DELL EMC

# DELL EMC ISILON BEST PRACTICES FOR HADOOP DATA STORAGE


## ABSTRACT

This white paper describes the best practices for setting up and managing the HDFS service on a Dell EMC Isilon cluster to optimize data storage for Hadoop analytics.



October 2016

## TABLE OF CONTENTS


**INTRODUCTION ..... 5**

OVERVIEW OF ISILON SCALE-OUT NAS FOR BIG DATA ..... 5

HOW HADOOP WORKS WITH ISILON SCALE-OUT NAS ..... 6

NAMENODE REDUNDANCY ..... 6

RACK AWARENESS ..... 7

THE HDFS ARCHITECTURE OF ONEFS ..... 7

**HDFS SETUP ..... 8**

SUPPORTED DISTRIBUTIONS ..... 8

OVERVIEW OF HOW TO INTEGRATE AN ISILON CLUSTER WITH HADOOP ..... 8

SIZING AN ISILON CLUSTER FOR A HADOOP COMPUTE GRID ..... 10

MAXIMIZING NETWORK THROUGHPUT WITH 10 GIGABIT ETHERNET ..... 10

DO NOT RUN THE NAMENODE AND DATANODE SERVICES ON CLIENTS ..... 10

OVERVIEW OF ISILON HDFS COMMANDS ..... 10

THE HDFS SERVICE AND LOGS ..... 10

CREATING DIRECTORIES AND SETTING PERMISSIONS ..... 11

SETTING THE ROOT PATH ..... 13

SETTING UP A SMARTCONNECT ZONE FOR NAMENODES ..... 13

**WORKING WITH DIFFERENT HADOOP DISTRIBUTIONS ..... 13**

CLOUDERA ..... 14

PIVOTAL HD ..... 14

**TUNING ONEFS FOR HDFS OPERATIONS ..... 14**

BLOCK SIZES ..... 14

TUNING THE NUMBER OF HDFS THREADS ..... 15

OBTAINING STATISTICS TO TUNE ONEFS FOR A HADOOP WORKFLOW ..... 15

STORING INTERMEDIATE JOBS ON AN ISILON CLUSTER ..... 17

DEALING WITH SHUFFLE SPACE BOTTLENECKS ..... 18

FEDERATION WITH DAS AND NAS ..... 18

STORAGE POOLS FOR DIFFERENT DATA SETS ..... 19

DATA PROTECTION BEST PRACTICES ..... 19
ALIGNING WORKLOADS WITH DATA ACCESS PATTERNS ..... 20
**SMARTCONNECT FOR HDFS ..... 20**
HANDLING NAMENODE AND DATANODE FAILURE SCENARIOS ..... 21
USE MULTIPLE IP ADDRESSES PER INTERFACE ..... 23
SERVICE SUBNETS AND DNS INTEGRATION ..... 23
**ALIGN DATASETS WITH STORAGE POOLS ..... 24**
SMARTPOOLS FOR ANALYTICS DATA ..... 24
WRITE CACHING WITH SMARTCACHE ..... 24
GUIDELINES FOR FILE POOL MANAGEMENT ..... 24
CHECK THE OPTIMIZATION SETTINGS OF DIRECTORIES AND FILES ..... 25
**SECURING HDFS CONNECTIONS WITH KERBEROS ..... 26**
SET UP KERBEROS AUTHENTICATION WITH ACTIVE DIRECTORY ..... 26
SET UP KERBEROS AUTHENTICATION WITH MIT KERBEROS 5 ..... 27
ALTERING HADOOP CONFIGURATION FILES FOR KERBEROS ..... 27
TEST KERBEROS AUTHENTICATION ..... 30
TROUBLESHOOTING KERBEROS AUTHENTICATION ..... 30
**CONCLUSION ..... 311**

# INTRODUCTION

The Dell EMC® Isilon® scale-out network-attached storage (NAS) platform provides Hadoop clients with direct access to big data through a Hadoop File System (HDFS) interface. Powered by the distributed Dell EMC Isilon OneFS® operating system, a Dell EMC Isilon cluster delivers a scalable pool of storage with a global namespace.

Hadoop compute clients access the data that is stored in an Isilon cluster by connecting to any node over HDFS. Every node in the cluster can act as a namenode and a datanode. Each node boosts performance and expands the cluster's capacity. For Hadoop analytics, the Isilon scale-out distributed architecture minimizes bottlenecks, rapidly serves big data, and optimizes performance for MapReduce jobs.

An Isilon cluster fosters data analytics without ingesting data into an HDFS file system. With a Dell EMC Isilon cluster, you can store data on an enterprise storage platform with your existing workflows and standard protocols, including SMB, HTTP, FTP, REST, and NFS as well as HDFS. Regardless of whether you store the data with SMB or NFS, however, you can analyze it with a Hadoop compute grid through HDFS. There is no need to set up an HDFS file system and then load data into it with tedious HDFS copy commands or specialized Hadoop connectors.

An Isilon cluster simplifies data management while cost-effectively maximizing the value of data. Although high-performance computing with Hadoop has traditionally stored data locally in compute clients' HDFS file system, the following use cases make a compelling case for coupling MapReduce with Isilon scale-out NAS:

- Store data in a POSIX-compliant file system with SMB and NFS workflows and then access it through HDFS for MapReduce
- Scale storage independently of compute as your data sets grow
- Protect data more reliably and efficiently instead of replicating it
- Eliminate HDFS copy operations to ingest data and hadoop fs commands to manage data
- Implement distributed fault-tolerant namenode services
- Manage data with enterprise storage features such as deduplication and snapshots

Storing data in an Isilon scale-out NAS cluster instead of HDFS clients streamlines the entire analytics workflow. Isilon's HDFS interface eliminates extracting the data from a storage system and loading it into an HDFS file system. Isilon's multiprotocol data access with SMB and NFS eliminates exporting the data after you analyze it. The result is that you can not only increase the ease and flexibility with which you analyze data, but also reduce capital expenditures and operating expenses.

This white paper describes the best practices for managing an Isilon cluster for Hadoop data analytics.

## OVERVIEW OF ISILON SCALE-OUT NAS FOR BIG DATA

The Dell EMC Isilon scale-out platform combines modular hardware with unified software to provide the storage foundation for data analysis. Isilon scale-out NAS is a fully distributed system that consists of nodes of modular hardware arranged in a cluster. The distributed Isilon OneFS operating system combines the memory, I/O, CPUs, and disks of the nodes into a cohesive storage unit to present a global namespace as a single file system.

The nodes work together as peers in a shared-nothing hardware architecture with no single point of failure. Every node adds capacity, performance, and resiliency to the cluster, and each node acts as a Hadoop namenode and datanode. The namenode daemon is a distributed process that runs on all the nodes in the cluster. A compute client can connect to any node in the cluster to access namenode services.

As nodes are added, the file system expands dynamically and redistributes data, eliminating the work of partitioning disks and creating volumes. The result is a highly efficient and resilient storage architecture that brings all the advantages of an enterprise scale-out NAS system to storing data for analysis.

Unlike traditional storage, Hadoop's ratio of CPU, RAM, and disk space depends on the workload—factors that make it difficult to size a Hadoop cluster before you have had a chance to measure your MapReduce workload. Expanding data sets also makes sizing decisions upfront problematic. Isilon scale-out NAS lends itself perfectly to this scenario: Isilon scale-out NAS lets you increase CPUs, RAM, and disk space by adding nodes to dynamically match storage capacity and performance with the demands of a dynamic Hadoop workload.

An Isilon cluster optimizes data protection. OneFS more efficiently and reliably protects data than HDFS. The HDFS protocol, by default, replicates a block of data three times. In contrast, OneFS stripes the data across the cluster and protects the data with forward error correction codes, which consume less space than replication with better protection.

An Isilon cluster also includes enterprise features to back up your data and to provide high availability. For example, in managing your datanode data, a best practice with a traditional Hadoop system is to back up your data to another system—an operation that must be performed with brute force by using a tool like DistCP. OneFS includes clones, NDMP backups, synchronization, geo- replication, snapshots, file system journal, virtual hot spare, antivirus, IntegrityScan, dynamic sector repair, and accelerated drive rebuilds. For complete information about the data availability features of OneFS, see the white paper titled Isilon: Isilon: Data Availability and Protection.

The enterprise features of OneFS ease data management. OneFS includes storage pools, deduplication, automated tiering, quotas, high-performing SSDs, capacity-optimized HDDs, and monitoring with InsightIQ. SmartPools, for example, provides tiered storage so that you can store current data in a high-performance storage pool while storing older data in a lower, more cost-effective pool in case you need to analyze it again later. For more information about the enterprise features of OneFS, see the white paper titled Hadoop on Isilon NAS.

For security, OneFS can authenticate HDFS connections with Kerberos. SmartLock can protect sensitive data from malicious, accidental, or premature alteration or deletion to help comply with SEC 17a-4 regulations.

## HOW HADOOP WORKS WITH ISILON SCALE-OUT NAS

An Isilon cluster separates data from compute. As Hadoop clients run MapReduce jobs, the clients access the data stored on an Isilon cluster over HDFS. OneFS becomes the HDFS file system for MapReduce clients.

OneFS implements the server-side operations of the HDFS protocol on every node, and each node functions as both a namenode and a datanode. An Isilon node, however, does not act as a job tracker or a task tracker; those functions remain the purview of Hadoop clients. OneFS contains no concept of a secondary namenode: Since every Isilon node functions as a namenode, the function of the secondary namenode—checkingpointing the internal namenode transaction log—is unnecessary.

The cluster load balances HDFS connections across all the nodes in the cluster. Because OneFS stripes Hadoop data across the cluster and protects it with parity blocks at the file level, any node can simultaneously serve datanode traffic as well as namenode requests for file blocks.

A virtual racking feature mimics data locality. You can, for example, create a virtual rack of nodes to assign compute clients to the nodes that are closest to a client's network switch if doing so is necessary to work with your network topology or to optimize performance.

Client computers can access any node in the cluster through dual 1 GigE or dual 10 GigE network connections. A SmartConnect license adds additional network resilience with IP address pools that support multiple DNS zones in a subnet as well as IP failover. In OneFS, an IP address pool appears as `subnet:poolname`. A best practice, which is discussed later in this paper, is to bind multiple IP addresses to each node interface in a Dell EMC Isilon SmartConnect™ network pool.

## NAMENODE REDUNDANCY

Every Isilon node acts as a namenode and a datanode. Because every node runs the OneFS HDFS service, every node can simultaneously serve namenode requests for file blocks and datanode traffic. A cluster thus inherently provides namenode redundancy as long as you follow the standard Isilon practice of setting your clients to connect to the cluster's SmartConnect zone's DNS entry. The result: There is no single point of failure.

SmartConnect uses a round-robin algorithm to distribute namenode sessions. When a Hadoop client first tries to connect to a namenode, OneFS routes the traffic to a node, which will serve as the client's namenode. The client's subsequent namenode requests go to the same node. When a second Hadoop client connects to the cluster's SmartConnect DNS entry, OneFS balances the traffic with, by default, round-robin, routing the connection to a different node than that used by the previous client. In this way, OneFS evenly distributes namenode connections across the cluster to significantly improve the performance of read-write intensive traffic like that of TeraSort.

If a node that a client is using as a namenode goes down, SmartConnect moves the IP address of the connection to another node, which then becomes the node that services namenode traffic for the Hadoop clients that had been connected to the failed node.

Although reassigning the namenode IP address to another node might temporarily disrupt the in-flight connections, the MapReduce job will continue. There may be tasks that need to be restarted, however.

## RACK AWARENESS

OneFS can contain a virtual rack of nodes to assign a pool of Hadoop compute clients to a pool of datanodes that are closest to the clients' main network switch. The virtual rack also ensures that OneFS does not route compute clients to unreachable datanodes in networks that lack full connectivity.

A virtual rack mimics data locality. The rack associates the clients' IP addresses with a pool of datanodes such that when a client connects to the cluster, OneFS assigns the connection to one of the datanodes in the pool, which is usually referred to in the following format `subnet:poolname`. In this way, a virtual rack can, for example, route a client's connection through its optimal network switch, which improves performance by reducing read throughput latency while minimizing traffic through a top-of-rack switch.

More specifically, the virtual rack simulates two datanodes in the rack and a third datanode in a different rack to optimize the arrangement of datanodes for your network switch topology. When a namenode returns three IP addresses to a Hadoop client's request to read data, the namenode selects the datanode. The OneFS HDFS daemon checks which client is connecting and then returns the IP addresses for the datanode and the secondary datanode from one rack and an IP address for the third datanode from another rack.

A Hadoop client, such as a Pivotal Data Computing Appliance, can use a virtual rack to connect to a node even when a networking switch fails. If, for example, a client connects to two networking switches, a main switch and a top-of-rack switch, a virtual rack ensures that the client can connect to a datanode even if the top-of-rack switch fails. In such a case, the client's connection comes into the Isilon cluster through the main switch.

A Hadoop client connects over HDFS to the datanodes with interfaces that are assigned to the pool. After you add a pool with SmartPools, you can change the allocation of IP address for clients that connect to the cluster. You could, for example, direct compute cluster 1 to S-series datanodes and compute cluster 2 to X-series datanodes with Isilon virtual racks. For more information, see the OneFS Command Reference or the OneFS Administration Guide.

The result is that virtual racks can help you obtain the performance that you expect from a Hadoop compute cluster. To support efforts to tune a OneFS cluster for compute performance, the OneFS daemon provides the following features:

- Ingests Hadoop-format rack-locality configurations
- Allows you to control which OneFS nodes are exposed as Datanodes to racks of clients
- Hands out datanodes to clients according to policies that you can set
- Attempts to handle old configuration values or misconfiguration gracefully

## THE HDFS ARCHITECTURE OF ONEFS

OneFS implements the server-side operations of the HDFS protocol on every node. The architecture employs a single thread pool with a daemon—named isi_hdfs_d—that allocates a thread to each HDFS connection to handle RPC calls to namenodes and read- write requests to datanodes.

A namenode resides on every node in the cluster. An HDFS client connects to a namenode to query or modify metadata in the OneFS file system. The metadata includes the logical location of data for the file stream—that is, the address of the datanode on which a block resides. An HDFS client can modify the metadata through the namenode's RPC interface. OneFS protects HDFS metadata at the same protection level as HDFS file data. In fact, OneFS handles all the metadata; you do not need to worry about managing it or backing it up.

With Isilon, the namenode daemon translates HDFS semantics and data layout into OneFS semantics and file layout. For example, the namenode translates a file's path, offset, and LEN into lists of block IDs and generation stamps. The namenode also translates a client's relative path request into a LIN and then returns to the client the address of a datanode and the location of a block.

A datanode stores blocks of files. More specifically, the datanode maps blocks to block data. With HDFS, a block is an inode-offset pair that refers to a part of a file. With OneFS, you can set the size of an HDFS block to optimize performance. A Hadoop client can connect to

a datanode to read or write a block, but the client may not write a block twice or delete a block. To transfer a block to a datanode, the HDFS client encapsulates the block in packets and sends them over a TCP/IP connection. The Isilon HDFS daemon performs zero-copy system calls to read and write blocks to the file system. On OneFS, the datanode reads packets from and writes packets to disk.

To manage writes, OneFS implements the same write semantics as the Apache implementation of HDFS: Files are append-only and may be written to by only one client at a time. Concurrent writes are permitted only to different files. As with the Apache implementation, OneFS permits one lock per file and provides a mechanism for releasing the locks or leases of expired clients.

# HDFS SETUP

The section describes the best practices that can help set up an Isilon cluster to work with Hadoop distributions, such as Apache Hadoop and Pivotal HD, to solve real-world problems in data analysis. This section does not, however, cover how to perform many of the tasks, such as creating a SmartConnect zone, that it refers to; the instructions are in the Dell EMC Isilon manuals for your version of OneFS.

## SUPPORTED DISTRIBUTIONS

A Dell EMC Isilon cluster is platform agnostic for compute, and there is no vendor lockin: You can run most of the common Hadoop distributions with an Isilon cluster, including Apache Hadoop, Hortonworks, Cloudera, and Pivotal HD. After you activate an Isilon Hadoop license, the cluster tries to automatically detect a client's Hadoop distribution.

Clients running different Hadoop distributions or versions can connect to the cluster simultaneously. For example, you can point both Cloudera and Pivotal HD at the same data on your Isilon cluster and run MapReduce jobs from both distributions at the same time.

A Dell EMC Isilon cluster running OneFS 7.0.2.2 or later works with the following Hadoop distributions and projects. Earlier versions of OneFS work with many of these distributions and projects, too; check with a Dell EMC Isilon representative.

**Supported Hadoop Distributions and Projects**

| | |
|---|---|
| Apache Hadoop 0.20.203 | Cloudera CDH4.2 |
| Apache Hadoop 0.20.205 | Cloudera Manager CDH4 |
| Apache Hadoop 1.0.0-1.0.3 | Greenplum HD 1.1 |
| Apache Hadoop 1.2.1 | Greenplum HD 1.2 |
| Apache Hadoop 2.0.x | Hortonworks/Apache 1.0.3 |
| Cloudera CDH3u2 | Pivotal HD 1.0.1 |
| Cloudera CDH3u3 | HAWQ 1.1.0.1 |
| Cloudera CDH3u4 | Apache HBase |
| Cloudera CDH3u5 | Apache Hive |
| | Pig |

OneFS 7.0.2.2 or later works with these Hadoop distributions and projects

## OVERVIEW OF HOW TO INTEGRATE AN ISILON CLUSTER WITH HADOOP

The process of installing a Hadoop distribution and integrating it with an Isilon cluster varies by distribution, requirements, objectives, network topology, security policies, and many other factors. The following overview assumes that you have installed a supported distribution of Hadoop and verified that it works.

Throughout this paper, the instructions and examples for configuring Hadoop abstract away from the Hadoop distributions—you might have to adapt the instructions and commands to your distribution, including translating the code in a configuration file to its corresponding field in a distribution's graphical user interface. For instructions on how to install and configure Hadoop on a compute client, see the documentation for your Hadoop distribution.

This overview also assumes that you have obtained an HDFS license from a Dell EMC Isilon representative and activated the license on the cluster. For more information about the OneFS commands in the following examples, see the OneFS CLI Administration Guide.

Most of the following steps are expanded on in later sections.

On your Isilon cluster, connect as root to any node in the cluster with SSH and create a directory dedicated to Hadoop data—for example:

```
pwd
/ifs
mkdir Hadoop
```

Add a text file to the directory so that when you return to your Hadoop clients after you finish the integration, you can confirm that you can connect to the cluster by seeing the file:

```
ls /ifs/hadoop
ThisIsIsilon.txt
```

Set the dedicated directory as the root path for HDFS connections:

```
isi hdfs --root-path=/ifs/hadoop
```

Create local users and groups on your Isilon cluster for Hadoop. The UIDs and GIDs of the local groups should exactly match those of the Hadoop user and group accounts on your Hadoop clients. The users and groups that you create depend on your needs, and you might have to add more users and groups than in the following example:

```
isi auth groups create --provider=local --name=hadoop --gid=489 --zone=system
isi auth users create mapred --zone=system --provider=local --uid=495 --primarygroup=hadoop --
homedirectory=/
ifs/hadoop/user/mapred --password=hadoop --enabled=true
isi auth users create hdfs --zone=system --provider=local --uid=501 --primary-group=hadoop --
homedirectory=/
ifs/hadoop/user/hdfs --password=hadoop --enabled=true
isi auth users create hadoop --zone=system --provider=local --uid=498 --primarygroup=hadoop --
homedirectory=/
 ifs/hadoop/user/hadoop --password=hadoop --enabled=true
```

You can verify that the groups and users were created by running the `isi auth users view` and `isi auth groups view` commands.

Make sure the directory that you created earlier for your Hadoop data is owned by the hadoop user and the hadoop group so that the hadoop user from the Hadoop cluster can read data from and write data to the directory.

```
cd /ifs
chown –R hadoop:hadoop Hadoop
```

Create a SmartConnect zone for the connections from your Hadoop clients. SmartConnect requires that you add a new name server (NS) record as a delegated domain to the authoritative DNS zone that contains the cluster. For instructions on how to set up a SmartConnect zone, see the OneFS guide.

On your Hadoop master, add the name of the DNS entry of the SmartConnect zone that you created for Hadoop to core-site.xml so that your Hadoop clients connect to a namenode with the DNS name of the zone. (Make sure to back up your core-site.xml file before you modify it; using a version control system to track changes to configuration files can make troubleshooting easier.) For example, if you named the DNS entry for the SmartConnect zone namenode.example.com, set the name as well as Port 8020 in the core-site.xml configuration file for your Hadoop distribution like this:

```
<?xml version="1.0"?>
<!-- core-site.xml -->
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://namenode.example.com:8020/</value>
<final>true</final>
</property>
</configuration>
```

Finally, on your Hadoop client, restart the Hadoop services as the hadoop user so that the changes to core-site.xml take effect.

```
su – hadoop
bin/stop-all.sh
bin/start-all.sh
```

Verify that your Hadoop client is now using the Isilon cluster as its data store:

```
bin/hadoop fs –ls /
```

You should see the text file that you created earlier, confirming the integration.

## SIZING AN ISILON CLUSTER FOR A HADOOP COMPUTE GRID

Finding an Isilon cluster's optimal point—the point at which it scales in processing MapReduce jobs and reduces runtimes in relation to other systems for the same workload—depends on the size of your compute grid, the number of jobs, the working datasets, and

other factors. The number of HDFS threads and the number of racks also play a role in sizing. Dell EMC Isilon recommends that you work with your Isilon representative to determine the number of nodes that will best serve your Hadoop workflow and your compute grid.

In general, nodes in the X-Series, such as the X400, deliver cost-effective capacity and performance for most Hadoop deployments. Hard-disk drives (HDDs) work well for MapReduce jobs that consist of large, sequential read operations. For datasets that consist of many small files, solid-state drives (SSDs) can improve performance.

With Hadoop, RAM matters. Hadoop jobs generally consist of many sequential reader threads. An Isilon cluster's large level-two cache, which is what most of a node's D-RAM is used for, supports MapReduce jobs with sequential reader threads. Each node that you add to the cluster increases the amount of the cluster's RAM and the size of the level-two cache, both of which can increase performance for Hadoop workflows.

## MAXIMIZING NETWORK THROUGHPUT WITH 10 GIGABIT ETHERNET

A cluster includes two networks: an internal InfiniBand network to exchange data between nodes and an external network to handle client connections. Each node includes redundant InfiniBand ports that enable you to add a second internal network in case the first one fails.

Clients reach the cluster with 1 GbE or 10 GbE. Every node includes Ethernet ports. The dual 10 Gigabit Ethernet connections on each Isilon node eliminate network bottlenecks. I/O-intensive operations like MapReduce jobs demand high-bandwidth, low-latency network cables, and switches. You should establish dual 10 GbE connections to each node to support the high levels of network utilization that take place during MapReduce jobs. You can also aggregate two 10 GbE links into a single logical 20 GbE link, which might benefit some workflows.

## DO NOT RUN THE NAMENODE AND DATANODE SERVICES ON CLIENTS

Because an Isilon cluster acts as the NameNode, Secondary NameNode, and DataNodes for Hadoop, these services should run only on the Isilon cluster, not on your compute clients. On your compute clients, the only daemons that you need to start are the jobtracker and tasktracker, which with most distributions of Hadoop can be done with the following script:

```
$HADOOP_HOME/bin/start-mapred.sh
```

## OVERVIEW OF ISILON HDFS COMMANDS

To manage the HDFS service on an Isilon cluster, you use the `isi hdfs` command. Running the command without options shows the values of the settings:

```
isi hdfs
block size: 4 KB
declared checksum type: NULL
hadoop authentication: permissive
log level: DEBUG
number of threads: auto
```

```
root path: /ifs/data/hadoop_long
version: attempt to support all clients
Rack Clients Datanode Pools
/default-rack { 0.0.0.0-192.0.2.2, 192.0.2.5-255.255.255.255 } { subnet0:BF_10gig, subnet0:BF_ext,
subnet0:SyncIQ, subnet0:VMware, subnet0:engineering, subnet0:pool0, subnet0:siq-longevity,
subnet0:test, subnet0:x400s }
```

For a description of the command's options, run the following command or see the OneFS Command-Line Administration Guide:

```
isi hdfs -h
isi hdfs: Command help
HDFS management interface
'isi hdfs' options are:
--force-version=<string> Force protocol version
--log-level=<string> Set log level [must be one of: EMERG, ALERT,
CRIT, ERR, WARNING, NOTICE, INFO, DEBUG]
--root-path=<string> Set the HDFS-exposed root path
--block-size=<size> Set the Hadoop block size used [4KB to 1GB]
--num-threads=<string> Tune the number of worker threads used [1 to 256
or 'auto']
--hadoop-security=<string> In enforcing mode, disallows insecure
authentication [must be one of 'enforcing',
'permissive']
--help, -h Print usage help and exit
'isi hdfs' sub-commands and [aliases] are:
racks [] Manage rack-local configuration
```

**krb5 [] Manage HDFS Kerberos configuration**

## THE HDFS SERVICE AND LOGS

Activating a Dell EMC Isilon HDFS license turns on the HDFS service. After you obtain a license key by contacting your Dell EMC Isilon representative, you activate it by running the `isi license activate` command followed by a license key. After you activate a license, you can verify that the HDFS service is running (the following output is abridged):

isi services Available Services:

```
    apache2             Apache2 Web Server                      Enabled
    isi cdate d         Compliance Clock Service                Disabled
    isi_hdfs_d          Hadoop FS Daemon                        Enabled
```

If the service is not running, turn it on:

isi services isi_hdfs_d enable

A Hadoop client negotiates an HDFS session with a namenode. When the client initiates communication with the namenode, the HDFS service selects the Hadoop distribution by default.

The HDFS service sends its log messages to /var/log/isi_hdfs_d.log:

cat /var/log/isi_hdfs_d.log

2013-06-20T11:18:08-07:00 <1.7> cp-dev-2(id2) isi_hdfs_d: isi_hdfs_d starting
2013-06-20T11:18:08-07:00 <1.5> cp-dev-2(id2) isi_hdfs_d: waiting for license activation 2013-06-20T11:49:30-07:00 <1.5> cp-dev-2(id2) isi_hdfs_d: license activated, starting
2013-06-20T11:49:30-07:00 <1.6> cp-dev-2(id2) isi_hdfs_d: Loaded libsasl "Cyrus SASL 2.1.23" v2.1.23.0
2013-06-20T11:49:30-07:00 <1.5> cp-dev-2(id2) isi_hdfs_d: Switching to process encoding of UTF-8 2013-06-20T11:49:30-07:00 <1.5> cp-dev-2(id2) isi_hdfs_d: started successfully

To change the log level to, for example, the warning level, run the following command:

isi hdfs --log-level=WARNING

## CREATING DIRECTORIES AND SETTING PERMISSIONS

OneFS controls access to directories and files with POSIX mode bits and access control lists. When you set up your directories, files, accounts, and permissions for Hadoop, you must make sure that they have the correct permissions so that users and applications can access their directories and files.

If you are not careful, problems can occur. For example, if the jobtracker is inadvertently run as root, it changes the tmp directory (for example, /ifs/Hadoop/tmp) to root ownership. When the jobtracker runs correctly as the mapred user, it can no longer write data to the tmp directory, and you end up with an error that looks something like this:

```
ERROR security.UserGroupInformation: PriviledgedActionException as:root
cause:org.apache.hadoop.ipc.RemoteException: org.apache.hadoop.mapred.SafeModeException: JobTracker
is in safe mode
```

The mapred user requires temporary space on HDFS when map jobs run. With Apache Hadoop, Hortonworks, and Pivotal HD, the /tmp folder is automatically created the first time a job is run and the mapred user is given permissions. For Cloudera, however, you must set up the /tmp directory manually and then assign it permissions, usually 1777, but the permissions depend on your security profile and other factors. If you inadvertently run the jobtracker as root, you might have to reset the permissions on the tmp directory.

Before deploying Hadoop, you should create the directories that you will need for Hadoop on the Isilon cluster. The directories that must be created and the permissions that must be assigned to them will vary by Hadoop distribution, environment, requirements, and security policies. Here is an example with loose permissions:

mkdir –p /ifs/data/hdfs (777)
mkdir –p /ifs/data/hdfs/user (1777)
mkdir –p /ifs/data/hdfs/user/history (1777) mkdir –p
/ifs/data/hdfs/user/gpadmin (1777)

mkdir –p /ifs/data/hdfs/tmp (1777) mkdir –p
/ifs/data/hdfs/yarn (1777) mkdir –p
/ifs/data/hdfs/apps (755)
mkdir –p /ifs/data/hdfs/apps/hbase/data (755) mkdir –p
/ifs/data/hdfs/hawq_data (777)
mkdir –p /ifs/data/hdfs/hive(1777)
mkdir –p /ifs/data/hdfs/hive/gphd/warehouse/ (1777)
Assigning directories with an open permission level of 777 makes troubleshooting easier if a problem occurs as you set up your Hadoop system. After you confirm that the cluster works for HDFS operations and MapReduce jobs, however, you should restrict the permissions as tightly as possible.

You must also create the user accounts that your Hadoop deployment requires. Again, the accounts that you need and their owner and group settings vary by distribution, requirements, security policies, and other factors. The profiles of the accounts on the Isilon cluster should match those of the accounts on your compute clients. Here are two examples of how to create the Hadoop accounts that you will need on OneFS:

isi auth users create --name="gpadmin" –-zone=system
isi auth groups create --name=hadoop --gid=489 --zone=system
For more information on the `isi auth` command, see the OneFS Command-Line Administration Guide.

Dell EMC Isilon recommends that the UIDs and GIDs for the Hadoop accounts are consistent across both the Isilon cluster and the clients in the compute grid. After you create your Hadoop users and groups, the owner and group information should initially look something like the following example:

```
drwxrwxrwx 3 root wheel 22B Aug 1 10:43 ..
drwxrwxrwt 3 yarn hadoop 22B Aug 28 12:20 yarn
drwxr-xr-x 2 hdfs hadoop 0B Aug 28 13:40 apps
drwxr-xr-x 2 hbase hadoop 0B Aug 28 13:40 apps/hbase
drwxrwxrwx 2 gpadmin hadoop 0B Aug 28 13:41 hawq_data
drwxrwxrwt 2 hdfs hadoop 0B Aug 28 14:04 tmp
drwxrwxrwx 8 root hadoop 136B Aug 28 14:05 .
```

```
drwxrwxrwt 4 mapred hadoop 50B Aug 28 14:23 user
drwxrwxrwt 4 gpadmin hadoop 50B Aug 28 14:23 user/gpadmin
```

**drwxrwxrwt 3 hdfs hadoop 22B Aug 28 15:57 hive**

### SETTING THE ROOT PATH

By default, a Hadoop client connects to the cluster's root directory: ifs. To change the root directory for Hadoop clients, run the following command, replacing the path in the example with the path that you want. In general, it is a best practice to set the root path before you configure a Hadoop distribution like Pivotal HD to work with OneFS.

```
isi hdfs --root-path=/ifs/hadoop
```

After you set the root path, you should also make the directory available as an export or share for NFS and SMB clients; see the OneFS guide for instructions.

OneFS can translate symbolic links for HDFS. The HDFS root can be /ifs/hadoop, for instance, but then a symlink can point from

/ifs/hadoop/data to /ifs/data to isolate the Hadoop application directories (/tmp, /mapred, /yarn, /hbase) from your actual data while keeping it accessible over HDFS.

### SETTING UP A SMARTCONNECT ZONE FOR NAMENODES

For namenode redundancy, you set up a SmartConnect zone in OneFS, create a DNS entry for the zone, and then set your Hadoop clients to connect to a namenode with the DNS name of the zone. The zone policy balances the namenode connections among all the nodes in the cluster.

For information about how namenodes and datanodes handle traffic on OneFS and for an explanation of why you should set up a SmartConnect zone for namenodes but not for datanodes, see Handling namenode and datanode failure scenarios.

SmartConnect requires that you add a new name server (NS) record as a delegated domain to the authoritative DNS zone that contains the cluster. For instructions on how to set up a SmartConnect zone, see the OneFS guide.

After you get the DNS name of your SmartConnect zone, set the default file system in core-site.xml to the name of the zone's DNS entry. (Make sure to back up your core-site.xml file before you modify it.) If you named the DNS entry for the SmartConnect zone

namenode.example.com, for instance, you set the name as well as Port 8020 in the core-site.xml configuration file for your Hadoop distribution like this:

```
<?xml version="1.0"?>
<!-- core-site.xml -->
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://namenode.example.com:8020/</value>
<final>true</final>
</property>
</configuration>
```

The value of fs.default.name specifies the default file system that Hadoop clients connect to. HDFS clients must access a Dell EMC Isilon cluster through Port 8020. After you modify core-site.xml, you must restart the Hadoop services on the client for the changes to take effect. For more information on fs.default.name, see the documentation for your Hadoop distribution.

## WORKING WITH DIFFERENT HADOOP DISTRIBUTIONS

This section lists solutions to common problems that can interfere with integrating an Isilon cluster with a compute grid running a Hadoop distribution such as Pivotal HD or Cloudera.

For all the distributions, do not run the NameNode, DataNode, and Secondary NameNode services on your compute clients; OneFS provides those services.

### CLOUDERA

#### Turning on checksums for Cloudera

To run Cloudera with an Isilon cluster, you must be running OneFS 7.0.2.2. Because Cloudera includes a default setting in hdfs- site.xml to support the WRT checksum type on datanodes, you must run the following OneFS command to support Cloudera:

```
isi hdfs --checksum-type=crc32
```

### PIVOTAL HD

Pivotal provides documentation on how to set up Pivotal HD to work with an Isilon cluster. To obtain the documentation, contact Pivotal.

When you run Pivotal HD with an Isilon cluster, you must remember to change the variables for namenodes in core-site.xml to point to the name of the SmartConnect zone for the Isilon cluster. You must also remember to edit

`/home/gpadmin/ClusterConfigDir/yarn/yarn-site.xml` to turn off Yarn log aggregation by setting its value to `false` so it looks like this:

```
<property>
<name>yarn.log-aggregation-enable</name>
<value>false</value>
<description>Whether to enable log aggregation
</description>
</property>
```

## TUNING ONEFS FOR HDFS OPERATIONS

This section describes strategies and options for tuning an Isilon cluster to improve performance for Hadoop data sets, workflows, and workloads.

### BLOCK SIZES

On an Isilon cluster, raising the HDFS block size from the default of 64 MB to 128 MB optimizes performance for most use cases. Boosting the block size lets Isilon nodes read and write HDFS data in larger blocks, which can decrease drive-seek operations and increase performance for MapReduce jobs.

To set the server-side HDFS block size, connect with SSH to any node in the cluster and run the `isi hdfs` command with the block-size option and a value from 4 KB to 1 GB—for example:

```
isi hdfs --block-size=128MB
```

You can also change the client-side block size on your Hadoop clients with the dfs.block.size parameter. On a client, you set the block size in the hdfs-site.xml configuration file with the following property:

```
dfs.block.size
```

Apache Hadoop 2.0 and CDH4 renamed the property to the following:

```
dfs.blocksize
```

The default of client block size is also 64 MB. The client-side block size and the server-side block size, however, can differ.

The client-side block size determines how an HDFS client writes a block of file data to the cluster. During a file-create operation, an HDFS client broadcasts its block size to the namenode, and subsequent writes to the file on a datanode take place with the same block size. For most workflows, setting the client's block size to 128 MB works well. As the optimal block size depends on your data, how you process your data, and other factors, you can tune performance by testing different client-side block sizes.

The server-side block size determines how the OneFS HDFS daemon returns data to read requests. The HDFS clients must contain enough memory to accept the server-side block size. If your server-side block size, for example, is set to 512 MB and if a client's maximum

map JVM size is set to 512 MB, the client will be unable to pull files from the cluster. The server-side block size also controls the number of map tasks and the file input size of each map task.

## TUNING THE NUMBER OF HDFS THREADS

The OneFS HDFS service runs a daemon named isi_hdfs_d that handles traffic from namenodes and datanodes. With its default setting of auto, the multithreaded daemon spawns as many as 30 threads on a node.

To support a large system of compute clients, however, you might need to increase the number of threads. Assuming that you are distributing namenode traffic across all the nodes in a cluster by using a SmartConnect zone, the total number of OneFS HDFS threads should equal at least half of the total number of map and reducer slots in your compute grid.

Here is an example that illustrates how to calculate the number of threads to set. If your compute grid contains 80 compute nodes with 10 map slots and 5 reduce slots each, your Isilon cluster would need at least 600 threads:

(80 * (10 + 5)) / 2 = 600

If your Isilon cluster contains 25 nodes, the minimum number of threads per node would equal 24: 600 / 25 = 24

Convention, however, dictates that you set the number of threads to a power of 2 (8, 16, 32, 64, 128, 256). So, for an example cluster of 25 nodes handling requests from 600 compute slots, you would set the value to 32 threads:

```
isi hdfs --num-threads=32
```

OneFS limits the number of threads to 256.

## OBTAINING STATISTICS TO TUNE ONEFS FOR A HADOOP WORKFLOW

With OneFS 6.0 or later, you can run the isi statistics command to obtain statistics for client connections, the file system, and protocols. To view HDFS protocol statistics, for example, you can run the `isi statistics` command as follows. For more information about the statistics commands and their options, see the OneFS Command-Line Administration Guide.

```
isi statistics pstat --protocol=hdfs
```

To obtain statistics for all the protocols, execute the following command, which can help you determine whether your MapReduce job is running:

```
isi statistics system --nodes --timestamp --noconversion --interval=30
```

When the nodes are actively handling HDFS connections, the output looks something like this:

```
Timestamp Node   CPU   SMB   FTP  HTTP ISCSI   NFS        HDFS       Total       NetIn      NetOut      DiskIn     DiskOut
          s  LNN %Used   B/s   B/s   B/s   B/s   B/s         B/s         B/s         B/s         B/s         B/s         B/s
1383938796.7  All  54.8   0.0   0.0   0.0   0.0   0.0 200848112.0 200848112.0 113712138.8 113873594.4 135343547.7 119578794.7
1383938796.7    1  67.1   0.0   0.0   0.0   0.0   0.0  80221048.0  80221048.0  32163148.8  20050643.2  23927398.4  21515776.0
1383938796.7    2  61.9   0.0   0.0   0.0   0.0   0.0         0.0         0.0  15856857.6  25100025.6  31218688.0  28093644.8
1383938796.7    3  56.0   0.0   0.0   0.0   0.0   0.0  91685468.0  91685468.0   8607364.8  48302576.0  21400064.0  19517030.4
1383938796.7    4  26.7   0.0   0.0   0.0   0.0   0.0         0.0         0.0   7468484.8  11882696.8  30017365.3  26413738.7
1383938796.7    5  62.3   0.0   0.0   0.0   0.0   0.0  28941596.0  28941596.0  49616282.8   8537652.8  28780032.0  24038604.8
```

By analyzing the columns titled NetIn and NetOut, you can determine whether your HDFS connections are predominantly reading or writing data. Looking at the distribution of input and output across all the nodes shows whether Hadoop is using all the nodes for a MapReduce job.

To obtain an HDFS protocol report that totals the statistics by operation, run the following command as one line. In the report that this command generates, the read and write operations take place on datanodes; all the other operations take place on namenodes. You should be able to see the average file size Hadoop uses by looking at the values for InAvg (for read) and OutAvg (for write)— values that show the average bytes transmitted per remote procedure call (RPC). If the files are large, the values of InAvg and OutAvg should be near the block size, which is 64 M by default.

```
isi statistics protocol --protocols=hdfs --long -- output=TimeStamp,NumOps,In,InAvg,Out,OutAvg,TimeMin,TimeMax,TimeAvg,Op --totalby=Op
--noconversion --interval=30
```

To obtain an HDFS protocol report that totals the statistics by node, run the following command as one line:

```
isi statistics protocol --protocols=hdfs --long --output=TimeStamp,In,Out,Op --totalby=node -- noconversion --interval=30
```

You can use the report that the following command creates to check if some nodes are responding to requests slowly and then identify why by looking at such columns as node.cpu.user.avg and node.cpu.sys.avg. The resulting column that is titled node.disk.busy.avg can reveal whether CPUs are a bottleneck. In addition, you should also make sure that the values for node.cpu.sys.avg are much greater than node.cpu.user.avg; otherwise, the Isilon HDFS daemon (isi_hdfs_d) might be calculating checksums when in fact it should not be.

```
isi_for_array 'isi statistics query -- stats=node.health,node.cpu.user.avg,node.cpu.sys.avg,node.cpu.idle.avg,node.memory.used,node.memory
.free,node.disk.busy.avg --nofooter --interval=30'
```

To see a report on the drives on all the nodes, run the following command.

```
isi statistics drive --nodes=all --timestamp --noconversion --interval=30
```

The output looks something like the following. These statistics, which can reveal whether disk input/output operations are creating a bottleneck, can help you tune the cluster to improve performance for MapReduce jobs.

```
Timestamp       Drive Type OpsIn          BytesIn OpsOut       BytesOut TimeAvg TimeInQ Queued Busy s
                LNN:bay                      N/s         B/s        N/s         B/s         ms      ms
                %
1383940353.7      1:1   SAS  23.8  1312563.2   38.2  2369331.2     1.6      1.4     0.2  11.8
1383940353.7      1:2   SAS  14.6   708608.0   50.2  2289049.6     1.7      3.3     0.3  14.0
1383940353.7      1:3   SAS  23.6  1400832.0   43.6  1817190.4     1.5      3.2     0.3  11.2
…
1383940353.7     2:12   SAS  25.0  1403289.6   12.2   763699.2     1.1      1.1     0.8   4.1
1383940353.7      3:1   SAS  30.2  1746739.2   28.6  1836646.4     1.7      1.4     0.2  11.3
1383940353.7      3:2   SAS  53.2  2735513.6   46.8  1322188.8     1.5      4.5     0.2  12.4
1383940353.7      3:3   SAS  31.6  1727795.2   44.2  2811494.4     1.6      1.7     0.3  14.9
…
1383940353.7     3:12   SAS  52.6  3201433.6   82.0  1915289.6     1.0      3.9     0.5  13.4
1383940353.7      4:1   SAS  58.8  3493068.8   36.0  1825382.4     1.6      2.2     0.3  14.9
1383940353.7      4:2   SAS  23.0  1361510.4   48.8  1541939.2     1.6      3.0     0.3  12.9
1383940353.7      4:3   SAS  33.2  1948057.6   24.8  1509171.2     1.8      1.8     0.2  11.5
…
1383940353.7     4:12   SAS  37.2  2058444.8   75.8  1502617.6     1.1      4.0     0.7  12.7
1383940353.7      5:1   SAS  39.4  2193817.6   51.0  2087321.6     1.4      4.7     0.5  12.7
1383940353.7      5:2   SAS  27.4  1417420.8   45.2  1792409.6     1.5      3.8     0.2   9.9
1383940353.7      5:3   SAS  17.8   944332.8   34.4  2131558.4     1.8      1.3     0.2  11.9
…
1383940353.7     5:12   SAS  55.2  3110604.8   65.0  2113536.0     1.3      2.9     0.6  19.1
```

To see a report that shows the active and the connected HDFS clients, run the following command as one line. This report can check whether clients are connecting to every node during a MapReduce job.

```
isi statistics query --nodes=all
--stats=node.clientstats.active.hdfs,node.clientstats.connected.hdfs
--nofooter --noconversion --interval=30
```

The output looks like this:

```
NodeID node.clientstats.active.hdfs node.clientstats.connected.hdfs
     1                            2                               23
     2                            6                               32
     3                            3                               24
     4                            2                               29
```

5 3 36

If your MapReduce jobs seem to be bogging down, you can check whether blocking is an issue. Other users or applications might be locking files to write data to them while a MapReduce job tries to process the files.

isi statistics heat --totalby=node,event --orderby=node,event --noconversion --interval=30

The output looks something like this:

```
   Ops    Node        Event               Class Path N/s
   8.3       1      blocked                     * *
   5.2       1    contended                     * *
   0.6       1         link                     * *
 777.2       1         lock                     * *
 463.2       1       lookup                     * *
 153.2       1         read                     * *
 172.1       1       rename                     * *
  45.3       1      setattr                     * *
   1.5       1            *                     * *
  22.4       2      blocked                     * *
   1.4       2    contended                     * *
 778.3       2         lock                     * *
 342.7       2       lookup                     * *
 314.3       2         read                     * *
  68.7       2       rename                     * *
  24.1       2      setattr                     * *
   0.5       2            *                     * *
  53.4       3      blocked                     * *
   4.6       3    contended                     * *
 979.5       3         lock                     * *
 284.4       3       lookup                     * *
 507.4       3         read                     * *
  67.3       3       rename                     * *
  25.1       3      setattr                     * *
   0.6       3            *                     * *
  11.6       4      blocked                     * *
   3.7       4    contended                     * *
   0.1       4         link                     * *
 741.8       4         lock                     * *
 346.2       4       lookup                     * *
 182.9       4         read                     * *
  75.2       4       rename                     * *
  27.2       4      setattr                     * *
   0.4       4            *                     * *
  25.8       5      blocked                     * *
   5.1       5    contended                     * *
   0.1       5         link                     * *
 885.4       5         lock                     * *
 383.1       5       lookup                     * *
 351.3       5         read                     * *
  78.3       5       rename                     * *
  28.4       5      setattr                     * *
   0.7       5            *                     * *
```

## STORING INTERMEDIATE JOBS ON AN ISILON CLUSTER

Storing intermediate map results on an Isilon cluster can solve two problems: performance and scalability. A Hadoop client typically stores its intermediate map results locally. The speed of the local storage, however, can slow down the performance of Hadoop tasks. The amount of local storage can also affect the client's ability to run some jobs. You could, of course, increase the space on the compute node by adding more drives, or you could compress the map results to reduce the amount of data that the client stores locally.

Another option is to offload the intermediate map results to your Isilon cluster--which is optimized to handle the large amounts of data that large jobs like TeraSort generate. You can offload the intermediate map results to an Isilon cluster by changing Hadoop's `mapred.local.dir` parameter for the tasktracker in mapred-site.xml. You might be able to boost performance by exporting a dedicated directory for intermediate map results and then creating subdirectories for each Hadoop client in it--for example:

/ifs/compute
/ifs/compute/client1
/ifs/compute/client2
/ifs/compute/clientn

Because this strategy leads to more network transfers, you should also turn on compression for mapred output to reduce the data that is moved into and out of the Isilon cluster. To turn on compression, set Hadoop's `mapred.compress.map.output` parameter to `true`.

To optimize performance for large jobs, you can also set the access pattern for the directories on OneFS to streaming and consider setting the level of data protection to N+1. Although the low protection level risks losing intermediate map results if a node fails, the risk is low if you can restart the intermediate job.

## DEALING WITH SHUFFLE SPACE BOTTLENECKS

Map tasks write output to a circular memory buffer, which is 100 MB by default. When the output in the buffer exceeds 80 percent by default, the contents of the buffer begin overflowing into a shuffle space on the local disk. The task continues to write output into the buffer even as data spills out of it and into the shuffle space. If the buffer fill ups during the spill, the map task will be halted until the overflowing data finishes spilling onto disk.

The problem is that with disk-starved tasktrackers, such as with a deployment that uses Sergengeti and VMware, the shuffle space on disk can become a bottleneck that undermines overall job performance. One solution might be to place the shuffle space on an Isilon cluster by altering `mapred.local.dir` to point to an NFS mount. If your jobs are bogging down because tasks are overflowing the memory buffer and spilling data into the shuffle space on disk more than once, you should test whether storing the data on an Isilon cluster improves overall job performance.

## FEDERATION WITH DAS AND NAS

This section illustrates how to configure Apache Hadoop to point to two namespaces and then aggregate them under the Hadoop virtual file system, viewFS. The capability of HDFS version 2 to point to two namespaces is a major improvement over HDFS version 1, which could use only one namespace. Pointing to two namespaces lays the foundation for simpler migrations, tiering, and other storage profiles.

The following setup assumes you have two DAS-based namenodes and a compute cluster with three datanodes running Apache Hadoop. The example also assumes that you have an Isilon cluster configured for Hadoop, including Hadoop user and group accounts.

Keep in mind that most of the settings in the following snippets are examples; your properties, directories, names, and values will be different. On your clients, you will have to restart the services that are affected by these changes, especially dfs.

First, add the following example code to hdfs-site.xml and copy it to all the clients:

```
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.permissions</name>
<value>false</value>
</property>
<property>
<name>dfs.federation.nameservices</name>
<value>ns1,ns2</value>
</property>
<property>
<name>dfs.namenode.rpc-address.ns1</name>
<value>apache2-1:9001</value>
</property>
<property>
<name>dfs.namenode.rpc-address.ns2</name>
<value>apache2-2:9001</value>
</property>
```

Second, create the subdirectories on the DAS namenodes that will be used as mountpoints. On DAS namenode number 1, make a subdirectory for home and another for a temporary directory:

hadoop fs -mkdir hdfs://apache2-1:9001/home hadoop fs -mkdir
hdfs://apache2-1:9001/tmp
On DAS namenode number 2, make a home directory:

hadoop fs -mkdir hdfs://apache2-2:9001/home
Third, modify core-site.xml as follows to add the name of the SmartConnect zone for your Isilon cluster:

```
<property>
<name>fs.defaultFS</name>
<value>viewfs:///</value>
</property>
<property>
<name>fs.viewfs.mounttable.default.link./NN1Home</name>
<value>hdfs://apache2-1:9001/home</value>
</property>
<property>
<name>fs.viewfs.mounttable.default.link./NN2Home</name>
<value>hdfs://apache2-2:9001/home</value>
</property>
<property>
<name>fs.viewfs.mounttable.default.link./tmp</name>
<value>hdfs://apache2-1:9001/tmp</value>
</property>
<property>
<name>fs.viewfs.mounttable.default.link./isilon</name>
<value>hdfs://Isilon.example.com:8020/</value>
</property>
```

Finally, distribute the core-site.xml file to all your clients and then restart the services by running sbin/stop-all.sh and sbin/start- all.sh

## STORAGE POOLS FOR DIFFERENT DATA SETS

A difficulty arises if you are analyzing two different data sets, one with large files and another with small files. In such a case, you might be able to cost-effectively store and optimally analyze both types by using storage pools. Storage pools let you group different files by attributes and then store them in different pools of storage: Small files can be routed by a OneFS policy to SSDs, for instance, and large files can be routed to X-Series nodes. And then you can use OneFS SmartConnect zones to associate a set of compute clients with each data set to optimize the performance of the MapReduce jobs that analyze each set. For more information, see Align datasets with storage pools. You could, for example, direct one compute cluster to S-series datanodes and another compute cluster to X-series datanodes with Isilon virtual racks; see Rack awareness.

## DATA PROTECTION BEST PRACTICES

OneFS takes a more efficient approach to data protection than HDFS. The HDFS protocol, by default, replicates a block of data three times to protect it and to make it highly available. Instead of replicating the data, OneFS stripes the data across the cluster over its internal InfiniBand network and protects the data with forward error correction codes.

Forward error correction, or FEC, is a highly efficient method of reliably protecting data. FEC encodes a file's data in a distributed set of symbols, adding space-efficient redundancy. With only a part of the symbol set, OneFS can recover the original file data. Striping data with FEC codes consumes much less storage space than replicating data three times, reducing the number of drives by a factor of 2.5. Striping also lets a Hadoop client connecting to any node take advantage of the entire cluster's performance to read or write data.

If you set the replication level from an HDFS client, OneFS ignores it and instead uses the OneFS protection level that you set for the directory or the file pool that contains your Hadoop data.

In general, a best practice for most environments entails using the default level of protection—typically, N+2:1. For larger clusters, you can set the level slightly higher, for example, to +2.

For most Hadoop workflows—in which optimizing for an aggregation of workloads takes precedence over optimizing for a single job— you should consider using +2:1 for clusters with fewer than 18 nodes. For clusters with more than 18 nodes, consider using +2.

An analytics workflow of many small files, however, might benefit from using mirroring instead of forward error correction codes, because the overhead for FEC codes is similar to what is used for mirroring small files anyway. In addition, mirroring might help boost performance, but the gain in performance might be at the expense of capacity.

In general, lower protection levels yield better throughput, but the amount varies by type of operation. Random write performance receives the biggest performance gain from using a lower protection level. Streaming writes, streaming reads, and random reads receive a smaller performance boost from lower protection levels.

The protection level that you choose depends on a variety factors, including the number of nodes in the cluster. The right protection level for you might also depend on which version of OneFS you are running. Because so many variables interact to determine the optimal protection level for an environment, a best practice is to consult with an Isilon representative about selecting a protection level. Isilon professional services can analyze the cluster and suggest an optimal protection policy.

### ALIGNING WORKLOADS WITH DATA ACCESS PATTERNS

Once you identify the dominant access pattern of your Hadoop workflow—concurrent, streaming, or random—you can optimize how OneFS lays out data to match it. By default, OneFS optimizes data layout for concurrent access. With Hadoop, however, the dominant data access pattern might be streaming—that is, the pattern has lower concurrency, higher single-stream workloads.

You can set OneFS to stripe data for streaming access patterns to increase sequential-read performance for MapReduce jobs. To better handle streaming access, OneFS stripes data across more drives and prefetches data well in advance of data requests.

Streaming is most effective on directories or subpools serving large files or handling large compute jobs.

If your workload is truly random, the random access pattern, which prefetches no data, might increase performance.

**Table 1**

| ONEFS ACCESS PATTERN | DESCRIPTION |
|---|---|
| Concurrent | Prefetches data at a medium level |
| Streaming | Prefetches data far in advance of requests |
| Random | Prefetches no data |

OneFS Access Pattern Options

With a SmartPools license, you can apply a different access pattern to each storage pool that you create. Selecting the access pattern enables you to set the optimal pattern for each dataset; for more information, see Align datasets with storage pools.

## SMARTCONNECT FOR HDFS

By default, the OneFS SmartConnect module balances connections among nodes by using a round-robin policy with static IP addresses and one IP address pool for each subnet. A SmartConnect Advanced license adds advanced balancing policies and lets you define IP address pools to support multiple DNS zones in a subnet. The licensed version supports dynamic IP failover to provide continuous access to namenodes when hardware or a network path fails. The rest of this section assumes that you have the licensed version of SmartConnect Advance in place.

If a node or one of its interfaces fails, SmartConnect reassigns the dynamic IP addresses to the remaining interfaces in the pool. OneFS then routes connections to the newly assigned member interface without interrupting access. When the failed interface or the failed node comes back online, OneFS updates the list of online pool members and redistributes the dynamic IP addresses across the list. A best practice is to employ dynamic IP failover to prevent interrupting client connections to namenodes but to use static IP addresses for datanodes; see Setting up a SmartConnect zone for namenodes.

All the nodes in a cluster can be configured to appear as a single network host (that is, a single fully qualified domain name, or FQDN, can represent the entire cluster), or one node can appear as several discrete hosts (that is, a cluster can have more unique FQDNs than nodes).

SmartConnect offers several connection-balancing strategies:

- **Zoning:** You can optimize storage performance by designating zones to support workloads or clients. For large clusters, you can use zoning to partition the cluster's networking resources and allocate bandwidth to each workload, which minimizes the likelihood that heavy traffic from one workload will affect network throughput for another. Zoning can help you segment a cluster for two disparate Hadoop workflows.
- **Inclusion and exclusion:** You can add node interfaces and interface types to SmartConnect pools. For example, in an inclusive approach, a pool includes all the 1 GbE interfaces in the cluster, regardless of node type. An exclusive strategy, by contrast, limits membership to only one interface type from one node type—for example, only 10 GbE interfaces on S-Series nodes. As a result, clients connecting to a pool receive similar performance from any node in the pool.

The licensed version of SmartConnect provides four policies for distributing traffic across the nodes in a network pool. Because you can set policies for each pool, static and dynamic SmartConnect address pools can coexist. The four policies are as follows:

- **Round robin:** This policy, which is the default, rotates connections among IP addresses in a list. Round robin is the best balancing option for most workflows.
- **CPU usage:** This policy examines average CPU usage in each node and then attempts to distribute the connections to balance the workload evenly across all the nodes.
- **Network throughput:** This policy evaluates the overall average network throughput per node and assigns new connections to the nodes with the lowest relative throughput in the pool.
- **Connection count:** This policy uses the number of open TCP connections on each node in the pool to determine which node a client will connect to.

For an HDFS workload, round robin is likely to work best. If you find that round robin does not suit your workflow, you can experiment with the other options to improve performance.

## HANDLING NAMENODE AND DATANODE FAILURE SCENARIOS

To gracefully handle namenode and datanode failures, you should set up a dynamic pool of IP addresses with a SmartConnect zone name for namenodes and a static pool of IP addresses without a SmartConnect zone name for datanodes.

First, a gloss on how OneFS handles requests:

1. The job tracker on a compute client resolves the default name of the file system to an IP address by using DNS:

```
<!-- core-site.xml -->
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://example.com:8020/</value>
<final>true</final>
</property>
</configuration>
```

2. The job tracker makes a namenode request to the IP address that DNS returned and initiates either a read or a write sequence.

For a read sequence, the job tracker requests the block locations of the data. OneFS returns a list of three IP addresses from the list defined in the SmartConnect pool for HDFS. OneFS sends back three IP addresses to mimic a traditional Hadoop file system, in which every block is replicated three times.

For writes, the job tracker requests a location to write its chunk of data. OneFS returns a list of three IP addresses that the job tracker can provide to tasks for writing data.

3. For both read and write requests, the job tracker provides the list of three IP addresses to a task tracker on the compute node.
4. The task tracker initiates a TCP connection with the first IP address in the list for that block, using the HDFS data-node protocol.
5. The task performs either a read or a write over the data-node protocol to the first IP address in the list.
6. Once done, the task reports back to the job tracker.

**Assign Dynamic IP Addresses with a SmartConnect Zone Name to NameNodes**

When a job tracker or a task tracker detects a failure to communicate with an Isilon cluster, the tracker retries the connection by using the same IP address for namenode requests. If the IP address is dynamic, the retry succeeds because OneFS automatically moves the IP address to another node--one that is available and processing requests.

#### Assign Static IP Addresses without a Zone Name to Datanodes

For datanodes, however, static IP addresses work best. HDFS already gracefully handles read and write failures by reissuing a failed request to another datanode. Thus, dynamic IP addresses deliver no advantage over static IP addresses. In fact, even when no failure occurs, assigning static IP addresses to datanodes can break a datanode session if you dynamically rebalance the IP addresses or if you set failback mode to automatic to forcibly move IP addresses to a node that comes back online.

#### Failure scenarios

The failure scenarios discussed in this section elucidate how OneFS processes HDFS requests and routes them among nodes.

When a job tracker submits a namenode request, the OneFS HDFS daemon returns a sliding list of three IP addresses; the sliding list balances connections across the cluster by responding to the first request with, for example, the IP addresses of nodes one, two, and three; the second request then responds with the IP addresses on nodes two, three, and four, and so forth. Because the task tracker usually ends up connecting to the first IP address in the list, this approach effectively balances connections across all the nodes with round robin.

In a failure scenario such as a node going offline during a read or write operation, the task running on the task tracker retries its connection after a short timeout to the next IP address in the list. If all three of the IP addresses do not respond, the task fails and job tracker is notified. At that point, a second series of retries obtains a new list of block locations. The job tracker, however, places the IP addresses that failed in a blacklist so that they are not retried. (For more information on the blacklist and how to clear it, see the section on dealing with a blacklisted tasktracker in *Hadoop Operations*, published by O'Reilly.)

In the case of a partially completed write operation, the entire block of data, usually either a 64 MB block or a 128 MB block, is discarded, and the write request is reissued to a working datanode. In the case of a partially completed read operation, a resume mechanism redirects the read operation to another datanode without the whole read operation having to start anew.

If you were using dynamic IP addresses for the datanodes and a node went offline, the IP address of the datanode would move to another node, and any task or session running on the datanode would fail. The tasktracker would reissue the session to the next IP address in the list. Although this scenario is similar to what would happen with static a IP address, there are two problems thrown into the mix if you assign dynamic IP addresses to the datanodes:

1. If you were to run a job to dynamically rebalance the IP addresses, the datanode sessions would break even though no failure actually occurred.
2. If you were to set failback mode to automatic, when the node came back online, OneFS would move IP addresses back to it, interrupting the datanode sessions associated with those IP addresses.

As a result, static IP addresses, not dynamic IP addresses, are recommended for datanode transactions.

The situation is different with namenodes. The jobtracker and tasks send remote procedure calls (RPCs) to a namenode to obtain metadata, such as the location of a block or a confirmation of a write operation. Because such sessions tend to be short, the likelihood that a dynamic IP address would cause issues is smaller than it is with datanodes. In contrast, using static IP addresses with namenodes can more likely lead to job and task failures than dynamic IP addresses.

As a result, dynamic IP addresses are recommended for namenode transactions.

#### Solution

Here is how to set up dynamic IP addresses for namenodes and static IP addresses for datanodes. First, create a SmartConnect zone for namenodes with IP addresses to cover all the interfaces of all the nodes in the cluster (see the section on optimizing the number of IP addresses below). Set load-balancing to round robin. (Do not configure the namenode SmartConnect zone with the isi hdfs --ip- pool command; that command is for the datanode IP address pool.) Point your HDFS configuration at this zone name in core- site.xml, not at an IP address—for example:

```
<!-- core-site.xml -->
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://namenode.example.com:8020/</value>
<final>true</final>
</property>
</configuration>
```

Now when the job tracker or the task trackers resolve hdfs://namenode.example.com, they receive a dynamic IP address from SmartConnect and connect to the cluster with a dynamic IP address. When the jobtracker detects a failure to communicate with a namenode, it retries its namenode requests with the same IP address, and if that address is dynamic, the retry will succeed.

Second, create an IP address pool for datanode traffic with exactly the same number of IP addresses as all the interfaces that are associated with pool. You do not need to give the IP address pool for datanodes a SmartConnect zone name because OneFS does not use SmartConnect for datanode traffic. Instead, configure this pool with following command. For more information on the command, see the OneFS Command-Line Administration Guide.

```
isi hdfs --add-ip-pool=subnet0:datanode0
```

Now, once the jobtracker communicates with the cluster over the dynamic IP addresses, the cluster returns three IP addresses from the datanode pool of static IP addresses that you set with the isi hdfs --ip-pool command. Since Hadoop performs retries by using the next IP address in the list of three addresses, the static IP addresses ensure that the retry connects to an available node that acts as a datanode.

## USE MULTIPLE IP ADDRESSES PER INTERFACE

This approach consists of binding multiple IP addresses to each node interface in a SmartConnect pool for namenodes. The ideal number of IP addresses per interface depends on the size of the pool. The following recommendations apply to dynamic pools only. Because static pools include no failover capabilities, a static pool requires only one IP address per interface.

Generally, you can achieve optimal balancing and failover when the number of IP addresses allocated to the pool equals N * (N – 1), where N equals the number of node interfaces in the pool.

If, for example, a pool contains five node interfaces, the optimal IP address allocation for the pool is 5 * (5 – 1) = 20 IP addresses. The equation allocates four IP addresses to each of the five-node interfaces in the pool.

For a namenode SmartConnect pool with four node interfaces in which the N * (N – 1) model is followed and results in three unique IP addresses being allocated to each node, a failure on one node interface results in each of that interface's three IP addresses failing over to a different node in the pool, ensuring that each of the three active interfaces remaining in the pool receives one IP address from the failed node interface. If client connections to that node were evenly balanced across its three IP addresses, SmartConnect distributes the workloads to the remaining pool members evenly.

Assigning each workload a unique IP address allows SmartConnect to move a workload to another interface, minimizing the additional workload that a remaining node in the pool must absorb and ensuring that SmartConnect evenly distributes the workload across the surviving nodes in the pool.

If you cannot use multiple IP addresses per interface—because, for example, you hold a limited number of IP addresses or because you must ensure that an interface failure within the pool does not affect performance for other workloads using the same pool—you should consider using a hot standby interface per pool.

## SERVICE SUBNETS AND DNS INTEGRATION

When you create a SmartConnect pool, you should set up a service subnet for DNS forwarding. A service subnet is the name of the external network subnet whose SmartConnect service responds to DNS requests for the IP address pool. For each pool, only one service subnet can answer DNS requests. As a best practice, you should designate a service subnet for each SmartConnect Advanced pool. You should also create a delegation for the subnet's service IP address on the DNS servers that provide name resolution services to the clients.

If you leave the service subnet option blank, OneFS ignores incoming DNS queries for the pool's zone name and restricts access to only the IP addresses that you enter manually. In addition, you cannot apply the SmartConnect load-balancing policies to a pool without setting a service subnet.

## ALIGN DATASETS WITH STORAGE POOLS

With a SmartPools license, you can create node pools, file policies, and tiers of storage. Node pools segment nodes into groups so that you can align a dataset with its performance requirements. File polices can isolate and store files by type, path, size, and other attributes. Tiers optimize the storage of data by need, such as a frequently used high-speed tier or a rarely accessed archive.

Because you can combine Dell EMC Isilon nodes from the S-Series, the X-Series, and the NL-Series into a cluster, you can set up pools of nodes to accelerate access to important working sets while placing inactive data in more cost-effective storage.

For example, you can group S-Series nodes with 900 GB SAS drives and 400 GB SSDs per node in one pool, while you put NL-Series nodes with 3 TB SATA drives in another. As each type of node has a different capacity-to-performance ratio, you should assign different datasets to node pools that meet the dataset's capacity or performance requirements. In this way, the node pools can isolate the datasets to increase performance for important data and decrease storage costs for less important data.

After you set up node pools, you can establish file pools and govern them with policies. The policies move files, directories, and file pools among node pools or tiers. Routing files with a certain file extension into a node pool containing S-Series nodes can deliver faster read and write performance. Another policy can evaluate the last-modified date to archive old files.

SmartPools also manages data overflow with a spillover policy. When a pool becomes full, OneFS redirects write operations to another pool. It is a best practice to apply a spillover policy to a pool that handles a critical workflow.

### SMARTPOOLS FOR ANALYTICS DATA

You can build node pools and policies to streamline an analytics workflow. For example, with SmartPools you can create two pools and one policy. One pool contains higher-performance SAS drives, while the other contains lower-cost SATA drives. Both pools are set at a high protection level. You need to create only one policy to manage the data—a policy that keeps current data on the faster drives. To place an active working set on faster drives, the policy can use file attributes to route files to a pool of S-Series nodes with SSDs to increase throughput rates and decrease latency levels.

At the same time, the policy can move older files, as identified by the last-modified or last-accessed date, to a data storage target of NL-Series disks to reserve the capacity in the S-Series node pool for newer data. When a policy moves a file or directory from one disk pool to another, the file's location in the file system tree remains the same, and you do not need to reconfigure clients to access the file in the new location. A file pool policy can apply to directories or files, and it can filter files by file name, path, type, and size, as well as other file attributes. You can also separate snapshot storage from data storage to route snapshots to a cost-effective target.

### WRITE CACHING WITH SMARTCACHE

Write caching improves performance for most workflows. To optimize I/O operations, write caching coalesces data into a write-back cache and writes the data to disk at the best time. Dell EMC Isilon calls write caching SmartCache. OneFS caches write operations sent over HDFS in the same way that OneFS caches write operations for other protocols.

In general, Dell EMC Isilon recommends that you leave write caching turned on. OneFS interprets writes as either synchronous or asynchronous, depending on a client's specifications. The impact and risk of write caching depend on the protocols with which your clients write data to the cluster and on whether the writes are interpreted as synchronous or asynchronous. If you disable write caching, OneFS ignores your clients' specifications and writes data synchronously. For more information, see the "OneFS Administration Guide."

### GUIDELINES FOR FILE POOL MANAGEMENT

Dell EMC Isilon recommends the following guidelines to improve overall performance:

- Plan file pool policies carefully, especially the sequence in which they are applied. Without proper planning and analysis, policies can conflict with or override one another.

- Note that directories with large amounts of stale data—for instance, data that has not been accessed for more than 60 days—can be migrated with a file pool policy to archive storage on an NL-Series pool. OneFS supplies a template for this policy.
- If directories take a long time to load in a file manager like Windows Explorer because there are a large number of objects—directories, files, or both—enable metadata acceleration for the data to improve load time.

For more information on SmartPools file pool policies, see the white paper titled “Next Generation Storage Tiering with Dell EMC Isilon SmartPools” on the Dell EMC website and the “OneFS Administration Guide.”

## CHECK THE OPTIMIZATION SETTINGS OF DIRECTORIES AND FILES

You can check the attributes of directories and files to see your current settings by running the following command and replacing

<data> in the example with the name of a directory or file. The command’s output, which shows the properties of a directory named data, is abridged to remove some immaterial information.

isi get -D data ❶

```
POLICY      W   LEVEL PERFORMANCE      COAL    ENCODING        FILE                IADDRS
default         4x/2 concurrency ❷ on          N/A             ./                  <1,1,202150912:512>,
<1,4,235175936:512>, <1,23,135037440:512>, <2,0,33851392:512>, <3,2,67409408:512> ct: 1369866689 rt:
0
************************************************
* IFS inode: [ 1,1,202150912:512, 1,4,235175936:512, 1,23,135037440:512, 2,0,33851392:512,
3,2,67409408:512 ]
************************************************
  *   Inode Version:          6
  *   Dir Version:            2
  *   Inode Revision:         7
  *   Inode Mirror Count: 5
  *   Physical Blocks:        0
*     LIN:                    1:0003:0000
*     Last Modified:          1370290930.593574196
*     Last Inode Change:      1370290930.593574196
*     Create Time:            1369866689.508040477
  *   Rename Time:            0
  *   Write Caching:          Enabled ❸
  *   Parent Lin              2
  *   Parent Hash:            763857
  *   Manually Manage:
  *           Access          False
  *           Protection      False
  *   Protection Policy:      Diskpool default
  *   Current Protection: 4x
  *   Future Protection:      0x
  *   Disk pools:             policy any pool group ID -> data target s200_13tb_400gb-ssd_48gb:6(6), metadata target
 s200_13tb_400gb-ssd_48gb:6(6)
  *   SSD Strategy:           metadata ❹
  *   SSD Status:             complete
  *   Layout drive count: 0
  *   Access pattern: 0
  *   File Data (118 bytes):
  *     Metatree Depth: 1
  *   Dynamic Attributes (37 bytes):
            ATTRIBUTE                OFFSET SIZE
            New file attribute       0      23
            Disk pool policy ID      23     5
            Last snapshot paint time 28     9
```

```
************************************************
  *    NEW FILE ATTRIBUTES ❺
  *    Access attributes:              active
  *    Write Cache:          on
  *    Access Pattern:             concurrency
  *    At_r: 0
  *    Protection attributes:                active
  *    Protection Policy:              Diskpool default
  *    Disk pools:                     policy any pool group ID
  *    SSD Strategy:                   metadata
************************************************
```

Here is what some of these lines mean:

1. This is a OneFS command to display the file system properties of a directory or file. For more information, see the "OneFS Command Reference."
2. The directory's data access pattern is set to concurrency. ❸Write caching (SmartCache) is turned on.
3. The SSD strategy is set to metadata, which is the default.
4. Files that are added to the directory are governed by these settings, most of which can be changed by applying a file pool policy to the directory.

## SECURING HDFS CONNECTIONS WITH KERBEROS

Although a best practice is to secure your Isilon cluster with Kerberos, setting up Kerberos authentication can rile the most seasoned system administrator. It's particularly difficult with a Hadoop compute grid.

A best practice is to add Kerberos authentication only after you have set up your Isilon cluster and your Hadoop clients and verified that the entire system works seamlessly. Permissions issues should be resolved before you set up Kerberos. Having to troubleshoot either a Hadoop system or permissions issues after you add the complexity of Kerberos is not a situation you want to find yourself in. Keep in mind that after you add Kerberos security, connections to the cluster that are not authenticated with Kerberos, including those of tools and scripts, will be denied access.

*Hadoop Operations*, published by O'Reilly Media, contains excellent sections on identities, permissions, authentication, and Kerberos. It is recommended that you read them closely before you tighten permissions or set up Kerberos authentication.

You can set up Kerberos authentication by using either Microsoft Active Directory or an MIT Kerberos 5 key distribution center.

### SET UP KERBEROS AUTHENTICATION WITH ACTIVE DIRECTORY

This section explains how to set up an Isilon cluster running OneFS 7.0.2.4 through 7.1 to authenticate and authorize HDFS connections with Kerberos 5 and Microsoft Active Directory. It is assumed that you know how to configure Linux and Unix systems to work with Kerberos and Active Directory and that you know how to manage an Isilon cluster with the OneFS command-line interface.

First, make sure you have created user accounts in Active Directory for HDFS, the job tracker, and the task tracker with a service principal name. Once you set up Kerberos, you cannot use a local user on a compute client to access HDFS data on an Isilon cluster; you must use an account in Active Directory (or LDAP). The following examples use three service principal names:

hdfs@EAST.EXAMPLE.COM jt@EAST.EXAMPLE.COM tt@EAST.EXAMPLE.COM

First, you must create keytab files for every computer or hostname in your Hadoop client cluster by using Microsoft's ktpass utility. You can then use the _HOST keyword in your mapred-site.xml file when you specify the tasktracker and jobtracker principals. Here is an example of the syntax of the ktpass command to create a keytab file:

```
ktpass princ <user>/<computername>@<REALM> mapuser <user>@<REALM> +rndPass -out Documents/user.keytab /ptype KRB5_NT_PRINCIPAL
```

For information on ktpass, see http://technet.microsoft.com/en-us/library/cc776746.aspx.

After you generate the keytab files, use secure FTP or another secure method to distribute them to the Hadoop compute clients that need them. The Kerberos keytab file contains an encrypted, local copy of the host's key and, if compromised, might allow unrestricted access to the host computer. It is therefore crucial to protect it with file-access permissions.

To avoid account name conflicts after you set up Kerberos, you might have to delete the local accounts on your Isilon cluster that you created for the Hadoop services—user accounts such as hdfs, mapred, and hadoop as well as the local group named hadoop.

Second, you must configure the krb5.conf file on your Hadoop compute clients to use Active Directory. Include the default domain and the server settings for the KDC. For more information about setting up Hadoop clients to use Kerberos authentication, see *Hadoop Operations* and *Kerberos: The Definitive Guide*, both published by O'Reilly Media.

Third, make sure that both your Isilon cluster as well as your compute clients use either Active Directory or the same NTP server as their time source. With Hadoop and Kerberos, a time skew among system components can wreak havoc.

Fourth, connect as root with SSH to any Isilon node and run the following commands, replacing EAST.EXAMPLE.COM with the realm of your Active Directory domain. The first command, which requires you to enter the password for an Active Directory administrator account, joins your Isilon cluster to an Active Directory domain; if you are already joined to a domain, you do not need to run this command. (For more information about the commands, see the OneFS guide for the command-line interface.)

```
isi auth ads create EAST.EXAMPLE.COM administrator
isi auth ads modify --provider-name=EAST.EXAMPLE.COM --assume-default-domain=true
```

Next, obtain the hostname of your Isilon cluster by running the following command and then confirm that DNS can resolve the hostname.

```
isi auth ads list -v | grep Hostname
```

Next, run the following commands, replacing hostname with the hostname of the cluster that you obtained with the grep command and replacing the example domain with your Active Directory domain:

```
isi hdfs krb5 --kerb-instance=hostname
isi hdfs krb5 --keytab=DYNAMIC:/usr/lib/kt_isi_pstore.so:hdfs:EAST.EXAMPLE.COM
isi auth ads spn create --spn=hdfs/hostname --domain=EAST.EXAMPLE.COM --user=administrator
```

The final steps are to modify your Hadoop configuration files on your Hadoop compute clients and then restart the Hadoop services so that the changes take effect; see Altering Hadoop configuration files for Kerberos.

## SET UP KERBEROS AUTHENTICATION WITH MIT KERBEROS 5

This section explains how to set up an Isilon cluster running OneFS 7.0.2.4 through 7.1 to authenticate HDFS connections with a stand-alone MIT Kerberos 5 key distribution center. The following instructions assume that you have already set up a Kerberos system with a resolvable hostname for the KDC and a resolvable hostname for the KDC admin server. This section also assumes that you know how to manage an Isilon cluster with the OneFS command-line interface and that you know how to configure client computers to work with Kerberos. The instructions also assume that you have already created a SmartConnect zone for your Isilon cluster.

Setting up a stand-alone KDC to process HDFS authentication requests instead of sending them straight to Active Directory might improve performance for heavy workloads. After you set up the KDC, you can establish a trust between the KDC and Active Directory so that you can centrally manage all the users, groups, and principals in Active Directory.

First, connect with SSH as root to any node in your Isilon cluster and run the following command to add your realm and set the hostnames of the KDC and the KDC admin server. Replace the descriptions in brackets with the information from your network.

```
isi auth krb5 add realm --realm=<REALM NAME> --kdc=<resolvable hostname of kdc> --adminserver=<
resolvable hostname of kdc admin server>
```

Second, set the default domain:

```
isi auth krb5 modify default --default-realm=<REALM>
```

Third, write a krb5.conf file:

```
isi auth krb5 write
```

You can now run the kadmin utility on the cluster to confirm that you added the realm correctly:

```
kadmin -p root/admin@<REALM>
kadmin: quit
```

Next, set the cluster to use the principal that matches the name of the SmartConnect zone:

```
isi hdfs krb5 --kerb-instance=<resolvableSmartConnectZoneName>
```
This command returns the principal that you must add to your KDC. You must also create a principal name with the principal by running the following commands on your Isilon cluster in the kadmin utility:

```
kadmin -p <admin credentials>
ank -randkey hdfs/<resolvableSmartConnectZoneName>
```
The result produces a key. In the kadmin utility, add it to a keytab file and export it with a unique file name in the kadmin utility:

```
ktadd -k <keytab /path/filename> hdfs/<resolvableSmartConnectZoneName>
```
Quit the kadmin utility and then, with the OneFS command-line interface, securely copy the keytab file to all the nodes in your Isilon cluster by running the following command after replacing the variables with your path and file name.

```
for ip in `isi_nodes %{internal}`; do scp <keytab /path/filename> $ip:/etc/;done
```
Test initializing a Kerberos ticket by running the following command with the OneFS command-line interface:

```
kinit -k -t <keytab /path/filename> hdfs/<resolvableSmartConnectZoneName>
```
View the ticket:

```
klist
```
Erase the ticket after the test:

```
kdestroy
```
Finally, set the OneFS HDFS daemon to use the keytab file that you exported:

```
isi hdfs krb5 --keytab=/etc/<filename>
```
**Example**

Here is a sample block of code that moves through the commands to set up OneFS to work with an MIT Kerberos 5 KDC. The

following example block assume that an Isilon SmartConnect zone named wai-kerb-sc was created like this:

```
kdc-demo-1# isi networks modify subnet --name=subnet0 --sc-service-addr=192.0.2.17
Modifying subnet 'subnet0':
Saving: OK
kdc-demo-1# isi networks modify pool --name=subnet0:pool0 --zone=wai-kerb-sc --sc-subnet=subnet0
Modifying pool 'subnet0:pool0':
Saving: OK
```
Here is the block of commands that was run on an Isilon node named kdc-demo-1. The name of the MIT Kerberos 5 KDC and adminserver is york.east.example.com.

```
kdc-demo-1# isi auth krb5 add realm --realm=EAST.EXAMPLE.COM --kdc=york.east.example.com --
adminserver=
york.east.example.com
kdc-demo-1# isi auth krb5 modify default --default-realm=EAST.EXAMPLE.COM
kdc-demo-1# isi auth krb5 write
kdc-demo-1# kadmin -p root/admin@EAST.EXAMPLE.COM
Authenticating as principal root/admin@EAST.EXAMPLE.COM with password.
Password for root/admin@EAST.EXAMPLE.COM:
kadmin: quit
kdc-demo-1# ping wai-kerb-sc
PING wai-kerb-sc.east.example.com (192.0.2.11): 56 data bytes
64 bytes from 192.0.2.11: icmp_seq=0 ttl=64 time=0.561 ms
kdc-demo-1# isi hdfs krb5 --kerb-instance=wai-kerb-sc
Add this principal to your KDC: hdfs/wai-kerb-sc.east.example.com@<YOUR-REALM.COM>
kdc-demo-1# kadmin -p root/admin
Authenticating as principal root/admin with password.
Password for root/admin@EAST.EXAMPLE.COM:
kadmin: ank -randkey hdfs/wai-kerb-sc.east.example.com
WARNING: no policy specified for hdfs/wai-kerb-sc.east.example.com@EAST.EXAMPLE.COM; defaulting to
no policy
Principal "hdfs/wai-kerb-sc.east.example.com@EAST.EXAMPLE.COM" created.
kadmin: ktadd -k /ifs/hdfs.keytab hdfs/wai-kerb-sc.east.example.com
Entry for principal hdfs/wai-kerb-sc.east.example.com with kvno 3, encryption type AES-256 CTS mode
```

```
with 96-bit SHA-1 HMAC added to keytab WRFILE:/ifs/hdfs.keytab.
Entry for principal hdfs/wai-kerb-sc.east.example.com with kvno 3, encryption type ArcFour with
HMAC/md5 added to keytab WRFILE:/ifs/hdfs.keytab.
Entry for principal hdfs/wai-kerb-sc.east.example.com with kvno 3, encryption type Triple DES cbc
mode with HMAC/sha1 added to keytab WRFILE:/ifs/hdfs.keytab.
Entry for principal hdfs/wai-kerb-sc.east.example.com with kvno 3, encryption type DES cbc mode
with CRC-32 added to keytab WRFILE:/ifs/hdfs.keytab.
kdc-demo-1# for ip in `isi_nodes %{internal}`;do scp /ifs/hdfs.keytab $ip:/etc/;done
Password:
hdfs.keytab 100% 666 0.7KB/s 0.7KB/s 00:00
Max throughput: 0.7KB/s
kdc-demo-1# kinit -k -t /etc/hdfs.keytab hdfs/wai-kerb-sc.east.example.com
kdc-demo-1# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: hdfs/wai-kerb-sc.east.example.com@EAST.EXAMPLE.COM
Valid starting Expires Service principal
01/28/14 15:15:34 01/29/14 01:15:34 krbtgt/EAST.EXAMPLE.COM@EAST.EXAMPLE.COM
renew until 01/29/14 15:13:46
kdc-demo-1# kdestroy
```

```
kdc-demo-1# isi hdfs krb5 --keytab=/etc/hdfs.keytab
```

## ALTERING HADOOP CONFIGURATION FILES FOR KERBEROS

To turn on Kerberos authentication with either Active Directory or a stand-alone KDC, you must modify three Hadoop configuration files on your Hadoop compute clients: mapred-site.xml, hdfs-site.xml, and core-site.xml. After you make the changes, you must restart the Hadoop daemons on the compute clients for the changes to take effect.

### Modify mapred-site.xml

Add the principal for the job tracker and the location of its keytab file to mapred-site.xml. Here is an example:

```
<?xml version="1.0" ?>
<!-- mapred-site.xml -->
<property>
<name>mapreduce.jobtracker.kerberos.principal</name>
<value>jt/hadoop-east.example.com@EAST.EXAMPLE.COM</value>
</property>
<property>
<name>mapreduce.jobtracker.keytab.file</name>
<value>/etc/jt.keytab</value>
</property>
```

Add the principal for the tasktracker and the location of its keytab file to mapred-site.xml. Example:

```
<property>
<name>mapreduce.tasktracker.kerberos.principal</name>
<value>tt/hadoop-east.example.com@EAST.EXAMPLE.COM</value>
</property>
<property>
<name>mapreduce.tasktracker.keytab.file</name>
<value>/etc/tt.keytab</value>
</property>
```

**Important:** To run Hadoop jobs or distcp, you must make sure that the principals that the Hadoop daemons are using—that is, the value of the tasktracker's and jobtracker's Kerberos principal in mapred-site.xml—map to users on your Isilon cluster and can be resolved on the cluster by using either OneFS local users or users from LDAP or Active Directory.

#### Modify hdfs-site.xml for Kerberos

Now, return to your Hadoop compute clients and add the following properties to hdfs-site.xml, replacing `hostname` with the hostname of the cluster from the grep command and replacing the example realm with your realm (do not use the _host variable):

```
<?xml version="1.0" ?>
<!-- hdfs-site.xml -->
<configuration><property>
<name>dfs.namenode.kerberos.principal</name>
<value>hdfs/hostname@EAST.EXAMPLE.COM</value>
</property><property>
<name>dfs.datanode.kerberos.principal</name>
<value>hdfs/hostname@EAST.EXAMPLE.COM</value>
</property></configuration>
```

**Modify core-site.xml for authentication and authorization**

Next, turn on Kerberos authentication by adding the following security properties to core-site.xml:

```
<?xml version="1.0"?>
<!-- core-site.xml -->
<configuration><property>
<name>hadoop.security.authentication</name>
<value>kerberos</value>
</property><property>
<name>hadoop.security.authorization</name>
<value>true</value>
</property></configuration>
```

Remember to restart your Hadoop daemons on the compute clients so that the changes to the Hadoop configuration files take effect.

## TEST KERBEROS AUTHENTICATION

Finally, on a compute client, validate the connection to your Isilon cluster:

```
# su hdfs
$ kinit hdfs@EAST.EXAMPLE.COM
$ ./hadoop-2.0.0-cdh4.0.1/bin/hadoop fs -ls /
```

And then run a sample MapReduce job to check the system. The following example uses a job from Cloudera 4 but you can substitute one of your own:

```
# passwd hdfs [hdfs needs a password]
# su – hdfs
$ ./hadoop-2.0.0-cdh4.0.1/sbin/start-yarn.sh
$ ./hadoop-2.0.0-cdh4.0.1/bin/hadoop jar hadoop-2.0.0-cdh4.0.1/share/hadoop/mapreduce/hadoop- mapreduce-examples-2.0.0-
cdh4.0.1.jar pi 100 1000000
```

## TROUBLESHOOTING KERBEROS AUTHENTICATION

Authentication problems can be difficult to diagnose. First, check all the configuration parameters, including the location and validity of the keytab file. Second, check your user and group accounts for permissions. Make sure there are no duplicates of the accounts across systems, such as a local hdfs account on OneFS and an hdfs account in Active Directory. Third, make sure none of the problems in the following table are sabotaging authentication.

| PROBLEM | SOLUTION |
|---|---|
| The system's clock is out of sync. | The Kerberos standard requires that system clocks be no more than 5 minutes apart. Make sure that the system clocks on the Active Directory domain controller, the Isilon nodes, and the Hadoop clients are synchronized with a formal time source like NTP. |

| | |
|---|---|
| The service principal name of a Hadoop service, such as the tasktracker, is mapped to more than one object in the Active Directory. | Although this problem is rare, it is difficult to diagnose because the error messages are vague. The problem can occur after the ktpass utility was used repeatedly to generate a Kerberos keytab file for a service.<br><br>To check for this problem, log on to your Active Directory domain controller and open the Event Viewer. Look for an event of type=Error, source=KDC, and event ID=11. The text of the event will be similar to this message:<br><br>There are multiple accounts with name HDFS/myserver.mydomain.com of type DS_SERVICE_PRINCIPAL_NAME.<br><br>To fix the problem, find the computer or user objects that were used to map the service principal name in Active Directory and then use the ADSI Edit tool to manually remove the (for example) “HDFS/myserver.mydomain.com” string from the servicePrincipalName object property. For more information, see the Microsoft documentation for Active Directory and ADSI Edit. You can also use the Microsoft Ldp utility to search for an object by name, such as hdfs. |

Situations to check to help troubleshoot Kerberos Authentication

## CONCLUSION

A Dell EMC Isilon cluster optimizes the storage of big data for data analysis. Combining Hadoop clients with Isilon scale-out NAS and the OneFS implementation of HDFS delivers the following solutions:

- Store your analytics data with existing workflows and protocols like NFS, HTTP, and SMB instead of spending time importing and exporting data with HDFS.
- Protect data efficiently, reliably, and cost-effectively with forward error correction instead of triple replication.
- Manage data with such enterprise features as snapshots, deduplication, clones, and replication.
- Receive namenode redundancy with a distributed namenode daemon that eliminates a single point of failure.
- Support HDFS 1.0 and 2.0 simultaneously without migrating data or modifying metadata.
- Run multiple Hadoop distributions—including Cloudera, Pivotal HD, Apache Hadoop, and Hortonworks—against the same data set at the same time.
- Implement security for HDFS clients with Kerberos and address compliance requirements with write-once, read-many (WORM) protection for Hadoop data.
- Scale storage independently of compute to handle expanding data sets.

By scaling multidimensionally to handle the exponential growth of big data, a Dell EMC Isilon cluster pairs with Hadoop to provide the best of both worlds: Data analytics and enterprise scale-out storage. The combination helps you adapt to fluid storage requirements, nondisruptively add capacity and performance in cost-effective increments, reduce storage overhead, and exploit your data through in-place analytics.