Technical Report

NetApp FAS NFS Connector for Hadoop

Karthikeyan Nagalingam and Xing Lin, NetApp
October 2015 | TR-4382

Abstract

This document introduces NetApp® FAS NFS Connector for Apache Hadoop, a product that allows open-source analytics frameworks such as Hadoop to run on NFS storage natively. The topics that we cover include configuration, underlying architecture, primary use cases, integration with Hadoop, and the benefits of using Hadoop with the NetApp clustered Data ONTAP® operating system and its data management capabilities. The NetApp FAS NFS Connector for Hadoop allows analytics to run NetApp FAS with clustered Data ONTAP as the underlying storage operating system. It is easy to install and works with Apache Hadoop, Apache Spark, Tachyon, and Apache HBase, enabling data on NFSv3 to be analyzed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

Big data is defined broadly as large datasets with the mix of structured, unstructured, and semi-structured types of data, which cannot be processed by traditional database systems. Businesses have turned to big data framework that supports analytical tools such as Apache Hadoop to help store and analyze these datasets. Apache Hadoop software is a framework that enables the distributed processing of large and varied datasets, across clusters of computers, by using programming models. Hadoop Distributed File

System (HDFS) provides high throughput of application data. Hadoop provides integration to enhance specific workloads, storage efficiency, and data management.

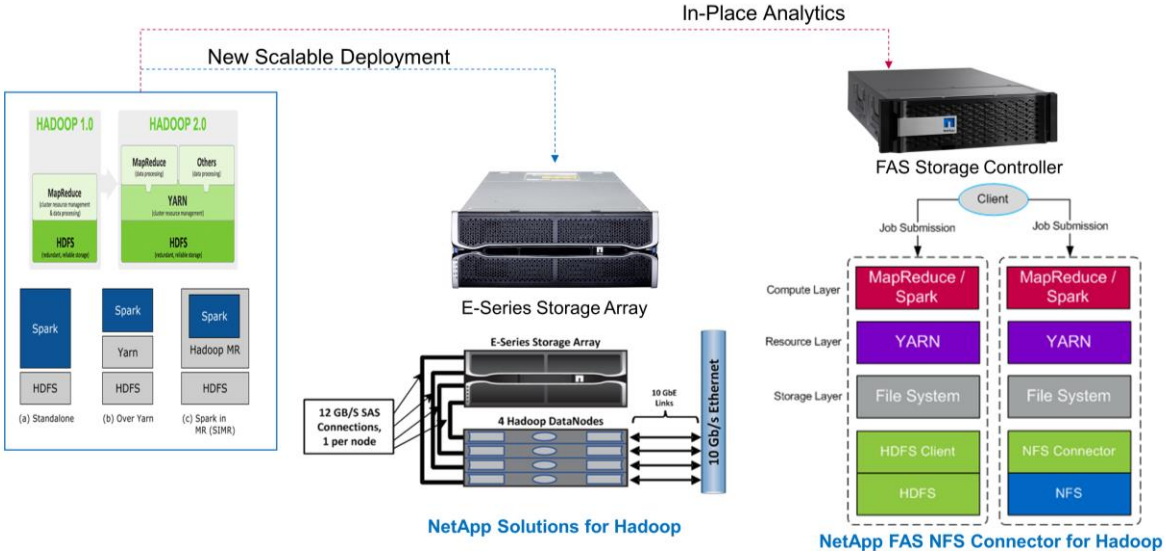
Historically, Hadoop has been used primarily on incoming, external data; however, there's been a need to use Hadoop on existing, internal data, typically stored in network-attached storage (NAS). Typically, this requires setting up another storage silo to host the HDFS and then running the Hadoop analytics on that storage. This, in turn, results in additional data management, more inefficiencies, and additional costs of moving the data between NAS and HDFS.

In this document, we introduce the NetApp NFS Connector for Hadoop, which allows Hadoop to run natively on NFS and without needing to move the data or create a separate data silo on HDFS. We also cover installation, underlying architecture, use cases, integration with Hadoop, and benefits of using analytics with clustered Data ONTAP and its data management capabilities.

1.1 Overview

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

Figure 1) E-Series and FAS arrays for Hadoop.

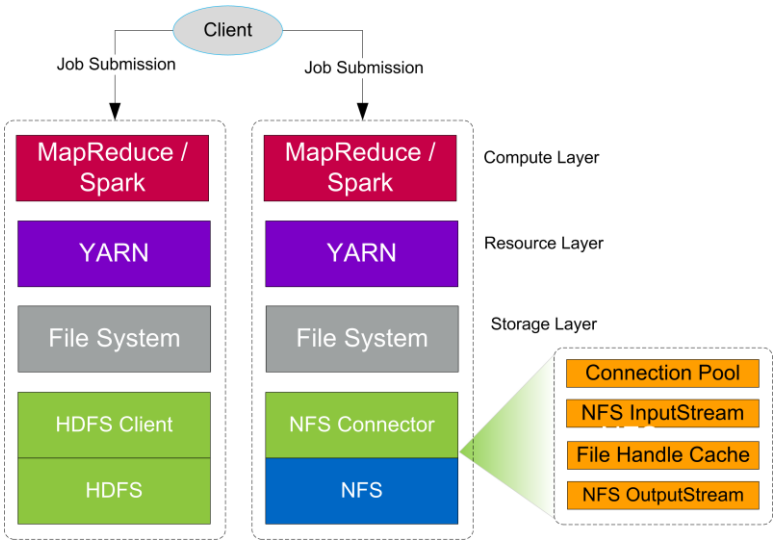


The NetApp NFS Connector for Hadoop allows analytics software to use clustered Data ONTAP. It works with Apache Hadoop and Apache Spark by using a simple configuration file change that enables data on NFSv3 storage to be analyzed. By using clustered Data ONTAP, the connector decouples analytics from storage, leveraging the benefits of NAS. Clustered Data ONTAP enables high availability and storage efficiency. For even higher performance, the NetApp NFS Connector for Hadoop can be combined with Tachyon to build a scale-out caching tier that is backed by clustered Data ONTAP.

The NetApp NFS Connector for Hadoop is an implementation of the file system interface defined by Apache Hadoop. The current implementation is based on the NFSv3 protocol by using the `AUTH_NONE` and `AUTH_SYS` security policies, which are UNIX specific authentications.

Figure 2 shows how the NetApp NFS Connector for Hadoop plugs into Apache Hadoop. This design allows NFS to run side by side with HDFS, or for HDFS to be swapped out for NFS. The selection depends on the use case and on the performance expected by applications that use Apache Hadoop. Regardless of the selection, user applications don't have to be modified.

Figure 2) NetApp NFS Connector for Hadoop plugs into Apache Hadoop.



2 Hadoop in the Enterprise

The use of big data analytics is becoming more popular, but running analytics in an enterprise environment still faces many challenges. Large enterprises have existing hardware and data, and traditional analytics platforms such as Hadoop cannot be easily installed in those environments. Here, we describe four main challenges in running analytics in an enterprise environment:

1. **Enterprises have storage and compute imbalance.** Some environments store large amounts of data but might not analyze all of it continuously; other environments might have smaller amounts of data and analyze it continuously. In a traditional architecture design, compute and storage are tightly linked. With a decoupled design, both the compute and storage tier can be scaled independently.
2. **Enterprises have existing hardware.** Many enterprise environments have a shared infrastructure that is used for home directories, databases, and many other services. In the traditional design, the enterprise must dedicate new hardware and storage for data analytics. In addition, the data must be moved from enterprise storage into siloed analytics storage before it can be used—a costly proposition. In contrast, a decoupled design uses existing storage hardware and runs analytics on data in place.

   Note: A decoupled design is a framework for complex work that allows components to remain completely autonomous and scale independently of each other.

   Note: Sqoop-like tools are required to extract data from databases like RDBMS and store it in a format that can be processed within the Hadoop framework.
3. **Analytics storage JBOD is not efficient.** The storage used for analytics (for example, HDFS) typically utilizes three copies of data, for reliability and for performance. This method is not storage efficient. An enterprise storage system provides reliability by using efficient approaches such as NetApp RAID DP® technology.
4. **Analytics storage JBOD lacks data management.** File systems used by analytics frameworks lack enterprise features such as deduplication, high availability, and disaster recovery. NetApp NFS Connector for Hadoop leverages NetApp clustered Data ONTAP nondisruptive operations to add storage on demand that is independent of data nodes. The connector facilitates the scalable distributed processing of datasets stored in existing enterprise storage systems. The connector also allows access to the analytics file system in the current analytics frameworks, while enabling data analytics on NFS storage. The connector enables NetApp data management features such as

Snapshot® technology, FlexClone® technology, and data protection. It simplifies job workflows and negates the need to copy data across different storage silos.

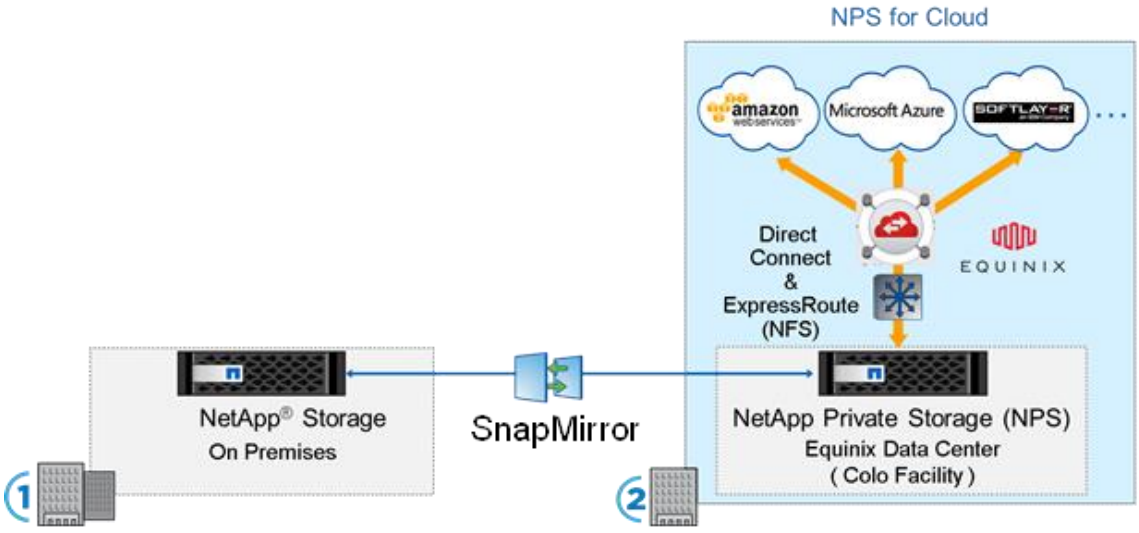
2.1 Benefits and Use Cases

The NetApp NFS Connector for Hadoop allows computations to analyze data stored on enterprise storage systems. The decoupled design of the NetApp NFS Connector for Hadoop provides high functional value in the following scenarios:

- **Analyzing data on enterprise storage.** Companies can leverage their existing investment in enterprise storage and enable analytics incrementally. Many file-based data sources exist, such as source-code repositories, e-mails, and log files. These files are generated by traditional applications but currently require a workflow to ingest the data into a separate analytics file system. NetApp NFS Connector for Hadoop allows a single storage back end to manage and service data for both enterprise and analytics workloads. Data analytics that use the same file system namespace can analyze enterprise data with no additional ingest workflows.
- **Cross data–center deployments.** The decoupled design of NetApp NFS Connector for Hadoop also allows independent scaling of compute and storage layers. This feature gives the flexibility of placing the analytics compute tier on cloud infrastructures, such as Amazon EC2, while keeping the data on premises. In this scenario, up-front hardware purchases are replaced by the pay-as-you-go cloud model. Cross data–center deployments benefit from products such as Amazon Web Services (AWS) Direct Connect and NetApp Private Storage (NPS) that enable high-bandwidth connections between private data centers and public clouds.

  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.

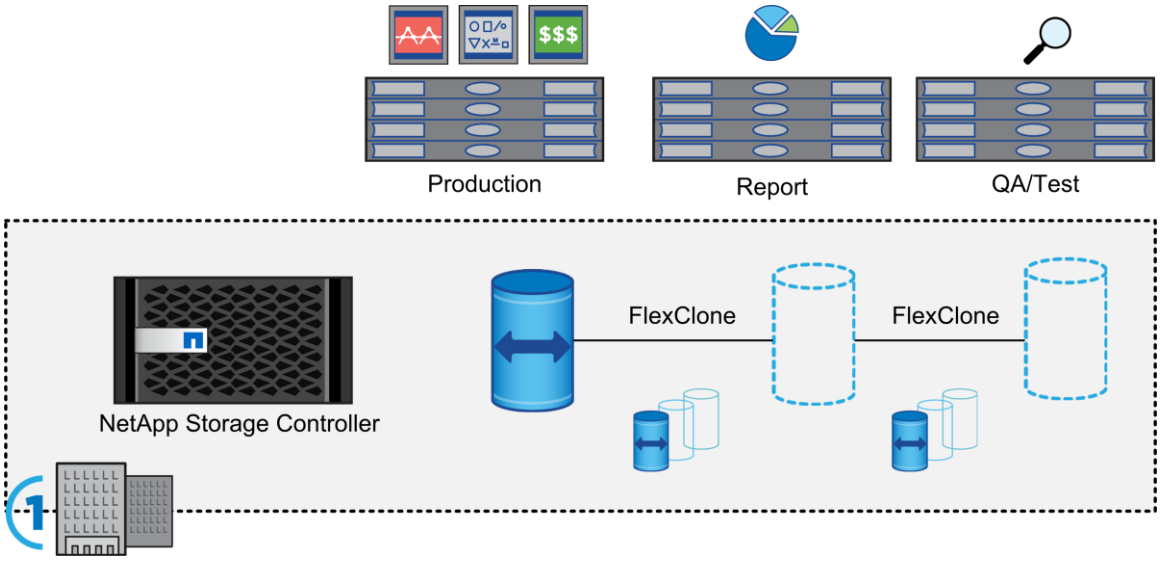
Figure 3) Cross data–center deployments.



Specifically, the NetApp NFS Connector for Hadoop is optimal for the following use cases:

- **Analyze data on existing NFS storage.** NetApp NFS Connector for Hadoop enables analytics on existing workflows and applications that write files to NFS, code repositories on NFS, and data in NetApp SnapLock® volumes.
- **Build testing and QA environments by using clone of existing data.** NetApp NFS Connector for Hadoop allows developers to run variations of analytics code on shared datasets. If the production data is on a FlexVol® volume, then the volume can simply be cloned and used for testing.

Figure 4) Duplicate Hadoop cluster using FlexClone.



- **Leverage storage-level caching for iterative machine learning algorithms.** The iterative machine learning algorithms are cache friendly and compute intensive. The NetApp NFS Connector for Hadoop can leverage storage caches such as NetApp Flash Cache™ caches for acceleration.
- **Use a backup site for analytics.** When data is stored near the cloud (NetApp Private Storage), NetApp NFS Connector for Hadoop allows the use of cloud resources such as Amazon EC2 or Microsoft Azure for analytics, while keeping data on clustered Data ONTAP.

2.2 Deployment Options

The NetApp NFS Connector for Hadoop 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 such as clustered Data 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 storage system.**

Regardless of the approach taken, the NetApp NFS Connector for Hadoop allows analytics to use existing technologies such as Snapshot, RAID DP, NetApp SnapMirror® data replication, and FlexClone.

2.3 Ease of Deployment

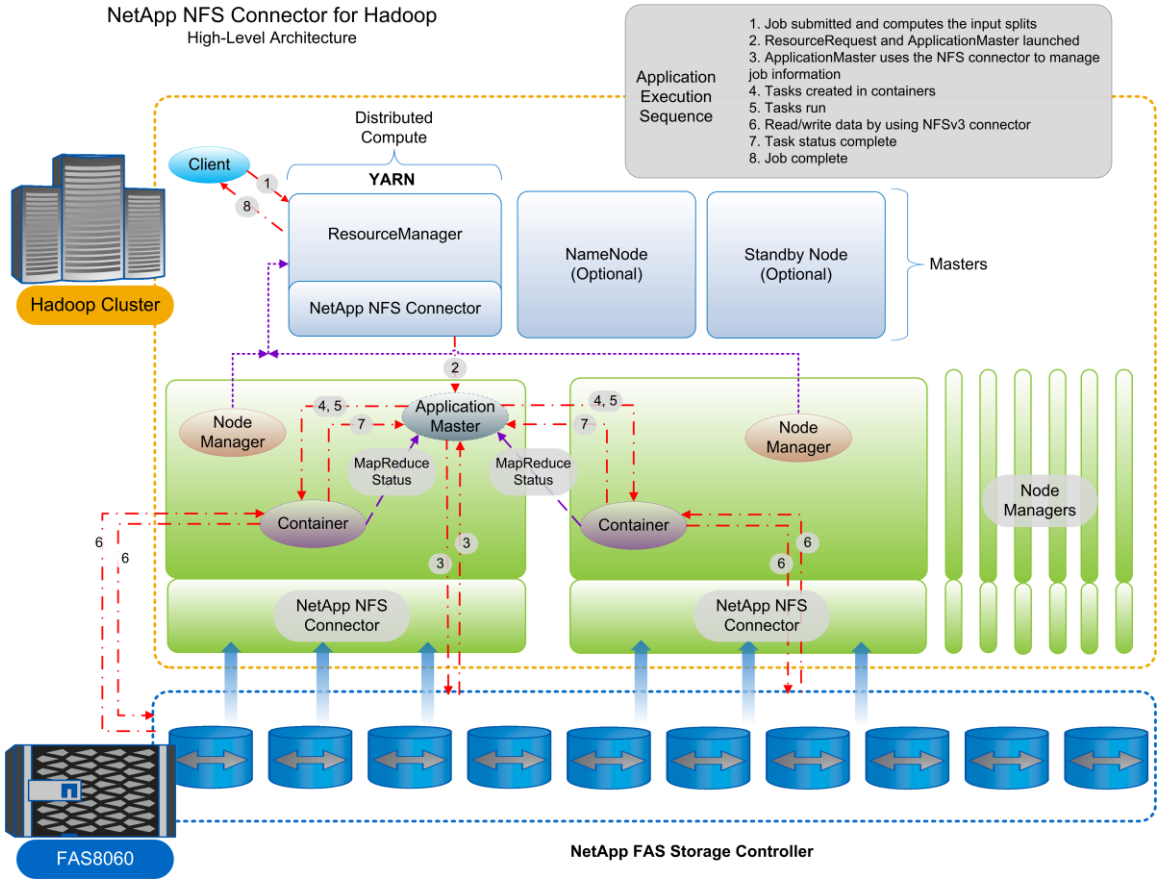
Installing the NetApp NFS Connector for Hadoop is simple. For Apache Hadoop, install the connector JAR file and modify the `core-site.xml` file. A similar change is needed for Apache HBase, where `hbase-site.xml` must be changed. After this modification, applications that use HDFS as their storage system can simply use NFS.

3 NetApp FAS NFS Connector for Hadoop—Architecture and Design

3.1 High-Level Architecture

Figure 5 shows a high-level architecture for NetApp FAS NFS Connector for Hadoop and the application execution sequence.

Figure 5) High-level architecture of NetApp FAS NFS Connector for Hadoop, with application execution sequence.



The high-level architecture of NetApp FAS NFS Connector for Hadoop can be explained through a client's application execution sequence:

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

3.2 Technical Advantages

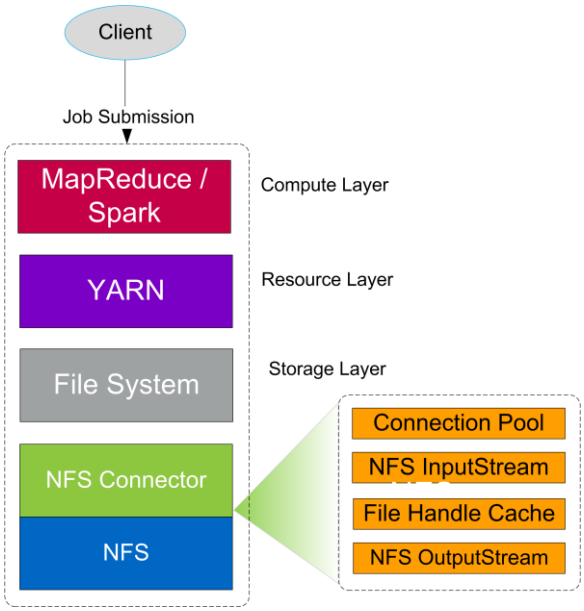
The NetApp FAS NFS Connector for Hadoop has the following technical advantages:

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

3.3 Design

Figure 6 shows the four main components of the NetApp FAS NFS Connector for Hadoop: connection pool, NFS InputStream, file handle cache, and NFS OutputStream. There are two other minor components: authentication and user and group name.

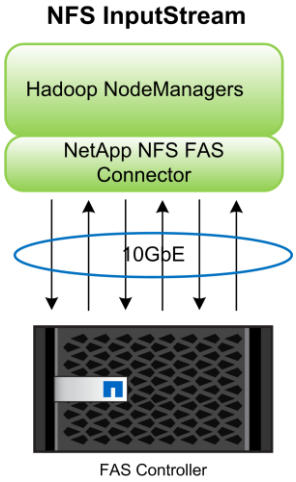
Figure 6) Components of the NetApp FAS NFS Connector for Hadoop.



Connection Pool

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

Figure 7) Connection pool.



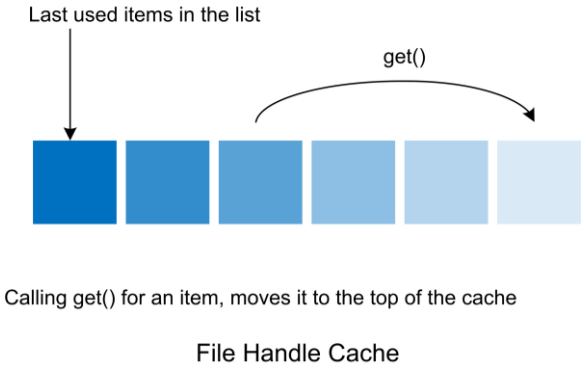
File Handle Cache

A Least Recently Used cache is implemented to cache recently used file handles. This lowers the need to issue frequent lookup operations. It works as follows:

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

The process repeats until it reaches the path that you are looking for.

Figure 8) LRU cache.



NFS InputStream

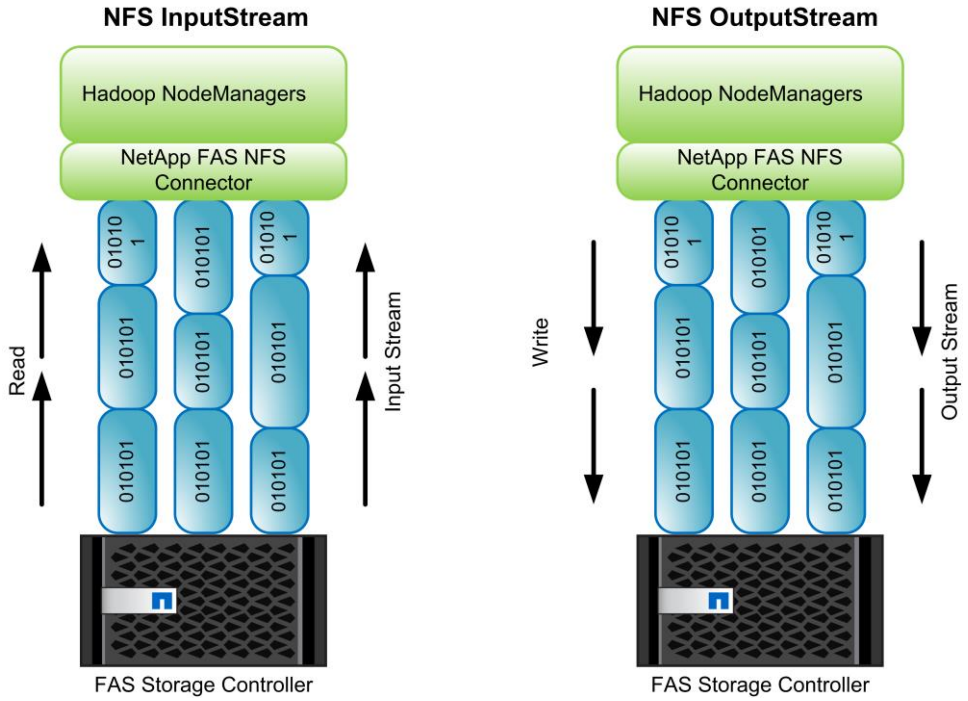
InputStream is the method used by Apache Hadoop to read data from files. The connector is optimized to take full advantage of Hadoop and the underlying network and file system in the following ways: The main optimizations include large sequential reads, multiple outstanding I/Os, and prefetching. Following is more specific information:

- **Large sequential reads.** Applications use InputStream to read data. They read it in bytes required by the application, ranging from a single byte to several kilobytes of data. However, this is not optimal for

NFS. The connector modifies the I/O size issued to NFS to optimize for the underlying network. The default read value is 1MB, but it is configurable by setting the `nfsReadSizeBits` option. If the block size is larger than the maximum read request that the NFS server can support, the connector automatically switches to use the smaller one of these two.

- **Multiple outstanding I/Os.** The connector uses a temporary cache and issues multiple I/O requests in parallel. This allows the amortization of the I/O time and enables you to prefetch aggressively.
- **Prefetching.** Prefetching is used to improve the performance of streaming reads. Upon receiving an on-demand read request, prefetching for the next 128 blocks is issued. To avoid unnecessary prefetching for Hadoop jobs, a heuristic is implemented. Upon receiving a seek request, it sets the last block to prefetch, based on the offset of the seek request and the split size. The connector stops prefetching when it reaches that block. It never prefetches beyond the boundary of a file split. In the other cases, in which the last block to prefetch is not set, the connector simply continues prefetching. The split size is configurable with the `nfsSplitSizeBits` option.

Figure 9) NFS input and output streams.



NFS OutputStream

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

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

Authentication

Currently, we support two types of authentication: none or UNIX. Authentication is configurable with the `nfsAuthScheme` option.

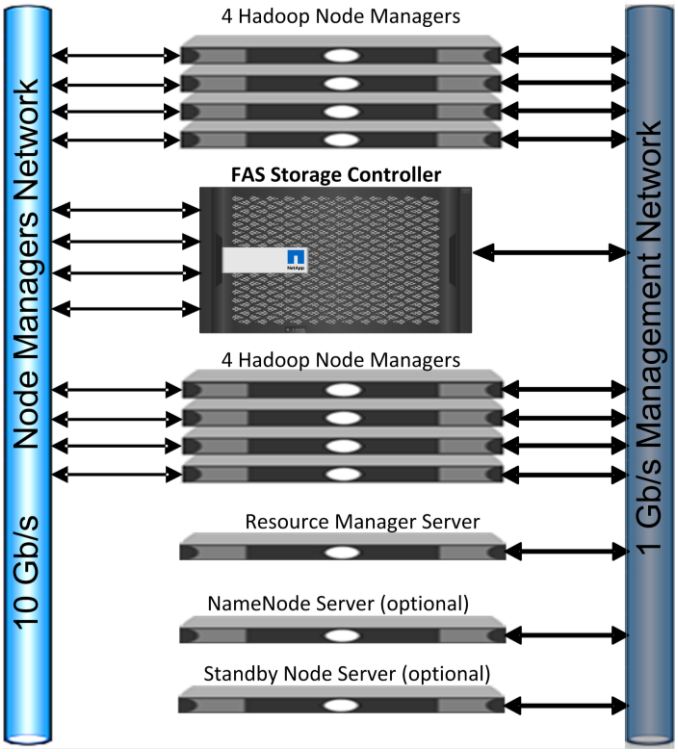
When creating new files or directories, users can set the UID, GID, user name, or group name to use by using the following options: `nfsUid`, `nfsGid`, `nfsUsername` and `nfsGroupname`. By default, root permission is required. NetApp is in the process of adding tighter integration with other authentication schemes such as NIS and Kerberos.

4 Solution Architecture

4.1 Network Architecture

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

Figure 10) Network connectivity.



Recommended Network Topology

Apache Hadoop with the NetApp FAS NFS Connector for Hadoop has the following network topology:

- NodeManagers (two 10GbE ports):
  - Primary purpose: data ingest and movement within a cluster
  - Private interconnection between all NodeManagers
  - Dedicated nonroutable VLAN
- Management Network (GbE):
  - Purpose: system administration network
  - Publicly routable subnet

- NodeManagers and ResourceManager (in Hadoop 2.0) also use GbE
- GbE interface for administration and data transfer purposes on all nodes
- Two GbE ports required by FAS storage systems for management purposes only

4.2 Storage Architecture

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

On a clustered Data ONTAP system, NAS clients access flexible volumes through a storage virtual machine (SVM). SVMs abstract storage services from their underlying hardware.

Flexible volumes containing NAS data are junctioned into the owning SVM in a hierarchy. This hierarchy presents NAS clients with a unified view of the storage, regardless of the physical location of flexible volumes inside the cluster.

When a flexible volume is created within the SVM, the administrator specifies the junction path for the flexible volume. The junction path is a directory location under the root of the SVM where the flexible volume can be accessed. A flexible volume's name and its junction path do not need to be the same.

Junction paths allow each flexible volume to be browsable, like a directory or folder. NFS clients can access multiple flexible volumes using a single mount point. CIFS clients can access multiple flexible volumes using a single CIFS share.

A namespace consists of a group of volumes connected using junction paths. It is the hierarchy of flexible volumes within a single SVM as presented to NAS clients.

The storage architecture can consist of one or more FAS storage controllers. Figure 11 shows a single NetApp FAS controller, with a volume mounted under its default or own namespace.

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

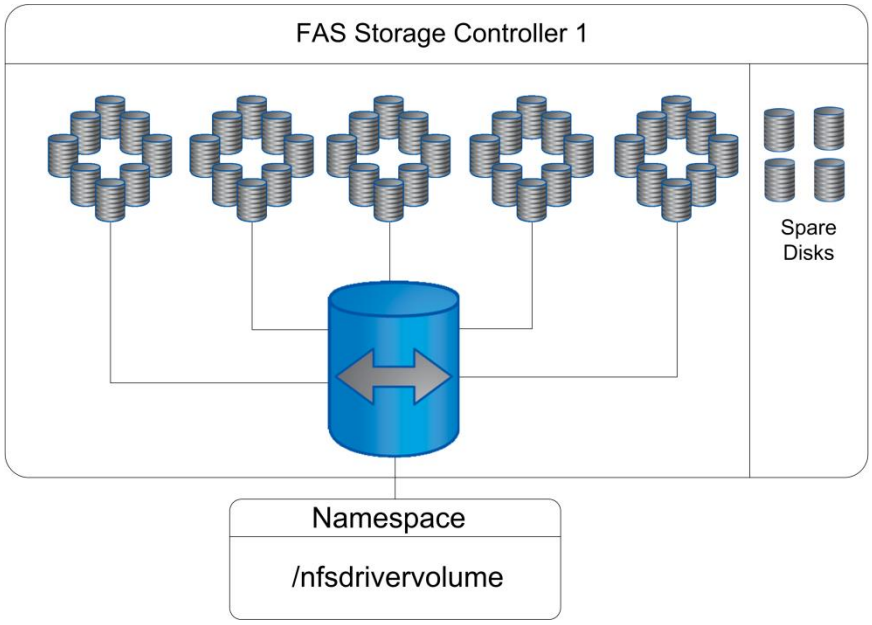
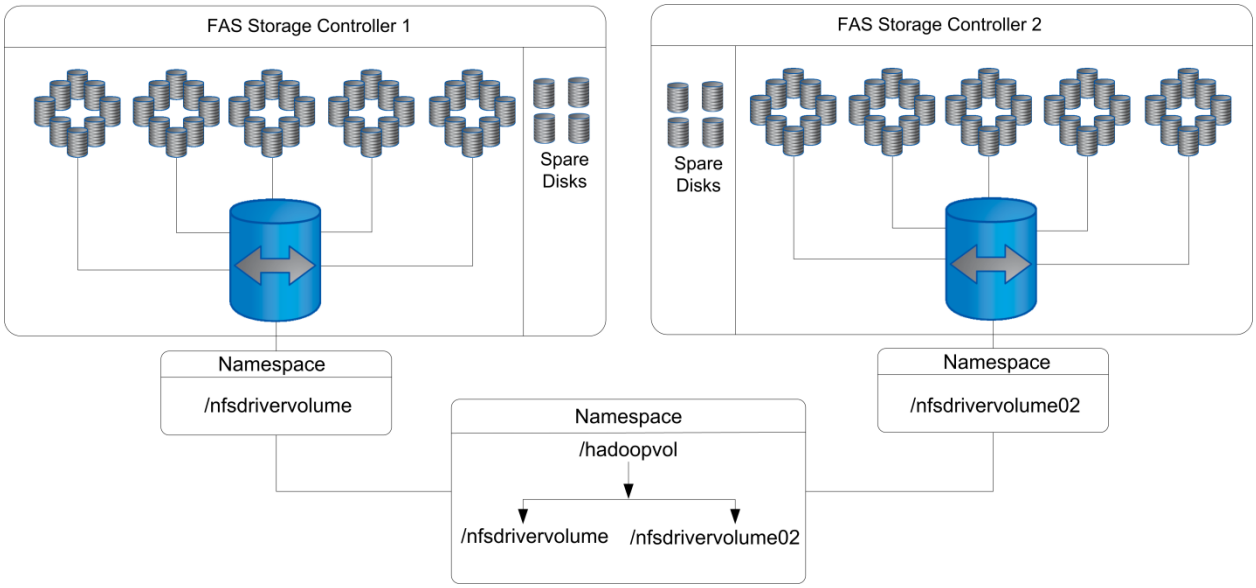


Figure 12 shows a FAS controller, in which each volume from each controller is mounted under one common namespace. The common namespace is a dummy volume that can be from any one of the controllers, and it's configured as `nfsExportPath` in `core-site.xml` in the Hadoop cluster.

NetApp recommends that you create a volume with RAID DP and add more disks for better performance, and keep two disks as global hot spares for up to 100 disk drives of same type.

Figure 12) NFS volumes from two FAS storage controllers.



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

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

4.3 Key Components of Validated Configuration

Table 1 lists the products used for the validation of the NetApp FAS NFS Connector for Hadoop. You can configure the storage based on the NetApp sizer recommendations for your bandwidth (in Mbps), IOPs, and capacity requirements.

Table 1) Components used for the NetApp FAS NFS Connector for Hadoop tests.

Component	Product or Solution	Details
Storage	NetApp FAS8060 storage array with clustered Data ONTAP 8.3 or higher	• 2 controllers, each with 40 x 900GB drives • 10,000 RPM drives • SAS disk type • 10 hot spares • 2 aggregates and 2 volumes • RAID DP
Servers	Intel Xeon E5-2400 series processors	• 4 nodes with two 2.4GHz (6-core) or 2.3GHz (8-core) processors • 12 DIMMs (up to 192GB), up to 1600MHz

Component	Product or Solution	Details
		• 2 x 3.5" HS HDDs • 2 x 1Gb/sec Ethernet ports, 1 x 10GbE network port
Networking	10GbE nonblocking network switch 1GbE network switch	A Cisco Nexus 5000 was used for testing (or a compatible 10GbE network switch)
Operating system	Red Hat Enterprise Linux Server 6.2 (x86_64) or later	Hadoop typically requires a Linux distribution
Hadoop distribution	Cloudera Distribution for Hadoop (also works)	• Cloudera Enterprise Core 5.4.x (recommended) • Cloudera Manager 5.4.x
	Hortonworks Data Platform 2.2.0.0-2041 (tested)	Apache Ambari 1.7

5 Installation and Configuration

The NetApp FAS NFS Connector for Hadoop installation is simple and easy to configure with a Hadoop cluster.

5.1 Installation

To install the NetApp FAS NFS Connector for Hadoop, complete the following steps:

1. Download the JAR file (`hadoop-connector-nfsv3-1.0.jar`) from https://github.com/NetApp/NetApp-Hadoop-NFS-Connector/releases.
2. Copy the JAR file to the ResourceManager and NodeManagers file path, `/usr/lib/hadoop/lib`, either by using the following example or by doing it manually.

```
[root@stlrx300s6-22 ~]# for i in 10.63.150.103 10.63.150.105 10.63.150.107 10.63.150.109
10.63.150.111 10.63.150.113 10.63.150.115 10.63.150.117 10.63.150.119 10.63.150.121 10.63.150.123
; do echo $i; scp hadoop-connector-nfsv3-1.0.jar $i:/usr/lib/hadoop/lib/hadoop-connector-nfsv3-
1.0.jar; done
10.63.150.103
hadoop-connector-nfsv3-1.0.jar                                100%   81KB  80.7KB/s   00:00
10.63.150.105
hadoop-connector-nfsv3-1.0.jar                                100%   81KB  80.7KB/s   00:00
10.63.150.107
hadoop-connector-nfsv3-1.0.jar                                100%   81KB  80.7KB/s   00:00
10.63.150.109
hadoop-connector-nfsv3-1.0.jar                                100%   81KB  80.7KB/s   00:00
10.63.150.111
hadoop-connector-nfsv3-1.0.jar                                100%   81KB  80.7KB/s   00:00
10.63.150.113
hadoop-connector-nfsv3-1.0.jar                                100%   81KB  80.7KB/s   00:00
10.63.150.115
hadoop-connector-nfsv3-1.0.jar                                100%   81KB  80.7KB/s   00:00
10.63.150.117
hadoop-connector-nfsv3-1.0.jar                                100%   81KB  80.7KB/s   00:00
10.63.150.119
hadoop-connector-nfsv3-1.0.jar                                100%   81KB  80.7KB/s   00:00
10.63.150.121
hadoop-connector-nfsv3-1.0.jar                                100%   81KB  80.7KB/s   00:00
10.63.150.123
hadoop-connector-nfsv3-1.0.jar                                100%   81KB  80.7KB/s   00:00
[root@stlrx300s6-22 ~]#
```

5.2 Configuration

To configure the FAS storage controller and the Hadoop cluster, complete the following steps:

1. In the FAS storage controller, do the following:
   a. Create a storage virtual machine (SVM) with NFS access, if the SVM does not exist.

   Note: SVMs were previously called, and are synonymous with, Vservers. The term Vserver is still used in commands and in GUIs.

   b. Create a volume in the SVM, if the in-place analytics volume does not exist.
   c. Create at least one LIF with data access to the volume from NodeManagers.
   d. NetApp recommends having a private network or separate VLAN with NodeManagers and FAS storage controllers.
   e. Allow `mount/nfs` from nonreserved ports (basically this allows Hadoop to mount from ports > 1024).
   f. Disable the `nfs-rootonly` and `mount-rootonly` options to Vserver.

```
FAS8060-cmode::> vserver nfs modify -vserver nfsdrivervserver -nfs-rootonly disabled

FAS8060-cmode::> vserver nfs modify -vserver nfsdrivervserver  -mount-rootonly disabled

FAS8060-cmode::>
```

   g. Increase the NFS read message size to 1MB and the write size to 65536 bytes.

```
FAS8060-cmode::> set advanced

Warning: These advanced commands are potentially dangerous; use them only when directed to do so
by NetApp personnel.
Do you want to continue? {y|n}: y

FAS8060-cmode::*> vserver nfs modify -vserver nfsdrivervserver -v3-tcp-max-read-size 1048576

FAS8060-cmode::*> vserver nfs modify -vserver nfsdrivervserver -v3-tcp-max-write-size 65536

FAS8060-cmode::*>
```

2. In the Hadoop cluster, configure the following parameters in nfs-mapping.json on all nodes:

Parameter	Default Value	Setting Used	Description
Global Options			
Name	NA	rtp	A human-readable name to describe the NFS data
URL	NA	nfs://192.168.120.207:2049	Primary IP of the NFS data
NFS Configuration			
nfsExportPath	/	/hadoopvol	The exported path of the NFS data
nfsReadSizeBits	20	20	The size of NFS read requests (will auto-negotiate with NFS server)
nfsWriteSizeBits	20	20	The size of NFS write requests (will auto-negotiate with NFS server)
nfsSplitSizeBits	28	30	The size of the InputSplit used

Parameter	Default Value	Setting Used	Description
nfsAuthScheme	AUTH_NONE	AUTH_SYS	Authentication scheme used for NFS
nfsUsername	Root	root	User name used to mount and access data (for `AUTH_SYS`)
nfsGroupname	Root	root	Group name used to mount and access data (for `AUTH_SYS`)
nfsUid	0	0	User id used to mount and access data (for `AUTH_SYS`)
nfsGid	0	0	User id used to mount and access data (for `AUTH_SYS`)
nfsPort	2049	2049	Port used for NFS connections (use in case Portmapper is not running)
nfsMountPort	-1	-1	Port used for MOUNT (in case Portmapper is not running)
nfsRpcbindPort	111	111	Port used for Portmapper connections
Endpoint Configuration (for each item in list)			
host	NA	nfs://192.168.150.210:2049	Secondary IP of the NFS data
path	NA	/nfsdrivervolume/	The path in the NFS SVM for which the IP is suited

a. Create a NFS configuration file (`nfs-mapping.json`).

Note: The NetApp FAS NFS connector uses a JSON (Javascript Object Notation) file to describe the NFS configuration and the layout of the clustered Data ONTAP nodes. Each NFS configuration is called a “space” and each space has its own options and endpoints. For more information about JSON, refer to www.json.org.

b. Modify `core-site.xml` in Hadoop and `hbase-site.xml` in Apache HBase.

Parameter	Setting Value	Description
fs.nfs.prefetch	True	Enables prefetch for the InputStream
fs.defaultFS	nfs://192.168.120.207:2049	Name of the default file system specified as a URL
fs.AbstractFileSystem.nfs.impl	org.apache.hadoop.fs.nfs.NFSv3AbstractFilesystem	Allows Hadoop 2.0 to find the connector for NFS
fs.nfs.impl	org.apache.hadoop.fs.nfs.NFSv3FileSystem	Allows Hadoop to find the NetApp NAS NFS connector
fs.nfs.configuration	/etc/Hadoop/conf/nfs-mapping.json	Defines the cluster architecture for NetApp FAS NFS connector

Best Practices

- Create separate networks between the volumes and NodeManager for improved network bandwidth for in-place analytics and new deployments.
- Spread the volumes across controllers equally to distribute the load.

```
 [root@stlrx300s6-25 conf]# cat /etc/hadoop/conf/nfs-mapping.json
{
    "spaces": [
        {
            "name": "rtp",
            "uri": "nfs://192.168.120.207:2049/",
            "options": {
                "nfsExportPath": "/hadoopvol",
                "nfsReadSizeBits": 20,
                "nfsWriteSizeBits": 20,
                "nfsSplitSizeBits": 30,
                "nfsAuthScheme": "AUTH_SYS",
                "nfsUsername": "root",
                "nfsGroupname": "root",
                "nfsUid": 0,
                "nfsGid": 0,
                "nfsPort": 2049,
                "nfsMountPort": -1,
                "nfsRpcbindPort": 111
            },
            "endpoints": [
                {
                    "host": "nfs://192.168.120.207:2049/",
                    "path": "/nfsdrivervolume/"
                },
                {
                    "host": "nfs://192.168.120.210:2049/",
                    "path": "/nfsdrivervolume/"
                },
                {
                    "host": "nfs://192.168.120.211:2049/",
                    "path": "/nfsdrivervolume/"
                },
                {
                    "host": "nfs://192.168.120.209:2049/",
                    "path": "/nfsdrivervolume/"
                },
                {
                    "host": "nfs://192.168.120.102:2049/",
                    "path": "/nfsdrivervolume02/"
                },
                {
                    "host": "nfs://192.168.120.100:2049/",
                    "path": "/nfsdrivervolume02/"
                },
                {
                    "host": "nfs://192.168.120.101:2049/",
                    "path": "/nfsdrivervolume02/"
                }
            ]
        }
    ]
}
[root@stlrx300s6-25 conf]#
```

5.3 Verification

To verify the NetApp FAS NFS connector for Hadoop installation and configuration, do the following:

1. Use `copyFromLocal` to test the NetApp FAS NFS Connector for Hadoop.

```
[root@stlrx300s6-22 conf]# hadoop fs -copyFromLocal ./slaves nfs://192.168.120.207:2049/slaves4
[root@stlrx300s6-22 conf]#
```

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

```
[root@stlrx300s6-22 ~]# hadoop fs -ls nfs://192.168.120.207:2049/
Store with ep Endpoint: host=nfs://192.168.120.207:2049/ export=/hadoopvol path=/ has fsId
2147484678
Found 9 items
-rw-r--r--   1 root root        831 2015-01-20 15:19
nfs://192.168.120.207:2049/5bresults_two_vols.txt
drwxrwxrwx   - root root       4096 2015-01-20 14:15 nfs://192.168.120.207:2049/app-logs
drwxrwxrwx   - root root       4096 2015-01-20 14:04 nfs://192.168.120.207:2049/mapred
drwxrwxrwx   - root root       4096 2015-01-20 14:04 nfs://192.168.120.207:2049/mr-history
drwxr-xr-x   - root root       4096 2015-02-09 14:00 nfs://192.168.120.207:2049/nfsdrivervolume
drwxr-xr-x   - root root       4096 2015-02-09 04:10 nfs://192.168.120.207:2049/nfsdrivervolume02
-rw-r--r--   1 root root        526 2015-02-06 14:12
nfs://192.168.120.207:2049/outputdir.xml.template
-rw-r--r--   1 root root 5368709120 2015-02-09 06:42 nfs://192.168.120.207:2049/slaves4
drwx------   - root root       4096 2015-01-20 14:15 nfs://192.168.120.207:2049/user
[root@stlrx300s6-22 ~]#
#
```

6 Product Validation

To validate NetApp FAS NFS Connector for Hadoop for real world deployments, we used the TeraGen tool to generate a Hadoop dataset. We then used the TeraSort tool to conduct a MapReduce/YARN process to verify that the configuration worked as expected.

The following subsections present the details of each test we conducted to validate the NetApp FAS NFS Connector for Hadoop with Hadoop 2.6.

6.1 Basic Hadoop Functionality Validation

Setup Procedure

To set up basic validation testing of Hadoop functionality, we followed these steps:

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

Run Procedure

To run the validation testing, we followed this step:

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

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

Test Information	Details
Test type	Initial tuning and full function
Execution type	Automated
Configuration	• Memory configured per the Hortonworks guide. – `yarn.nodemanager.resource.memory-mb = 43008` – `yarn.scheduler.minimum-allocation-mb = 2048` – `yarn.scheduler.maximum-allocation-mb = 43008`

Test Information	Details
	– yarn.app.mapreduce.am.resource.mb = 4096 – mapreduce.map.java.opts = -Xmx1638m – mapreduce.reduce.java.opts = -Xmx3276m – mapreduce.map.memory.mb = 2048 – mapreduce.reduce.memory.mb = 4096 • TeraGen options: – -Dmapreduce.job.maps=167 • TeraSort options: – -Dmapreduce.job.reduces=167
Duration	Multiple runs, one day total
Description	This test runs a TeraGen job with duration greater than 10 minutes to generate a substantial dataset. It then runs a TeraSort job on the dataset created by TeraGen.
Prerequisites	The NodeManagers components have been started.
Test results	• Proper output results are received from the TeraSort reduce stage. • No tasks on individual task nodes (NodeManagers) fail. • The file system (NFS) maintains integrity and is not corrupted. • All test environment components are still running.
Notes	• Use integrated web and UI tools to monitor the Hadoop tasks and file system. • Use Ganglia to monitor the Linux server in general.

7 Hadoop TeraGen and TeraSort Validation

7.1 TeraGen and TeraSort Validation with FAS8060

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

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

Figure 13 shows the elapsed time to create the different datasets by using TeraGen. Creating a 1TB dataset took over 11 minutes, and no issues were logged during the TeraGen operations. Also, the time required to generate the datasets increased proportionally with the size of the dataset, indicating that the cluster maintained data ingest rates over time.

Note: We used two volumes from two FAS8060 storage controllers for this testing with TeraGen and TeraSort.

The performance validation done based on Table 1.

Figure 13) TeraGen dataset creation report.

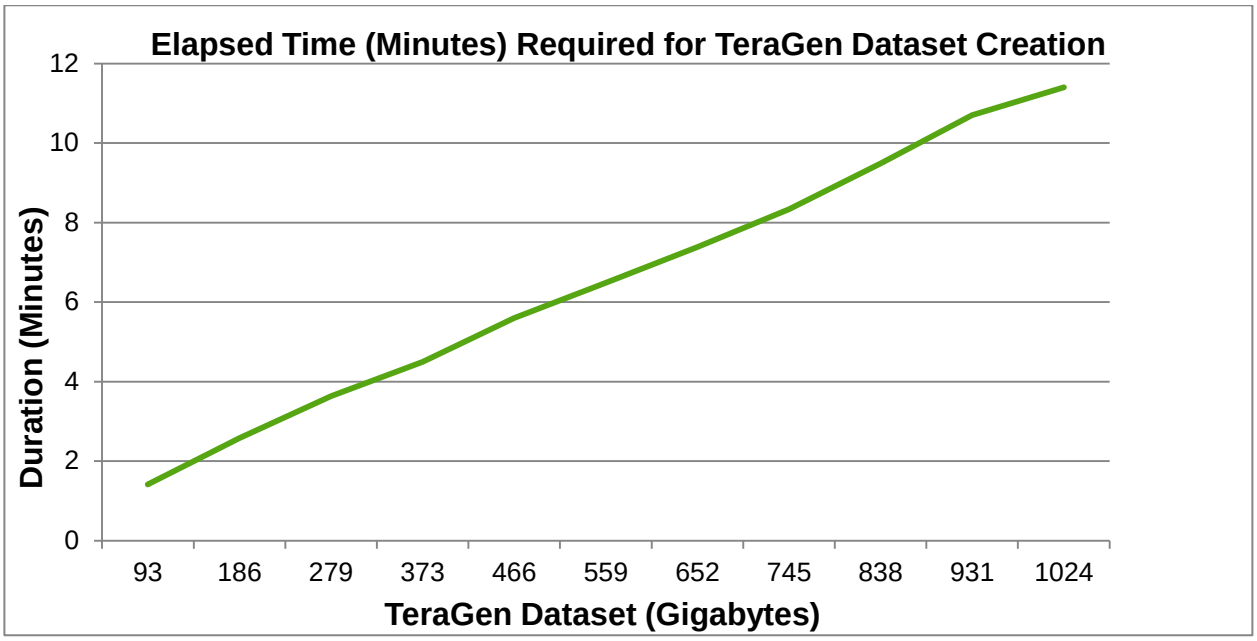


Figure 14 shows the elapsed time required to complete a TeraSort job on each of the increasingly larger datasets described in the preceding paragraphs. The 1TB dataset required 1 hour, 50 minutes, completing the process, and no issues were logged during the TeraSort operations. These results demonstrate that the Hadoop cluster maintained comparable processing rates as the size of the dataset increased. They also demonstrate the stability of the overall Hadoop cluster.

Figure 14) TeraSort job completion report.

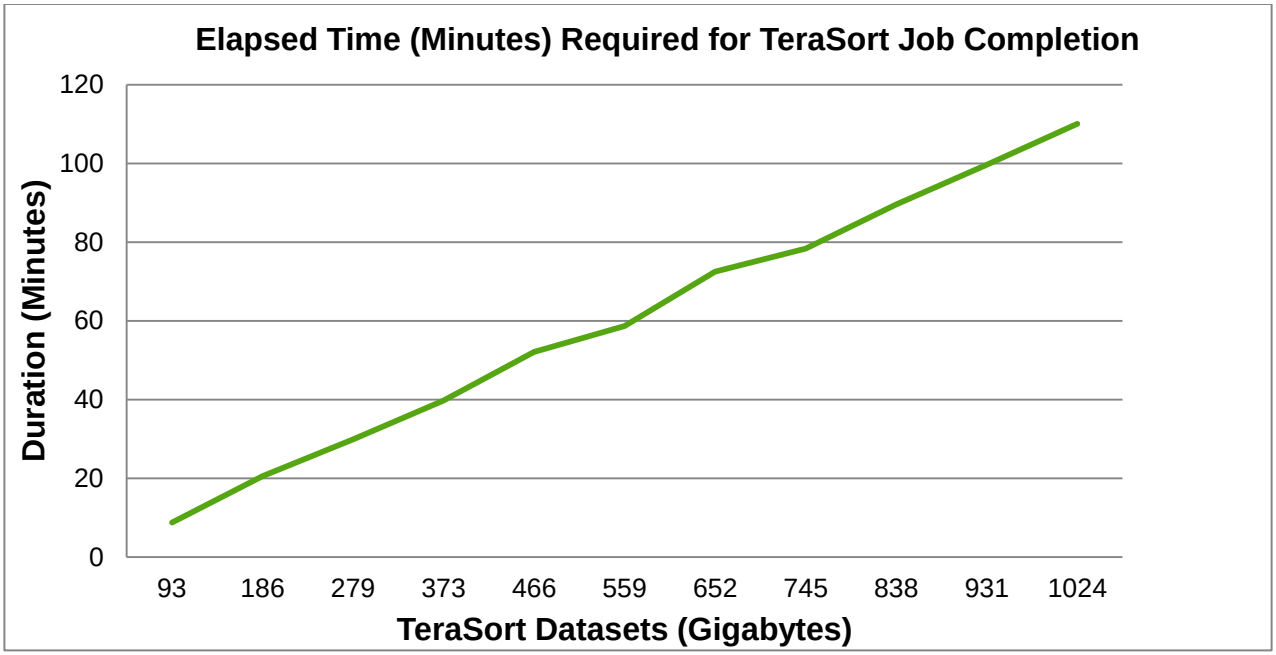
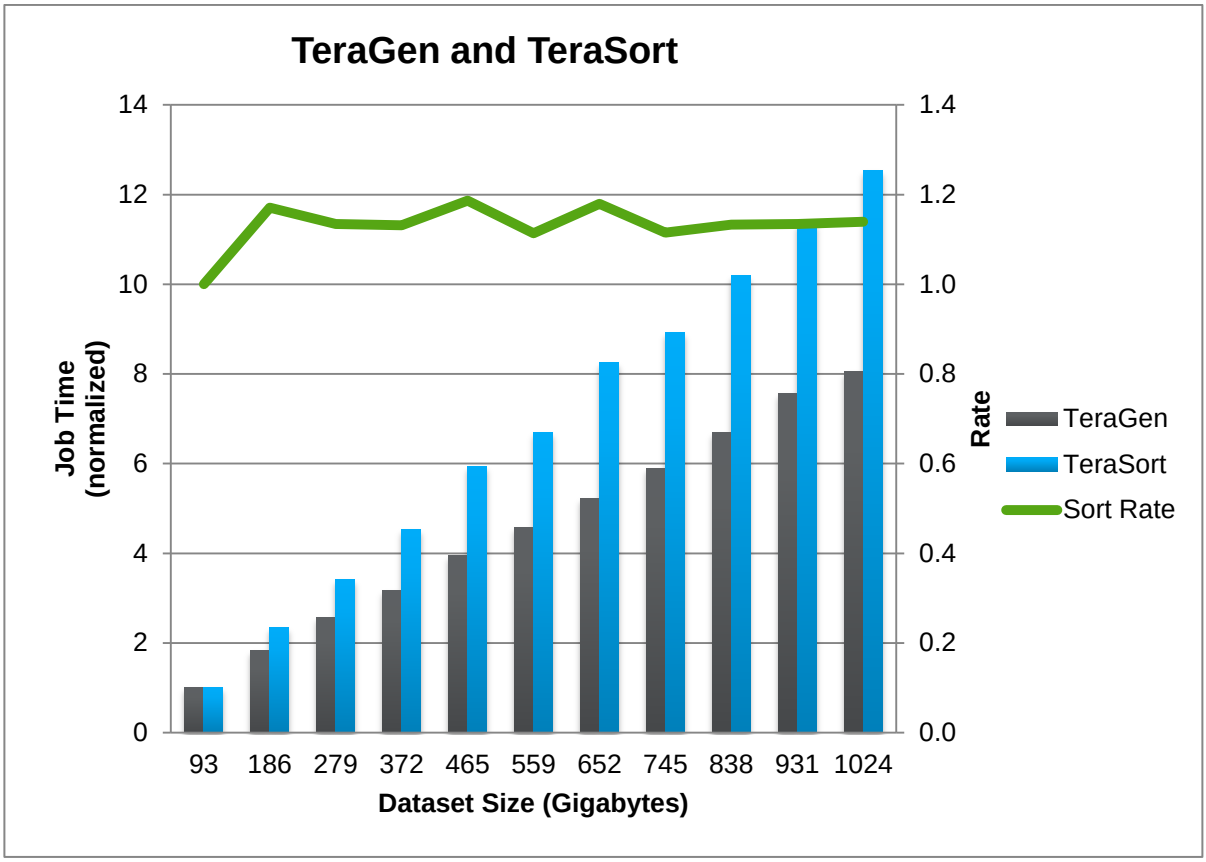


Figure 15 shows the normalized performance of TeraGen and TeraSort as the dataset size is increased. We run the benchmarks for 1,000,000,000 to 11,000,000,000 records dataset. The results show that the

validated configuration performs well and the data generation time and time to sort is linear to the dataset size. This is further supported by the sort rate, defined as the normalized time to sort one billion rows, which stays between 1.1 and 1.2 as the dataset size is increased.

Figure 15) TeraGen and TeraSort normalized report.



The tests are based on four Hadoop NodeManagers and one FAS8060HA array with two storage controllers. During the test, we observed that the storage controllers and disk utilization was less than 20%. Essentially there was a lot of headroom in the storage system to perform additional operations.

7.2 Teragen and TeraSort Validation with NetApp FAS8080

In addition to the basic validation with 4-node Hadoop cluster, a scale-out validation was also performed using the 28-node Hadoop cluster on FAS8080EX cluster with 8 storage controllers. The configuration included 3 masters, 24 slaves, and 1 remote server. The performance was validated by using TeraGen and TeraSort.

Table 2) Hardware configuration used for the testing.

Component	Product or Solution	Details
Storage	NetApp FAS8080 storage array with clustered Data ONTAP 8.3	• 8 controllers(4 HA pairs), each with 48 x 900GB, SAS drives (10K RPM) • 2 hot spares per disk shelf • 1 data aggregate (40 drives) per controller • 1 volume per controller (total of 8 volumes)

Component	Product or Solution	Details
Servers	Intel Xeon E5-2400 series processors	• 28 servers and each have two 2.4GHz (6-core) or 2.3GHz (8-core) processors (40 cores) • 12 DIMMs (up to 192GB), up to 1600MHz (256GB/node) • 2 x 1Gb/sec Ethernet ports, 1 x 10GbE network port
Networking	• 10GbE nonblocking network switch • 1GbE network switch	A Cisco Nexus 5000 switch was used for testing. Any compatible 10GbE network switch can also be used.
Server operating system	Red Hat Enterprise Linux Server 6.2 (x86_64) or later	Hadoop typically requires a Linux distribution
Hadoop distribution used in the testing	Hortonworks Data Platform 2.2.0.0-2041 (Also works with Cloudera distribution)	Apache Ambari 1.7

Figure 16) Server connections.

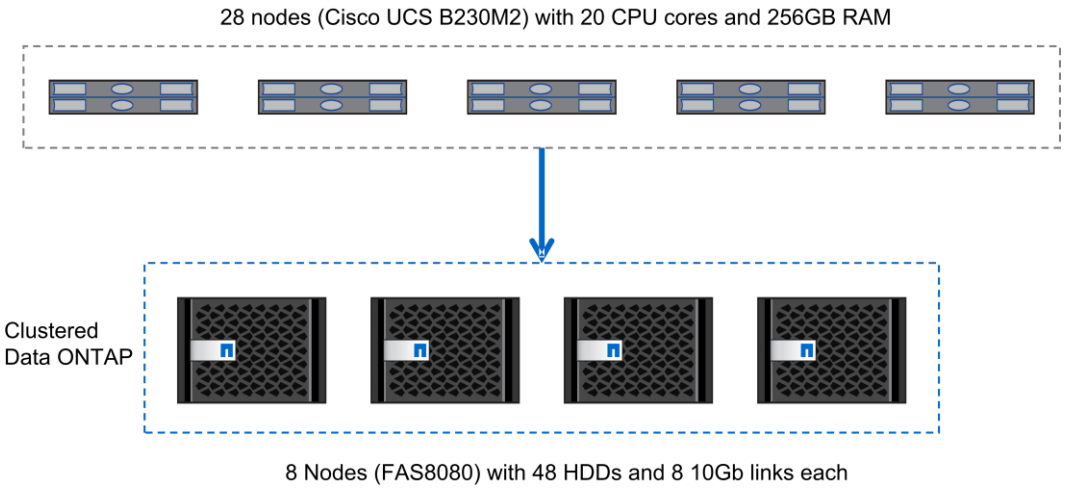
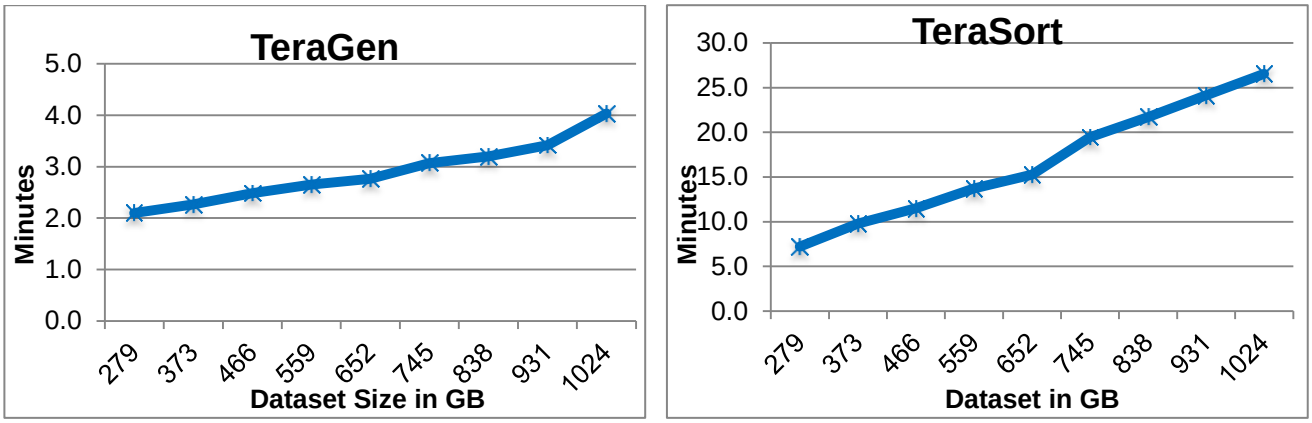


Figure 17) TeraGen and TeraSort results (SAS drives).



Optimal TeraGen and TeraSort results were achieved as shown in the previous graphs.

The eight-node FAS8080EX storage configuration was underutilized throughout this testing and had a lot of performance headroom. Controller and disk utilization were less than 10% and the throughputs less than 2GB/sec for write and less than 3GB/sec from reads are much lower than the maximum.

The following Json file was used for testing:

```
{
    "spaces": [
        {
            "name": "test",
            "uri": "nfs://10.235.208.183:2049/",
            "options": {
                "nfsExportPath": "/mnt/ram",
                "nfsReadSizeBits": 23,
                "nfsWriteSizeBits": 23,
                "nfsSplitSizeBits": 29,
                "nfsAuthScheme": "AUTH_NONE",
                "nfsUsername": "root",
                "nfsGroupname": "root",
                "nfsUserConfigFile":"/etc/hadoop/conf/users.json",
                "nfsGroupConfigFile":"/etc/hadoop/conf/groups.json",
                "nfsUid": 0,
                "nfsGid": 0,
                "nfsPort": 2049,
                "nfsMountPort": -1,
                "nfsRpcbindPort": 111
            },
            "endpoints": [
                 {
                     "host": "nfs://10.235.208.183:2049/",
                     "exportPath": "/mnt/ram",
                     "path": "/"
                 }
            ]
        },
        {
            "name": "default",
            "uri": "nfs://10.237.21.88:2049/",
            "options": {
                "nfsExportPath": "/",
                "nfsReadSizeBits": 20,
                "nfsWriteSizeBits": 20,
                "nfsSplitSizeBits": 29,
                "nfsAuthScheme": "AUTH_SYS",
```

```
                "nfsUsername": "root",
                "nfsGroupname": "root",
                "nfsUserConfigFile":"/etc/hadoop/conf/users.json",
                "nfsGroupConfigFile":"/etc/hadoop/conf/groups.json",
                "nfsUid": 0,
                "nfsGid": 0,
                "nfsPort": 2049,
                "nfsMountPort": -1,
                "nfsRpcbindPort": 111
            },
            "endpoints": [
                {
                    "host": "nfs://10.237.21.87:2049/",
                    "exportPath": "/nfsconnectorvol1_aggr1_sti8080_265_1/",
                    "path": "/nfsconnectorvol1_aggr1_sti8080_265_1"
                },
                {
                    "host": "nfs://10.237.21.88:2049/",
                    "exportPath": "/nfsconnectorvol1_aggr1_sti8080_265_1/",
                    "path": "/nfsconnectorvol1_aggr1_sti8080_265_1"
                },
                {
                    "host": "nfs://10.237.21.91:2049/",
                    "exportPath": "/nfsconnectorvol1_aggr1_sti8080_265_1/",
                    "path": "/nfsconnectorvol1_aggr1_sti8080_265_1"
                },
                {
                    "host": "nfs://10.237.21.92:2049/",
                    "exportPath": "/nfsconnectorvol1_aggr1_sti8080_265_1/",
                    "path": "/nfsconnectorvol1_aggr1_sti8080_265_1"
                },
                {
                    "host": "nfs://10.237.21.149:2049/",
                    "exportPath": "/nfsconnectorvol1_aggr1_sti8080_265_1/",
                    "path": "/nfsconnectorvol1_aggr1_sti8080_265_1"
                },
```

Removed long list of endpoints entries….

```
                {
                    "host": "nfs://10.237.21.164:2049/",
                    "exportPath": "/nfsconnectorssdvol4_sti8080_272_ssd_1/",
                    "path": "/nfsconnectorssdvol4_sti8080_272_ssd_1"
                }
            ]
        }
    ]
}
```

7.3 TeraGen and TeraSort Validation with NetApp All-Flash FAS

This section describes the TeraGen and TeraSort validation with NetApp all-flash FAS8080 in a virtualized environment with Hadoop cluster running on VMware vSphere.

Figure 18) Hadoop infrastructure.

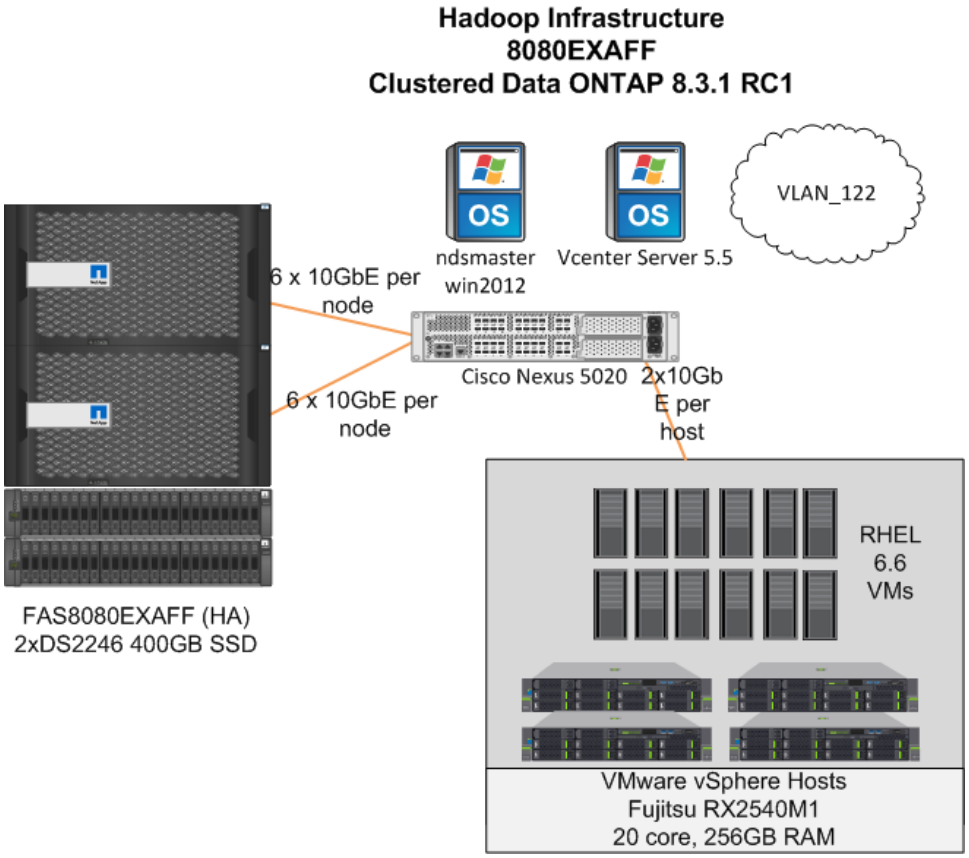


Figure 19) Network connectivity.

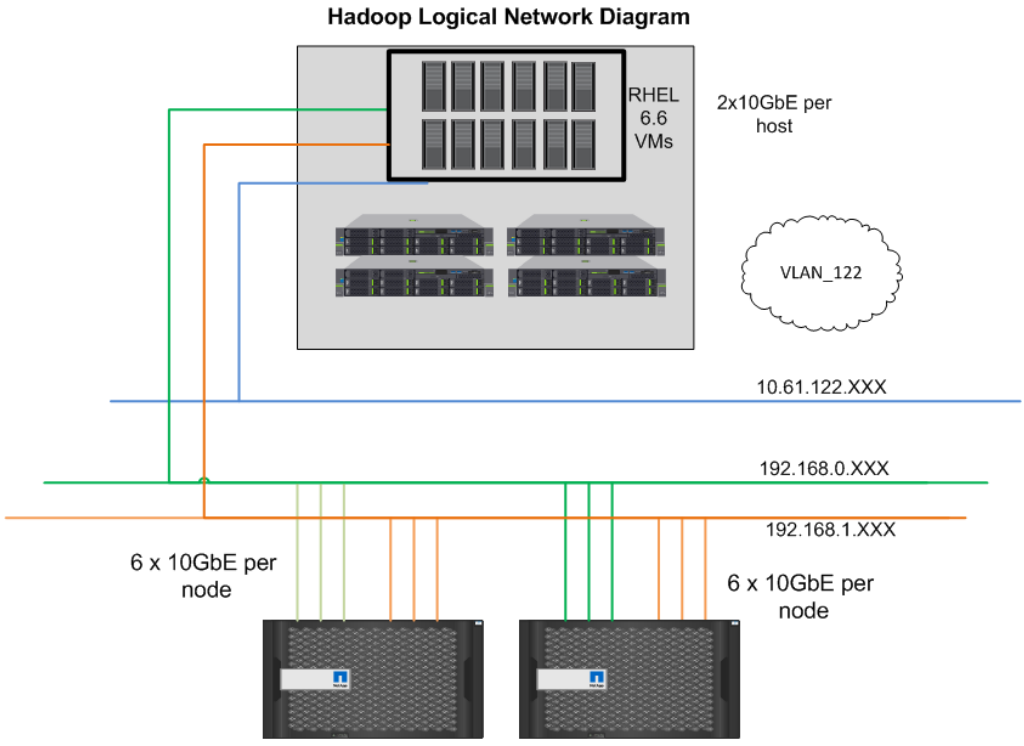


Figure 20) Volume layout.

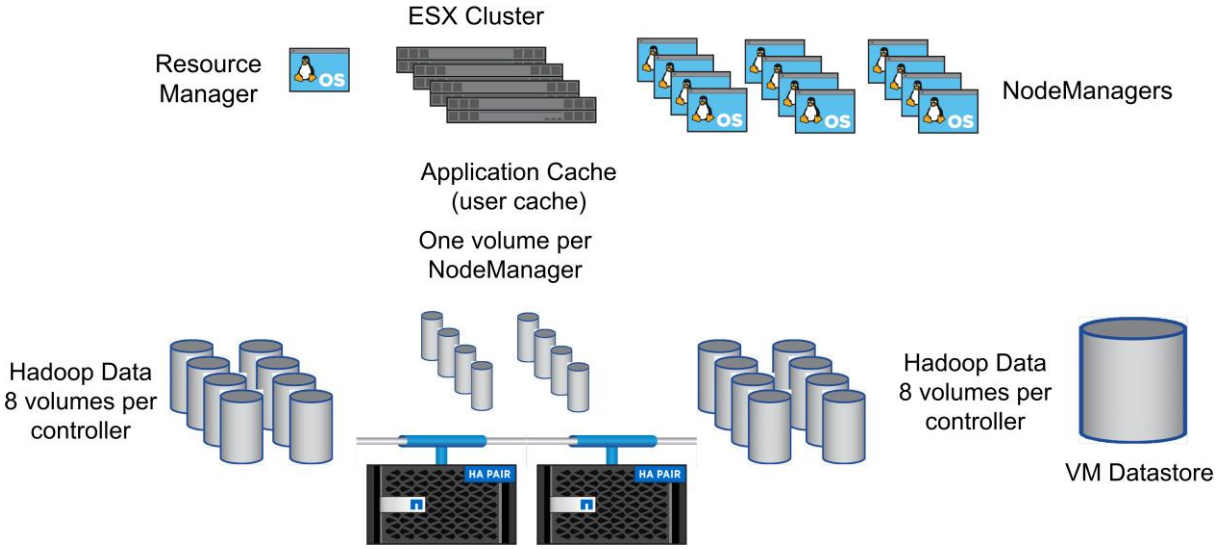


Figure 21 shows the performance results from TeraGen, TeraSort, and TeraValidate, which is based on 10 virtual machines (NodeManagers).

Figure 21) Teragen, TeraSort, and TeraValidate results.

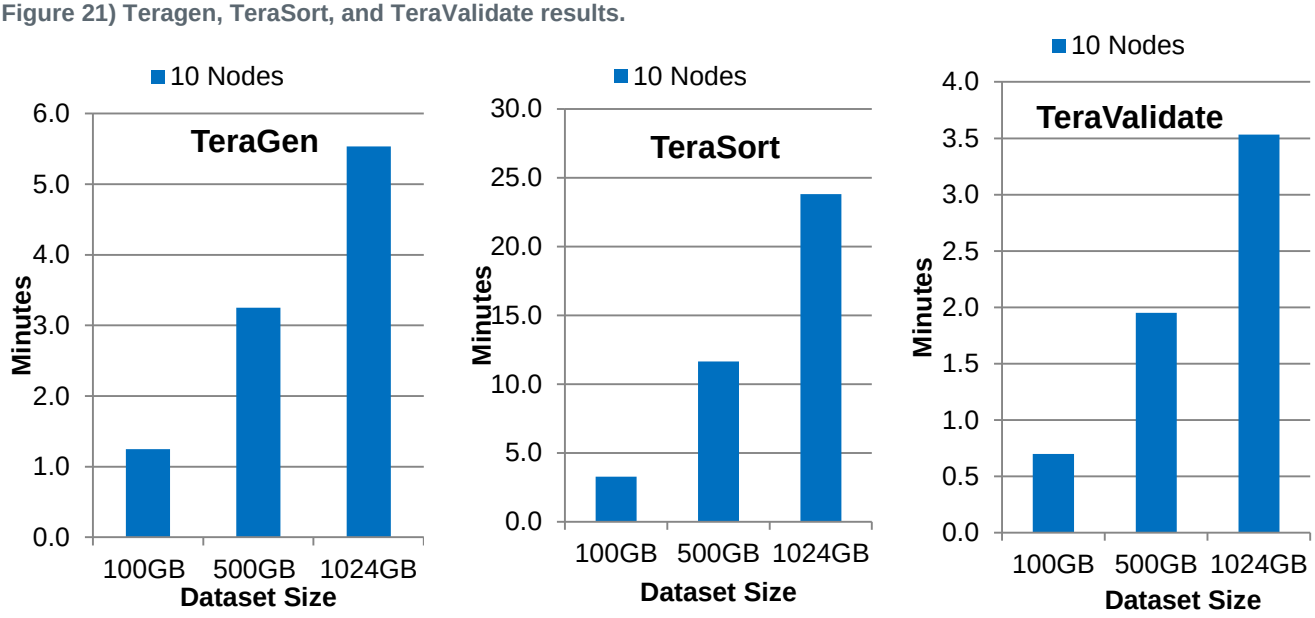


Figure 22 and Figure 23 show the performance results of two-node FAS8080FF for NFS read, write, and latency. Based on the results, the two-node FAS8080AFF storage configuration was underutilized throughout the testing.

Figure 22) Read/write throughput and latency during TeraGen operation.

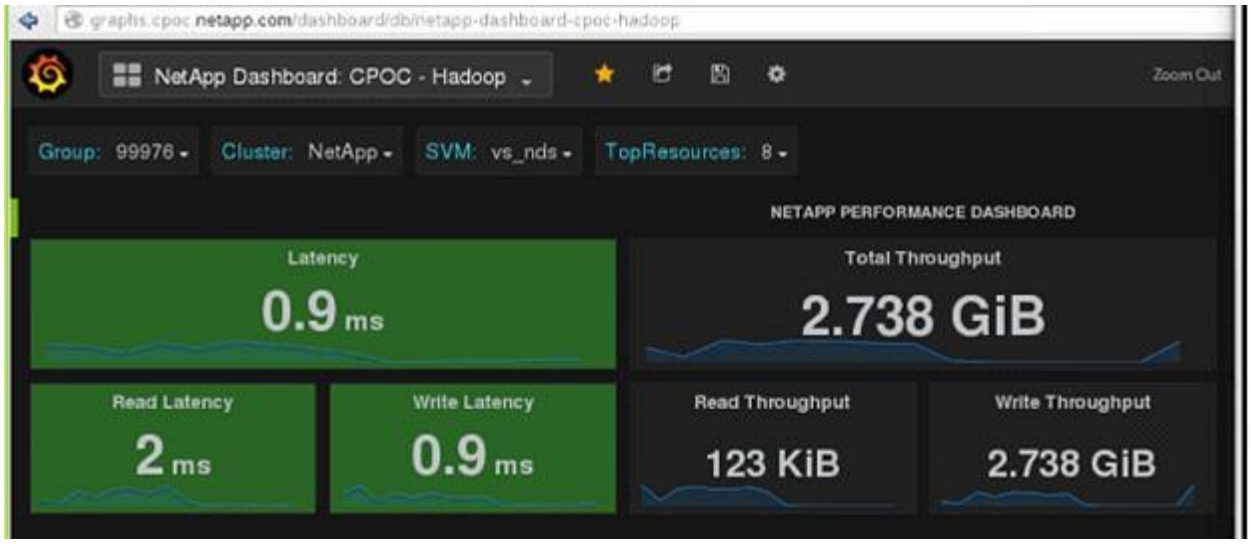
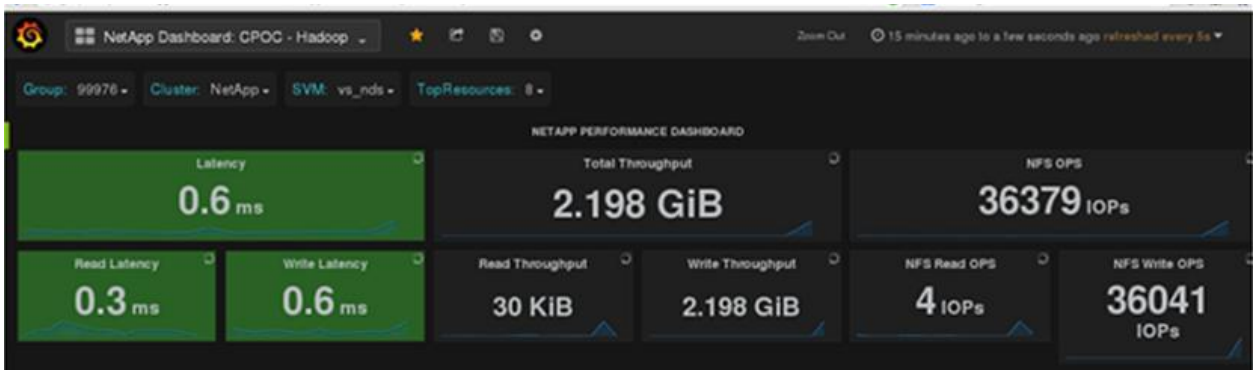


Figure 23) NFS read/write during TeraGen operation.



In the `nfs-mapping.json` file, we used 16 volumes, 8 volumes from each controller. For better network bandwidth, we used 2 networks for 8 volumes from a single controller.

/etc/name-mapping/nfs-mapping.json

```
[root@rm bin]# cat /etc/name-mapping/nfs-mapping.json
{
    "spaces": [
        {
            "name": "vs_nds_nfs",
            "uri": "nfs://192.168.0.111:2049/",
            "options": {
                "nfsExportPath": "/",
                "nfsReadSizeBits": 20,
                "nfsWriteSizeBits": 20,
                "nfsSplitSizeBits": 29,
                "nfsAuthScheme": "AUTH_SYS",
                "nfsUserConfigFile":"/etc/name-mapping/users.json",
                "nfsGroupConfigFile":"/etc/name-mapping/groups.json",
                "nfsUsername": "root",
                "nfsGroupname": "root",
                "nfsUid": 0,
                "nfsGid": 0,
                "nfsPort": 2049,
                "nfsMountPort": -1,
                "nfsRpcbindPort": 111
```

```
            },
            "endpoints": [
                {
                    "host": "nfs://192.168.0.111:2049/",
                    "path": "/nfs_aggr1_vol1/"
                },
                {
                    "host": "nfs://192.168.1.111:2049/",
                    "path": "/nfs_aggr1_vol2/"
                },
                {
                    "host": "nfs://192.168.0.112:2049/",
                    "path": "/nfs_aggr1_vol3/"
                },
                {
                    "host": "nfs://192.168.1.112:2049/",
                    "path": "/nfs_aggr1_vol4/"
                },
                {
                    "host": "nfs://192.168.0.113:2049/",
                    "path": "/nfs_aggr1_vol5/"
                },
                {
                    "host": "nfs://192.168.1.113:2049/",
                    "path": "/nfs_aggr1_vol6/"
                },
                {
                    "host": "nfs://192.168.0.112:2049/",
                    "path": "/nfs_aggr1_vol7/"
                },
                {
                    "host": "nfs://192.168.1.112:2049/",
                    "path": "/nfs_aggr1_vol8/"
                },
                {
                    "host": "nfs://192.168.0.121:2049/",
                    "path": "/nfs_aggr2_vol1/"
                },
                {
                    "host": "nfs://192.168.1.121:2049/",
                    "path": "/nfs_aggr2_vol2/"
                },
                {
                    "host": "nfs://192.168.0.122:2049/",
                    "path": "/nfs_aggr2_vol3/"
                },
                {
                    "host": "nfs://192.168.1.122:2049/",
                    "path": "/nfs_aggr2_vol4/"
                },
                {
                    "host": "nfs://192.168.0.122:2049/",
                    "path": "/nfs_aggr2_vol5/"
                },
                {
                    "host": "nfs://192.168.1.122:2049/",
                    "path": "/nfs_aggr2_vol6/"
                },
                {
                    "host": "nfs://192.168.0.123:2049/",
                    "path": "/nfs_aggr2_vol7/"
                },
                {
                    "host": "nfs://192.168.1.123:2049/",
                    "path": "/nfs_aggr2_vol8/"
                }
            ]
        }
    ]
}
```

```
[root@rm bin]#
```

```
[root@rm bin]# cat /etc/name-mapping/groups.json
{
  "groupnames":[
    {
      "groupName":"root",
      "groupID":"0"
    },
    {
      "groupName":"hadoop",
      "groupID":"490"
    },
    {
      "groupName":"mapred",
      "groupID":"483"
    },
    {
      "groupName":"nobody",
      "groupID":"99"
    },
    {
      "groupName":"yarn",
      "groupID":"481"
    },
    {
      "groupName":"hdfs",
      "groupID":"488"
    }
  ]
}
```

```
 [root@rm bin]# cat /etc/name-mapping/users.json
{
  "usernames":[
    {
      "userName":"root",
      "userID":"0"
    },
    {
      "userName":"mapred",
      "userID":"488"
    },
    {
      "userName":"hdfs",
      "userID":"493"
    },
    {
      "userName":"nobody",
      "userID":"99"
    },
    {
      "userName":"yarn",
      "userID":"486"
    }
  ]
}
[root@rm bin]#
```

core-site.xml

In addition to the default settings, the parameters in Table 3 were updated in `core-site.xml`.

Table 3) core-site.xml parameters.

Parameter	Setting Value	Description
fs.nfs.prefetch	true	Whether to do prefetch for InputStream
fs.defaultFS	nfs://192.168.0.111:2049/	Name of the default file system specified as a URL
fs.AbstractFileSystem.nfs.impl	org.apache.hadoop.fs.nfs.NFSv3AbstractFilesystem	Allows Hadoop to find the connector for NFS
fs.nfs.impl	org.apache.hadoop.fs.nfs.NFSv3FileSystem	Allows Hadoop to find the NFS connector
fs.nfs.configuration	/etc/Hadoop/conf/nfs-mapping.json	It defines the cluster architecture for NetApp FAS NFS Connector

yarn-site.xml

In addition to the default settings, the parameters in Table 4 were updated in `yarn-site.xml`.

Table 4) yarn-site.xml parameters.

Parameter	Setting Value
yarn.resourcemanager.address	rm.cpoc.local:8032
yarn.resourcemanager.admin.address	rm.cpoc.local:8033
yarn.resourcemanager.scheduler.address	rm.cpoc.local:8030
yarn.resourcemanager.resource-tracker.address	rm.cpoc.local:8031
yarn.resourcemanager.webapp.address	rm.cpoc.local:8088
yarn.resourcemanager.webapp.https.address	rm.cpoc.local:8090
yarn.resourcemanager.client.thread-count	50
yarn.resourcemanager.admin.client.thread-count	1
yarn.scheduler.minimum-allocation-mb	1024
yarn.scheduler.maximum-allocation-mb	49152
yarn.scheduler.minimum-allocation-vcores	0
yarn.scheduler.increment-allocation-vcores	1
yarn.scheduler.maximum-allocation-vcores	8
yarn.resourcemanager.amliveliness-monitor.interval-ms	1000
yarn.resourcemanager.scheduler.class	org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler

mapred-site.xml

In addition to the default settings, the parameters in Table 5 were updated in `mapred-site.xml`.

Table 5) mapred-site.xml parameters.

Parameter	Setting Value
mapreduce.output.fileoutputformat.compress	False
mapreduce.output.fileoutputformat.compress.type	BLOCK
mapreduce.task.io.sort.factor	100
mapreduce.map.sort.spill.percent	0.8
mapreduce.reduce.shuffle.parallelcopies	10
mapreduce.task.io.sort.mb	1024
yarn.app.mapreduce.am.resource.mb	6144
yarn.app.mapreduce.am.resource.cpu-vcores	1
mapreduce.jobhistory.address	rm.cpoc.local:10020
mapreduce.jobhistory.webapp.address	rm.cpoc.local:19888
mapreduce.jobhistory.webapp.https.address	rm.cpoc.local:19890
mapreduce.jobhistory.admin.address	rm.cpoc.local:10033
yarn.app.mapreduce.am.command-opts	-Djava.net.preferIPv4Stack=true -Xmx3221225472
mapreduce.map.java.opts	-Djava.net.preferIPv4Stack=true -Xmx3221225472
mapreduce.reduce.java.opts	-Djava.net.preferIPv4Stack=true -Xmx6442450944
mapreduce.map.memory.mb	4096
mapreduce.map.cpu.vcores	1

7.4 TeraGen and TeraSort Validation with FAS8060 and E5660

This section describes the TeraGen and TeraSort test results of FAS NFS connector for Hadoop (FAS) and NetApp E-Series solutions for Hadoop.

Table 6) Hardware used for E-Series and FAS testing.

Component	E-Series (NSH)	NetApp FAS NFS Connector for Hadoop
Servers	• 4 servers - Intel Xeon E5-2400 series processors • 16-core CPU per server • 48GB RAM per server • Red Hat Enterprise Linux Server 6.5 (x86_64) • Red Hat Enterprise Linux Server 6.5 (x86_64)	
	• 1 x LSI 9207-8e SAS HBA per server • 1 x 10GbE per server	• 1 x 10GbE per server
Storage	NetApp E-Series 5660 storage array • 6TB NL SAS drives (7200 RPM) • RAID 5 with 48 data drives; 8 parity	• FAS8060 HA pair • 900GB SAS drives (10K RPM) • RAID DP with 48 data drives, 8 parity

Component	E-Series (NSH)	NetApp FAS NFS Connector for Hadoop
	drives • 4 x 8Gb/s SAS connectivity (2 SAS connection from each controller to 2 data nodes)	drives, • 8 x 10GbE with 8 LIFs (4 from each controller)
Hadoop distribution	Cloudera Distribution for Hadoop Cloudera Enterprise Core and Manager 5.4.2	
Tools used for Testing	TeraGen and TeraSort with dataset size from 93GB to 1TB	
Hadoop replication factor	2	Replication not required with FAS NFS Connector for Hadoop
Maps and reduce	maps=83, reduce=41	

Note: For these tests, we did not attempt to maximize the utilization of either E5660 or FAS8060.The goal was to observe the TeraGen and TeraSort results for processing datasets up to 1TB.

Based on the test results, we can conclude that both configurations perform optimally for Hadoop workloads, with FAS NFS Connector for Hadoop targeted for in-place analytics use case, and the E-Series solution targeted for new scalable Hadoop deployments with HDFS.

Figure 24) TeraGen dataset creation report.

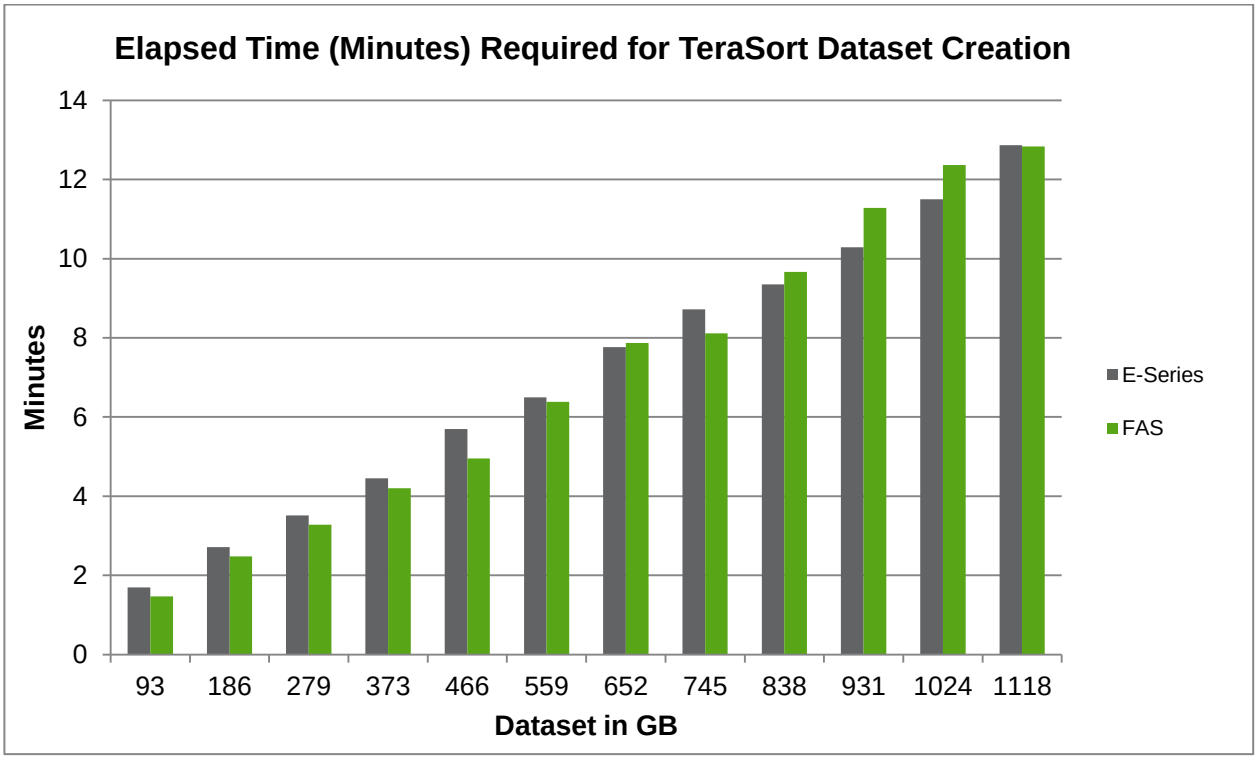
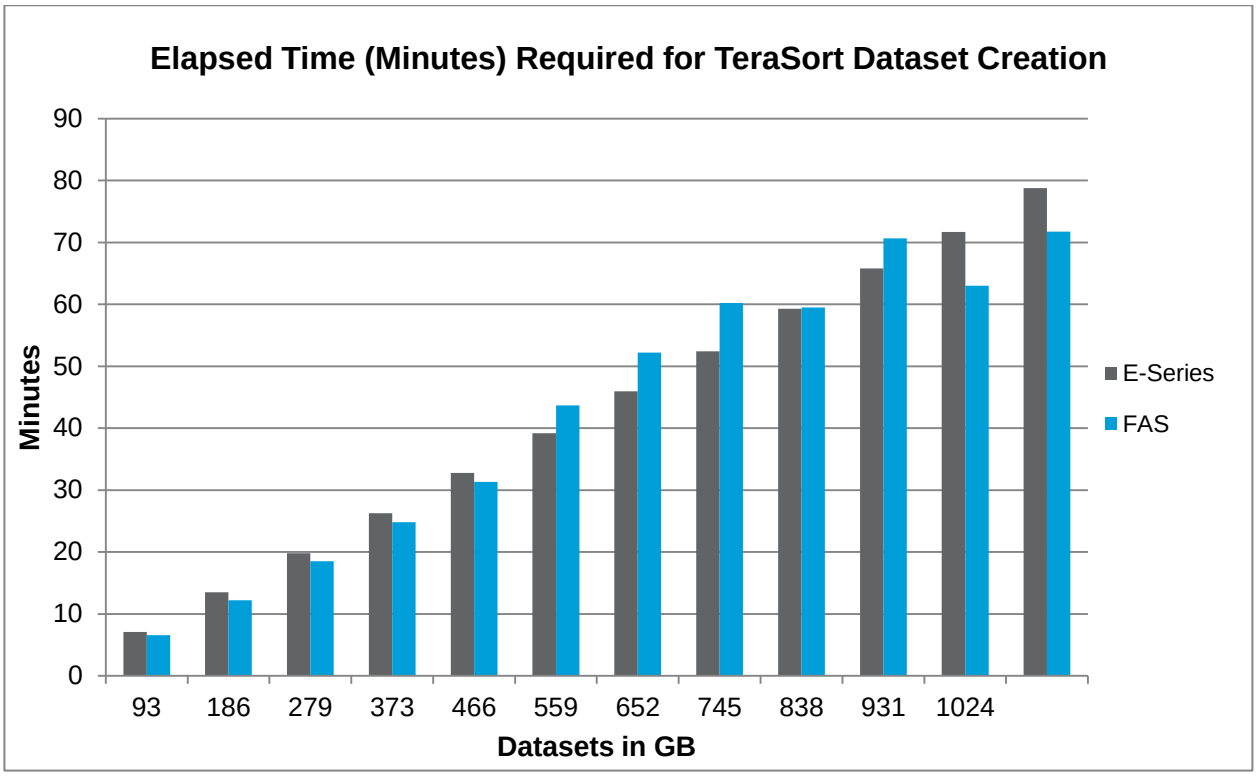


Figure 25) TeraSort dataset creation report.



8 Conclusion

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

References

The following references were used in this report:

- NetApp Solutions for Hadoop Reference Architecture
  www.netapp.com/us/media/wp-7196.pdf
- NetApp Solutions for Hadoop Reference Architecture: Cloudera
  http://www.netapp.com/us/media/wp-7217.pdf
- MixApart: Decoupled Analytics for Shared Storage Systems
  https://www.usenix.org/system/files/conference/fast13/fast13-final58.pdf
- Apache Hadoop YARN—Concepts & Applications
  http://hortonworks.com/blog/apache-hadoop-yarn-concepts-and-applications/

Acknowledgements

- Jeffrey Heller, senior director, Advanced Technology Group
- Beth Schwartz, engineering program manager, Third-Party Process
- Scott Dawkins, vice president, Advanced Technology Group

- Iyer Venkatesan, solutions marketing, Big Data Analytics
- AJ Mahajan, senior director, Product Management
- Chris Lemmons, director, DSG Technical Marketing for Business Processing Workloads
- Abhinav Joshi, senior product manager, Big Data
- John Elliott, senior technical marketing engineer, Oracle database
- Pranoop Erasani, technical director, NFS
- Sunitha Rao, senior manager, Product Management
- PradeepNayak UdupiKadbet, software engineer
- Paramita Das, senior manager
- Randy Cubrilovic, CPOC architect

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

TR-4382-1015