White Paper

# NetApp Solutions for Hadoop

## Reference Architecture

Gus Horn, Iyer Venkatesan, and Karthikeyan Nagalingam, NetApp
January 2015 | WP-7196


## Abstract

Today's businesses must store an unprecedented volume of data and control and analyze the speed and complexity of the data that they are capturing. Apache Hadoop has gained in popularity due to its ability to handle large and diverse data types. However, enterprises face several technical challenges when deploying Hadoop, specifically in the areas of cluster availability, operations, and scaling. NetApp has developed reference architectures with leading Hadoop vendors to deliver a solution that overcomes some of these challenges so that business can ingest, store, and manage big data with greater reliability and scalability and with less time spent on operations and maintenance.

This white paper discusses a flexible, validated, enterprise-class Hadoop architecture that is based on NetApp® storage in a managed or external direct-attached storage (DAS) configuration with any compatible distribution of Hadoop. The paper includes guidelines and best practices on how to build and configure modular Hadoop systems, including recommendations on servers, file systems, and networking topology. This information will enable groups or organizations to gain insights for competitive advantage, in-depth customer knowledge, and many other business benefits.

TABLE OF CONTENTS



## LIST OF TABLES



## LIST OF FIGURES



# 1 Introduction

Big data is having a significant impact on businesses that need to collect, store, manage, and analyze large amounts of complex data. Apache Hadoop and its growing ecosystem of products have enabled businesses to handle large volumes and diverse types of data so that they can start gaining valuable insights from this data. However, Hadoop poses some challenges for enterprise data centers regarding operations, availability, and implementation.

Validated reference architectures for Hadoop are an effective way to overcome some of the challenges that arise from implementing Hadoop. NetApp Solutions for Hadoop validate NetApp storage systems with the leading Hadoop distributions in a flexible and open environment and include recommendations for servers and networking so that a fully configured cluster can be built.

As groups begin to move Hadoop pilots or proofs of concept into production, they typically expect a certain degree of scalability, manageability, and availability as well as consistent, deterministic performance. The NetApp reference architecture is designed for high-availability (HA) scale-out and provides reliable SLAs and deterministic performance. In short, it is an open enterprise-grade scale-out solution.

## 1.1 Benefits of NetApp Solutions for Hadoop

NetApp Solutions for Hadoop offer the following key benefits:

- Enterprise-class implementation of Hadoop with lower cluster downtime, higher data availability, and linear scalability
- Fast time to value with validated, presized configurations that are ready to deploy, thereby reducing the risks that are traditionally associated with Hadoop
- High storage efficiency and lower operational expenses for Hadoop
- An open solution built with NetApp storage and best-in-class products from partners, providing choice without vendor lock-in or performance compromise

NetApp has partnered with major Hadoop distribution providers and other analytics platform vendors to deliver reliable and efficient storage for Hadoop solutions. These validated solutions, when implemented in a Hadoop cluster, provide the capabilities for ingesting, storing, and managing large datasets with high efficiency, reliability, and performance.

## 1.2 Brief Hadoop Overview

**Note:** Readers familiar with Hadoop can skip to section 1.3, “Basic Hadoop 1.0 Architecture,” or section 2, “Hadoop Challenges.”

Hadoop is an open-source analytical framework and an ecosystem of products and technologies that contain, among other things, databases, scripting languages, and management tools. The two main components of Hadoop are MapReduce and the Hadoop Distributed File System (HDFS). MapReduce is a framework for parallel processing and HDFS is a distributed file system that provides petabyte-size storage and data management.

Hadoop is designed to run on a large number of parallel machines. When you load all of your organization’s data into Hadoop, the software makes three copies of the data, breaks that data into pieces, and then spreads it across different servers that are set up to run in parallel. There is no one place where you can go to manage all of your data; Hadoop NameNode, the repository for all HDFS metadata, keeps track of where the data resides. And because there are multiple copy stores, data stored on a server that goes offline or dies can be automatically replicated from a known good copy.

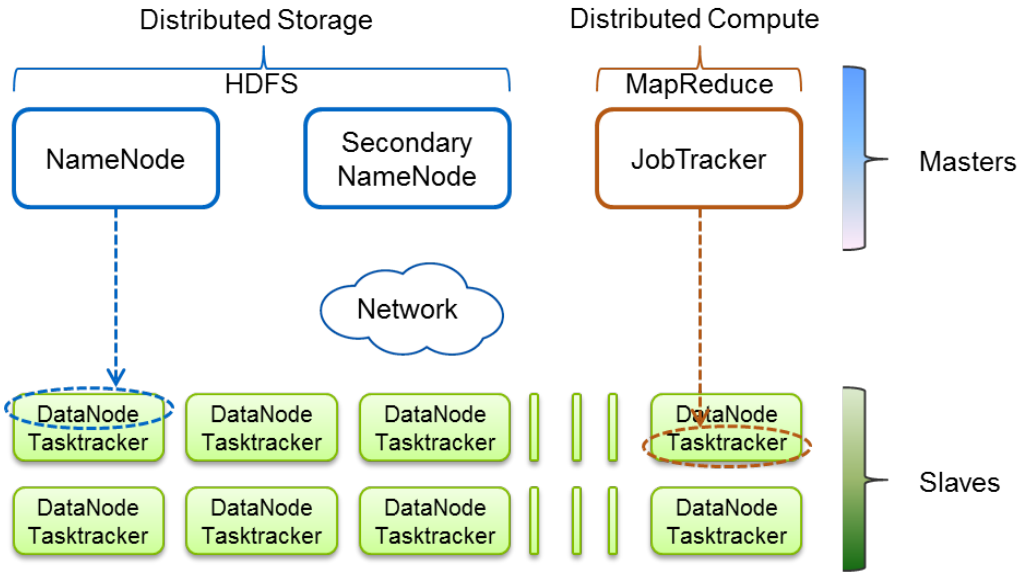
## 1.3 Basic Hadoop 1.0 Architecture

The basic Hadoop 1.0 architecture is built around commodity servers, internal direct-attached storage (DAS), and networking. A conventional Hadoop cluster consists of the following basic components:

- **NameNode or active NameNode**. This node manages the HDFS namespace. It runs all of the services needed to manage the HDFS data storage and MapReduce tasks in the cluster, including data and task distribution and tracking.
- **Checkpoint node**. This node is a secondary NameNode that manages the on-disk representation of the NameNode metadata. In active or standby HA mode, this node runs a second NameNode process, usually called the standby NameNode process. In quorum-based HA mode, this node runs the second journal node.
- **JobTracker node**. This node manages all of the jobs submitted to the Hadoop cluster and facilitates job and task scheduling.
- **DataNodes**. DataNodes are slave nodes that provide both DataNode functionality and TaskTracker functionality. These nodes perform all of the real processing work done by the cluster. As DataNodes, they store all of the data blocks that make up the file system, and they serve I/O requests. As TaskTrackers, they perform the job tasks assigned to them by the JobTracker.

Figure 1 shows the basic architecture of Hadoop 1.0.

**Figure 1) Basic Hadoop 1.0 architecture.**



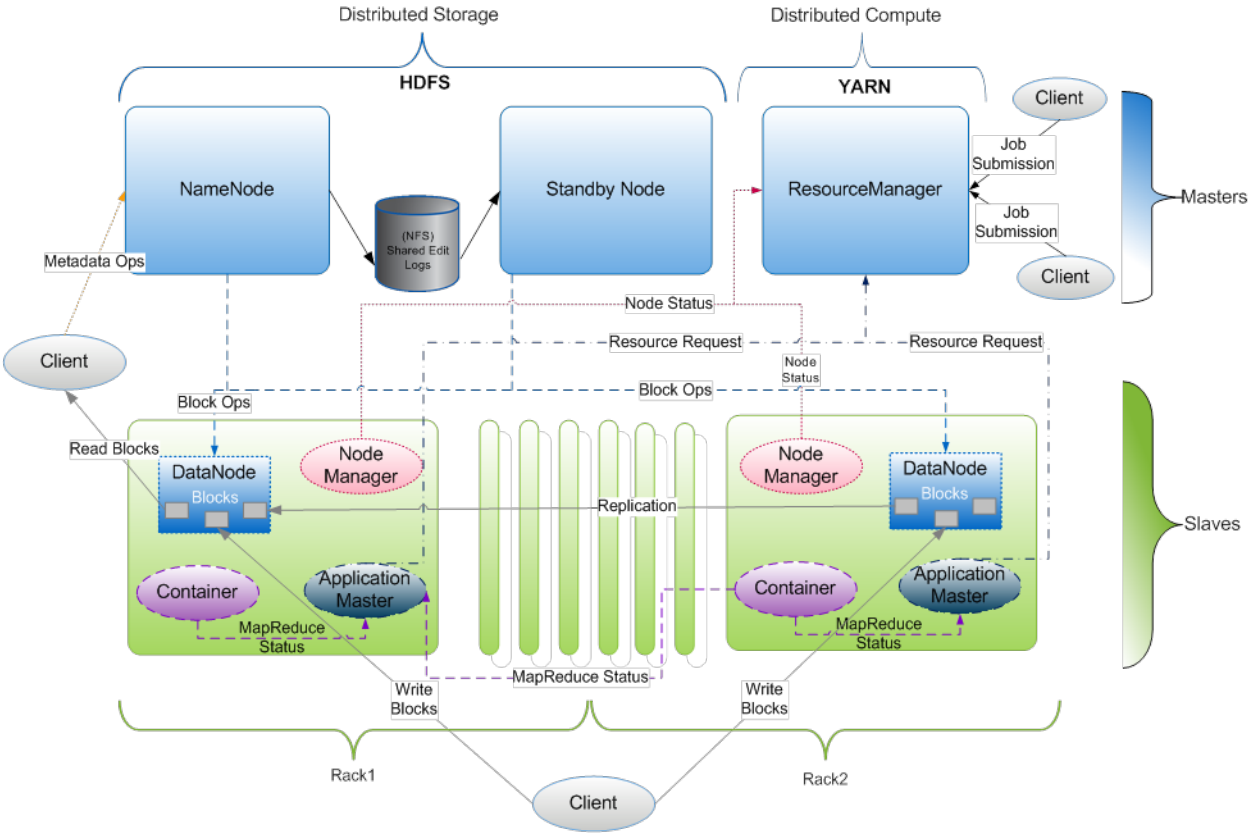

## 1.4 Basic Hadoop 2.0 Architecture

The Hadoop 2.0 architecture has YARN (Yet Another Resource Negotiator), which splits JobTracker functionalities such as resource management and job scheduling into two services, the ResourceManager and the ApplicationMaster. The ResourceManager and the per-node slave NodeManager form the data-computation framework. The ResourceManager manages the resources for all applications in the cluster. The ApplicationMaster negotiates for resources from the ResourceManager and works with the NodeManager to run and monitor tasks or jobs.

Figure 2 shows the basic architecture of Hadoop 2.0.

**Figure 2) Basic Hadoop 2.0 architecture.**




The ResourceManager includes a scheduler and the ApplicationManager. The scheduler allocates resources to applications based on capacities, queues, and other criteria, but it does not restart failed tasks caused by application or hardware failures. It provides resources to applications by using containers; a container consists of memory, CPU, disks, network, and so on. The scheduler has a policy plug-in that partitions the cluster resources between various queues and applications. The ResourceManager can use two schedulers, the CapacityScheduler or the FairScheduler.

The ApplicationManager accepts job submissions, negotiating the first container for executing the application-specific ApplicationMaster tasks and restarting the ApplicationMaster on failure.

NodeManager is a per-machine framework agent that is responsible for container management, for monitoring the resource usage (CPU, memory, disks, and network) of individual containers, and for reporting this data to the ResourceManager and the scheduler.

# 2 Hadoop Challenges

Hadoop was incubated and developed at Internet companies such as Google and Yahoo, and their goal was to analyze billions of files and petabytes of data by using thousands of low-cost servers. They achieved this goal in an environment in which development and operations coexisted.

As Hadoop has gained traction in enterprise data centers, operational requirements have changed over time: a cluster with a few hundred nodes suffices for most businesses; enterprise customers do not have active developers working on Hadoop; enterprises are driven by very strict SLAs. Therefore, any solution put into such environments must meet enterprise-level operational requirements and data center standards and SLAs. Enterprises face some common challenges when implementing Hadoop:

- Operational challenges:
  - Limited flexibility when scaling; inability to independently add servers or storage (that is, when servers are added, storage must be added as well)
  - Lack of high availability; jobs are interrupted during failures, and recovering these jobs from drive failures can be time consuming
  - Difficulty meeting enterprise-level SLAs
- Efficiency challenges:
  - Three copies of data (also called triple mirroring)
  - Network congestion and overdrain of system resources
  - Poor job efficiency due to storage fragmentation over time
- Management challenges:
  - Requirement for skilled resources
  - Lack of flexibility for changes to the architecture
  - Growth of the data center footprint over time

## 2.1 HDFS Replication Trade-Offs

Although HDFS is distributed, feature rich, and flexible, there are both throughput and operational costs associated with server-based replication for data ingest, redistribution of data following the recovery of a failed DataNode, and capacity utilization.

A standard Hadoop installation creates at least three copies of every piece of stored data. Triple replication imposes the following costs:

- For each usable terabyte of data, 3TB of raw capacity are required.
- Server-based triple replication creates a significant load on the servers themselves. At the beginning of an ingest task, before any data blocks are actually written to a DataNode, the NameNode creates a list of DataNodes to which all three of the first block replicas will be written, forming a pipeline and allocating blocks to those nodes. The first data block is then written to the first DataNode. That DataNode forwards the same block to the second node in the pipeline. Finally, the second DataNode sends the block to the third DataNode, and the write-to-storage process follows the same I/O path as the former writes. This process results in a significant load on server resources.
- Server-based triple replication also creates a significant load on the network. For ingest, three network trips are required to accomplish a single block write. All writes compete for and consume the same network, CPU, memory, and storage resources allocated to process the Hadoop analysis tasks.

Anything done to reduce the consumption of server and network resources during the loading of data increases cluster efficiency. The NetApp solution can provide better reliability with a one-third reduction in replication; that is, the NetApp architecture requires only two copies of the data with the NetApp shared-nothing architecture.

## 2.2 When Disk Drives Fail

The most-fault prone components in the Hadoop architecture are the disk drives. It is simply a matter of time before a disk will fail, and the larger the cluster, the more likely it is that a disk failure will occur. If one or more disks fail, all tasks accessing data on those disks fail as well and are reassigned to other DataNodes in the cluster. This operation results in a severe degradation of completion times for running jobs.

In addition, the failed disk must be physically replaced, and the new disk must be mounted, partitioned, formatted, and reloaded with data from nearby data copies before HDFS can make full use of that disk. Repopulating the new disk with data can be accomplished by using the Hadoop balancer, which

redistributes HDFS data across all DataNodes by moving data blocks from overutilized to underutilized DataNodes.

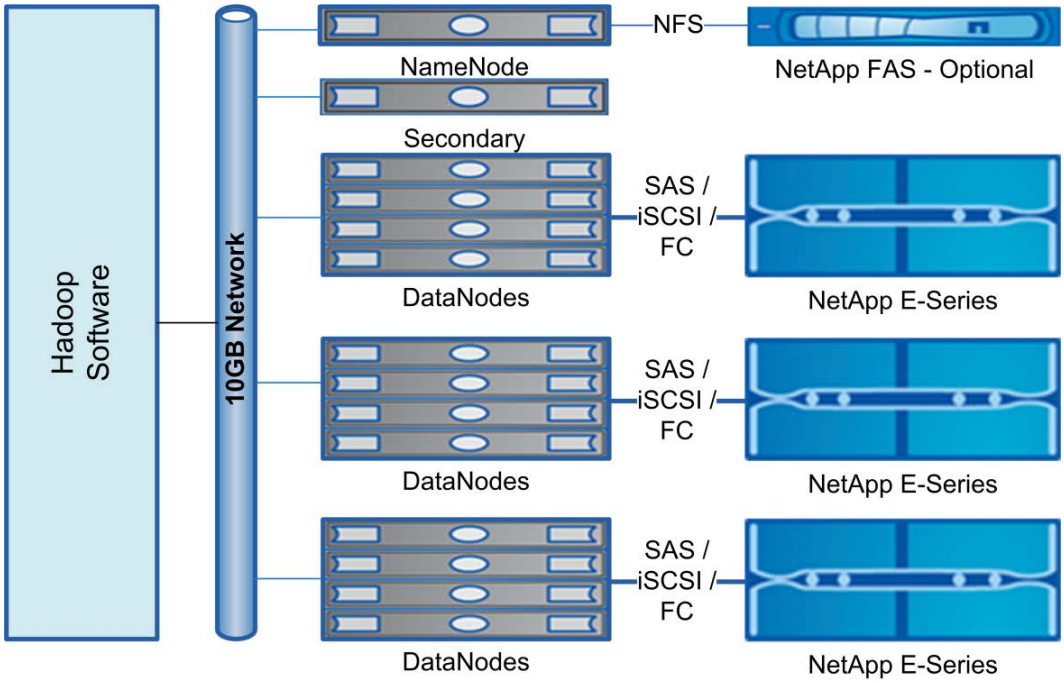
For every full 3TB SATA disk that fails, several hours are required for rebalancing the distribution of blocks following the replacement of that disk, depending on the tolerance for network resource consumption in the cluster. The trade-off is between rebalancing time, network utilization, and disk I/O bandwidth utilization. Higher resource utilization affects MapReduce job performance, especially during data ingest operations. In some cases, rebalancing might also require some manual movement of blocks between disk drives.

# 3 NetApp Solutions for Hadoop Architecture

## 3.1 High-Level Architecture

Figure 3 shows the general architecture of the NetApp Solutions for Hadoop reference design.

**Figure 3) Architecture diagram for NetApp solutions for Hadoop.**




## 3.2 Technical Advantages of NetApp Solutions for Hadoop

NetApp Solutions for Hadoop have the following technical advantages:

- Lower cluster downtime with transparent RAID operations and faster rebuild of failed media
- Less storage required with a Hadoop replication count of two
- Enhanced robustness for HDFS through reliable NFS server-based metadata protection
- Disk drives that can be replaced while the DataNode is running; this operation cannot be performed on internal DAS-based DataNode configurations
- Performance for the next generation of DataNodes (SMP) and networks (10GbE)
- Highly reliable and manageable Hadoop platform
- Server, storage, and networking hardware preconfigured and validated with Apache-compatible Hadoop distributions

## 3.3 Validated Base Configuration

NetApp Solutions for Hadoop have a validated base reference architecture that comprises the hardware, software, and networking components listed in the following subsections. This recommended configuration has been tested, validated, and optimized for running a Hadoop cluster.

### Servers

- Any Intel® Xeon® E5-2400 series processor, up to two 2.4GHz (6-core) or 2.3GHz (8-core) processors
- Memory capacity of 12 DIMMs (up to 192GB), up to 1600Mhz
- 2 x 3.5" HS HDDs
- 2 x 1Gbps Ethernet ports, 10GbE network port
- LSI 9207-8e SAS HBA or compatible iSCSI/FC HBA
- SFF-8088 external mini-SAS cable (for maximum signal integrity, the cable should not exceed 5m)

### Storage

- NetApp E5460 storage array:
  - 1 DE6600 4U chassis with rail kit and power cords
  - 2 E-5460 series controllers, each configured with dual SAS ports and SAS disk drives
  - 60 3TB, 7.2K-RPM, 3.5" NL-SAS disks
  - NetApp SANtricity® operating system (10.84 or later release)
  - SANtricity Storage Manager
  - Turbo feature enabled
- Optional: NetApp FAS2220 NFS storage system

  **Note:** NetApp FAS2220 is recommended if customers require HA protection of the NameNode or need simplified management for the Hadoop cluster.

### Network

- 10GbE nonblocking network switch
- 1GbE network switch

### Software

- Red Hat Enterprise Linux® 6.2 or later
- One of the following Hadoop distributions:
  - Cloudera Distribution 4.1.3 or later; includes Cloudera Manager and Cloudera Enterprise
  - Hortonworks Data Platform HDP 1.3 or later
- SANtricity 10.84 or later storage management software (bundled with E-Series storage arrays)

## 3.4 Other Supported Configurations

Apart from the validated base configuration, this solution offers flexibility for choosing other models and versions of each component. You can select the model of E-Series storage array that best fits your requirements. You might also consider an HA pair of network switches; some users prefer a bonded 1GbE NIC pair instead of a 10GbE network. Multiple options for storage-to-DataNode connectivity are also available, such as direct-connect SAS, FCP, or iSCSI.

Section 6 of this paper, “Other Supported Features, Products, and Protocols,” provides a detailed list of additional supported configurations.

# 4 Solution Architecture Details

The reference design shown in Figure 3 combines two types of NetApp storage controllers. The first type is the E-Series storage array that provides data storage services, with one E-Series array dedicated to four Hadoop DataNodes. The second, optional type of storage is the FAS2220, which provides enterprise-grade protection of the NameNode metadata.

Each DataNode has its own dedicated, nonshared set of disks. This configuration is set up in the exact same way in which Hadoop DataNodes configure capacity when disks are physically collocated in the chassis, as in internal JBOD (just a bunch of disks) configurations.

In practical terms, Linux is not able to distinguish between a set of LUNs presented by the LSI 1068 controller chip on the HBA and a set of disks presented by either an embedded SAS or SATA controller, because there is no logical difference in LUN presentation. The two types of presentation do differ, however, in performance and reliability.

The set of disks in an embedded controller is configured as two blocks, each one using one of two parity-based protection schemes. Eight volume groups are configured in the E-Series storage, and they are configured so that each DataNode sees only its set of two private blocks. This design is an improved alternative to packaging JBOD DAS media for four DataNodes, offering better performance, reliability, data availability, uptime, and manageability.

Each block is a RAID-protected volume group that contains a single virtual logical volume, which can be exported to a given DataNode as a LUN. This LUN appears as a physical disk in Red Hat Enterprise Linux. It can be partitioned, and a file system can be created and mounted on it to store HDFS blocks.

**Note:** HDFS blocks can be 64MB, 128MB, 256MB, or even 512MB in size.

## 4.1 Network Architecture

The network architecture of a Hadoop cluster is critical. With a default replication count of three, more than two-thirds of all the I/O traffic must traverse the network during ingest, and perhaps a third of all MapReduce I/O operations during processing runs could originate from other nodes. The NetApp E-Series storage modules provide a level of performance that is significantly better than the performance provided by a JBOD SATA solution. The E-Series storage array offers a performance platform based on four 6Gb/sec SAS ports, supported by proven caching and best-in-class hardware RAID.

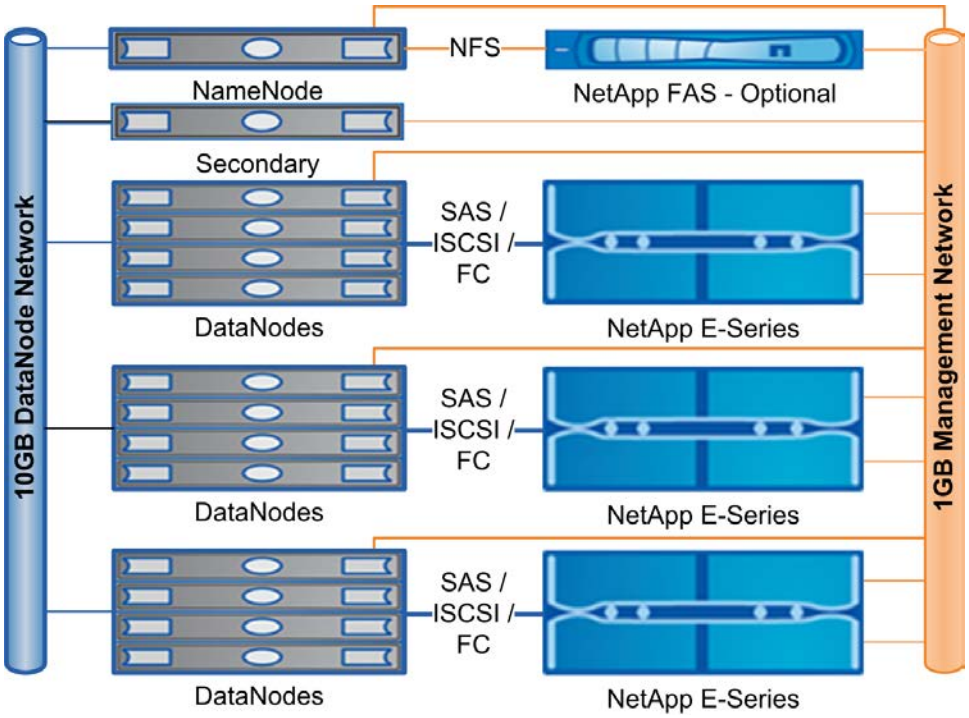
The NetApp solution uses two network backplanes. The core of the solution is the HDFS network, which is a 10GbE network backplane served by an extremely high-performance network 10/40GbE switch solution for a top-of-rack switch. The other network is a 1GbE network, which caters to the administration network and also connects the NameNode, the ResourceManager (in Hadoop 2.0) or the JobTracker (in Hadoop 1.0), and the FAS storage system, if a FAS system is being used for NameNode high availability.

On the 10GbE network, all components are configured for jumbo frames. This requirement also applies to the switch ports to which the components are connected. Any ports on the 10GbE switches that are configured for 1GbE connections (such as the FAS NFS ports and uplinks) should not have jumbo frames enabled.

Figure 4 shows the network architecture of the NetApp Solutions for Hadoop.

Figure 4) Network architecture.




## Recommended Network Topology

NetApp Solutions for Hadoop have the following network topology:

- Data network (10GbE):
  - Primary purpose: Data ingest and movement within cluster
  - Private interconnect between all DataNodes
  - 10GbE nonblocking switch
  - 40GbE uplink between the racks
  - Dedicated nonroutable VLAN

  **Note:** When the iSCSI protocol is used, a dedicated separate VLAN is required for the iSCSI network.

- Management network (1GbE):
  - Purpose: System administration network
  - Publically routable subnet
  - The NameNode and the ResourceManager (in Hadoop 2.0) or the JobTracker (in Hadoop 1.0) also use 1GbE
  - All nodes have a public 1GbE interface for administration and data transfer purposes
  - E-Series storage systems also require two 1GbE ports for management purposes only

## 4.2 Storage Architecture

Each NetApp E-Series storage array is configured as eight independent RAID groups of seven disks that can be set up in a RAID 5 (2 x 6+1) configuration. This setup consumes 56 (8 x 7) disks. The remaining four disks are global hot spares.

If customers deploy all of their files with a replication count of two, then using a single-parity drive over six spindles provides a good balance between storage space utilization and RAID protection. As a result, two forms of protection are constantly available when the E-Series storage arrays are running with files set to a replication count of two.

Customers have the option of selecting two different fan-in ratios depending on their requirements. The more common configuration is one E-Series storage array serving four DataNodes, as shown in Figure 5. For workloads in which lower storage density is enough, a lighter configuration of a single E-Series storage array serving eight DataNodes may be considered. This option is depicted in Figure 6.

**Figure 5) E-Series configured with four hosts per array.**

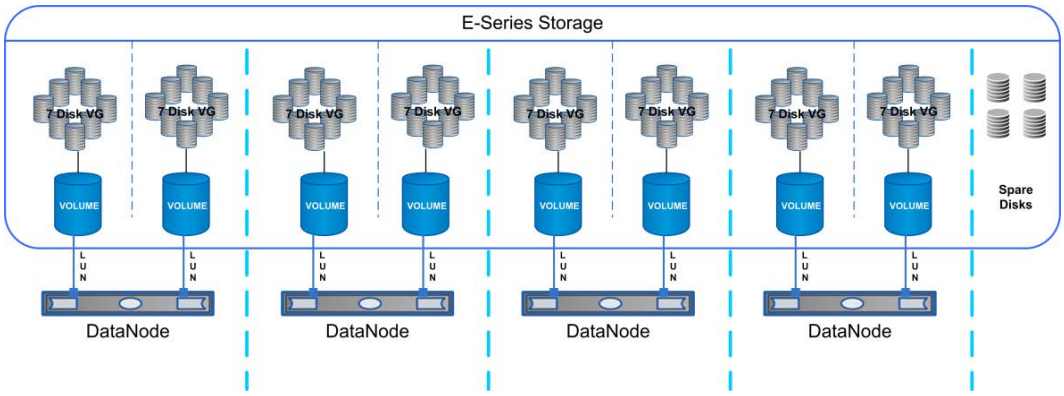
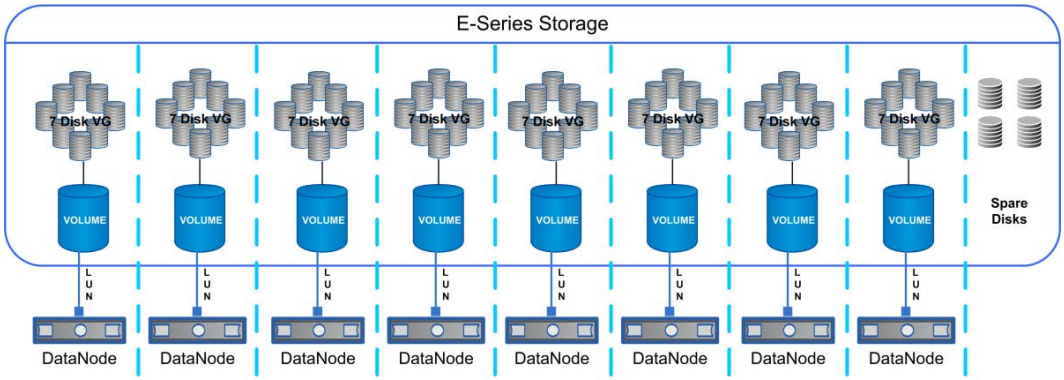



**Figure 6) E-Series configured with eight hosts per array.**




## 4.3 NameNode Metadata Protection

Prior to Hadoop 2.0, the NameNode was a single point of failure in an HDFS cluster. Each cluster had a single NameNode, and if that machine or process became unavailable, the cluster as a whole would be unavailable until the NameNode was either restarted or brought up on a separate machine.

To mitigate the risk of a NameNode failure and the accompanying data loss, two options are available: using backup and restore with Hadoop 1.x or using the HDFS HA feature with Hadoop 2.0 or later. For either option, a FAS system is recommended because it helps manage the Hadoop cluster by storing boot images and binaries, thereby simplifying administration tasks.

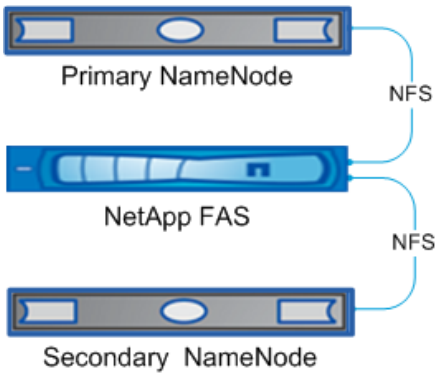
### Option 1: Backup and Restore Method

If you are planning to use the traditional Hadoop method for protection against NameNode failure, then the NetApp solution offers a unique feature for NameNode metadata protection. This protection is provided by introducing a FAS NFS storage system into the architecture.

The NameNode is configured to keep a copy of its HDFS metadata (FSImage) in the NFS-mounted storage. The same NFS-mounted storage is also made available to the secondary NameNode. If a NameNode failure occurs, the critical metadata is still safe on the FAS storage system. At this point, the metadata can be recovered and services restarted on the secondary NameNode to get the HDFS running promptly. Figure 7 illustrates the backup and restore method.

**Figure 7) NameNode backup configuration using NFS storage.**

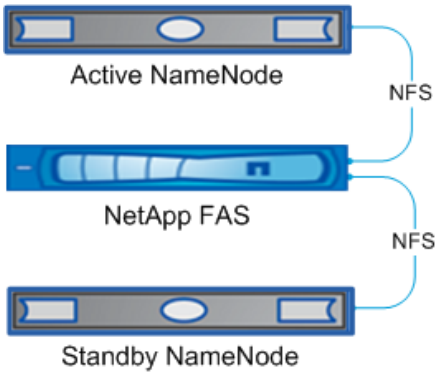



### Option 2: HDFS High Availability

Versions 2.0 and later of HDFS offer an HA NameNode feature that mitigates NameNode failures. The HDFS HA feature addresses the NameNode failure problem by providing the option of running two redundant NameNodes in the same cluster in an active-passive configuration with a hot standby. This configuration allows a fast failover to a new NameNode in the case of a machine crash or a graceful administrator-initiated failover for the purpose of planned maintenance.

For the standby node to keep its state synchronized with the active node, the current implementation of the HA feature requires that the two nodes both have access to a directory on a shared storage device. This access is facilitated by the FAS NFS storage device in the solution. Figure 8 illustrates this option.

**Figure 8) NameNode HA configuration using NFS storage.**




For more details about this configuration, refer to HDFS High Availability.

## 4.4 Rack Awareness Implementation

Implementing rack awareness in a Hadoop solution results in important benefits. This feature allows Hadoop to be configured to know the topology of the network:

- Rack awareness enables Hadoop to maximize network utilization by favoring block transfers within a rack over transfers between racks. The ResourceManager (in Hadoop 2.0) and the JobTracker (in

Hadoop 1.0) are able to optimize MapReduce job performance by assigning tasks to nodes that are closer to their data in terms of network topology.

- By default, during HDFS writes, the NameNode assigns the second and third block replicas to nodes located in a different rack from the first replica. Such assignment provides data protection even against rack failure. However, this protection is possible only if Hadoop has been configured with knowledge of its rack configuration.
- By using the rack awareness feature, a Hadoop engineer can set up the second copy of each block such that it always gets stored on a different E-Series-based DataNode. This feature increases tolerance to storage array and DataNode failures.

Care must be taken to ensure that, for each HDFS block, the corresponding replica is stored on a different E-Series controller. In multirack clusters, this mapping is accomplished by providing the actual rack location of each DataNode. If all DataNodes are located in the same rack, rack mapping must be extended to reflect the E-Series enclosure or E-Series controller that is used for DataNode storage. In this case, the HDFS rack awareness configuration is used to prevent data loss in the unlikely event of storage controller failure.

**Note:** For more information about how to set up the rack awareness feature, refer to "Appendix: iSCSI Validation."

# 5 Key Components of Validated Configuration

Table 1 lists the recommended products for the NetApp validated reference architecture.

**Table 1) Recommended products for the validated reference architecture.**

| Component | Product or Solution | Details |
|---|---|---|
| Storage | NetApp E-Series 5460 storage array<br><br>NetApp FAS2220 (if used for NameNode protection) | • 60 3TB drives for 180TB of capacity<br>• 4 hot spares<br>• 8 volume groups; 8 volumes<br>• RAID 5 volume groups, 7 disks each |
| Servers | Intel Xeon E5-2400 series processors | • Up to two 2.4GHz (6-core) or 2.3GHz (8-core) processors<br>• 12 DIMMs (up to 192GB), up to 1600Mhz<br>• 2 x 3.5" HS HDDs<br>• 2 x 1Gb/sec Ethernet ports, 1 x 10GbE network port<br>• LSI 9207-8e SAS HBA |
| Networking | 10GbE nonblocking network switch<br>1GbE network switch | |
| OS | Red Hat Enterprise Linux Server 6.2 (x86_64) or later | Hadoop typically requires a Linux distribution |
| Hadoop distribution | Cloudera Distribution for Hadoop | • Cloudera Enterprise Core 4.1.3<br>• Cloudera Manager 4.1.3 |
| | Hortonworks Data Platform 1.3 | |

| Component | Product or Solution | Details |
|---|---|---|
| Management software | NetApp SANtricity 10.84 | SANtricity software is bundled with E-Series storage arrays. |

# 6 Other Supported Features, Products, and Protocols

Although NetApp recommends the validated configuration in Table 1, customer requirements might call for different features, products, and protocols. Table 2 lists the alternative options that are supported by NetApp and its partners.

**Table 2) Alternative products supported by NetApp and its partners.**

| Component or Feature | Supported Options | Details |
|---|---|---|
| Storage arrays | E5412<br>E5424<br>E27xx<br>E55xx<br>EF5xx | Any E5400 storage array is supported as well as E2700, E5500, and EF500 series arrays. |
| Disk drives and types | 12, 24, 60<br>1TB, 2TB, 4TB | 12, 24, and 60 disk drives are supported with a capacity of 1TB, 2TB, or 4TB. |
| Protocols | Fibre Channel<br>iSCSI<br>InfiniBand | |
| Other Hadoop distributions | MapR<br>BigInsights<br>Intel | Any Apache-compatible Hadoop distribution is supported. |

# 7 iSCSI Validation

NetApp has conducted a series of proof-of-concept tests using the iSCSI protocol with this reference design. The tests demonstrated that the use of iSCSI with Hadoop resulted in a stable, high-performance configuration that was able to effectively handle a variety of dataset sizes.

The following sections describe these tests and present their results, using TeraGen and TeraSort to demonstrate the scalability of the configuration. Additionally, specific details concerning how the environment was configured and tuned are included in the Appendix of this document (“Appendix: iSCSI Validation”).

Table 3 lists the specific components that were used for the iSCSI tests, including the Hadoop version and host operating system used on the DataNodes.

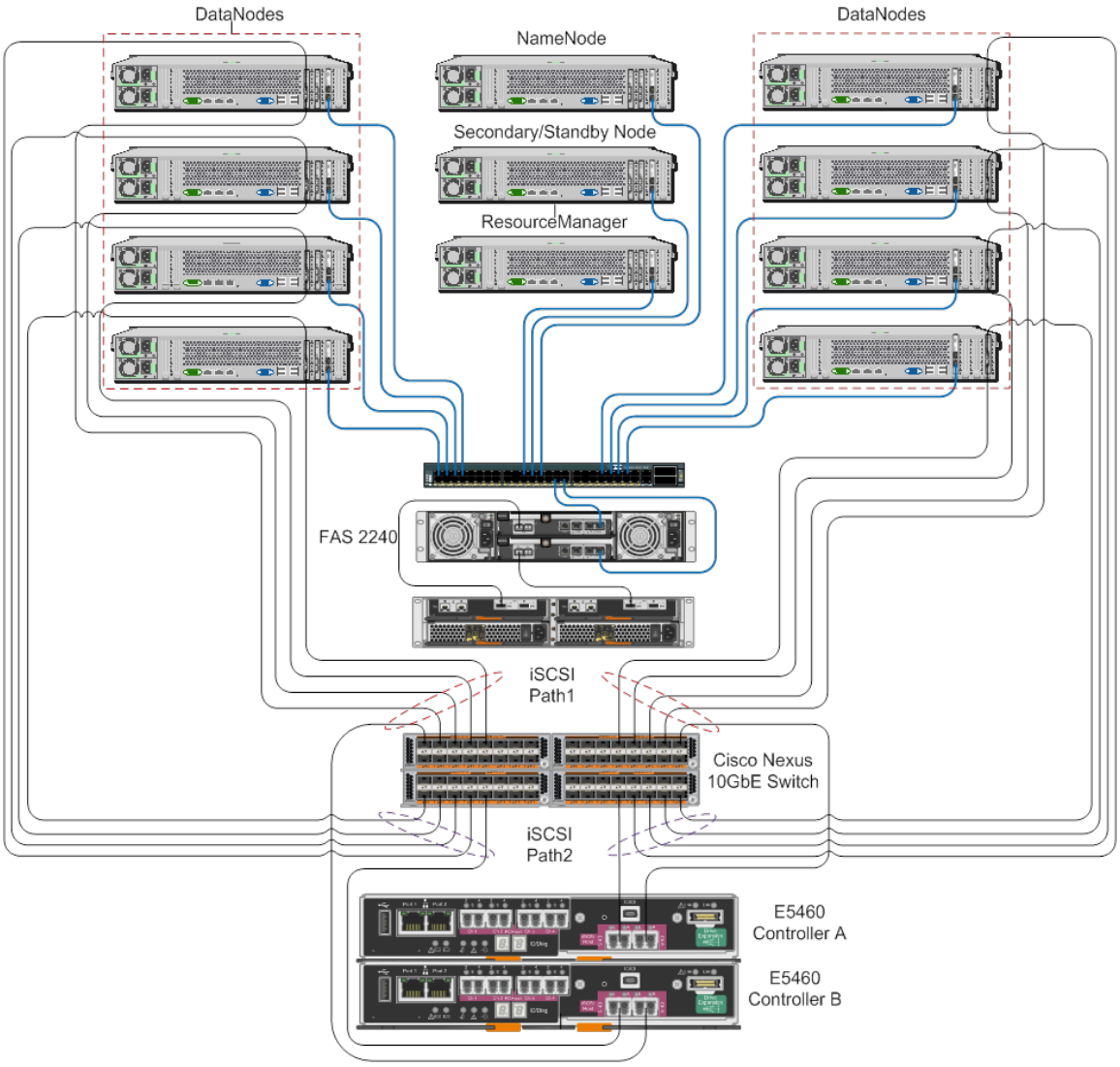
**Table 3) Components used for the iSCSI-validation tests.**

| Component | Product or Solution | Details |
|---|---|---|
| Storage | NetApp E-Series 5460 | • 60 2TB drives for 72.75TB capacity<br>• 4 hot spares<br>• 8 volume groups; 8 volumes<br>• RAID 5 volume groups, 7 disks each |

| Component | Product or Solution | Details |
|---|---|---|
| Servers | Fujitsu PRIMERGY RX300 S6 Gs01 | • Up to two 2.4GHz (4-core) or 2.3GHz (8-core) processors<br>• 6 x 8GB DDR3 memory, 1066 MHz<br>• 2 x 3.5" HS HDDs<br>• 2 x 1Gb/sec Ethernet ports, 1 x 10GbE network port<br>• Direct-access LSI / Symbios Logic SAS1068E PCI-Express Fusion-MPT SAS HBA |
| Networking | 10GbE nonblocking network switch<br>1GbE network switch | Cisco Nexus® 5000 was used for testing (or a compatible 10GbE network switch). |
| OS | Red Hat Enterprise Linux Server 6.5 (x86_64) or later | Hadoop typically requires a Linux distribution. For testing, we used Red Hat Enterprise Linux 6.5. |
| Hadoop distribution | Hortonworks Data Platform 2.1.5 (tested) | Ambari 1.6.1 |
| Management software | NetApp SANtricity 11.10 | SANtricity software is bundled with E-Series storage arrays. |

The iSCSI-validation testing was performed in a Hadoop 2.0 environment that had the architecture described in section 1.4 (“Basic Hadoop 2.0 Architecture”). In accordance with section 4.2 (“Storage Architecture”), we configured the Hadoop environment by using eight DataNodes connected to an E5460 controller that used the iSCSI protocol. Figure 9 shows how the components in the environment were connected. Additionally, we used an optional FAS2240 storage controller to provide failover capability for the NameNode metadata.

**Figure 9) iSCSI cabling diagram.**

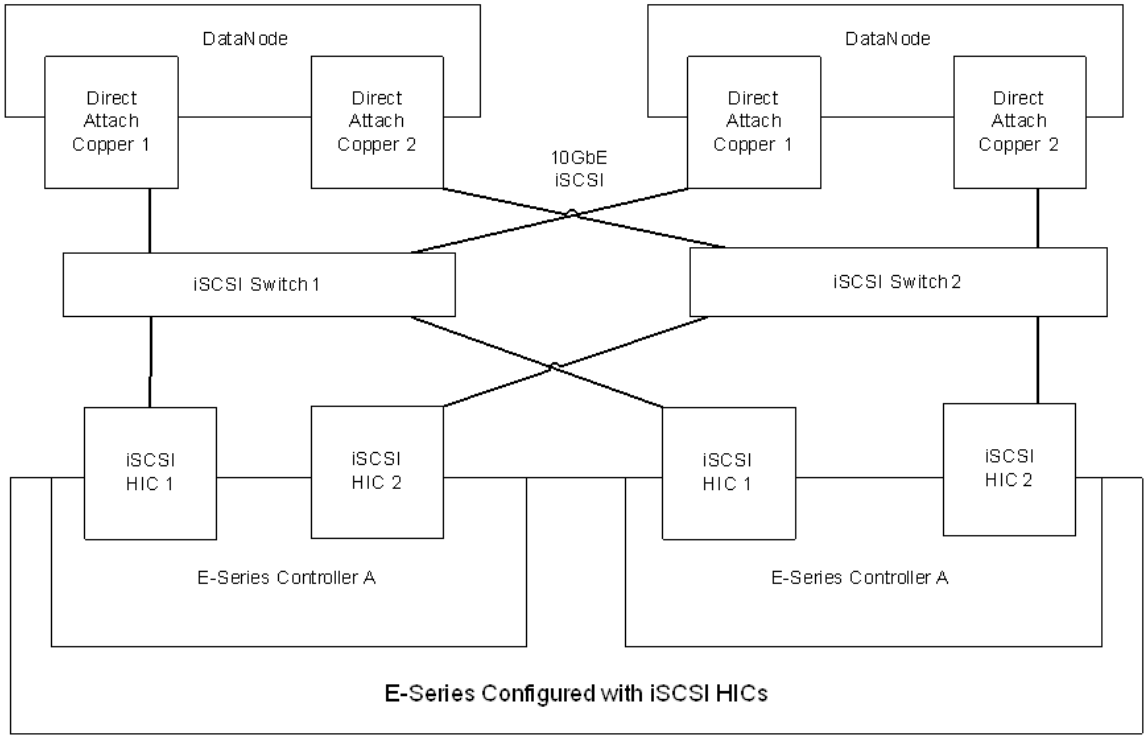



When setting up the iSCSI environment, we configured multipathing in the environment so that multiple active data paths were available from each DataNode to the E5460 storage controllers. Figure 10 shows how this setup was configured. Each DataNode accessed the volume (LUN) from the E-Series controller through four physical paths, with two active-optimized paths through the controller with LUN ownership and two active-nonoptimized paths through an alternative controller in the storage system.

**Note:** For specific information about the multipathing configuration, refer to “Appendix: iSCSI Validation.”

**Figure 10) iSCSI multipathing overview.**




In the test environment, the E-Series storage system consisted of a pair of controllers connected to eight Hadoop DataNode servers through four iSCSI ports. As described in section 4.2, we configured eight RAID 5 volume groups, each comprising seven physical disks. On each volume group, we created a single volume (LUN) with approximately 10.913TB of usable storage.

Each volume was configured with a segment size of 512KB to maximize the use of disk-streaming I/O capabilities. Each volume was mapped as a LUN to the Hadoop DataNode server to which it was assigned.

**Note:** For specific information about the E-Series storage configuration parameters, refer to “Appendix: iSCSI Validation”.

# 8 Results of Validation Testing Using iSCSI

## 8.1 Basic Hadoop Functionality Validation

To validate the stability of the iSCSI configuration, we used the TeraGen tool to generate a moderately sized Hadoop dataset, and then we used the TeraSort tool to conduct a MapR process to verify that the configuration was working as expected. In addition, we conducted tests that injected drive failures on the E5460 storage controllers while an active TeraSort job was executing; we also tested the failure and recovery of the NameNode during an active TeraSort job.

The test cases are described in Table 4 along with the test results. In all cases, the Hadoop environment configured with iSCSI performed as expected.

**Note:** For full details on the test cases and on the testing methodologies, refer to “Appendix: iSCSI Validation.”

Table 4) Test cases.

| Test Case ID | Test Case Name | Pass/Fail (P/F) |
|---|---|---|
| 1 | Initial tuning and full functionality as baseline | P |
| 2 | Full functionality with fault injection | P |
| 3 | NameNode metadata recovery | P |

## 8.2 Performance Validation UsingTeraGen and TeraSort Utilities

In addition to the basic functionality and fault injection testing described in section 8.1, 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 3TB to 10TB and then using TeraSort to conduct a map-reduce function on each dataset, using all eight DataNodes in the cluster. We recorded the elapsed time required to complete the process, and we observed that the duration (in minutes) of TeraGen and TeraSort responses were 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 turning.

Figure 11 shows the elapsed time required to create the different datasets by using TeraGen. Creating the 10TB dataset took over 4 hours and no issues were logged during the TeraGen operations. Additionally, 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.

Figure 11) TeraGen report for iSCSI tests.

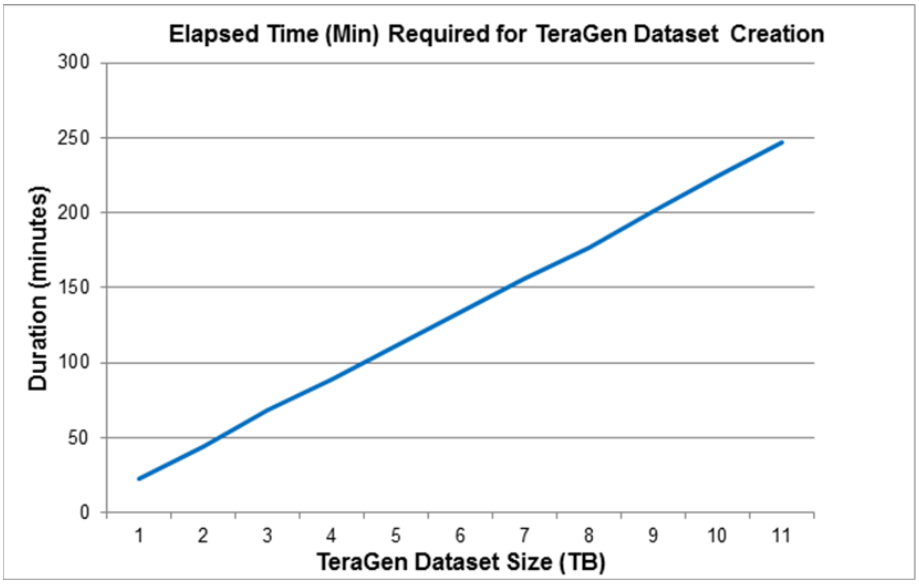
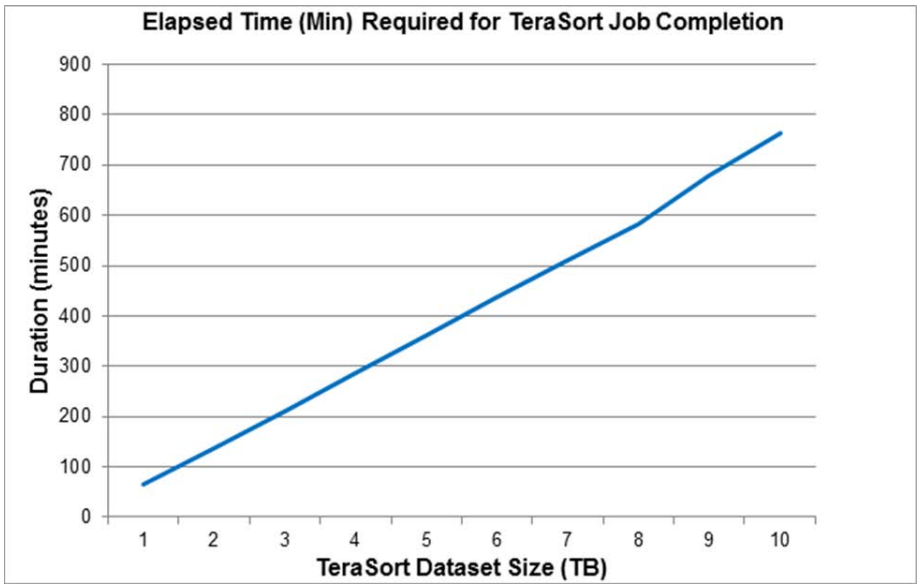



Figure 12 shows the elapsed time required to complete a TeraSort job on each of the increasingly larger datasets described in the preceding paragraphs. The 10TB dataset required over 12 hours to complete the process and no issues were logged during the TeraSort operations. These results demonstrate that the Hadoop cluster maintained comparable processing rates as the size of the dataset increased. They also demonstrate the stability of the overall Hadoop cluster.

Figure 12) TeraSort report for iSCSI tests.




# 9 Conclusion

With big data growing from its roots in Internet companies and becoming more established in data centers, businesses are turning to Hadoop to help them ingest, store, manage, and analyze all the data that they are collecting. Typically, organizations that are deploying or considering Hadoop use traditional server-based storage in an internal DAS configuration. However, this storage configuration can lead to challenges in data centers because internal DAS might not be as reliable for an organization's SLAs, is harder to scale, and is not as flexible.

The NetApp Solutions for Hadoop reference design employs external or managed DAS configurations that provide dedicated storage, with higher scalability and reliability. This design also offers architectural flexibility by decoupling compute nodes from storage to enable the ability to scale one without the need to scale the other. Such flexibility also allows the customer to choose any NetApp E-Series storage system and use most servers and networking technologies with the solution.

The NetApp Solutions for Hadoop reference architecture is validated to help customers get started with Hadoop or deploy Hadoop in production so that they can begin mining their data for insights.

# Appendix: iSCSI Validation

This appendix describes the settings that were applied to the NetApp Solutions for Hadoop reference architecture to test and validate it with the iSCSI protocol.

## Jumbo Frames Configuration

The maximum transmission unit (MTU) size affects network throughput and efficiency. Jumbo frames are network-layer protocol data units (PDUs) that have a size much larger than the typical 1,500-byte Ethernet MTU size. A larger MTU size of 9,000 bytes provides better network throughput and efficiency. The steps in this procedure describe how jumbo frames with an MTU size of 9,000 bytes were configured for traffic from the storage to the host in the iSCSI tests.

To configure jumbo frames, complete the following steps:

1. Open SANtricity Storage Manager.

2. From the Array Management Window (AMW) for the E-Series storage array, click the Setup tab and select Configure iSCSI Host Ports. Make the following selections:
   a. From the iSCSI Port list, select Controller A, HIC 1, Port 1.
   b. From the Configured Ethernet Port Speed list, select 10 Gbps.
   c. Select the Enable IPV4 checkbox.
   d. Configure the IP address, subnet mask, and gateway, if not yet configured.
   e. Select the Enable ICMP Ping Responses checkbox.
   f. Click the Advanced Port Settings button:
   - Enable jumbo frames.
   - For MTU size, select 9,000 bytes/frame.
3. In the DataNodes, configure the Ethernet port (direct attach copper) and restart the network service.

```
[root@stlrx300s6-21 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth3
DEVICE=eth3
HWADDR=00:1B:21:AE:A3:D1
TYPE=Ethernet
UUID=6bb5b352-fecc-48e7-a83d-ca4ed66ab4c6
ONBOOT=yes
NM_CONTROLLED=no
BOOTPROTO=none
IPV6INIT=no
USERCTL=no
IPADDR=192.168.131.109
NETMASK=255.255.255.0
MTU="9000"
[root@stlrx300s6-21 ~]#
```

4. Configure the iSCSI switch.

   **Note:** The iSCSI setup was tested with a Cisco Nexus 5000 switch.

```
STL5596-L3F2# show interface
STL5596-L3F2# show interface brief

STL5596-L3F2# config
Enter configuration commands, one per line.  End with CNTL/Z.
STL5596-L3F2(config)# policy-map type network-qos jumbo
STL5596-L3F2(config-pmap-nq)# class type network-qos class-default
STL5596-L3F2(config-pmap-nq-c)# mtu 9216
STL5596-L3F2(config-pmap-nq-c)# exit
STL5596-L3F2(config-pmap-nq)# exit
STL5596-L3F2(config)# system qos
STL5596-L3F2(config-sys-qos)# service-policy type network-qos jumbo
STL5596-L3F2(config-sys-qos)# exit
STL5596-L3F2(config)# exit
STL5596-L3F2#

STL5596-L3F2# show queuing interface ethernet 2/1
Ethernet2/1 queuing information:
  TX Queuing
    qos-group  sched-type  oper-bandwidth
        0       WRR            100

  RX Queuing
    qos-group 0
    q-size: 470080, HW MTU: 9216 (9216 configured)
    drop-type: drop, xon: 0, xoff: 470080
    Statistics:
        Pkts received over the port             : 117884
        Ucast pkts sent to the cross-bar        : 0
        Mcast pkts sent to the cross-bar        : 117884
        Ucast pkts received from the cross-bar  : 0
        Pkts sent to the port                   : 1158410
        Pkts discarded on ingress               : 0
```

```
        Per-priority-pause status               : Rx (Inactive), Tx (Inactive)

  Total Multicast crossbar statistics:
    Mcast pkts received from the cross-bar      : 1158410
STL5596-L3F2#
```

5. Check the jumbo frames configuration from the DataNode.

```
[root@stlrx300s6-23 ~]# ping 192.168.130.101 -s 8792
PING 192.168.130.101 (192.168.130.101) 8792(8820) bytes of data.
8800 bytes from 192.168.130.101: icmp_seq=1 ttl=64 time=1.74 ms
8800 bytes from 192.168.130.101: icmp_seq=2 ttl=64 time=1.00 ms
8800 bytes from 192.168.130.101: icmp_seq=3 ttl=64 time=1.01 ms
8800 bytes from 192.168.130.101: icmp_seq=4 ttl=64 time=1.00 ms
^C
--- 192.168.130.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3582ms
rtt min/avg/max/mdev = 1.003/1.190/1.745/0.322 ms
[root@stlrx300s6-23 ~]#
```

## iSCSI Network Configuration – DataNodes

To configure the iSCSI network with redundancy, we connected each DataNode port and one port from each controller to separate switches and partitioned each set of host ports and controller ports on separate VLANs. The steps in this procedure describe how the iSCSI network was configured for the DataNodes in the iSCSI tests.

To configure the iSCSI network, complete the following steps:

1. Change `node.session.nr_sessions = 1` in the `/etc/iscsi/iscsid.conf` file, if this is not yet configured.
2. Enable `iscsi` and `iscsid` in runlevels 2, 3, 4, and 5 by running `chkconfig`. For example, run `chkconfig iscsi on` and `chkconfig iscsid on`.
3. Find the iSCSI Qualified Name (IQN) initiator from the `/etc/iscsi/initiatorname.iscsi` file.

```
[root@stlrx300s6-21 ~]# cat /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.1994-05.com.redhat:5f327355672b
[root@stlrx300s6-21 ~]#
```

4. Configure an IP address for the iSCSI initiator port and verify that the subnet is the same as the one used for the iSCSI target ports. Restart the network service by running the `service network restart` command and verify that the service is able to ping the iSCSI targets.

```
[root@stlrx300s6-21 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth3
DEVICE=eth3
HWADDR=00:1B:21:AE:A3:D1
TYPE=Ethernet
UUID=6bb5b352-fecc-48e7-a83d-ca4ed66ab4c6
ONBOOT=yes
NM_CONTROLLED=no
BOOTPROTO=none
IPV6INIT=no
USERCTL=no
IPADDR=192.168.131.109
NETMASK=255.255.255.0
MTU="9000"
[root@stlrx300s6-21 ~]# service network restart
```

5. Create iSCSI interfaces by creating iSCSI iface bindings. For example:
   - For E5400 storage, create two bindings.
   - For E5500 storage, create four bindings.

```
[root@stlrx300s6-21 ~]# iscsiadm -m iface -I iface3 -o new
New interface iface3 added
[root@stlrx300s6-21 ~]# iscsiadm -m iface -I iface3 -o update -n iface.net_ifacename -v eth3
```

```
iface3 updated.
[root@stlrx300s6-21 ~]# iscsiadm -m iface
default tcp,<empty>,<empty>,<empty>,<empty>
iser iser,<empty>,<empty>,<empty>,<empty>
iface3 tcp,<empty>,<empty>,eth3,<empty>
[root@stlrx300s6-21 ~]#
```

6. Establish iSCSI sessions between initiators and targets. For example:
   - For E5400 storage, establish two sessions.
   - For E5500 storage, establish a total of four sessions.

```
[root@stlrx300s6-21 ~]# iscsiadm -m discovery -t sendtargets -p 192.168.131.101 -I iface3 -P 1
Target: iqn.1992-08.com.netapp:5481.60080e50001ff86400000000543ea12e
        Portal: 192.168.130.101:3260,1
                Iface Name: iface3
        Portal: [fe80:0000:0000:0000:0280:e5ff:fe19:34e9]:3260,1
                Iface Name: iface3
        Portal: 192.168.131.101:3260,1
                Iface Name: iface3
        Portal: [fe80:0000:0000:0000:0280:e5ff:fe19:34ec]:3260,1
                Iface Name: iface3
        Portal: 192.168.130.102:3260,2
                Iface Name: iface3
        Portal: [fe80:0000:0000:0000:0280:e5ff:fe19:39b1]:3260,2
                Iface Name: iface3
        Portal: 192.168.131.102:3260,2
                Iface Name: iface3
        Portal: [fe80:0000:0000:0000:0280:e5ff:fe19:39b4]:3260,2
                Iface Name: iface3
[root@stlrx300s6-21 ~]# iscsiadm -m node -T iqn.1992-
08.com.netapp:5481.60080e50001ff86400000000543ea12e -p 192.168.131.101 -I iface3 -l
Logging in to [iface: iface3, target: iqn.1992-
08.com.netapp:5481.60080e50001ff86400000000543ea12e, portal: 192.168.131.101,3260] (multiple)
Login to [iface: iface3, target: iqn.1992-08.com.netapp:5481.60080e50001ff86400000000543ea12e,
portal: 192.168.131.101,3260] successful.
[root@stlrx300s6-21 ~]#

[root@stlrx300s6-21 ~]# iscsiadm -m session
tcp: [2] 192.168.131.101:3260,1 iqn.1992-08.com.netapp:5481.60080e50001ff86400000000543ea12e
```

## Multipath Configuration Used for iSCSI Validation – DataNodes

Multipathing software is responsible for maintaining an active path to the underlying network storage in the event of disruption of one or more physical paths. It does this by presenting the DataNode operating system with a single virtual device that represents the active physical paths to the storage and manages the failover process that updates the virtual device in the event of a disruption. We used device mapper multipathing with the DataNodes. The following `multipath.conf` configuration was used in the tests:

```
devices {
        device {
                vendor                  "(NETAPP|LSI)"
                product                 "INF-01-00"
                product_blacklist       "Universal Xport"
                path_grouping_policy    group_by_prio
                path_checker            rdac
                features                "2 pg_init_retries 50"
                hardware_handler        "1 rdac"
                prio                    rdac
                failback                immediate
                no_path_retry           30
        }
}


blacklist {
}
```

## Additional Tuning for Block Devices

The additional tuning of the Linux servers that we performed for testing is described in Table 5. The sample commands are for the block device `/dev/sdb`. These settings must be implemented for all block devices used for HDFS data storage. These changes are not persistent across reboots, so they must be reset after each reboot. A good way to accomplish this is to include the commands in the Linux `/etc/rc.local` file.

**Table 5) Additional tuning for block devices.**

| Description of Setting | Suggested Value | Command Line Example |
|---|---|---|
| Block device kernel request queue length | 128 | echo 128 > /sys/block/sdb/queue/nr_requests |
| Block device queue depth | 128 | echo 128 > /sys/block/sdb/device/queue_depth |
| Block device read ahead in KB | 3072 | echo 3072 > /sys/block/sdb/queue/read_ahead_kb |
| Maximum number of kilobytes that the block layer allows for a file system request | 1024 | echo 1024 > /sys/block/sdb/queue/max_sectors_kb |
| Disable Red Hat transparent huge pages | Never | echo never >/sys/kernel/mm/redhat_transparent_hugepage/enabled |

## E-Series Storage Configuration Parameters

We modified the E-Series storage configuration for Hadoop by using the parameter settings described in Table 6.

**Table 6) E-Series storage configuration parameters.**

| Storage Parameter | Recommended Setting | Default Setting | Description |
|---|---|---|---|
| Cache block size | 32KB (or highest available setting) | 4KB | Block size for E-Series read/write cache. The largest size available was chosen to optimize I/O performance for the large block I/O generated by HDFS. This parameter is set for the entire array. |
| Read cache | Enabled | Enabled | Enables caching of prefetch data blocks to improve read performance. Read cache is set for each individual volume. |
| Write cache | Enabled | Enabled | Enables caching of writes, thereby improving write performance. Write cache is set for each individual volume. |
| Write cache without batteries | Disabled | Disabled | By default, batteries provide backup power to cache memory to avoid data loss during a power outage. This parameter is set at the volume level. |
| Write cache with mirroring | Disabled | Enabled | Maintains cache consistency between controllers to enable controller failover, but results in less available write cache and increased operational overhead. The NetApp Solution for Hadoop does not use the controller failover capability, so this capability is |

| Storage Parameter | Recommended Setting | Default Setting | Description |
|---|---|---|---|
| | | | disabled. It is enabled or disabled on a per-volume basis. |
| Dynamic cache read prefetch | Enabled | Enabled | Provides dynamic, intelligent read-ahead for improved read performance. Enabled at the volume level. |
| Volume segment size | 512KB (or largest size available) | 128KB | The amount of I/O written to a single disk drive before moving to the next drive in a RAID array. The largest segment size available was chosen to optimize data layout for the large block I/O generated by HDFS. The segment size is set for each individual volume. |

## E-Series Storage Provisioning for iSCSI

For Hadoop, NetApp recommends using volume groups to improve sequential workload, maximum system bandwidth, and storage-tuning customization. The steps in this procedure describe how the E-Series storage was provisioned for the iSCSI tests.

To provision E-Series storage for the iSCSI protocol, complete the following steps:

1. Create a volume group for each volume:

   **Note:** For our tests, we created eight volume groups for eight volumes; each group contained seven disks.

   a. In SANtricity Storage Manager, open the AMW for the E-Series storage array.

   b. Click the Storage & Copy Services tab, right-click Total Unconfigured Capacity, and select Create Volume Group.

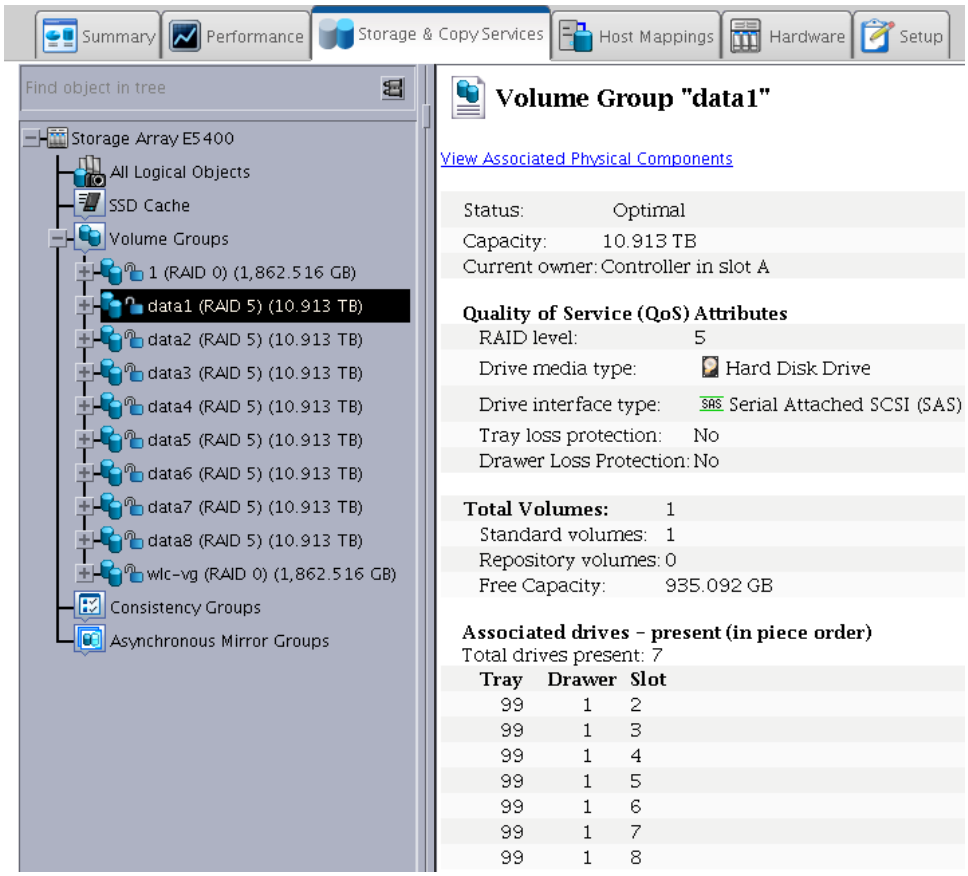
   c. Make the following selections:

   – Volume Group Name: `data1` (example)

   – Select RAID Level: RAID 5

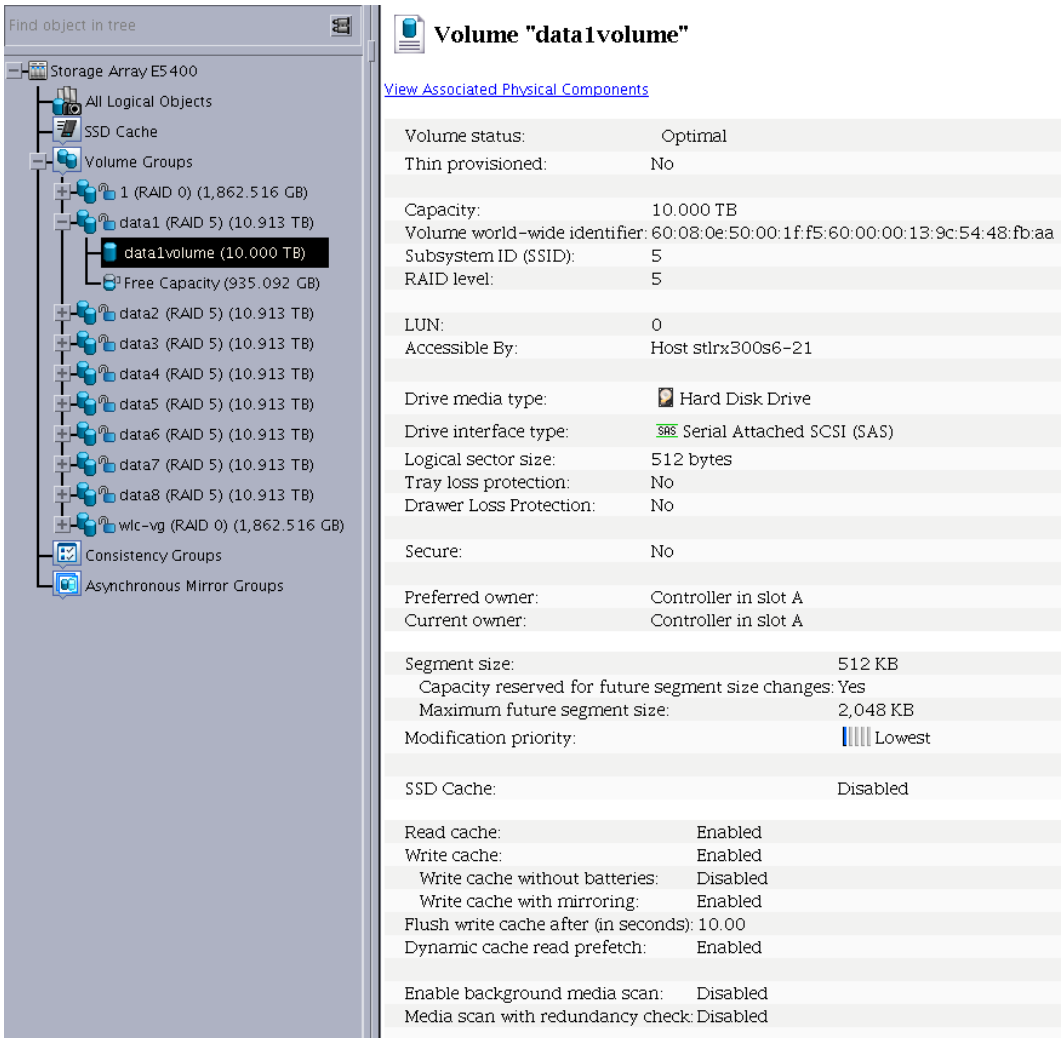
   – Drive Selection: 7 disks

   d. Click Finish to create the volume group. Verify that the new volume group is displayed in the storage array tree.




2. Create a volume:
   a. Click the Storage & Copy Services tab and expand the volume group created in step 1 (`data1` in this example).
   b. Right-click Free Capacity and select Create Volume.
   c. Configure the volume parameters in the following way:
   – New Volume Capacity: 10 TB
   – Volume Name: `data1volume` (example)
   – Map to Host: Map Later
   d. Configure the quality of service (QoS) attributes:
   – Volume I/O Characteristics Type: Custom
   – Select the Enable Dynamic Cache Read Prefetch checkbox.
   – Segment Size: 512 KB
   e. Click Finish to create the volume. Verify that the new volume is displayed under the correct volume group in the storage array tree.




3. Define the DataNode host in SANtricity Storage Manager:
   a. In the AMW, click the Host Mappings tab and expand the storage array tree.
   b. Select Default Group, right-click it, and select Define > Host.
   c. Make the following selections:
   – Select a host name: `stlrx300s6-21` (example).
   – Select Yes for the option Do You plan to Use Storage Partitions on This Storage Array?
   – For the host interface type, select iSCSI.
   – Select the option Add by Selecting a Known Unassociated Host Port Identifier.
   – Select the initiator name. In this example, it is `InitiatorName=iqn.1994-05.com.redhat:5f327355672b`.
   – Recommended: Enter an alias for the host port identifier.
   – For host type, select Linux (DM – MP).
   – Select the option No – This Host Will Not Share Access to the Same Volumes with Other Hosts.
4. Map a volume to the DataNode host:
   a. In the AMW, click the Host Mappings tab.
   b. Right-click the newly added DataNode host and select a host (`stlrx300s6-23` in this example).

c. Select a volume (`data1volume` in this example).

d. Click Finish.

e. Verify that the volume is mapped to the DataNode host. Select the volume and check the following settings:

- Volume Name: `data1volume`
- Accessible by: Host `stlrx300s6-21`
- LUN: 1
- Volume Capacity: 10.000 TB
- Type: Standard

5. Discover the mapped volume in the DataNode:

a. Run the `rescan-scsi-bus.sh` command. Ensure that the `sg3_utils` package is installed before you run the `rescan-scsi-bus.sh` command. Repeat this step for all the scans, from host0 to host10.

```
echo "- - - " > /sys/class/scsi_host/host10/scan
```

b. Verify the disk by running `multipath –ll`.

```
 [root@stlrx300s6-21 ~]# multipath -ll
mpathb (360080e50001ff5600000139c5448fbaa) dm-0 LSI,INF-01-00
size=10T features='3 queue_if_no_path pg_init_retries 50' hwhandler='1 rdac' wp=rw
|-+- policy='round-robin 0' prio=14 status=active
| |- 9:0:0:0  sdc 8:32 active ready running
| `- 11:0:0:0 sde 8:64 active ready running
`-+- policy='round-robin 0' prio=9 status=enabled
  |- 8:0:0:0  sdd 8:48 active ready running
  `- 10:0:0:0 sdb 8:16 active ready running
[root@stlrx300s6-21 ~]#
```

6. Create a file system and mount it in the DataNode:

a. Create a new partition:

- Install the XFS package (`xfsprogs`).

```
 [root@stlrx300s6-21 GA]# rpm -ivh /selinux/Packages/xfsprogs-3.1.1-14.el6.x86_64.rpm
warning: /selinux/Packages/xfsprogs-3.1.1-14.el6.x86_64.rpm: Header V3 RSA/SHA256 Signature, key
ID fd431d51: NOKEY
Preparing...                ########################################### [100%]
   1:xfsprogs               ########################################### [100%]
[root@stlrx300s6-21 GA]#
```

- Create a partition by running `parted`.

```
 [root@stlrx300s6-21 ~]# parted -s /dev/sdb mklabel gpt mkpart /dev/mapper/mpathbp1 xfs 6144s
10.0TB
```

b. Create the file system.

```
 [root@stlrx300s6-21 ~]# mkfs.xfs -f -L DISK1 -l size=65536b,sunit=256,lazy-count=1 -d
sunit=1024,swidth=6144 /dev/mapper/mpathbp1
meta-data=/dev/mapper/mpathbp1   isize=256    agcount=32, agsize=76294016 blks
         =                       sectsz=512   attr=2, projid32bit=0
data     =                       bsize=4096   blocks=2441405440, imaxpct=5
         =                       sunit=128    swidth=768 blks
naming   =version 2              bsize=4096   ascii-ci=0
log      =internal log           bsize=4096   blocks=65536, version=2
         =                       sectsz=512   sunit=32 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
[root@stlrx300s6-21 ~]#
```

c. Update `/etc/fstab`, create a folder, mount the partition, and check the mount point.

```
 [root@stlrx300s6-21 ~]# cat /etc/fstab | grep DISK1
LABEL=DISK1     /datadir        xfs     allocsize=128m,noatime,nobarrier,nodiratime     0       0
```

```
[root@stlrx300s6-21 ~]#
[root@stlrx300s6-21 ~]# mkdir /datadir
[root@stlrx300s6-21 ~]# mount /datadir
[root@stlrx300s6-21 ~]# touch /datadir/testfile
```

## Linux Kernel and Device Tuning for Hadoop

For the tests, we modified the default configuration of the Linux kernel and the device parameters for the operating system with the settings in Table 7.

**Table 7) Linux `/etc/sysctl.conf` settings used in tests.**

| Parameter | Actual Setting/Default Value | Description |
|---|---|---|
| net.ipv4.ip_forward | 0/0 | Controls IP packet forwarding. |
| net.ipv4.conf.default.rp_filter | 1/0 | Controls source route verification. |
| net.ipv4.conf.default.accept_source_route | 0/1 | Does not accept source routing. |
| kernel.sysrq | 0/1 | Controls the system-request debugging functionality of the kernel. |
| kernel.core_uses_pid | 1/0 | Controls whether core dumps append the PID to the core file name. Useful for debugging multithreaded applications. |
| kernel.msgmnb | 65536/16384 | Controls the maximum size of a message, in bytes. |
| kernel.msgmax | 65536/8192 | Controls the default maximum size of a message queue. |
| kernel.shmmax | 68719476736/33554432 | Controls the maximum shared segment size, in bytes. |
| kernel.shmall | 4294967296/2097512 | Controls the maximum number of shared memory segments, in pages. |
| net.core.rmem_default | 262144/129024 | Sets the default OS receive buffer size. |
| net.core.rmem_max | 16777216/131071 | Sets the maximum OS receive buffer size. |
| net.core.wmem_default | 262144/129024 | Sets the default OS send buffer size. |
| net.core.wmem_max | 16777216/131071 | Sets the maximum OS send buffer size. |
| net.core.somaxconn | 1000/128 | Sets the maximum number of sockets that the kernel can serve at one time. It is set on:<br>• NameNode, secondary NameNode, and JobTracker in Hadoop 1.0<br>• Standby node and ResourceManager in Hadoop 2.0 |
| fs.file-max | 6815744/4847448 | Sets the total number of file descriptors. |
| net.ipv4.tcp_timestamps | 0/1 | Turns off the TCP time stamps. |
| net.ipv4.tcp_sack | 1/1 | Turns on select ACK for TCP. |

| Parameter | Actual Setting/Default Value | Description |
|---|---|---|
| net.ipv4.tcp_window_scaling | 1/1 | Turns on TCP window scaling. |
| kernel.shmmni | 4096/4096 | Sets the maximum number of shared memory segments. |
| kernel.sem | 250 32000 100 128/250 32000 32 128 | Sets the maximum number and size of semaphore sets that can be allocated. |
| fs.aio-max-nr | 1048576/65536 | Sets the maximum number of concurrent I/O requests. |
| net.ipv4.tcp_rmem | 4096 262144 16777216/ 4096 87380 4194304 | Sets the minimum, default, and maximum receive window size. |
| net.ipv4.tcp_wmem | 4096 262144 16777216/4096 87380 4194304 | Sets the minimum, default, and maximum transmit window size. |
| net.ipv4.tcp_syncookies | 0/0 | Turns off the TCP syncookies. |
| sunrpc.tcp_slot_table_entries | 128/16 | Sets the maximum number of in-flight remote procedure call (RPC) requests between a client and a server. This value is set on the NameNode and secondary NameNode to improve NFS performance. |
| vm.dirty_background_ratio | 1/10 | Sets the maximum percentage of active system memory that can be used for dirty pages before dirty pages are flushed to storage. Lowering this parameter results in more frequent page-cache flushes to storage, resulting in a more constant I/O write rate to storage and better storage performance for writes. |
| vm.dirty_ratio | 20/40 | Decreases the amount of memory available for dirty pages. |
| vm.nr_hugepages | 0/memory dependent | Forces the number of Red Hat huge pages to 0. |
| fs.xfs.rotorstep | 254/1 | Increases the number of files to be written to an XFS allocation group before moving to the next allocation group. |

## /etc/rc.local Settings

Sometimes, the XFS and some other parameters are not updated after reboot, so NetApp recommends keeping their settings in the `/etc/rc.local` file.

```
[root@stlrx300s6-32 ~]# cat /etc/rc.local
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local
/sbin/modprobe sunrpc
/sbin/modprobe xfs
/sbin/sysctl -p
```

```
/sbin/iptables -F
echo never > /sys/kernel/mm/redhat_transparent_hugepage/enabled
echo 128 > /sys/block/sda/queue/nr_requests
echo 128 > /sys/block/sda/device/queue_depth
echo 3072 > /sys/block/sda/queue/read_ahead_kb
echo 1024 > /sys/block/sda/queue/max_sectors_kb

[root@stlrx300s6-32 ~]#
```

## Hadoop Configuration and Parameter Settings Used for iSCSI

The `hadoop-env.sh` configuration file typically contains commands to set environmental variables for the Hadoop environment. Check and modify the `JAVA_HOME` and `HADOOP_HEAPSIZE` parameters in the default configuration, if required.

```
export HADOOP_HEAPSIZE="2048"
export JAVA_HOME=/usr/jdk64/jdk1.7.0_45
```

**Note:** To improve cluster efficiency, change the `HADOOP_HEAPSIZE` setting from 1000MB to 2048MB.

## core-site.xml and hdfs-site.xml Settings Used for iSCSI

The `core-site.xml` parameter settings used in the test environment are described in Table 8. These settings are important configurations used by other components in the cluster.

**Table 8) Hadoop `core-site.xml` parameter settings used in tests.**

| Parameter | Actual Setting/Default Value | Description |
|---|---|---|
| fs.defaultFS | hdfs://stlrx300s6-31.stl.netapp.com:8020/<br><br>Default value: file:/// | Name of the default file system specified as a URL (IP address or host name of the NameNode along with the port to be used). |
| mapreduce.jobtracker.webinterface.trusted | true/false | Enables or disables certain management functions in the Hadoop web UI, including the ability to kill jobs and modify job priorities. |
| io.file.buffer.size | 262144/4096 | Size in bytes of the read/write buffer. |
| topology.script.file.name | /etc/hadoop/conf/topology_script<br><br>Default value: null | Script used to resolve the slave node name or IP address to a rack ID. Used to invoke Hadoop rack awareness. The default value of null results in all slaves being given a rack ID of `/default-rack`. |
| topology.script.number.args | 1/100 | Sets the maximum acceptable number of arguments to be sent to the topology script at one time. |

The `hdfs-site.xml` parameter settings used in the test environment are described in Table 9.

**Table 9) Hadoop `hdfs-site.xml` parameter settings used in tests.**

| Parameter | Actual Setting/Default Value | Description |
|---|---|---|
| dfs.namenode.name.dir | /local/hadoop/hdfs/namenode,/fsimage_bkp/hadoop/hdfs/namenode | Path on the local file system where the NameNode persistently stores the namespace and transaction logs. |

| Parameter | Actual Setting/Default Value | Description |
|---|---|---|
| | Default value: ${hadoop.tmp.dir}/dfs/name | **Note:** If this path is a comma-delimited list of directories (as in this configuration), then the name table is replicated in all of the directories for redundancy.<br>The directory `/fsimage_bkp/hadoop/hdfs/` is a location on NFS-mounted FAS storage where NameNode metadata is mirrored and protected, a key feature of the Hadoop solution. |
| dfs.datanode.data.dir | /datadir/hadoop/hdfs/data<br><br>Default value: ${hadoop.tmp.dir}/dfs/data | Directory path locations on the DataNode local file systems where HDFS data blocks are stored. |
| dfs.namenode.checkpoint.dir | /local/hadoop/hdfs/namesecondary<br><br>Default value: ${hadoop.tmp.dir}/dfs/namesecondary | Path to the location where checkpoint images are stored (used by the secondary NameNode). |
| dfs.replication | 2/3 | HDFS block replication count. The Hadoop default count is 3. The NetApp Solution for Hadoop uses a default of 2. |
| dfs.blocksize | 134217728 (128MB)/67108864 | HDFS data storage blocks size in bytes. |
| dfs.namenode.handler.count | 128/10 | Number of server threads for the NameNode. |
| dfs.datanode.max.transfer.threads | 4096 | Specifies the maximum number of threads to use for transferring data in and out of the DataNode. |
| dfs.namenode.replication.max-streams | 8/2 | Maximum number of replications a DataNode is allowed to handle at one time. |

## Slave Configuration Used for iSCSI

In a Hadoop 2.0 cluster, the NameNode, the ResourceManager, and the standby node are considered master nodes. Other nodes such as the NodeManager and the DataNodes are referred to as slaves.

The following DataNodes and NodeManagers were used in the iSCSI tests:

```
[root@stlrx300s6-22 conf]# cat slaves
stlrx300s6-25.stl.netapp.com
stlrx300s6-28.stl.netapp.com
stlrx300s6-24.stl.netapp.com
stlrx300s6-21.stl.netapp.com
stlrx300s6-26.stl.netapp.com
stlrx300s6-29.stl.netapp.com
stlrx300s6-27.stl.netapp.com
stlrx300s6-23.stl.netapp.com
[root@stlrx300s6-22 conf]#
```

## mapred-site.xml Settings Used for iSCSI

In addition to the default parameters, the `mapred-site.xml` parameters listed in Table 10 were used in the iSCSI testing.

**Table 10) Hadoop `mapred-site.xml` parameters used in tests.**

| Parameter | Actual Setting/Default Value | Description |
|---|---|---|
| mapreduce.jobhistory.address | stlrx300s6-22.stl.netapp.com:10020 / 0.0.0.0:10020 | The MapReduce JobHistory server IPC host port. |
| mapreduce.framework.name | yarn/local | The runtime framework for executing MapReduce jobs. Its value can be `local`, `classic`, or `yarn`. The following settings were used in the tests:<br>• `yarn` for MapReduce version 2 (MRv2)<br>• `classic` for MRv1<br>• `local` for local runs on MapReduce jobs |
| mapreduce.map.speculative | false/true | If set to `true`, then multiple instances of some map tasks may be executed in parallel. |
| yarn.app.mapreduce.am.resource.mb | 4096/1536 | The amount of memory the MapReduce AppMaster needs for executing a job. |
| mapreduce.map.java.opts | -Xmx1638m | Hadoop Mapper is a Java® process that owns heap memory allocation through mapreduce.map.java.opts. NetApp recommends having 20% of memory for the Java code or 0.8 x RAM per container. |
| mapreduce.cluster.administrators | hadoop | Hadoop group name. |
| mapreduce.application.classpath | $HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*,$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/lib/* / <emptry> | CLASSPATH for MapReduce applications. |
| mapreduce.output.fileoutputformat.compress.type | BLOCK/RECORD | Setting for compressing job outputs as SequenceFiles. Should be set to either `NONE`, `RECORD` or `BLOCK`. NetApp recommends the `BLOCK` setting. |
| mapreduce.reduce.speculative | false/true | If true, then multiple instances of some reduce tasks may be executed in parallel. |
| mapreduce.reduce.java.opts | -Xmx3276m | Larger heap-size for child Java virtual machines (JVMs) of reduces. The calculation is 0.8 x 2 x RAM per container. |
| yarn.app.mapreduce.am.admin-command-opts | -server -XX:NewRatio=8 -Djava.net.preferIPv4Stack=true - | Java opts for the MapReduce AppMaster processes (for administrative purposes). |

| Parameter | Actual Setting/Default Value | Description |
|---|---|---|
| | Dhadoop.metrics.log.level=WARN / <empty> | |
| mapreduce.map.sort.spill.percent | 0.7/0.8 | The soft limit in the serialization buffer. When the limit is reached, a thread begins to spill the contents to disk in the background.<br>**Note:** Collection is not blocked if the threshold is exceeded while a spill is already in progress, so spills may be larger than the limit when the value is set to less than 0.5. |
| mapreduce.task.timeout | 300000/600000 | The number of milliseconds before a task is terminated if it neither reads an input, writes an output, nor updates its status string. A value of 0 disables the timeout. |
| mapreduce.map.memory.mb | 2048 | Larger resource limit for maps. It is equal to the RAM-per-container value. |
| mapreduce.task.io.sort.factor | 100/10 | The number of streams to merge at once while sorting files. This number determines the number of open file handles. |
| mapreduce.jobhistory.intermediate-done-dir | /mr-history/tmp / ${yarn.app.mapreduce.am.staging-dir}/history/done_intermediate | Directory where history files are written by MapReduce jobs. |
| mapreduce.reduce.memory.mb | 4096 | Larger heap-size for child JVMs of maps. It is two times the RAM-per-container value. |
| mapreduce.admin.user.env | LD_LIBRARY_PATH=/usr/lib/hadoop/lib/native:/usr/lib/hadoop/lib/native/`$JAVA_HOME/bin/java -d32 -version &amp;gt; /dev/null;if [ $? -eq 0 ]; then echo Linux-i386-32; else echo Linux-amd64-64;fi` / <empty> | Additional execution environment entries for map and reduce task processes. |
| yarn.app.mapreduce.am.staging-dir | /user / /tmp/hadoop-yarn/staging | The staging directory used when submitting jobs. |
| mapreduce.reduce.shuffle.parallelcopies | 30/5 | The number of parallel transfers run by reduce during the copy (shuffle) phase. |
| mapreduce.jobhistory.done-dir | /mr-history/done / ${yarn.app.mapreduce.am.staging-dir}/history/done | Directory where history files are managed by the MapReduce JobHistory server. |
| mapreduce.admin.reduce.child.java.opts | -server -XX:NewRatio=8 -Djava.net.preferIPv4Stack=true -Dhadoop.metrics.log.level=W | Setting that configures MapReduce to use snappy compression. |

| Parameter | Actual Setting/Default Value | Description |
|---|---|---|
| | ARN | |
| mapreduce.task.io.sort.mb | 1024/100 | The total amount of buffer memory used for sorting files, in megabytes. By default, gives each merge stream 1MB, which should minimize seeks. |
| yarn.app.mapreduce.am.command-opts | -Xmx3277m / -Xmx5734m | Java opts for the MapReduce AppMaster processes. |
| mapreduce.admin.map.child.java.opts | -server -XX:NewRatio=8 -Djava.net.preferIPv4Stack=true -Dhadoop.metrics.log.level=WARN | Setting that configures MapReduce to use snappy compression. |

## yarn-site.xml Settings Used for iSCSI

Some of the `yarn-site.xml` default parameters were changed for testing. Table 11 describes the parameters that were changed.

**Table 11) Hadoop `yarn-site.xml` parameters used in tests.**

| Parameter | Actual Setting/Default Value | Description |
|---|---|---|
| yarn.nodemanager.local-dirs | /datadir/hadoop/yarn/local / ${hadoop.tmp.dir}/nm-local-dir | List of directories in which to store localized files. |
| yarn.resourcemanager.resource-tracker.address | stlrx300s6-22.stl.netapp.com:8025 / ${yarn.resourcemanager.hostname}:8031 | YARN server name with port. |
| yarn.resourcemanager.hostname | stlrx300s6-22.stl.netapp.com / 0.0.0.0 | YARN server name or IP address. |
| yarn.nodemanager.health-checker.script.timeout-ms | 60000/1200000 | Script time-out period.. |
| yarn.nodemanager.resource.memory-mb | 43008/8192 | Amount of physical memory, in megabytes, that can be allocated for containers. |
| yarn.timeline-service.ttl-ms | 2678400000/604800000 | Time-to-live for timeline store data, in milliseconds. |
| yarn.timeline-service.webapp.address | stlrx300s6-32.stl.netapp.com:8188 / ${yarn.timeline-service.hostname}:8188 | The HTTP address of the timeline service web application. |
| yarn.timeline-service.enabled | true/false | Indicates to clients whether the timeline service is enabled. If enabled, clients put entities and events into the timeline server. |
| yarn.nodemanager.log.retain- | 604800/10800 | Time in seconds to retain user logs. Only |

| Parameter | Actual Setting/Default Value | Description |
|---|---|---|
| second | | applicable if log aggregation is disabled. |
| yarn.nodemanager.log-aggregation.compression-type | gz/none | T-file compression types used to compress aggregated logs. |
| yarn.nodemanager.log-dirs | /datadir/hadoop/yarn/log / ${yarn.log.dir}/userlogs | Location to store container logs. An application's localized log directory can be found in `${yarn.nodemanager.log-dirs}/application_${appid}`. Individual container log directories are under this location, in directories named `container_{$contid}`. Each container directory contains the `stderr`, `stdin`, and `syslog` files generated by that container. |
| yarn.nodemanager.health-checker.interval-ms | 135000/600000 | Frequency for running the node-health script. |
| yarn.nodemanager.remote-app-log-dir | /app-logs / /tmp/logs | Location for aggregating logs. |
| yarn.nodemanager.aux-services | mapreduce_shuffle/<empty> | The valid service name should only contain `a-zA-Z0-9_` and cannot start with numbers. |
| yarn.nodemanager.vmem-check-enabled | false/true | Determines whether virtual memory limits are enforced for containers. |
| yarn.admin.acl | <empty>/* | ACL for who can be the administrator of the YARN cluster. |
| yarn.resourcemanager.webapp.address | stlrx300s6-22.stl.netapp.com:8088 / ${yarn.resourcemanager.hostname}:8088 | The HTTP address of the ResourceManager web application. |
| yarn.timeline-service.leveldb-timeline-store.path | /local/hadoop/yarn/timeline / ${hadoop.tmp.dir}/yarn/timeline | Store file name for leveldb timeline store. |
| yarn.resourcemanager.admin.address | stlrx300s6-22.stl.netapp.com:8141 / ${yarn.resourcemanager.hostname}:8033 | The address of the ResourceManager administrator interface. |
| yarn.scheduler.minimum-allocation-mb | 1024/2048 | The minimum allocation for every container request at the RAM, in megabytes. Memory requests lower than this value will not take effect, and the specified value gets allocated at a minimum. |

## Rack Awareness Example in iSCSI Setup

This procedure describes how the rack awareness feature was configured in the iSCSI setup.

To configure rack awareness, complete the following steps:

1. Copy the `rack-topology.sh` and `rack-topology.sh` files to the `/etc/Hadoop/conf` directory on all nodes in the cluster.

```
[root@stlrx300s6-22 conf]# cat rack-topology.sh
#!/bin/bash

# Adjust/Add the property "net.topology.script.file.name"
# to core-site.xml with the "absolute" path the this
# file.  ENSURE the file is "executable".

# Supply appropriate rack prefix
RACK_PREFIX=default

# To test, supply a hostname as script input:
if [ $# -gt 0 ]; then

CTL_FILE=${CTL_FILE:-"rack_topology.data"}

HADOOP_CONF=${HADOOP_CONF:-"/etc/hadoop/conf"}

if [ ! -f ${HADOOP_CONF}/${CTL_FILE} ]; then
  echo -n "/$RACK_PREFIX/rack "
  exit 0
fi

while [ $# -gt 0 ] ; do
  nodeArg=$1
  exec< ${HADOOP_CONF}/${CTL_FILE}
  result=""
  while read line ; do
    ar=( $line )
    if [ "${ar[0]}" = "$nodeArg" ] ; then
      result="${ar[1]}"
    fi
  done
  shift
  if [ -z "$result" ] ; then
    echo -n "/$RACK_PREFIX/rack "
  else
    echo -n "/$RACK_PREFIX/rack_$result "
  fi
done

else
  echo -n "/$RACK_PREFIX/rack "
fi
[root@stlrx300s6-22 conf]#
```

2. Create a `rack_topology.data` file.

```
[root@stlrx300s6-22 conf]# cat rack_topology.data
10.63.150.109 01
10.63.150.111 01
10.63.150.113 01
10.63.150.115 01
10.63.150.117 02
10.63.150.119 02
10.63.150.121 02
10.63.150.123 02

[root@stlrx300s6-22 conf]#
```

3. Stop the HDFS and update the `core-site.xml file` for the topology script properties, such as `net.topology.script.file.name` and `net.topology.script.number.args`.

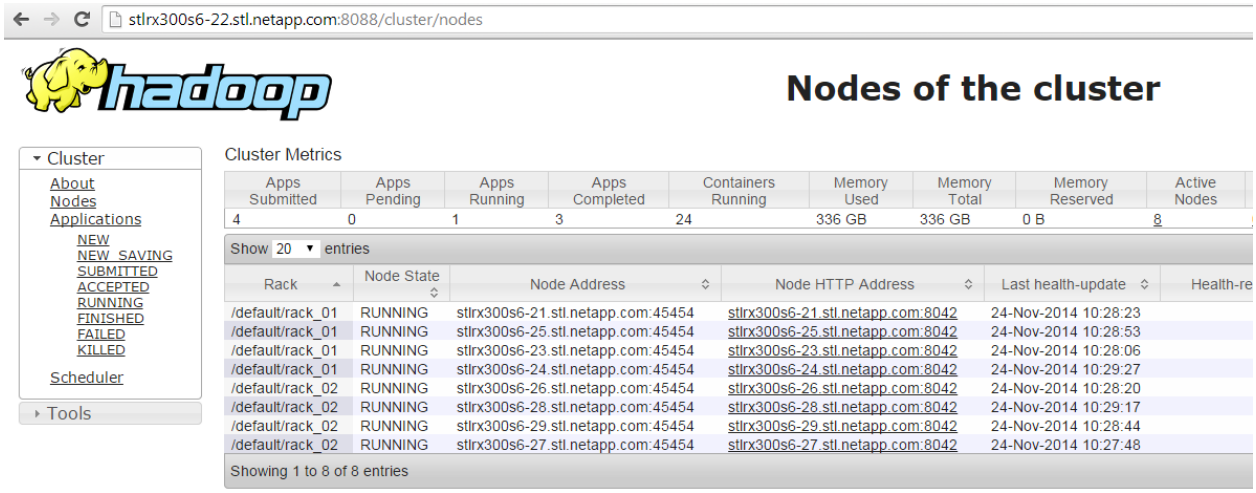
```
<property>
<name>topology.script.file.name</name>
<value>/etc/hadoop/conf/rack-topology.sh</value>
</property>
```

```
<property>
<name>ipc.client.connect.max.retries</name>
<value>50</value>
</property>
```

4. Restart HDFS and MapReduce.
5. Verify rack awareness:
   a. To verify the number of racks, run the command `sudo –u hdfs hadoop fsck`. For example, Number of Racks: 2.
   b. To verify rack location, run the command `sudo –u hdfs hadoop dfsadmin –report`. For example, Rack: `/default/rack_02`.

**Note:** You can also check rack details from the ResourceManager server for the cluster.




**Note:** For more information about how to configure rack awareness, refer to Configuring Rack Awareness on HDP.

## Test Cases Results

The following subsections present the details of each test conducted to validate the iSCSI protocol with Hadoop 2.0.

### Test Case 1: Initial Tuning and Full Functionality

**Table 12) Test case 1 details.**

| Test Information | Details |
|---|---|
| **Test case ID** | 1 |
| **Test type** | Initial tuning and full function |
| **Execution type** | Automated |
| **Duration** | Multiple runs, one day total |
| **Description** | This test runs a TeraGen job with a duration greater than 10 minutes to generate a substantial dataset. It then runs a TeraSort job on the dataset created by TeraGen. |

| Test Information | Details |
|---|---|
| **Prerequisites** | The HDFS components have been started. |
| **Expected results** | • Proper output results are received from the TeraSort Reduce stage.<br>• No tasks on individual task nodes (DataNodes) fail.<br>• The file system (HDFS) maintains integrity and is not corrupted.<br>• All test environment components are still running. |
| **Notes** | • Use integrated web and UI tools to monitor the Hadoop tasks and file system (see example).<br>• Use Ganglia to monitor the Linux server in general. |

### Setup Procedure

1. Remove any previous HDFS artifacts before each run.

### Execution Procedure

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 Case 2: Full Functionality with Fault Injection

**Table 13) Test case 2 details.**

| Test Information | Details |
|---|---|
| **Test case ID** | 2 |
| **Test type** | Full function with fault injection |
| **Execution type** | Manual |
| **Duration** | Multiple runs, one day total |
| **Description** | This test runs a TeraGen job with a duration greater than 10 minutes to generate a substantial dataset. It then runs a TeraSort job on the dataset created by TeraGen. |
| **Prerequisites** | • The HDFS components have been started.<br>• Test case 1 has been completed successfully. |
| **Expected results** | • Proper output results are received from the TeraSort Reduce stage.<br>• No tasks on individual task nodes (DataNodes) fail.<br>• The file system (HDFS) maintains integrity and is not corrupted. |
| **Notes** | • Use integrated web and UI tools to monitor the Hadoop tasks and file system (see example).<br>• Use Ganglia to monitor the Linux server in general. |

### Setup Procedure

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

### Execution Procedure

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.

2. After TeraGen has been running for approximately 23 minutes, fault a disk in each of the four controllers on the two E-Series arrays.

## Test Case 3: NameNode Metadata Recovery

**Table 14) Test case 3 details.**

| Test Information | Details |
|---|---|
| **Test case ID** | 3 |
| **Test type** | Recovery |
| **Execution type** | Manual |
| **Duration** | 2 days |
| **Description** | This scenario tests the failure of the Hadoop NameNode and the procedures to recover the cluster from the failure. |
| **Prerequisites** | Test cases 1 and 2 have been completed successfully. |
| **Expected results** | The output from TeraSort matches the original output from test case 1. |
| **Notes** | • Use integrated web and UI tools to monitor the Hadoop tasks and file system.<br>• Use Ganglia to monitor the Linux server in general. |

### Setup Procedure

1. Repeat test 1.
2. Note the total reduce input record count from the TeraSort job.
3. Run the `hadoop fsck /` command and note the following:
   - Health status of HDFS
   - Number of files
   - Total size of the file system
   - Corrupt blocks

### Execution Procedure

1. Repeat the TeraSort job.
2. While it is running, halt the NameNode to simulate loss of HDFS metadata.
3. Shut down all Hadoop daemons on all servers in the cluster.
4. NFS-mount the FAS volume containing the NameNode metadata onto the secondary NameNode, and copy the entire directory tree to the secondary NameNode.
5. Modify the `core-site.xml` file, making the secondary NameNode server the new NameNode server.
6. Replicate that file to all servers in the cluster.
7. Start the NameNode daemon on the secondary NameNode server.
8. Restart the secondary NameNode daemon on the new NameNode server.
9. Start the DataNode daemons on all DataNodes.
10. Start the ResourceManager daemon on the ResourceManager node.
11. Start the NodeManager daemons on all NodeManager nodes (DataNodes).

12. From any node in the cluster, run the `hadoop dfs –ls` command to verify that the data file created by TeraGen exists.
13. Verify the total amount of HDFS storage used.
14. Run the `hadoop fsck /` command to compare to results recorded before the NameNode was halted.
15. Run TeraSort against the file originally generated by TeraGen and monitor it for error-free completion.

    **Note:** For further information, refer to NameNode High Availability for Hadoop.

## Additional Information

To learn more about NetApp Hadoop solutions, visit the NetApp Open Solution for Hadoop webpage. This page contains additional information about the base reference design and about FlexPod® Select with Hadoop, which is the validated base reference architecture with Cisco® servers and networking. Two Cisco Validated Designs for FlexPod Select with Hadoop are available:

- FlexPod Select with Cloudera Enterprise
- FlexPod Select with Hortonworks Data Platform

For information about how to contact a NetApp sales representative or a NetApp partner to either investigate NetApp solutions further or start a pilot or proof of concept, visit the How to Buy page.

More information about the Hadoop project, technologies, and ecosystem can be found at www.Hadoop.apache.org. You can also contact the authors directly at iyerv@netapp.com, gustav.horn@netapp.com, or karthikeyan.nagalingam@netapp.com.

## Version History

| Version | Date | Document Version History |
|---|---|---|
| Version 1.0 | April 2014 | Initial release |
| Version 2.0 | January 2015 | Added information about Hadoop 2.0 with iSCSI |

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.