technical Director, NAS, NetApp
- Paramita Das, Senior Manager, NetApp
- Sunitha Rao, Director Product Management, NetApp
- Christopher Lemmons, Director, DSG Technical Marketing for Business Processing Workloads, NetApp
- Akshay Anant Patil, TME, NetApp
- Ankita Dhawale, TME, NetApp
- Nilesh Bagad, Sr. Product Manager, NetApp

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.

Copyright Information

Copyright © 1994–2016 NetApp, Inc. All rights reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7103 (October 1988) and FAR 52-227-19 (June 1987).

Trademark Information

NetApp, the NetApp logo, Go Further, Faster, AltaVault, ASUP, AutoSupport, Campaign Express, Cloud ONTAP, Clustered Data ONTAP, Customer Fitness, Data ONTAP, DataMotion, Flash Accel, Flash Cache, Flash Pool, FlashRay, FlexArray, FlexCache, FlexClone, FlexPod, FlexScale, FlexShare, FlexVol, FPolicy, GetSuccessful, LockVault, Manage ONTAP, Mars, MetroCluster, MultiStore, NetApp Fitness, NetApp Insight, OnCommand, ONTAP, ONTAPI, RAID DP, RAID-TEC, SANshare, SANtricity, SecureShare, Simplicity, Simulate ONTAP, SnapCenter, SnapCopy, Snap Creator, SnapDrive, SnapIntegrator, SnapLock, SnapManager, SnapMirror, SnapMover, SnapProtect, SnapRestore, Snapshot, SnapValidator, SnapVault, SolidFire, StorageGRID, Tech OnTap, Unbound Cloud, WAFL, and other names are trademarks or registered trademarks of NetApp, Inc. in the United States and/or other countries. All other brands or products are trademarks or registered trademarks of their respective holders and should be treated as such. A current list of NetApp trademarks is available on the web at <http://www.netapp.com/us/legal/netapptmlist.aspx>. TR-4529-0716