IBM® Spectrum Scale™

HDFS Transparency Guide

IBM Spectrum Scale

June 23, 2017

Contents

1. Overview 4
2. Supported IBM Spectrum Scale Storage modes 5
   2.1. Local storage mode 5
   2.2. Shared storage mode 5
3. Hadoop cluster planning 8
   3.1. Node roles planning 9
      3.1.1. The FPO mode 9
      3.1.2. The shared storage mode 10
      3.1.3. Integration with Hadoop distributions 11
   3.2. Hardware and software requirements 12
   3.3. Hadoop service roles 12
   3.4. Dual network interfaces 13
4. Installation and configuration 14
   4.1. Installation 14
   4.2. Configuration 14
      4.2.1. OS tuning for all nodes 15
      4.2.2. Configure Hadoop nodes 15
      4.2.3. Configure HDFS Transparency nodes 17
         4.2.3.1 Synchronization of Hadoop configurations 17
         4.2.3.2 Configuring the storage mode 18
         4.2.3.3 Update other configuration files 18
         4.2.3.4 Update environment variables for HDFS Transparency service 20
   4.3. Start and stop the service 21
   4.4. Health check the service 21
5. How application interact with HDFS Transparency 21
   5.1. Application interface 22
   5.2. Command line 22
6. Upgrading the HDFS Transparency cluster 23
   6.1. Removing IBM Spectrum Scale Hadoop connector 23
      6.1.1. Removing IBM Spectrum Scale Hadoop connector 2.4 over IBM Spectrum Scale 4.1.0.4, 4.1.0.5, 4.1.0.6, or 4.1.0.7 releases 23
      6.1.2. Removing IBM Spectrum Scale Hadoop connector 2.4 over IBM Spectrum Scale 4.1.0.7 (efix3) or 4.1.0.8 releases 24
      6.1.3. Removing IBM Spectrum Scale Hadoop connector 2.4 or 2.5 over IBM Spectrum

Scale 4.1.1+ releases 25
6.1.4. Removing IBM Spectrum Scale Hadoop connector 2.7 (earlier release) over IBM Spectrum Scale 4.1.0.7 (efix3+), 4.1.0.8, or 4.1.1+ releases 26
6.2. Upgrading the HDFS Transparency cluster 27
7. Security 27
8. Advanced features 27
8.1. High Availability 27
8.1.1. Configuration for manual HA switch 27
8.1.2. Configuration for automatic NameNode HA 31
8.2. Short circuit read configuration 35
8.2.1. For HDFS Transparency version 2.7.0-x 36
8.2.2. For HDFS Transparency version 2.7.2-x 38
8.3. Configuring multiple Hadoop clusters over the same file system 39
8.4. Docker support 40
8.5. HDFS Transparency federation 44
8.5.1. Federating IBM Spectrum Scale and HDFS 45
8.5.1.1 Federating native HDFS with HDFS Transparency 45
8.5.1.2 Federating HDFS Transparency into native HDFS 48
8.5.2. Federating two IBM Spectrum Scale file systems 50
8.5.3. Known limits 55
8.6. Hadoop `distcp` support 55
8.7. Automatic configuration refresh 57
8.8. Ranger support 58
8.8.1. Installing Ranger in native HDFS 58
8.8.2. Configuring Ranger with HDFS Transparency 64
8.8.3. Using Ranger to secure HDFS 67
8.8.4. Enabling Ranger auditing 70
8.8.5. Disable Ranger Support 70
8.8.6. Known issues 70
8.9. Rack locality support for shared storage 71
8.10. Accumulo support 74
9. Hadoop distribution support 74
9.1. Replacing native HDFS service with HDFS transparency 75
9.2. IBM BigInsights IOP support 76
10. Limitations and differences from native HDFS 76
11. Problem determination 78
12. Revision history 78

1. Overview

IBM Spectrum Scale HDFS Transparency, also known as HDFS Protocol, offers a set of interfaces that allows applications to use HDFS clients to access IBM Spectrum Scale through HDFS RPC requests.

All data transmission and metadata operations in HDFS are through the RPC mechanism and processed by the NameNode and the DataNode services within HDFS. IBM Spectrum Scale HDFS protocol implementation integrates both the NameNode and the DataNode services and responds to the request as if it were HDFS. Advantages of HDFS transparency are as follows:

- HDFS compliant APIs or shell-interface command
- Application client isolation from storage. Application client can access data in the IBM Spectrum Scale filesystem without having a GPFS client installed.
- Improved security management by Kerberos authentication and encryption for RPCs
- Simplified file system monitoring by Hadoop Metrics2 integration

The following *Figure 1* shows the framework of HDFS transparency over IBM Spectrum Scale:

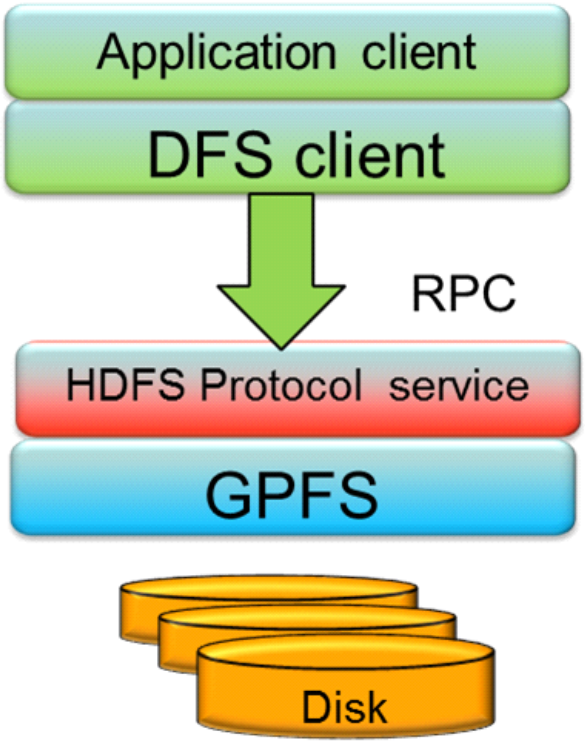


Figure 1 IBM Spectrum Scale HDFS Transparency Framework

Visit the Spectrum Scale developerWorks wiki website for more information and download the HDFS

Transparency package.

2. Supported IBM Spectrum Scale Storage modes

2.1. Local storage mode

HDFS Transparency allows big data applications to access IBM Spectrum Scale local storage - File Placement Optimizer (FPO) mode and enables the support for shared storage mode, such as SAN-based storage, IBM Elastic Storage Server (ESS) , starting with the gpfs.hdfs-protocol.2.7.0-1 package.

In FPO mode, the data blocks are stored in chunks in IBM Spectrum Scale and are replicated to protect against disk and node failures. DFS clients run on the storage node so that they can leverage the data locality for executing tasks quickly. For the local storage mode configuration, short-circuit read is recommended to improve the access efficiency.

Figure 2 illustrates the HDFS Transparency over FPO:

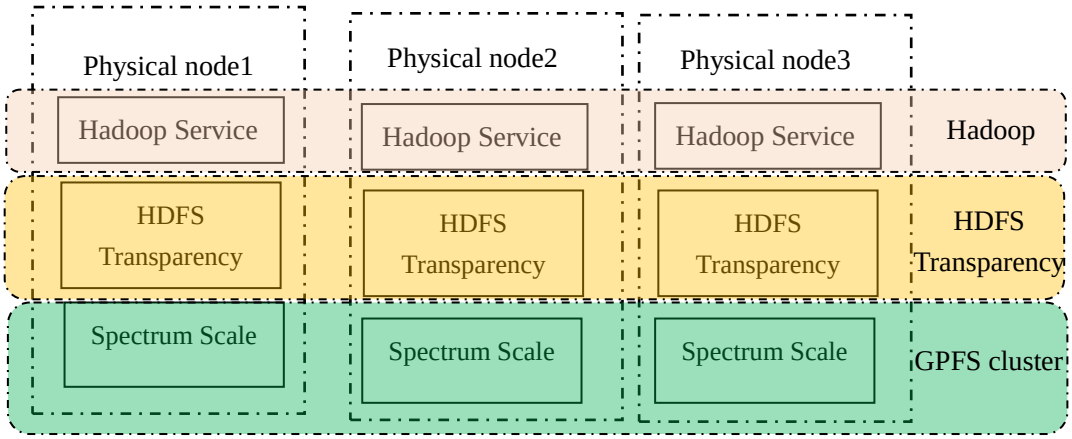


Figure 2: HDFS Transparency over IBM Spectrum Scale FPO

2.2. Shared storage mode

HDFS Transparency allows big data applications to access data stored in the shared storage mode, such as SAN-based storage and IBM Elastic Storage Server.

In this mode, data is stored in shared storage systems that can offer better storage efficiency than the local storage. RAID and other technologies can be used to protect against hardware failure instead of using data replication.

DFS clients access data through the HDFS protocol remote procedure call (RPC). When a DFS Client requests to write blocks to IBM Spectrum Scale, the HDFS Transparency NameNode selects a DataNode randomly for this request. When the DFS Client is located on a DataNode, that node will be selected for this request. When the DFS client requests to `getBlockLocation` of an existing block, NameNode selects a DataNode randomly for this request. For example, if the `dfs.replication` parameter is set to 3, then 3 DataNodes are returned for a `getBlockLocation` request. If `dfs.replication` parameter is set to 1, then a single DataNode is returned for the getBlockLocation request.

HDFS Transparency allows the Hadoop application to access data stored in both a local IBM Spectrum Scale file system or a remote IBM Spectrum Scale file system from multiple clusters.

Figure 3 ,***Figure 4*** and Figure 5 illustrate HDFS Transparency over shared storage.

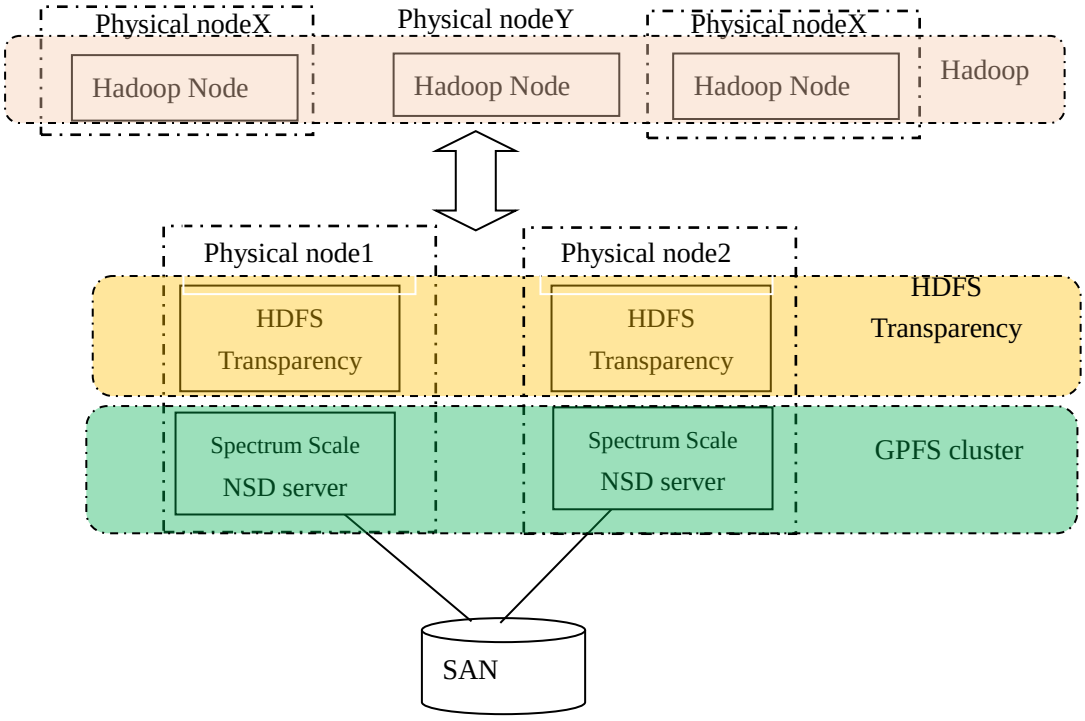


Figure 3: HDFS Transparency over Spectrum Scale NSD servers for shared storage

In this mode, HDFS Transparency is deployed over Spectrum Scale NSD servers and all Hadoop components will take HDFS client to talk with HDFS Transparency over HDFS RPC. the data traffic will go from Hadoop client nodes, HDFS Transparency nodes/Spectrum Scale server nodes and then

into/from SAN storage. If not considering short circuit read, this mode will give you best performance. Note: HDFS Transparency daemons are ligh weight processes and the system resource (1 CPU logic processor and 4GB~8GB physical memory are good for HDFS Transparency).

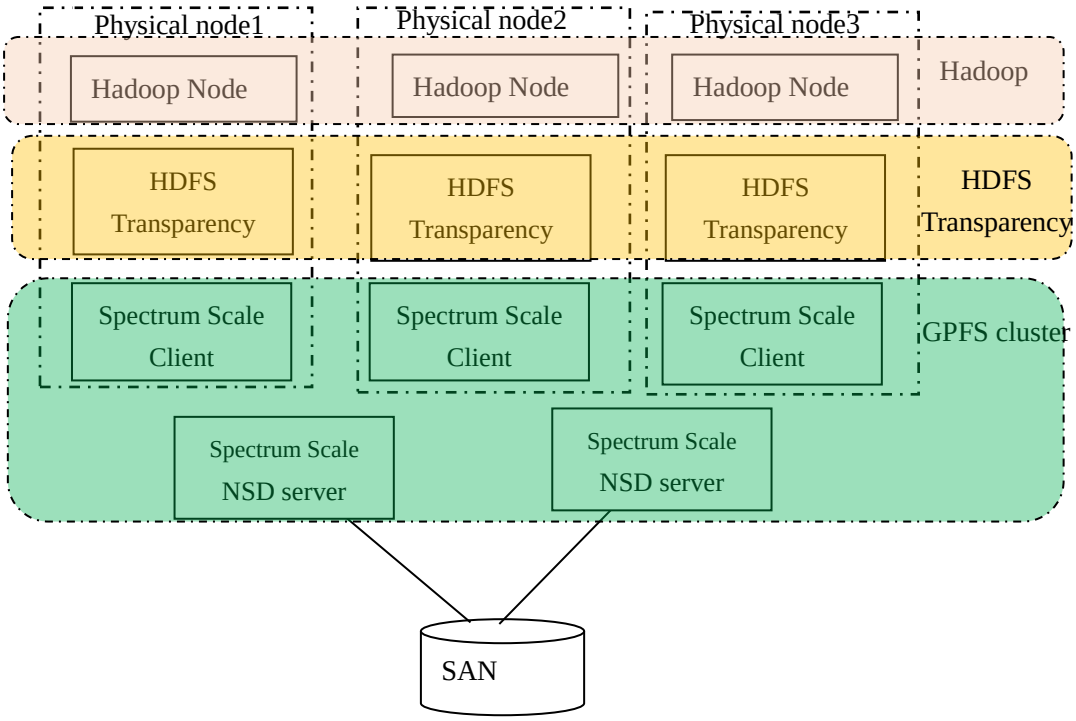


Figure 4: HDFS Transparency over Spectrum Scale Clients for shared storage

If the deployment in Figure3 is not accepted, the deployment in Figure 4 will be recommended: install Spectrum Scale clients on all Hadoop nodes. The data traffic in Figure 4 will go from Hadoop node, local lo(loop) network adapter, HDFS Transparency nodes/Spectrum Scale Clients, Spectrum Scale NSD servers and SAN storages. If short circuit read is enabled for this mode, the network traffic cost for local lo(loop) network adapter could be avoided.

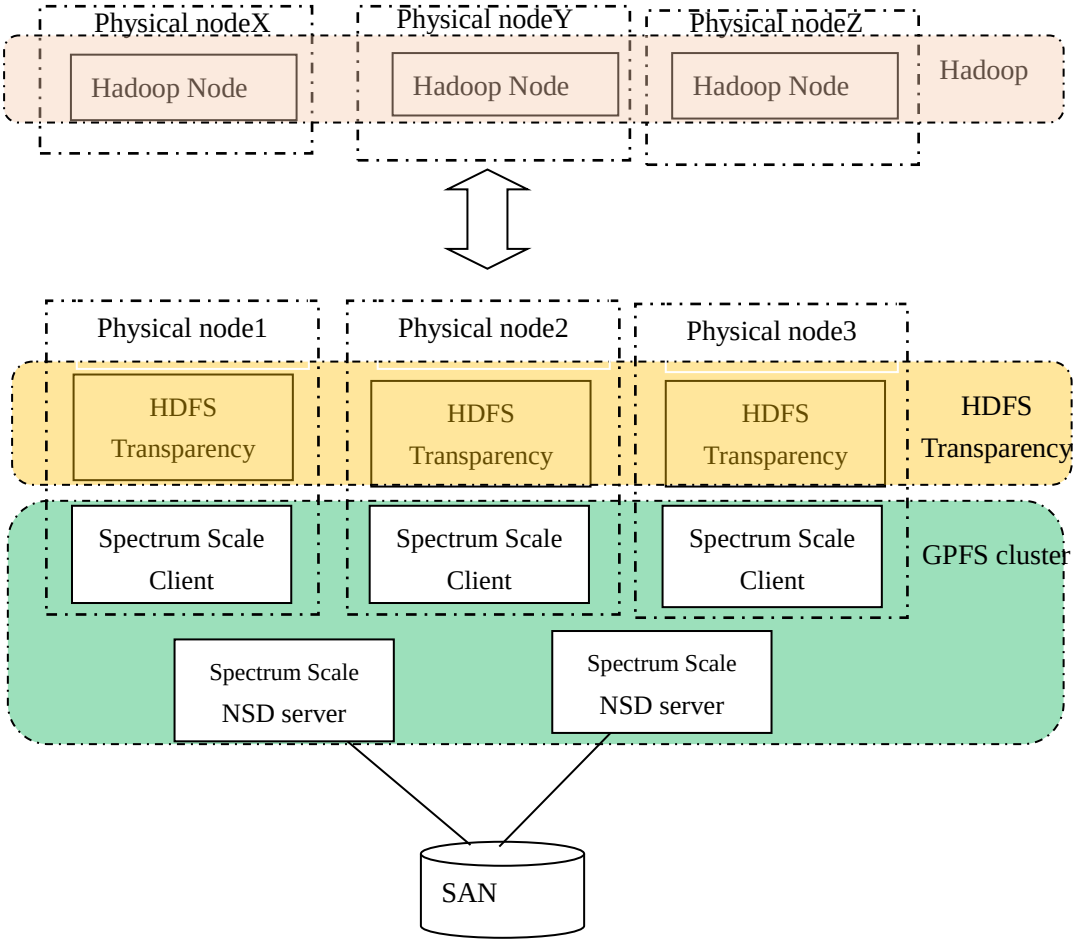


Figure 5: HDFS Transparency over limited Spectrum Scale Client for shared storage
In this deployment of Figure5, the data traffic will go from Hadoop nodes, netwprk RPC, HDFS Transparency nodes/Spectrum Scale Clients, network RPC, Spectrum Scale NSD servers and SAN storage. Short circuit read won't help data reading performance.

In Figure 4 or Figure5, the Spectrum Scale Clients and Spectrum Scale NSD servers could be configured with multi-cluster.

3. Hadoop cluster planning

In a Hadoop cluster that runs the HDFS protocol, a node can perform the role of a DFS client, a NameNode, or a DataNode, or all of them. The Hadoop cluster can contain nodes that are part of an IBM Spectrum Scale cluster or where only some of the nodes belong to an IBM Spectrum Scale cluster.

NameNode:
You can specify a single NameNode or multiple NameNodes to protect against a single point of failure in the cluster. For more information, see High availability configuration. The NameNode must be a part of an IBM Spectrum Scale cluster and must have a robust configuration to reduce the chances of a single-node failure. The NameNode is defined by setting the `fs.defaultFS` parameter to the hostname of the NameNode in the `core-site.xml` file in Hadoop 2.4, 2.5, and 2.7 releases.

Note: The Secondary NameNode in native HDFS is not needed for HDFS Transparency because the HDFS Transparency NameNode is stateless and does not maintain an FSImage or EditLog like state information.

DataNode:
You can specify multiple DataNodes in a cluster. The DataNodes must be a part of an IBM Spectrum Scale cluster. The DataNode are specified by listing their hostnames in the slaves configuration file.

DFS client
The DFS clients can be part of an IBM Spectrum Scale cluster. When the DFS Client is a part of an IBM Spectrum Scale cluster, it can read data from IBM Spectrum Scale through an RPC or use the short-circuit mode. Otherwise, the DFS client can access data from IBM Spectrum Scale only through an RPC. You can specify the NameNode address in the DFS Client configuration so that the DFS client can communicate with the appropriate NameNode service.

The purpose of cluster planning is to define the node roles: Hadoop node, HDFS transparency node, and GPFS node.

3.1. Node roles planning

3.1.1. The FPO mode

In the FPO mode, all nodes are IBM Spectrum Scale (GPFS) nodes in the FPO mode, Hadoop nodes, and HDFS Transparency nodes:

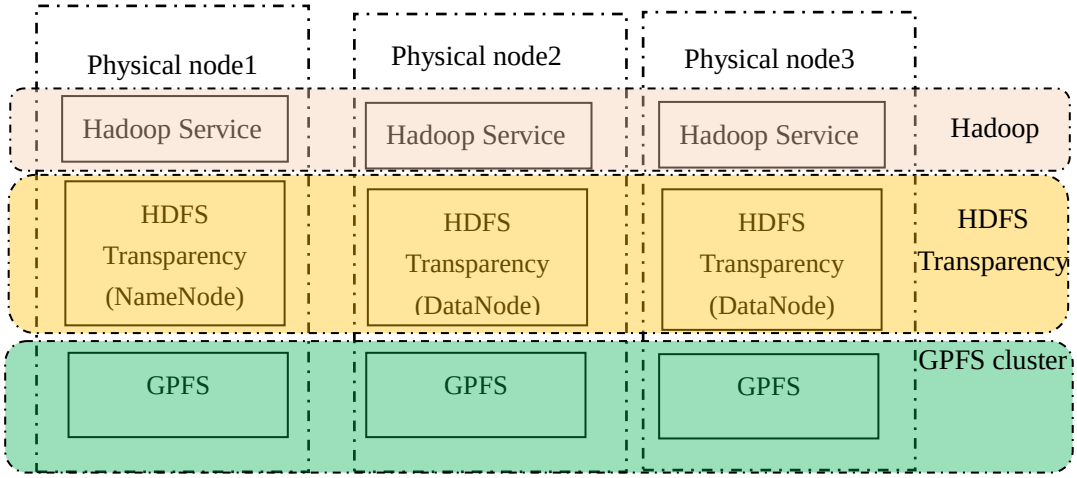


Figure 5: Typical Cluster Planning for Spectrum Scale FPO mode

In ***Figure 5***, one node is selected as the HDFS Transparency NameNode. All other nodes are HDFS Transparency DataNodes. Also, the HDFS Transparency NameNode can be an HDFS Transparency DataNode. Any one node can be selected as HDFS Transparency HA NameNode. The administrator must ensure that the primary HDFS Transparency NameNode and the standby HDFS Transparency NameNode are not the same node.

In this mode, the Hadoop cluster can be equal or larger than the HDFS transparency cluster.
Note: The Hadoop cluster can be smaller than the HDFS transparency cluster but this configuration is not typical and not recommended. Also, the HDFS transparency cluster can be smaller than or equal to the IBM Spectrum Scale cluster because HDFS Transparency must read and write data to the local mounted file system. Usually, in the FPO mode, the HDFS Transparency cluster is equal to the IBM Spectrum Scale cluster.

Note: Some nodes in the Spectrum Scale (GPFS) FPO cluster can be GPFS clients without any disks in the file system.

3.1.2. The shared storage mode

In the shared storage mode, Figure 6 depicts a typical configuration. If the IO stress is heavy, you can exclude the NSD servers from the HDFS Transparency cluster. Typically, in a shared storage mode, all nodes in the Hadoop cluster can be GPFS-client free, which means GPFS is not required to be installed on those Hadoop nodes.

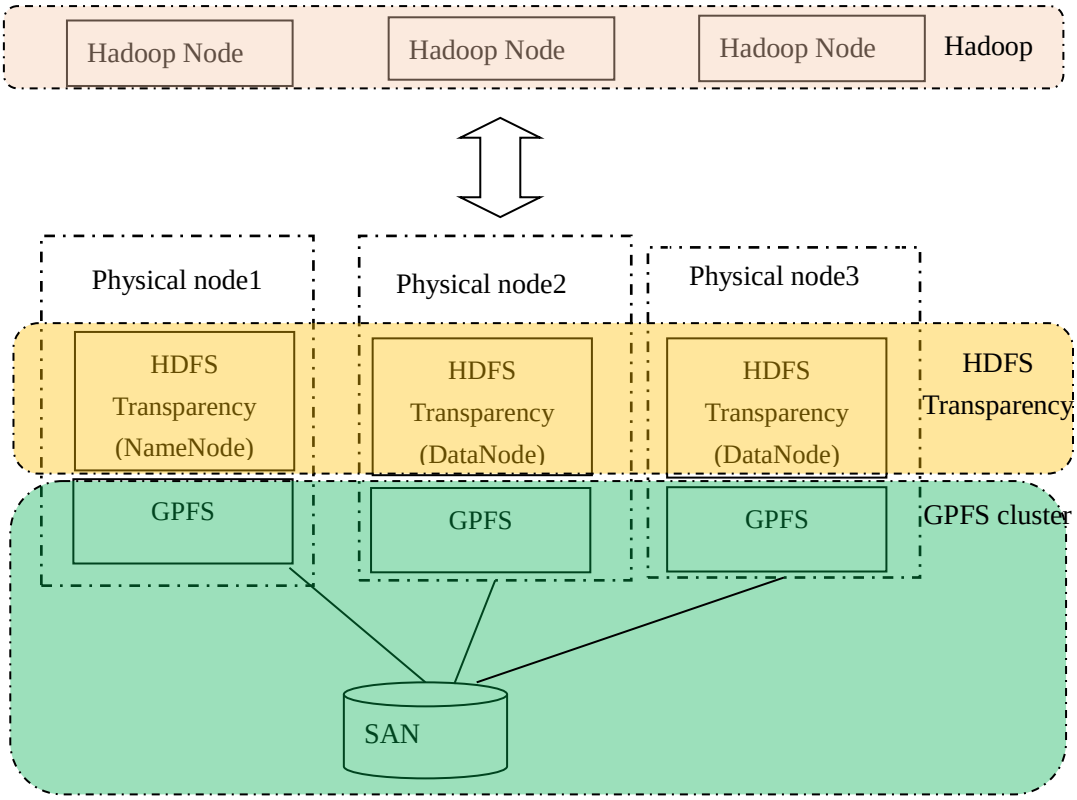


Figure 6: Typical Cluster Planning for shared storage mode

If using Spectrum Scale multi-cluster mode, please refer to ***Figure 4*** in section 2.2.

Note: All HDFS Transparency nodes require GPFS to be installed and mounted. For shared storage, deploy HDFS Transparency NameNode and DataNode services over nodes with local disk access path because this can reduce the amount of network traffic.

After HDFS transparency nodes are selected, follow Section 4 to configure HDFS Transparency on these nodes.

3.1.3. Integration with Hadoop distributions

If you deploy HDFS transparency with a Hadoop distribution, such as IBM BigInsights IOP, configure the native HDFS NameNode as the HDFS Transparency NameNode and add this node to the IBM Spectrum Scale cluster. This setup results in fewer configuration changes.

If the HDFS Transparency NameNode is not the same as the native HDFS NameNode, some services

might fail to start and can require additional configuration changes.

3.2. Hardware and software requirements

The recommended configuration for Hadoop nodes is a 10Gb network, 100GB physical memory, 10~20 internal SAS/SATA disks per node, and 8+ physical cores.

The following table is the Hardware & OS Matrix supported by HDFS Transparency:

HDFS Transparency	X86_64	ppc64	ppc64le
2.7.0-x	RHEL6+ RHEL7+ SLES11 SP3+ SLES12+ Ubuntu14.04+	RHEL7+	RHEL7+ SLES12+ Ubuntu14.04+
2.7.2-0	RHEL 6.7+ RHEL 7.2+	RHEL7.2+	RHEL7.2+ SLES12+ Ubuntu 14.04+
2.7.3-x	RHEL 6.7+ RHEL 7.2+	RHEL7.2+	RHEL7.2+ SLES12+ Ubuntu 14.04+

Note:

1. OpenJDK 1.8+ is required for HDFS Transparency versions 2.7.0-3+ and 2.7.2-0. For HDFS Transparency 2.7.2-0 on RHEL7/7.1, ensure that the glibc version is at level 2.14 or above. Use the rpm -qa | grep glibc command to check the glibc version.
2. The version number RHEL 6.7+ means RHEL 6.7 and later. Others are similar.

3.3. Hadoop service roles

In a Hadoop ecosystem, there are a lot of different roles for different components, such as HBase Master Server, Yarn Resource Manager, and Yarn Node Manager. Spread these master roles over different nodes as evenly as possible. If you put all master roles onto a single node, memory might become an issue.

While running Hadoop over IBM Spectrum Scale, it is recommended that up to 25% of the physical memory be reserved for the GPFS `pagepool` with a maximum of 16GB. If HBase is being used, it is recommended that up to 30% of the physical memory be reserved for the GPFS `pagepool`.
If the node has less than 100GB of physical memory, then the heap size for Hadoop Master services must be planned for carefully. If HDFS transparency NameNode service and HBase Master service are resident on the same physical node, HBase workload stress can result in Out of Memory (OOM)

exceptions.

3.4. Dual network interfaces

The following section is only applicable for IBM Spectrum Scale FPO (local storage) mode, and does not impact Hadoop clusters running over a shared storage configuration (e.g. SAN-based cluster, or ESS).

If the FPO cluster has a dual 10Gb network, you have two configuration options.The first option is to bind the two network interfaces and deploy the IBM Spectrum Scale cluster and the Hadoop cluster over the bonded interface.The second option is to configure one network interface for the Hadoop services including the HDFS transparency service and configure the other network interface for IBM Spectrum Scale to use for data traffic. This configuration can minimize interference between disk I/O and application communication.

For the second option perform the following steps to ensure that the Hadoop applications can use data locality for better performance.

1. Configure the first network interface with one subnet address (e.g. 192.168.1.0)and the second network interface as another subnet address (e.g. 192.168.2.0).
2. Create the IBM Spectrum Scale cluster and NSDs with the IP or hostname from the first network interface.
3. Install the Hadoop cluster and HDFS transparency services by using the IP addresses or hostnames from the 1st network interface.
4. Run `mmchconfig subnets=192.168.2.0 -N all.`

Note: 192.168.2.0 is the subnet used for IBM Spectrum Scale data traffic.

For Hadoop map/reduce jobs, the scheduler Yarn will check the block location. HDFS Transparency will return the hostname which is used to create IBM Spectrum Scale cluster as block location to Yarn. Yarn checks the hostname within the NodeManager host list. If Yarn cannot find the hostname within the NodeManager list, Yarn cannot schedule the tasks according to data locality. The suggested configuration can ensure that, the hostname for block location can be found in the Yarn NodeManager list and Yarn can schedule the task according to data locality.

For a Hadoop distribution like IBM BigInsights IOP, all Hadoop components are managed by Ambari™. In this scenario, all Hadoop components, HDFS transparency and IBM Spectrum Scale cluster must be created by using one network interface and the second network interface must be used for GPFS data traffic.

4. Installation and configuration

This section describes the installation and configuration of HDFS Transparency with IBM Spectrum Scale.

Note: For details on how to install and configure IBM Spectrum Scale, see IBM Spectrum Scale Concepts, Planning and Installation Guide and The IBM Spectrum Scale Advanced Administration Guide in IBM Knowledge Center. Also, see the best practices guide on the IBM DeveloperWorks GPFS Wiki.

4.1. Installation

IBM Spectrum Scale HDFS Transparency must be installed on nodes that serve as a NameNode or DataNodes.

Use the following command to install the HDFS Transparency RPM:

```
# rpm -hiv gpfs.hdfs-protocol-2.7.0-0.x86_64.rpm
```

This package has the following dependencies:

- libacl
- libattr
- openjdk 7.0+

HDFS Transparency files are installed under the `/usr/lpp/mmfs/hadoop` directory. To list the contents of this directory, use the following command:

```
# ls /usr/lpp/mmfs/hadoop/
bin etc  lib  libexec  README_dev  sbin  share
```

The following directories can be added to the system shell path for convenience; `/usr/lpp/mmfs/hadoop/bin` and `/usr/lpp/mmfs/hadoop/.`

4.2. Configuration

In this section, it is assumed that the Hadoop distribution is installed under $YOUR_HADOOP_PREFIX on each machine in the cluster. The configuration files for the IBM Spectrum Scale HDFS Transparency are located under `/usr/lpp/mmfs/hadoop/etc/hadoop` regardless of the Hadoop distribution. Configuration files for the Hadoop distribution might be located in different directories. For example, for IBM BigInsights IOP the configuration files can be found in `/etc/hadoop/conf`.

The `core-site.xml` and `hdfs-site.xml` configuration files must be synchronized between all

the nodes and kept identical for IBM Spectrum Scale HDFS Transparency and Hadoop distribution. The `log4j.properties` configuration file can differ between the IBM Spectrum Scale HDFS transparency and the native the Hadoop distribution.

4.2.1. OS tuning for all nodes

ulimit tuning

For all nodes, ulimit -n and ulimit -u must be equal or larger than *65536*. Too small a value will make the Hadoop java processes report unexpected exceptions.

In Redhat, add the following lines at the end of the `/etc/security/limits.conf` file:

```
*       soft nofile 65536
*       hard nofile 65536

*       soft nproc 65536
*       hard nproc 65536
```

For other Linux distributions, see the relevant documentation. After the above change, all the Hadoop services must be restarted for the change to take effect.

Note: This must be done on all nodes including the Hadoop client nodes and the HDFS Transparency nodes.

kernel.pid_max

usually, the default value is 32K. if you see the error “allocate memory”or “unable to create new native thread”, you could try to increase kernel.pid_max by adding “kernel.pid_max=99999” at the end of /etc/sysctl.conf and then “sysctl -p”.

4.2.2. Configure Hadoop nodes

If you are not familiar with HDFS/Hadoop, set up the native HDFS first by seeing the Hadoop cluster setup guide. After you set up HDFS/Hadoop, it will be easier for you to set up the HDFS Transparency to replace the native HDFS.

Here are basic examples of a `core-site.xml` and `slaves` configuration files for open source Apache© Hadoop:

In this example, the hostname of the NameNode service is *hs22n44*. Edit the following files for a standard Hadoop configuration.

In `$YOUR_HADOOP_PREFIX/etc/hadoop/core-site.xml`, ensure that the `fs.defaultFS` parameter is set to point to the HDFS Transparency NameNode:

```
<property>
```

```
<name>fs.defaultFS</name>
<value>hdfs://hs22n44:9000</value>
</property>
```

Replace *hs22n44:9000* with the hostname of your NameNode service and preferred port number.

Administrators can customize other configuration parameters like service ports. For more information, see http://hadoop.apache.org.

In `$YOUR_HADOOP_PREFIX/etc/hadoop/slaves` file, ensure that all the DataNodes are listed in the file, for example:

```
# cat $YOUR_HADOOP_PREFIX/etc/hadoop/slaves
hs22n44
hs22n54
hs22n45
```

As for `hdfs-site.xml` and other detailed configuration settings in `core-site.xml`, see http://hadoop.apache.org to configure the Hadoop nodes. Minimally, the following parameters must be configured to avoid unexpected exceptions from Hadoop:

```
<property>
  <name>dfs.datanode.handler.count</name>
  <value>40</value>
</property>
<property>
  <name>dfs.namenode.handler.count</name>
  <value>400</value>
</property>
<property>
  <name>dfs.datanode.max.transfer.threads</name>
  <value>8192</value>
</property>
```

After editing the configuration files, synchronize them to all Hadoop nodes.

For a Hadoop distribution, like IBM® BigInsights™ or Hortonworks Data Platform® (HDP), configure the Hadoop components (HBase, Hive, oozie) in the management GUI. For example: Ambari for IBM BigInsights or HDP.

For HDFS Transparency 2.7.0-x, 2.7.2-0, 2.7.2-1, do not export the Hadoop environment variables on the HDFS Transparency nodes because this can lead to issues when the HDFS Transparency uses the Hadoop environment variables to map to its own environment. The following Hadoop environment variables can affect HDFS Transparency:

HADOOP_HOME

HADOOP_HDFS_HOME

HADOOP_MAPRED_HOME
HADOOP_COMMON_HOME
HADOOP_COMMON_LIB_NATIVE_DIR
HADOOP_CONF_DIR
HADOOP_SECURITY_CONF_DIR

For HDFS Transparency versions 2.7.2-3+ and 2.7.3-x, the environmental variables above can be exported except for **HADOOP_COMMON_LIB_NATIVE_DIR.** This is because HDFS Transparency uses its own native .so library.

For HDFS Transparency versions 2.7.2-3+ and 2.7.3-x:

- If you did not export HADOOP_CONF_DIR, then HDFS Transparency will read all the configuration files under /usr/lpp/mmfs/hadoop/etc/hadoop
  such as the gpfs-site.xml file and the hadoop-env.sh file.
- If you export HADOOP_CONF_DIR, then HDFS Transparency will read
  all the configuration files under $HADOOP_CONF_DIR. Since gpfs-site.xml
  is required for HDFS Transparency, it will only read the gpfs-site.xml file from the
  /usr/lpp/mmfs/hadoop/etc/hadoop directory.

For questions or issues with HDFS Transparency configuration, send an email to scale@us.ibm.com for assistance.

4.2.3. Configure HDFS Transparency nodes

4.2.3.1 Synchronization of Hadoop configurations

By default, HDFS Transparency uses `core-site.xml` and the `hdfs-site.xml` configuration files from the Hadoop distribution, along with the `gpfs-site.xml` configuration file located in the `/usr/lpp/mmfs/hadoop/etc/hadoop` directory.

The following configuration files must be distributed to all the HDFS Transparency nodes:

```
core-site.xml
hdfs-site.xml
slaves
log4j.properties
```

If the HDFS Transparency nodes are also running Hadoop, you can use the following command to sync the configuration files to the rest of the HDFS Transparency nodes. Run this command on one of the HDFS Transparency nodes (e.g. *hdfs_transparency_node1)* after ensuring that the HDFS Transparency service is running:

```
[hdfs_transparency_node1#/usr/lpp/mmfs/hadoop/sbin/mmhadoopctl
connector syncconf <hadoop-conf-dir>
```

If the HDFS Transparency nodes are not running Hadoop as well, use a tool like `scp` (secure copy) to distribute the following files to the `/usr/lpp/mmfs/hadoop/etc/hadoop/` directory on all the HDFS Transparency nodes:

```
<hadoop-conf-dir>/core-site.xml
<hadoop-conf-dir>/hdfs-site.xml
<hadoop-conf-dir>/log4j.properties
```

4.2.3.2 Configuring the storage mode

1. Modify the `/usr/lpp/mmfs/hadoop/etc/hadoop/gpfs-site.xml` file on the *hdfs_transparency_node1* node:

```
<property>
<name>gpfs.storage.type</name>
<value>local</value>
</property>
```

The property `gpfs.storage.type` is used to specify the storage mode: ***local*** or ***shared***. This is a required configuration parameter and the `gpfs-site.xml` configuration file must be synchronized with all the HDFS Transparency nodes after the modification.

4.2.3.3 Update other configuration files

Note: For details on how to configure Hadoop HDFS, Yarn see the hadoop.apache.org website.

ConfiguringApache Hadoop

1. Modify the `/usr/lpp/mmfs/hadoop/etc/hadoop/gpfs-site.xml` file on the *hdfs_transparency_node1* node:

```
<property>
<name>gpfs.mnt.dir</name>
<value>/gpfs_mount_point</value>
</property>

<property>
<name>gpfs.data.dir</name>
<value>data_dir</value>
</property>

<property>
<name>gpfs.supergroup</name>
<value>hadoop</value>
```

```
</property>

<property>
<name>gpfs.replica.enforced</name>
<value>dfs</value>
</property>
```

In `gpfs-site.xml`, all Hadoop data is stored under the `/gpfs_mount_point/data_dir` directory. So, you can have two Hadoop clusters over the same file system and they will be isolated from each other. One limitation is that if there is a link under the `/gpfs_mount_point/data_dir` directory that points to a file outside the `/gpfs_mount_point/data_dir` directory, when Hadoop tries to operate on that file, HDFS Transparency will report an exception as that file is not accessible by Hadoop.

If you don't want to explicitly configure the *gpfs.data.dir parameter,* you can leave it as null, e.g. Keep the value as `<value></value>`.
Note: Do not configure it as `<value>/</value>`.

2. The `gpfs.supergroup` must be configured as per your cluster. Add some Hadoop users, such as hdfs, yarn, hbase, hive, oozie, , under the same group named *hadoop* and configure `gpfs.supergroup` as *hadoop*. You can specify two or more comma separated groups as `gpfs.supergroup`. For example, *group1,group2,group3*.

Note: Users in `gpfs.supergroup` are super users who can control all the data in the `/gpfs_mount_point/data_dir` directory, similar to the *root* user in Linux.

The `gpfs.replica.enforced` parameter is used to control the replication rules. Hadoop controls the data replication through the `dfs.replication` parameter. When running Hadoop over IBM Spectrum Scale, IBM Spectrum Scale has its own replication rules. If `gpfs.replica.enforced` is set to *dfs*, then the `dfs.replication` setting will always be in effect unless the `dfs.replication` parameter is specified in the command line options when submitting jobs. If `gpfs.replica.enforced` is set to *gpfs*, all data will be replicated according to the IBM Spectrum Scale configuration settings. The default value for this parameter is set to *dfs*.

Usually, the `core-site.xml` and `hdfs-site.xml` configuration files located under `/usr/lpp/mmfs/hadoop/etc/hadoop/` must not be changed. These two files must be consistent because they are used by the Hadoop nodes.

The `/usr/lpp/mmfs/hadoop/etc/hadoop/slaves` configuration file must be modified to add in all the HDFS Transparency DataNode hostnames. Add one hostname per line, for example:

```
# cat /usr/lpp/mmfs/hadoop/etc/hadoop/slaves
hs22n44
hs22n54
hs22n45
```

Check the `/usr/lpp/mmfs/hadoop/etc/hadoop/log4j.properties` and modify it as

needed.This file can be different than the log4j.properties used by the Hadoop nodes.

After all configuration changes have been made, use the following command to synchronize the configuration files to all the IBM Spectrum Scale HDFS Transparency nodes:
```
[hdfs_transparency_node1#/usr/lpp/mmfs/hadoop/sbin/mmhadoopctl
connector syncconf /usr/lpp/mmfs/hadoop/etc/hadoop
```

Configuring IBM BigInsights IOP

In IBM BigInsights IOP 4.0, IOP and IBM Spectrum Scale HDFS Transparency are integrated manually.

For IBM BigInsights IOP 4.1, if the deployment was done with the old Hadoop connector with IBM Spectrum Scale Ambari integration package gpfs.ambari-iop_4.1-X.X.noarch.bin, see *Upgrade IOP 4.1 + Ambari from Hadoop connector to HDFS Transparency Guide*.

If you deployed IOP 4.1 with the old Hadoop connector but without IBM Spectrum Scale Ambari integration package gpfs.ambari-iop_4.1-X.X.noarch.bin, perform the following steps to move to HDFS Transparency:

1. On *hdfs_transparency_node1*, run the following command to synchronize IBM BigInsights IOP configuration into the IBM Spectrum Scale HDFS Transparency configuration directory:
   `/usr/lpp/mmfs/hadoop/sbin/mmhadoopctl connector syncconf /etc/hadoop/conf/`
2. On *hdfs_transparency_node1*, create the `/usr/lpp/mmfs/hadoop/etc/hadoop/gpfs-site.xml file` and update the `/usr/lpp/mmfs/hadoop/etc/hadoop/slaves` and `/usr/lpp/mmfs/hadoop/etc/hadoop/log4j.properties` files.
   Note: See Configurations for Apache Hadoop.
3. On the node *hdfs_transparency_node1*, run the `/usr/lpp/mmfs/hadoop/sbin/mmhadoopctl connector syncconf /usr/lpp/mmfs/ hadoop/etc/hadoop/` command to synchronize the `gpfs-site.xml`, `core-site.xml`, `hdfs-site.xml`, `slaves` and `log4j.properties` to all the IBM Spectrum Scale HDFS Transparency nodes.

If you plan to deploy IBM BigInsights 4.1/4.2, follow the corresponding Deploying BigInsights IBM Spectrum Scale HDFS Transparency with Ambari guide under the IBM developerWorks Reference wiki page.

4.2.3.4 Update environment variables for HDFS Transparency service

In some situations, the administrator might have to update some environment variables for the HDFS Transparency service. For example, change the JVM options or the Hadoop environment variables like

HADOOP_LOG_DIR.

Follow these steps:

1. On the HDFS Transparency NameNode, modify the /usr/lpp/mmfs/Hadoop/etc/hadoop/hadoop-env.sh and other files as necessary.
2. Sync the changes to all the HDFS Transparency nodes by executing the following command:
   ```
   #/usr/lpp/mmfs/hadoop/sbin/mmhadoopctl connector syncconf
   /usr/lpp/mmfs/hadoop/etc/hadoop
   ```

4.3. Start and stop the service

On the NameNode, start the HDFS Transparency service by running the following command:

```
# /usr/lpp/mmfs/hadoop/sbin/mmhadoopctl connector start
```

Stop the service by running the following command:

```
# /usr/lpp/mmfs/hadoop/sbin/mmhadoopctl connector stop
```

Note: Only the *root* user can start and stop the HDFS Transparency services. Also, you must keep native HDFS service turned off because the HDFS Transparency provides the same services. If you keep both services running, it will report a conflict in the service network port number. You need to restart all other Hadoop services, such as Yarn, Hive, HBase after you replace the native HDFS service with HDFS Transparency.

4.4. Health check the service

Run the following commands:

```
# /usr/lpp/mmfs/hadoop/sbin/mmhadoopctl connector getstate
# hadoop dfs -ls /
```

If you see the configured nodes are running HDFS Transparency NameNode and DataNode services and there are files output from the "`hadoop dfs -ls /`" command, then the setup is successful.

Note: All users can run the above commands.

5. How application interact with HDFS Transparency

The Hadoop applications interact with HDFS Transparency similar to their interactions with native

HDFS. They can access data in the IBM Spectrum Scale filesystem using Hadoop file system APIs and Distributed File System APIs.

The application might have its own cluster that is larger than the HDFS Transparency cluster. However, all the nodes within the application cluster must be able to connect to all the nodes in the HDFS Transparency cluster by RPC.

Yarn can define the nodes in the cluster using the slave files. However, HDFS protocol can use a set of configuration files that are different from Yarn. In this scenario, the slave files in HDFS Transparency can be different from the ones in Yarn.

5.1. Application interface

In HDFS Transparency, applications can use the APIs defined in the `org.apache.hadoop.fs.FileSystem` class and the `org.apache.hadoop.fs.AbstractFileSystem` class to access the file system.

5.2. Command line

The HDFS shell command line can be used with HDFS Transparency.

You can run commands from the HDFS command shell:

$YOUR_HADOOP_PREFIX/bin/hdfs

```
Usage: hdfs [--config confdir] COMMAND
       where COMMAND is one of:
  dfs                  run a filesystem command on the file systems
supported in Hadoop.
  namenode -format     format the DFS filesystem
  secondarynamenode    run the DFS secondary namenode
  namenode             run the DFS namenode
  journalnode          run the DFS journalnode
  zkfc                 run the ZK Failover Controller daemon
  datanode             run a DFS datanode
  dfsadmin             run a DFS admin client
  haadmin             run a DFS HA admin client
  fsck                run a DFS filesystem checking utility
  balancer            run a cluster balancing utility
  jmxget               get JMX exported values from NameNode or DataNode.
  mover                run a utility to move block replicas across
                       storage types
  oiv                 apply the offline fsimage viewer to an fsimage
  oiv_legacy           apply the offline fsimage viewer to an legacy
fsimage
  oev                 apply the offline edits viewer to an edits file
  fetchdt              fetch a delegation token from the NameNode
  getconf             get config values from configuration
  groups              get the groups which users belong to
  snapshotDiff         diff two snapshots of a directory or diff the
```

```
                    current directory contents with a snapshot
  lsSnapshottableDir   list all snapshottable dirs owned by the current
user
                                                Use -help to see options
  portmap            run a portmap service
  nfs3               run an NFS version 3 gateway
  cacheadmin          configure the HDFS cache
  crypto             configure HDFS encryption zones
  storagepolicies     list/get/set block storage policies
  version            print the version
```

Most commands print help when invoked without parameters.

Note: All commands from `hdfs dfs` are supported ("`hdfs dfs -du`" and "`hdfs dfs -df`" are not exact in the output in HDFS Transparency 2.7,0-x. However, these issues have been fixed in HDFS Transparency version 2.7.2-0). **Other commands from HDFS interface are not supported**, such as `hdfs namenode -format`, because these commands are not needed for IBM Spectrum Scale.

6. Upgrading the HDFS Transparency cluster

Before upgrading the HDFS Transparency connector, you need to remove the older IBM Spectrum Scale Hadoop connector.

6.1. Removing IBM Spectrum Scale Hadoop connector

The following sections describe how to remove IBM Spectrum Scale Hadoop connector based on the version of the Hadoop connector and the IBM Spectrum Scale version. Refer to the section that pertain to your setup environment.

6.1.1. Removing IBM Spectrum Scale Hadoop connector 2.4 over IBM Spectrum Scale 4.1.0.4, 4.1.0.5, 4.1.0.6, or 4.1.0.7 releases

For users who are using IBM Spectrum Scale Hadoop connector 2.4 over IBM Spectrum Scale 4.1.0.4, 4.1.0.5, 4.1.0.6, or 4.1.0.7 releases, this section explains the steps required to remove the old connector over each node in the cluster.

Before you begin

If you are using Hadoop 1.x, IBM Spectrum Scale Hadoop Connector 2.7 does not support Hadoop

1.x and you need to upgrade your Hadoop version first.

Procedure

1. Remove any links or copies of the `hadoop-gpfs-2.4.jar` file from your Hadoop distribution directory. Also, remove any links or copies of the `libgpfshadoop.64.so` file from your Hadoop distribution directory.

   Note: For IBM BigInsights IOP 4.0, the distribution directory is under `/usr/iop/4.0.0.0`.
2. Stop the current connector daemon:

   ```
   # ps -elf | grep gpfs-connector-daemon
   # kill -9 <pid-of-connector-daemon>
   ```
3. Run the following commands, to remove callbacks from IBM Spectrum Scale:

   ```
   # cd /usr/lpp/mmfs/fpo/hadoop-2.4/install_script
   # ./gpfs-callbacks.sh --delete
   ```

   Run the `mmlscallbacks all` command to check whether the connector-related callbacks, such as callback ID start-connector-daemon and stop-connector-daemon, have been removed. The IBM Spectrum Scale Hadoop connector callbacks are cluster-wide and this step is required to be done from any one of the nodes.
4. Remove the following files:

   ```
   # rm -f /var/mmfs/etc/gpfs-callbacks.sh
   # rm -f /var/mmfs/etc/gpfs-callback_start_connector_daemon.sh
   # rm -f /var/mmfs/etc/gpfs-callback_stop_connector_daemon.sh
   # rm -f /var/mmfs/etc/gpfs-connector-daemon
   ```
5. Remove the IBM Spectrum Scale-specific configuration from your Hadoop `core-site.xml` file. Modify the `fs.defaultFS` into a HDFS schema format after removing the following configurations:

   ```
   fs.AbstractFileSystem.gpfs.impl,
   fs.AbstractFileSystem.hdfs.impl, fs.gpfs.impl, fs.hdfs.impl,
   gpfs.mount.dir, gpfs.supergroup
   ```

Next steps

Install and set up the HDFS Transparency, see “Upgrading the HDFS Transparency cluster”.

6.1.2. Removing IBM Spectrum Scale Hadoop connector 2.4 over IBM Spectrum Scale 4.1.0.7 (efix3) or 4.1.0.8 releases

For users who are using IBM Spectrum Scale Hadoop connector 2.4 over IBM Spectrum Scale 4.1.0.7 (efix3) or 4.1.0.8 releases, this section explains the steps required to remove the old connector over each node in the cluster.

Before you begin

If you are using Hadoop 1.x, IBM Spectrum Scale Hadoop Connector 2.7 does not support Hadoop 1.x and you must upgrade your Hadoop version first.

Procedure

1. To stop the connector services, run "mmhadoopctl connector stop" on all nodes.
2. To detach the connector, run "mmhadoopctl connector detach --distribution BigInsights" on any one node.
3. To uninstall the connector, run "rpm -e gpfs.hadoop-2-connector" on all nodes.
4. Remove the IBM Spectrum Scale-specific configuration from the Hadoop `core-site.xml` file. Modify the `fs.defaultFS` into a HDFS schema format after removing the following configurations:

   `fs.AbstractFileSystem.gpfs.impl, fs.AbstractFileSystem.hdfs.impl, fs.gpfs.impl, fs.hdfs.impl, gpfs.mount.dir, gpfs.supergroup`

Next steps

Install and set up the HDFS protocol, see "Upgrading the HDFS Transparency Cluster".

6.1.3. Removing IBM Spectrum Scale Hadoop connector 2.4 or 2.5 over IBM Spectrum Scale 4.1.1+ releases

For users who are using IBM Spectrum Scale Hadoop connector 2.4 or 2.5 over IBM Spectrum Scale 4.1.1+ releases, this section explains the steps required to remove the old connector over each node in the cluster.

Before you begin

If you are using Hadoop 1.x, IBM Spectrum Scale Hadoop Connector 2.7 does not support Hadoop 1.x and you need to upgrade your Hadoop version first.

Procedure

1. To stop the connector service, run "mmhadoopctl connector stop" on all nodes.
2. To detach the connector, run "mmhadoopctl connector detach --distribution BigInsights" on all nodes.
3. To detach the connector, run "rpm -e gpfs.hadoop-2-connector" on all nodes.

4. Remove the IBM Spectrum Scale-specific configuration from the Hadoop `core-site.xml` file. Modify the `fs.defaultFS` to a HDFS schema format after removing the following configurations:

   ```
   fs.AbstractFileSystem.gpfs.impl,
   fs.AbstractFileSystem.hdfs.impl, fs.gpfs.impl, fs.hdfs.impl,
   gpfs.mount.dir, gpfs.supergroup
   ```

Next steps

Install and set up the HDFS protocol, see "Upgrading the HDFS Transparency Cluster" for more information.

6.1.4. Removing IBM Spectrum Scale Hadoop connector 2.7 (earlier release) over IBM Spectrum Scale 4.1.0.7 (efix3+), 4.1.0.8, or 4.1.1+ releases

For users who are using IBM Spectrum Scale Hadoop connector 2.7 (earlier release) over IBM Spectrum Scale 4.1.0.7 (efix3+), 4.1.0.8, or 4.1.1+ releases, this section explains the steps required to remove the old connector over each node in the cluster.

Before you begin

If you are using Hadoop 1.x, IBM Spectrum Scale Hadoop Connector 2.7 does not support Hadoop 1.x and you need to upgrade your Hadoop version first.

Procedure

1. `mmhadoopctl connector stop`
2. `mmhadoopctl connector detach --distribution BigInsights`
3. `rpm -e gpfs.hadoop-2-connector`
4. Remove the IBM Spectrum Scale-specific configuration from the Hadoop `core-site.xml` file. Modify the `fs.defaultFS` into a HDFS schema format after removing the following configurations:

   ```
   fs.AbstractFileSystem.gpfs.impl,
   fs.AbstractFileSystem.hdfs.impl, fs.gpfs.impl, fs.hdfs.impl,
   gpfs.mount.dir, gpfs.supergroup
   ```

Next steps

Install and set up the HDFS protocol, see "Upgrading the HDFS Transparency Cluster" for more information.

6.2. Upgrading the HDFS Transparency cluster

Procedure

1. Back up the configuration, in case of failures.
2. Stop the HDFS protocol service on all nodes by running the following command: `/usr/lpp/mmfs/hadoop/sbin/ mmhadoopctl connector stop`
3. Upgrade the RPM package on each node by running the command: `rpm -U gpfs.hdfs-protocol-2.7.<x>-<y>.x86_64.rpm`. It does not update any configuration files under the `/usr/lpp/mmfs/hadoop/etc/hadoop` directory. The `core-site.xml`, `hdfs-site.xml`, and `slaves` files will not be removed during the upgrade.
4. Start the HDFS Transparency service on all nodes by running the following command: `/usr/lpp/mmfs/hadoop/sbin/ mmhadoopctl connector start`

7. Security

HDFS Transparency has full Kerberos support and has been verified over IBM BigInsights IOP 4.1.0.2/4.2. Refer to the IBM *Spectrum Scale HDFS Transparency Security Guide* in the IBM DeveloperWorks GPFS Wiki.

Also, HDFS Transparency is certificated with IBM Security Guardium DAM (Database Activity Monitoring) to monitor the Hadoop Data Access over IBM Spectrum Scale. See link for more information.

8. Advanced features

8.1. High Availability

8.1.1. Configuration for manual HA switch

High Availability (HA) is implemented in HDFS Transparency by using a shared directory in the IBM Spectrum Scale filesystem.

In the following configuration example, the HDFS nameservice ID is *mycluster* and the NameNode

IDs are *nn1* and *nn2*.

Step1: Define the nameservice ID in the `core-site.xml` file that is used by your Hadoop distribution. If you are using IBM BigInsights IOP, change this configuration in the Ambari GUI and restart the HDFS services to synchronize it with all the Hadoop nodes.

```
<property>
<name>fs.defaultFS</name>
<value>hdfs://mycluster</value>
</property>
```

Step2: Configure the `hdfs-site.xml` file that is used by your Hadoop distro. If you are using IBM BigInsights IOP, change these configurations in the Ambari GUI and restart the HDFS services to synchronize it with all the Hadoop nodes.

```
<property>
<!--define dfs.nameservices ID-->
<name>dfs.nameservices</name>
<value>mycluster</value>
</property>

<property>
<!--define name nodes ID for HA-->
<name>dfs.ha.namenodes.mycluster</name>
<value>nn1,nn2</value>
</property>

<property>
<!--Actual hostname and rpc address of name node ID-->
<name>dfs.namenode.rpc-address.mycluster.nn1</name>
<value>c8f2n06.gpfs.net:8020</value>
</property>

<property>
<!--Actual hostname and rpc address of name node ID-->
<name>dfs.namenode.rpc-address.mycluster.nn2</name>
<value>c8f2n07.gpfs.net:8020</value>
</property>

<property>
<!--Actual hostname and http address of name node ID-->
<name>dfs.namenode.http-address.mycluster.nn1</name>
<value>c8f2n06.gpfs.net:50070</value>
</property>

<property>
```

```
<!--Actual hostname and http address of name node ID-->
<name>dfs.namenode.http-address.mycluster.nn2</name>
<value>c8f2n07.gpfs.net:50070</value>
</property>

<property>
<!--Shared directory used for status sync up-->
<name>dfs.namenode.shared.edits.dir</name>
<value>/<gpfs.mnt.dir>/<gpfs.data.dir>/HA</value>
</property>

<property>
<name>dfs.ha.standby.checkpoints</name>
<value>false</value>
</property>


<property>
<name>dfs.client.failover.proxy.provider.mycluster</name>
<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverP
roxyProvider</value>
</property>
```

The configuration `dfs.namenode.shared.edits.dir` must be consistent with `gpfs.mnt.dir` and `gpfs.data.dir` defined in `/usr/lpp/mmfs/hadoop/etc/hadoop/gpfs-site.xml`. You can create the directory `/<gpfs.mnt.dir>/<gpfs.data.dir>/HA` and change the ownership to *hdfs:hadoop* before starting the HDFS transparency services.

The `dfs.ha.standby.checkpoints` must be set to *false*. Otherwise, you will see a log of exceptions in the standby NameNode logs, such as:

ERROR ha.StandbyCheckpointer (StandbyCheckpointer.java:doWork(371)) - Exception in doCheckpoint.

In HDFS transparency, the NameNode does not maintain any state like fsImage or editLogs as there is in native HDFS. So, there is no need to do checkpoints from the standby NameNode service.

The `dfs.client.failover.proxy.provider.mycluster` configuration parameter must be changed according to the name service ID. In the above example, the name service ID is configured as *mycluster* in `core-site.xml`. Therefore, the configuration name is `dfs.client.failover.proxy.provider.`**`mycluster`**.

Note: If you enable Short Circuit Read in the ***Short Circuit Read Configuration*** section**,** the value for the configuration parameter must be `org.apache.hadoop.`**`gpfs`**`.server.namenode.ha.ConfiguredFailoverProxyProvider`**.**

Step3: See the ***Sync Hadoop configurations*** section to synchronize `core-site.xml` and `hdfs-site.xml` from your Hadoop distribution to any one node that is running HDFS

transparency services, e.g. *HDFS_Transparency_node1*.

Step4: for ***HDFS Transparency 2.7.0-x***, on *HDFS_Transparency_node1*, modify the `/usr/lpp/mmfs/hadoop/etc/hadoop/hdfs-site.xml`:

```
<property>
<name>dfs.client.failover.proxy.provider.mycluster</name>
<value>org.apache.hadoop.gpfs.server.namenode.ha.ConfiguredFailoverP
roxyProvider</value>
</property>
```

With this configuration, WebHDFS service functions correctly when NameNode HA is enabled.

Note: On HDFS transparency nodes, the configuration value of the `dfs.client.failover.proxy.provider.mycluster` key in `hdfs-site.xml` is different from that in **Step2**.

Note: this step shouldn't be done since ***HDFS Transparency 2.7.2-x***.

Step5: On *HDFS_Transparency_node1*, run the command as the root user to synchronize the HDFS Transparency configuration to all the HDFS transparency nodes:

```
# mmhadoopctl connector syncconf /usr/lpp/mmfs/hadoop/etc/hadoop
```

Step6: Start the HDFS Transparency service by running the `mmhadoopctl` command:

```
# mmhadoopctl connector start
```

Step7: After the service starts, both NameNodes are in the standby mode by default. You can activate one NameNode by using the following command so that it responds to the client:

```
# /usr/lpp/mmfs/hadoop/bin/gpfs haadmin -transitionToActive
--forceactive [name node ID]
```

For example, you can activate the nn1 NameNode by running the following command:

```
# /usr/lpp/mmfs/hadoop/bin/gpfs haadmin -transitionToActive
-forceactive nn1
```

If the nn1 NameNode fails, you can activate another NameNode and relay the service by running the following command:

```
# /usr/lpp/mmfs/hadoop/bin/gpfs haadmin -transitionToActive
-forceactive nn2
```

Note: The switch must be done manually. Automatic switch will be supported in the future releases.

Use the following command to view the status of the NameNode:

```
# /usr/lpp/mmfs/hadoop/bin/gpfs haadmin -getServiceState [name node ID]
```

You could check your /usr/lpp/mmfs/hadoop/etc/hadoop/hdfs-site.xml or run the following commands to figure out the [name node ID]:

```
#/usr/lpp/mmfs/hadoop/bin/gpfs getconf -confKey fs.defaultFS
hdfs://mycluster
#hdfs getconf -confKey dfs.ha.namenodes.mycluster
nn1,nn2
```

After one NameNode becomes active, you can start the other Hadoop components, such as hbase and hive and run the Hadoop jobs.

Note: When HA is enabled for HDFS transparency, you might see the following exception in the logs:

Get corrupt file blocks returned error: Operation category READ is not supported in state standby.

These are known HDFS issues HDFS-3447 and HDFS-8910.

8.1.2. Configuration for automatic NameNode HA

Automatic NameNode Service HA is supported in gpfs.hdfs-protocol 2.7.0-2 and later. The implementation of high availability (HA) is similar to NFS-based HA in native HDFS except that the NFS shared directory is not needed for HDFS Transparency. The prerequisite to configure automatic NameNode HA is that the zookeeper services must be running in the cluster.

If a Hadoop distribution, such as IBM BigInsights IOP, is running, the zookeeper service is deployed by default. However, if you are running open source Apache Hadoop, you must set up the zookeeper service by following the instructions on the zookeeper website.

After you set up the zookeeper service, perform the following steps to configure automatic NameNode HA.

Note: In the following configuration example, HDFS Transparency NameNode service ID is *mycluster* and NameNode IDs are *nn1* and *nn2*. ZooKeeper server *zk1.gpfs.net*, *zk2.gpfs.net* and *zk3.gpfs.net* are configured to support automatic NameNode HA. The ZooKeeper servers must be started before starting the HDFS Transparency cluster.

Step1: Define the NameNode service ID in `core-site.xml` used by your Hadoop distribution (if IBM BigInsights IOP is running, this configuration change can be made in the Ambari GUI, followed by a restart of the HDFS service to sync the change to all the Hadoop nodes:

```
<property>
<name>fs.defaultFS</name>
<value>hdfs://mycluster</value>
</property>
```

Step2: Configure the `hdfs-site.xml` file used by your Hadoop distribution. If IBM BigInsights IOP is running, these configuration changes can be made in the Ambari GUI, followed by a restart of the HDFS service to synchronize the changes to all the Hadoop nodes:

```
<property>
<!--define dfs.nameservices ID-->
<name>dfs.nameservices</name>
<value>mycluster</value>
</property>

<property>
<!--define name nodes ID for HA-->
<name>dfs.ha.namenodes.mycluster</name>
<value>nn1,nn2</value>
</property>

<property>
<!--Actual hostname and rpc address of name node ID-->
<name>dfs.namenode.rpc-address.mycluster.nn1</name>
<value>c8f2n06.gpfs.net:8020</value>
</property>

<property>

<!--Actual hostname and rpc address of name node ID-->
<name>dfs.namenode.rpc-address.mycluster.nn2</name>
<value>c8f2n07.gpfs.net:8020</value>
</property>

<property>
<!--Actual hostname and http address of name node ID-->
<name>dfs.namenode.http-address.mycluster.nn1</name>
<value>c8f2n06.gpfs.net:50070</value>
</property>

<property>
<!--Actual hostname and http address of name node ID-->
<name>dfs.namenode.http-address.mycluster.nn2/name>
 <value>c8f2n07.gpfs.net:50070</value>
</property>

<property>
<!--Shared directory used for status sync up-->
<name>dfs.namenode.shared.edits.dir</name>
```

```
<value>/<gpfs.mnt.dir>/<gpfs.data.dir>/HA</value>
</property>

<property>
<name>dfs.ha.standby.checkpoints</name>
<value>false</value>
</property>


<property>
<name>dfs.client.failover.proxy.provider.mycluster</name>
<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverP
roxyProvider</value>
</property>

<property>
<name>dfs.ha.fencing.methods</name>
<value>shell(/bin/true)</value>
</property>

<property>
<name>dfs.ha.automatic-failover.enabled</name>
<value>true</value>
</property>

<property>
<name>ha.zookeeper.quorum</name>
<value>zk1.gpfs.net:2181,zk2.gpfs.net:2181,zk3.gpfs.net:2181</value>
</property>
```

The configuration `dfs.namenode.shared.edits.dir` must be consistent with `gpfs.mnt.dir` and `gpfs.data.dir` defined in `/usr/lpp/mmfs/hadoop/etc/hadoop/gpfs-site.xml`. You can create the `/<gpfs.mnt.dir>/<gpfs.data.dir>/HA` directory and change the ownership to *hdfs:hadoop* before starting the HDFS transparency services.

The `dfs.ha.standby.checkpoints` must be set to *false*. If it is not set to *false*, a log of exceptions will be generated in the standby NameNode logs.

ERROR ha.StandbyCheckpointer (StandbyCheckpointer.java:doWork(371)) - Exception in doCheckpoint.

In HDFS transparency, the NameNode does not maintain any state like fsImage or editLogs as there is in native HDFS. So, there is no need to perform checkpoints from the standby NameNode service.

The configuration parameter `dfs.client.failover.proxy.provider.mycluster` must be changed according to the nameservice ID. In the above example, the nameservice ID is configured as mycluster in `core-site.xml`. Therefore, the configuration name is `dfs.client.failover.proxy.provider.`**`mycluster`**.

Note: If you enable Short Circuit Read in the ***Short Circuit Read Configuration*** section**,** the value for this configuration parameter must be `org.apache.hadoop.gpfs.server.namenode.ha.ConfiguredFailoverProxyProvider`**.**

Step3: Follow the guide in the ***Sync Hadoop configurations*** section to synchronize `core-site.xml` and `hdfs-site.xml` from your Hadoop distribution to any one node running HDFS transparency services, such as *HDFS_Transparency_node1*.

Step4: for ***HDFS Transparency 2.7.0-x***, on *HDFS_Transparency_node1,* modify the `/usr/lpp/mmfs/hadoop/etc/hadoop/hdfs-site.xml`*:*

```
<property>
<name>dfs.client.failover.proxy.provider.mycluster</name>
<value>org.apache.hadoop.gpfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
</property>
```

This configuration ensures that the WebHDFS service is working when the NameNode HA is enabled.

Note: On HDFS transparency nodes, the above configuration value in `hdfs-site.xml` is different from that in **Step2**.

Note: This step shouldn't be done since HDFS Transparency 2.7.2-0.

Step5: On *HDFS_Transparency_node1,* run the following command as the *root* user to synchronize the HDFS Transparency configuration to all HDFS transparency nodes:

```
# mmhadoopctl connector syncconf /usr/lpp/mmfs/hadoop/etc/hadoop
```

Step6: Start the HDFS Transparency service by using the `mmhadoopctl` command:

```
# mmhadoopctl connector start
```

Step 7: Format the zookeeper data structure:

```
/usr/lpp/mmfs/hadoop/bin/gpfs --config /usr/lpp/mmfs/hadoop/etc/hadoop/ zkfc -formatZK
```

This step is only needed when you start HDFS Transparency service for the first time. After that, this step is not needed when restarting HDFS Transparency service.

Step 8: Start the zkfc daemon:

```
/usr/lpp/mmfs/hadoop/sbin/hadoop-daemon.sh start zkfc
```

Run `jps` on the name node nn1 and nn2 to check that a process named DFSZKFailoverController has been started by the command.

Note: If the option -formatZK is not specified during the first run, it will result in the following exception:

FATAL org.apache.hadoop.ha.ZKFailoverController: Unable to start failover controller. Parent znode does not exist

Step 9: Check the state of the services

Run the following command to check that all NameNode services and DataNode services are up:

```
# mmhadoopctl connector getstate
```

Run the command to check the state of NameNode services:

```
# /usr/lpp/mmfs/hadoop/bin/gpfs haadmin -getServiceState [name node ID]
```

You could check your /usr/lpp/mmfs/hadoop/etc/hadoop/hdfs-site.xml or run the following commands to figure out the [name node ID]:

```
#/usr/lpp/mmfs/hadoop/bin/gpfs getconf -confKey fs.defaultFS
hdfs://mycluster
#hdfs getconf -confKey dfs.ha.namenodes.mycluster
nn1,nn2
```

Note: When HA is enabled for HDFS transparency, you might see the following exception in logs: *Get corrupt file blocks returned error: Operation category READ is not supported in state standby.* These are known HDFS issues HDFS-3447 and HDFS-8910.

8.2. Short circuit read configuration

In HDFS, read requests go through the DataNode. When the client asks the DataNode to read a file, the DataNode reads the file off of the disk and sends the data to the client over a TCP socket. The short-circuit read obtains the file descriptor from the DataNode, allowing the client to read the file directly.

This is possible only in cases where the client is co-located with the data and used in the FPO mode. Short-circuit reads provide a substantial performance boost to many applications.

8.2.1. For HDFS Transparency version 2.7.0-x

Note: **Short-circuit local read can only be enabled on Hadoop 2.7.0**. HDFS Transparency versions 2.7.0-x does not support this feature in Hadoop 2.7.1/2.7.2. IBM BigInsights IOP 4.1 uses Hadoop version 2.7.1. Therefore, short circuit cannot be enabled over IBM BigInsights IOP 4.1 if HDFS Transparency 2.7.0-x is used. For more information on how to enable short-circuit reads on other Hadoop versions, contact scale@us.ibm.com.

Configuring short-circuit local read:

To configure short-circuit local reads, enable `libhadoop.so` and use the DFS client shipped by IBM Spectrum Scale HDFS Transparency, the package name is `gpfs.hdfs-protocol`. You cannot use the standard HDFS DFS client to enable the short-circuit mode over HDFS Transparency.

About this task

To enable `libhadoop.so`, compile the native library on the target machine or use the library shipped by IBM Spectrum Scale HDFS Transparency. To compile the native library on the specific machine, do the following steps:

Procedure

1. Download the Hadoop source code from Hadoop community. Untar the package and cd to that directory
2. Build by mvn: `$ mvn package -Pdist,native -DskipTests -Dtar`
3. Copy `hadoop-dist/target/hadoop-2.7.1/lib/native/libhadoop.so.*` to `$YOUR_HADOOP_PREFIX/lib/native/`

   To use the `libhadoop.so` delivered by the HDFS Transparency, copy `/usr/lpp/mmfs/hadoop/lib/native/ libhadoop.so` to `$YOUR_HADOOP_PREFIX /lib/native/libhadoop.so`

   The shipped libhadoop.so is built on x86_64, ppc64, or ppc64le respectively.

   Note: This step must be performedon all nodes running the Hadoop tasks.

Enabling DFS Client:

Procedure

1. On each node that accesses IBM Spectrum Scale in the short-circuit mode, back up **hadoop-hdfs-2.7.0.jar** by using `$ mv $YOUR_HADOOP_PREFIX/share/hadoop/hdfs/hadoop-hdfs-2.7.0.jar $YOUR_HADOOP_PREFIX/share/hadoop/hdfs/hadoop-hdfs-2.7.0.jar.backup`
2. Link **hadoop-gpfs-2.7.0.jar** to classpath using `$ ln -s /usr/lpp/mmfs/hadoop/share/hadoop/hdfs/hadoop-gpfs-2.7.0.jar`

```
$YOUR_HADOOP_PREFIX/share/hadoop/hdfs/hadoop-gpfs-2.7.0.jar
```

3. Update the `core-site.xml` file with the following information:

```
<property>
  <name>fs.hdfs.impl</name>
  <value>org.apache.hadoop.gpfs.DistributedFileSystem</value>
</property>
```

Results

Short-circuit read make use of a UNIX domain socket. This is a special path in the file system that allows the client and the DataNodes to communicate. Set a path to this socket. The DataNode must be able to create this path. However, users other than the HDFS user or *root* must not be able to create this path. Therefore, paths under `/var/run` or `/var/lib` folders are often used.

The client and the DataNode exchange information through a shared memory segment on the `/dev/shm` path. Short-circuit local reads must be configured on both the DataNode and the client. Here is an example configuration.

```
<configuration>
<property>
<name>dfs.client.read.shortcircuit</name>
<value>true</value>
</property>
<property>
<name>dfs.domain.socket.path</name>
<value>/var/lib/hadoop-hdfs/dn_socket</value>
</property>
</configuration>
```

Synchronize all these changes on the entire cluster, and if needed, restart the service.

Note: The `/var/lib/hadoop-hdfs` and `dfs.domain.socket.path` must be created manually by the *root* user before running the short-circuit read. The `/var/lib/hadoop-hdfs` folder must be owned by the *root* user. If not, the DataNode service fails when starting up.

```
# mkdir –p /var/lib/hadoop-hdfs
# chown root:root /var/lib/hadoop-hdfs
# touch /var/lib/hadoop-hdfs/${dfs.dome.socket.path}
# chmod 666 /var/lib/hadoop-hdfs/${dfs.dome.socket.path}
```

The permission control in short-circuit reads is similar to the common user access in HDFS. If you have the permission to read the file, then you can access it through short-circuit read.

8.2.2. For HDFS Transparency version 2.7.2-x

Note: Short Circuit Read configurations in this section is applicable to Apache Hadoop 2.7.1+. For Apache Hadoop, you could take the following steps to enable it. However, for IBM BigInsights IOP 4.2, we recommend to disable this by default and contact scale@us.ibm.com for more information.

Perform the following steps to e**nable the DFS client** and ensure that hdfs-site.xml is configured with the correct `dfs.client.read.shortcircuit and dfs.domain.socket.path` values.

Note: For configuring short-circuit local read,

glibc version must be at least version 2.14.

Enabling the DFS client:

Procedure

1. On each node that accesses IBM Spectrum Scale in the short-circuit mode, back up **hadoop-hdfs-2.7.2-IBM-12.jar** by running

```
$mv
$YOUR_HADOOP_PREFIX/hadoop-hdfs/hadoop-hdfs-2.7.2-IBM-12.jar
$YOUR_HADOOP_PREFIX/hadoop-hdfs/hadoop-hdfs-2.7.2-IBM-12.jar
   .backup
```

2. Link **hadoop-hdfs-2.7.2-IBM-12.jar** to classpath by using `$ ln -s`

```
/usr/lpp/mmfs/hadoop/share/hadoop/hdfs/hadoop-hdfs-2.7.2.jar
$YOUR_HADOOP_PREFIX/hadoop-hdfs/hadoop-hdfs-2.7.2-IBM-12.jar
```

3. Update the `core-site.xml` file with the following information:

```
<property>
  <name>fs.hdfs.impl</name>
  <value>org.apache.hadoop.hdfs.DistributedFileSystem</value>
</property>
```

Results

Short-circuit reads make use of a UNIX domain socket. This is a special path in the file system that allows the client and the DataNodes to communicate. You will need to set a path to this socket. The DataNode must be able to create this path. However, it should not be possible for any user except the HDFS user or *root* to create this path. Therefore, paths under the `/var/run` or `/var/lib` folders are often used.

The client and the DataNode exchange information through a shared memory segment on the `/dev/shm` path. Short-circuit local reads must be configured on both the DataNode and the client. Here is an example configuration.

```
<configuration>
```

```
<property>
<name>dfs.client.read.shortcircuit</name>
<value>true</value>
</property>
<property>
<name>dfs.domain.socket.path</name>
<value>/var/lib/hadoop-hdfs/dn_socket</value>
</property>
</configuration>
```

Synchronize the changes in the entire cluster and restart the Hadoop cluster to ensure that all services are aware of this configuration change. Restart the HDFS Transparency cluster or follow the section Automatic Cconfiguration Rrefresh to refresh the configuration without interrupting the HDFS Transparency service.

Note: The `/var/lib/hadoop-hdfs` and `dfs.domain.socket.path` must be created manually by the *root* user before running the short-circuit read. The `/var/lib/hadoop-hdfs` folder must be owned by the *root* user. If not, the DataNode service will fail while starting up.

```
# mkdir –p /var/lib/hadoop-hdfs
# chown root:root /var/lib/hadoop-hdfs
# touch /var/lib/hadoop-hdfs/${dfs.dome.socket.path}
# chmod 666 /var/lib/hadoop-hdfs/${dfs.dome.socket.path}
```

The permission control in the short-circuit reads is similar to the common user access in HDFS. If you have the permission to read the file, you can access it through short-circuit read.

8.3. Configuring multiple Hadoop clusters over the same file system

Using HDFS Transparency, you can configure multiple Hadoop clusters over the same IBM Spectrum Scale file system. For each Hadoop cluster, you need one HDFS Transparency cluster to provide the filesystem service as shown in Figure 7:

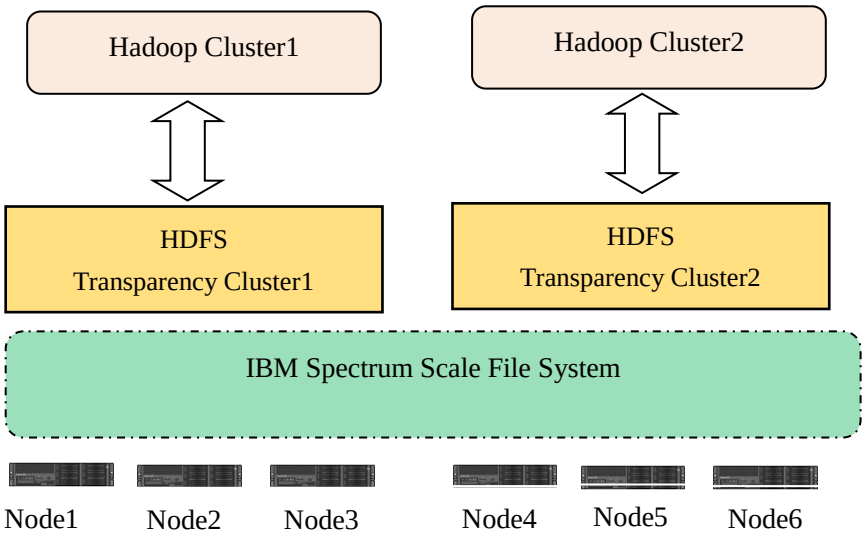


Figure 7: Two Hadoop Clusters over the same IBM Spectrum Scale file system

In this example, configure Node1-Node6 as an IBM Spectrum Scale cluster (FPO or shared storage mode). Configure Node1-Node3 as one HDFS Transparency clusterand Node4-Node6 as another HDFS Transparency cluster. HDFS Transparency Cluster1 and HDFS Transparency Cluster2 take different configurations by changing `/usr/lpp/mmfs/hadoop/etc/hadoop/gpfs-site.xml` to the following:

Step1) Change the `gpfs-site.xml` for HDFS Transparency Cluster1 to store the data under `/<gpfs-mount-point>/<hadoop1>`(gpfs.data.dir=hadoop1 in `gpfs-site.xml`)

Step2) Run `mmhadoopctl connector syncconf /usr/lpp/mmfs/hadoop/etc/hadoop` to synchronize the `gpfs-site.xml` from Step1 to all other nodes in HDFS Transparency Cluster1.

Step3) Change the `gpfs-site.xml` for HDFS Transparency Cluster2 to store the data under `/<gpfs-mount-point>/<hadoop2>`(gpfs.data.dir=hadoop2 in `gpfs-site.xml`).

Step4) Run "`mmhadoopctl connector syncconf /usr/lpp/mmfs/hadoop/etc/hadoop`" to synchronize the `gpfs-site.xml` from Step3 to all other nodes in HDFS Transparency Cluster2

Step5) Restart the HDFS Transparency services

8.4. Docker support

HDFS Transparency supports running the Hadoop Map/Reduce workloads inside Docker containers. See this website for an overview of the Docker technology.

With HDFS Transparency, you can run Hadoop Map/Reduce jobs in Docker and use the IBM Spectrum Scale as a uniform data storage layer over the physical machines, as shown in Figure 8.4-1:

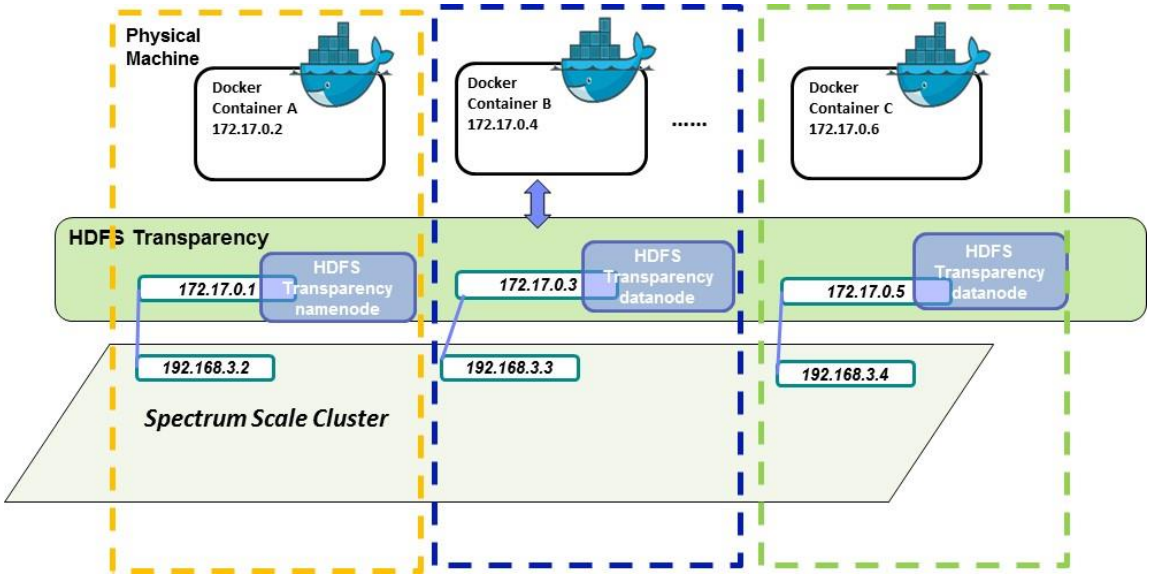


Figure 8.4-1: HDFS Transparency and Docker

You can configure different Docker instances from different physical machines as one Hadoop cluster and run Map/Reduce jobs on the virtual Hadoop clusters. All Hadoop data is stored in the IBM Spectrum Scale file system over physical machines. The 172.17.0.x IP address over each physical machine is one network bridge adapter used for network communication among Docker instances from different physical machines. HDFS Transparency services must be configured to monitor the network bridge and process requests from Docker instances. After receiving the requests from Hadoop jobs running in Docker instances, HDFS Transparency handles the IO requests for the mounted IBM Spectrum Scale file system on the node.

Configuring the Docker instance and HDFS Transparency:

Step1) Docker (version 1.9+) requires Redhat7+. Modify the Redhat Yum Repos to upgrade the selinux-policy and device-mapper-libs by running the following commands:

```
# yum upgrade selinux-policy
# yum upgrade device-mapper-libs
```

Step2) To install Docker engine (version 1.9+), see link .

Step3) Configure the network bridge adapter on physical machines. There can be only one network bridge adapter on one physical machine. Note: These configurations must be changed under /etc/sysconfig/network-scripts/:

```
[root@c3m3n04 network-scripts]# cat ifcfg-br0
DEVICE=br0
TYPE=Bridge
BOOTRPOTO=static
IPADDR=172.17.0.1
NETMASK=255.255.255.0
```

```
ONBOOT=yes

[root@c3m3n04 network-scripts]# cat ifcfg-enp11s0f0
# Generated by dracut initrd
DEVICE="enp11s0f0"
ONBOOT=yes
NETBOOT=yes
UUID="ca481ab0-4cdf-482e-b5d3-82be13a7621c"
IPV6INIT=yes
BOOTPROTO=static
HWADDR="e4:1f:13:be:5c:28"
TYPE=Ethernet
NAME="enp11s0f0"
IPADDR=192.168.3.2
BROADCAST=192.168.255.255
NETMASK=255.255.255.0
```

Note: You must modify the IPADDR, BROADCAST, and NETMASK according to your network configuration.

In this example, the br0 bridge adapter is bundled with the enp11s0f0 physical adapter. You must modify the above configuration for all physical machines on which the Docker instances will be run.

Step4) Modify the Docker service script and start the Docker engine daemons on each physical node:

```
# vim /usr/lib/systemd/system/docker.service
ExecStart=/usr/bin/docker daemon -b br0 -H fd://

# service docker stop
# service docker start
```

Step5) Configure the network route table on each physical machine:

```
# route add -net 172.17.1.0/24 gw <replace-physical-node-ip-here> dev
enp11s0f0
```

where <*replace-physical-node-ip-here*> is the IP address of your machine.

Step6). The IP addresses of the different physical nodes must be different so that the Docker instances from one physical node can access the Docker instances on another physical node.

Check if you can connect to the br0 IP address from another node. If you cannot, that means, you have problems in the network configuration and you need to fix them first.

Step7) Configure HDFS Transparency and start the HDFS Transparency services.

Modify `/usr/lpp/mmfs/hadoop/etc/hadoop/core-site.xml` and `/usr/lpp/mmfs/hadoop/etc/hadoop/slaves`. You must take the IP address from the Docker network bridge adapter (e.g. 172.17.0.x in *Figure8.4-1*).

Pull the Hadoop Docker image onto each physical nodes:

```
# docker pull sequenceiq/hadoop-docker:2.7.0
```

Note: In our verification, we selected the Hadoop Docker image from sequenceiq.

Step8) Start all Docker instances on each physical node by running the following command:

```
# docker run -h <this-docker-instance-hostname> -it
sequenceiq/hadoop-docker:2.7.0 /etc/bootstrap.sh -bash
```

You can start multiple Docker instances over the same physical node. This command will start a Docker instance with the hostname <this-docker-instance-hostname>.

Step9) For each Docker instance, change the `/etc/hosts` to map the Docker instance IP addresses to the hostname:

```
# vi /etc/hosts
172.17.0.2 node1docker1.gpfs.net node1docker1
172.17.0.4 node2docker1.gpfs.net node2docker1
172.17.0.6 node3docker1.gpfs.net node3docker1
```

Note: This must be done on the console of each Docker instance. You must add all the Docker instances if you want to set them up as one Hadoop cluster.

After a Docker instance is stopped, all changes are lost and you have to make this change again after a new Docker instance is started.

Step10) Select a Docker instance and start Yarn ResourceManager on it:

```
# cd /usr/local/hadoop-2.7.0/sbin ./start-yarn.sh
```

You cannot run two ResourceManagers in the same Hadoop cluster. Therefore, run ResourceManager in the selected Docker instance.

Step11) Start Yarn NodeManager on other Docker instances by running the following command:

```
# /usr/local/hadoop-2.7.0/sbin/yarn-daemon.sh --config
/usr/local/hadoop/etc/hadoop/ start nodemanager
```

Step12) Run `hadoop dfs -ls /` to check if you can run Map/Reduce jobs in Docker.

Note: To stop the Yarn services running inside Docker, following these steps:

→on Yarn ResourceManager Docker instance:

```
    # cd /usr/local/hadoop-2.7.0/sbin ./stop-yarn.sh
```

→on Yarn NodeManager Docker instances:

```
    # /usr/local/hadoop-2.7.0/sbin/yarn-daemon.sh --config
/usr/local/hadoop/etc/hadoop/ stop nodemanager
```

Note: When running HDFS transparency with Docker instances, data locality is not supported for the Map/Reduce jobs.

8.5. HDFS Transparency federation

Federation was added to HDFS to improve the HDFS NameNode horizontal scaling. In HDFS transparency, federation is used to make IBM Spectrum Scale filesystems and HDFS filesystem coexist. The Hadoop applications can get input data from the native HDFS, analyze the input and write the output to the IBM Spectrum Scale filesystem. This feature is available in HDFS transparency version 2.7.0-2 (gpfs.hdfs-protocol-2.7.0-2) and later.

Also, the HDFS transparency federation can allow two or more IBM Spectrum Scale file systems act as one uniform file system for Hadoop applications. These file systems can belong to the same cluster or be part of different Spectrum Scale clusters. A typical scenario could be, you need to read data from an existing file system, analyze it, and write the results to a new IBM Spectrum Scale file system.

Note: If you want your applications running in clusterA to process the data in clusterB, only update the configuration for federation in clusterA. This is call federating clusterB with clusterA. If you want your applications running in clusterB to process data from clusterA, you need to update the configuration for federation in clusterB. This is called federating clusterA with clusterB.

This guide provides an overview of the HDFS Federation feature, configuration, and management of the federated cluster.

8.5.1. Federating IBM Spectrum Scale and HDFS

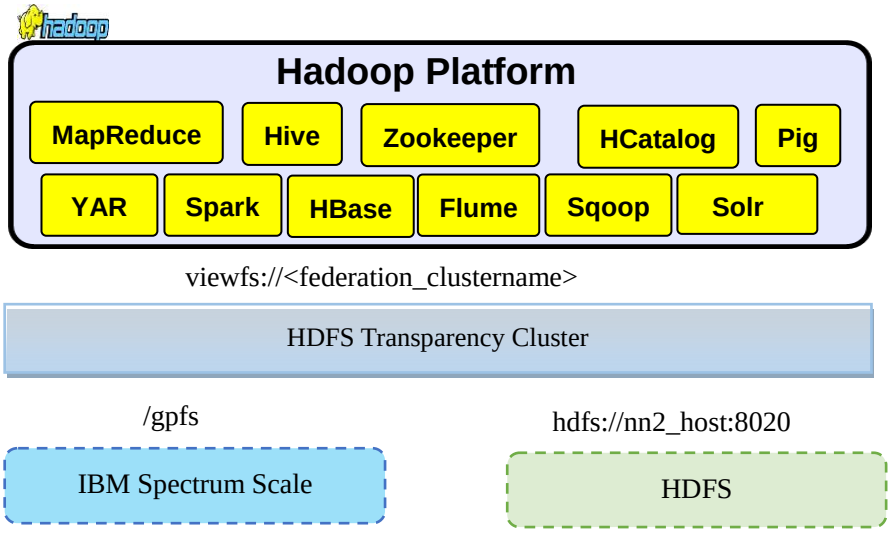


Figure 8.5-1: IBM Spectrum Scale and Native HDFS federation

Before configuring federation, ensure that you have configured the HDFS Transparency cluster (see Section 3 and Section 4). See the following sections to configure federation for IBM Spectrum Scale and Native HDFS.

Note: Section 8.5.1.1 federate native HDFS into HDFS Transparency while section 8.5.1.2 federate HDFS Transparency into native HDFS. Depending on how you want to federate your environment, follow the steps in section 8.5.1.1 or the steps in section 8.5.1.2 or see both sections 8.5.1.1 and 8.5.1.2 so that both clusters are federated with each other.

8.5.1.1 Federating native HDFS with HDFS Transparency

Step 1: Shut down the HDFS Transparency cluster daemon by running the following command from one of the HDFS Transparency nodes in the cluster:

```
# mmhadoopctl connector stop
```

Step 2: On nn1_host, add the following configuration settings in `/usr/lpp/mmfs/hadoop/etc/hadoop/core-site.xml`:

```
<configuration>
<property>
   <name>fs.defaultFS</name>
   <value>viewfs://<federation_clustername> </value>
   <description>The name of the federation file system</description>
</property>
```

```
<property>

<name>fs.viewfs.mounttable.<federation_clustername>.link./<federated_dir1></name>
    <value>hdfs://nn1_host:8020/<mount_dir></value>
    <description>The name of the Spectrum Scale file system</description>
</property>

<property>

<name>fs.viewfs.mounttable.<federation_clustername>.link./<federated_dir2></name>
   <value>hdfs://nn2_host:8020/<mount_dir></value>
   <description>The name of the hdfs file system</description>
</property>
</configuration>
```

Note: Change ***<federation_clustername>*** and ***<mount_dir>*** according to your cluster configuration. In this example, the *nn1_host* refers to the HDFS Transparency NameNode and the *nn2_host* refers to the native HDFS NameNode.

Once the federation configuration changes are in effect on the node, the node will only see the directories that are specified in the core-site.xml file. For the above configurations, you can only see the two directories /<federated_dir1> and /<federated_dir2>.

Step 3: On *nn1_host*, add the following configuration settings in `/usr/lpp/mmfs/hadoop/etc/hadoop/hdfs-site.xml`:

```
<configuration>
<property>
   <name>dfs.nameservices</name>
   <value>nn1,nn2</value>
</property>
<property>
   <name>dfs.namenode.rpc-address.nn1</name>
   <value>nn1-host:8020</value>
</property>
<property>
   <name>dfs.namenode.rpc-address.nn2</name>
   <value>nn2-host:8020</value>
</property>
<property>
   <name>dfs.namenode.http-address.nn1</name>
   <value>nn1-host:50070</value>
```

```
</property>
<property>
    <name>dfs.namenode.http-address.nn2</name>
    <value>nn2-host:50070</value>
</property>

</configuration>
```

Step 4: On *nn1_host*, synchronize the configuration changes with the other HDFS Transparency nodes by running the following command:

```
# mmhadoopctl connector syncconf /usr/lpp/mmfs/hadoop/etc/hadoop/
```

Note: The following output messages from the above command for the native HDFS NameNode, nn2-host, can be seen:

```
scp: /usr/lpp/mmfs/hadoop/etc/hadoop//: No such file or directory
scp: /usr/lpp/mmfs/hadoop/etc/hadoop//: No such file or directory
scp: /usr/lpp/mmfs/hadoop/etc/hadoop//: No such file or directory
scp: /usr/lpp/mmfs/hadoop/etc/hadoop//: No such file or directory
scp: /usr/lpp/mmfs/hadoop/etc/hadoop//: No such file or directory
scp: /usr/lpp/mmfs/hadoop/etc/hadoop//: No such file or directory
scp: /usr/lpp/mmfs/hadoop/etc/hadoop//: No such file or directory
scp: /usr/lpp/mmfs/hadoop/etc/hadoop//: No such file or directory
```

The output messages above are seen because during the synchronization of the configuration to all the nodes in the cluster, the /usr/lpp/mmfs/Hadoop/etc/hadoop directory does not exist in the nn2-host native HDFS NameNode,. This is because the HDFS Transparency is not installed on the native HDFS NameNode. Therefore, these messages for the native HDFS NameNode can be ignored.

Another way to synchronize the configuration files is by using the scp command to copy the following files under /usr/lpp/mmfs/hadoop/etc/hadoop/ into all the other nodes in HDFS Transparency cluster: slaves, log4j.properties, hdfs-site.xml, hadoop-policy.xml, hadoop-metrics.properties, hadoop-metrics2.properties, core-site.xml, and gpfs-site.xml.

Step 5: On *nn1_host*, start all the HDFS Transparency cluster nodes by running the following command:

```
# mmhadoopctl connector start
```

Note: The following warning output messages from the above command for the native HDFS NameNode, nn2-host can be seen:

```
nn2-host: bash: line 0: cd: /usr/lpp/mmfs/hadoop: No such file or directory
nn2-host: bash: /usr/lpp/mmfs/hadoop/sbin/hadoop-daemon.sh: No such file or
directory
```

These messages are displayed because HDFS Transparency is not installed on the native HDFS NameNode. Therefore, these messages can be ignored.

To avoid the above messages, run the following commands:

Step 5.1) On nn1-host, run the following command as root to start the HDFS Transparency NameNode:

```
# cd /usr/lpp/mmfs/hadoop; /usr/lpp/mmfs/hadoop/sbin/hadoop-daemon.sh --config /usr/lpp/mmfs/hadoop/etc/hadoop --script /usr/lpp/mmfs/hadoop/sbin/gpfs start namenode
```

Step5.2) On nn1-host, run the following command as root to start the HDFS Transparency DataNode:

```
# cd /usr/lpp/mmfs/hadoop; /usr/lpp/mmfs/hadoop/sbin/hadoop-daemons.sh --config /usr/lpp/mmfs/hadoop/etc/hadoop --script /usr/lpp/mmfs/hadoop/sbin/gpfs start datanode
```

Warning: If you deployed IBM BigInsights IOP, the Spectrum Scale Ambari integration module (gpfs.hdfs-transparency.ambari-iop_4.1-0**)** does not support federation configuration in Ambari. Therefore, starting the HDFS Transparency service or other services will regenerate the core-site.xml and hdfs-site.xml from the Ambari database and will overwrite the changes that were done from Step 1 to Step 4. HDFS Transparency and all other services will have to be started in command mode.

Step 6: Update the configuration changes in Step 2 and Step 3 in your Hadoop client configurations so that the Hadoop applications can view all the directories in federation.

Note: If you deployed IBM BigInsights IOP, update the core-site.xml and the hdfs-site.xml in Step 2 and Step 3 accordingly from the /etc/hadoop/conf directory on each of the node so that the Hadoop applications are able to see the directories in federation.

If you deployed Open Source Apache Hadoop, and update the core-site.xml and the hdfs-site.xml according to the Apache Hadoop location configured in your site.

Step7: From one of the Hadoop client, verify that the federated directories are available by running the following command: *hadoop dfs -ls /*

8.5.1.2 Federating HDFS Transparency into native HDFS

Step 1: Stop the Hadoop applications and the native HDFS services on the native HDFS cluster. The detailed command is dependent on the Hadoop distro. For example, for IBM BigInsights IOP, stop all services from the Ambari GUI.

Step 2: Perform Step 2 and Step 3 in the section *8.5.1.1 Federate native HDFS into HDFS Transparency* on the node *nn2-host* with the correct path for `core-site.xml` and the `hdfs-site.xml` according to the Hadoop distribution.

If running with the open source Apache Hadoop, the location of the `core-site.xml` and the `hdfs-site.xml` is in `$YOUR_HADOOP_PREFIX/etc/hadoop/`. The `$YOUR_HADOOP_PREFIX` is the location of the Hadoop package. If running with IBM BigInsights IOP, then Ambari currently does not support federation configuration. You will have to manually update the configurations under /etc/Hadoop/conf/.

Note: If you want to see all the directories from the native HDFS after federation, define all the native HDFS directories in the core-site.xml.

If you have a secondary NameNode configured in native HDFS, update the following configuration in the hdfs-site.xml:

```
<property>
  <name>dfs.namenode.secondary.http-address.nn2-host</name>
  <value>secondaryNameNode-host:50090</value>
</property>

<property>
  <name>dfs.secondary.namenode.keytab.file.nn2-host</name>
    <value>/etc/security/keytabs/nn.service.keytab</value>
</property>
```

Note: If you have deployed IBM BigInsights IOP, it will generate the key dfs.namenode.secondary.http-address and dfs.secondary.namenode.keytab.file by default. For federation, modify the hdfs-site.xml file with the correct values according to your environment.

Step 3: Synchronize the updated configurations from the *nn2-host* node to all the other native HDFS nodes and start the native HDFS services.

If running with open source Apache Hadoop, you need to use the scp command to synchronize the `core-site.xml` and the `hdfs-site.xml` from the host *nn2-host* to all the other native HDFS nodes. Start the native HDFS service by running the following command:
`$YOUR_HOME_PREFIX/sbin/start-dfs.sh`

If IBM BigInsights IOP is running, synchronize the updated configurations manually to avoid the updated federation configurations overwritten by Ambari.

Note: Check the configurations under /etc/hadoop/conf to ensure that all the changes have been synchronized to all the nodes.

Step 4: Start the native HDFS service.
If you are running open source Hadoop, start the native HDFS service on the command line:
`$YOUR_HADOOP_PREFIX/bin/start-dfs.sh`

If you deployed IBM BigInsights IOP, Ambari does not support federation configuration. Therefore, you must start the native HDFS services manually.

Starting the native HDFS
Step 4.1) Start native HDFS NameNode.

Log in to nn2-host as root, run su - hdfs to switch to the hdfs uid and then run the following command:

```
/usr/iop/current/hadoop-client/sbin/hadoop-daemon.sh --config
/usr/iop/current/hadoop-client/conf start namenode
```

Step 4.2) Start the native HDFS DataNode.

Log in to the DataNode, run su - hdfs to switch to the hdfs uid and then run the following command:

```
/usr/iop/current/hadoop-client/sbin/hadoop-daemon.sh --config /usr/iop/current/hadoop-client/conf
start datanode
```

Note: Run the above command on each DataNode.

Log in to the SecondaryNameNode, run su - hdfs to switch to the hdfs UID and run the following command to start SecondaryNameNode:

```
/usr/iop/current/hadoop-client/sbin/hadoop-daemon.sh --config /usr/iop/current/hadoop-client/conf
start secondarynamenode
```

Step 5: Update `core-site.xml` and `hdfs-site.xml` used by the Hadoop clients on which the Hadoop applications will run over federation.

If the open source Apache Hadoop is running, the location of `core-site.xml` and `hdfs-site.xml` is in `$YOUR_HADOOP_PREFIX/etc/hadoop/`. The `$YOUR_HADOOP_PREFIX` is the location of the Hadoop package. If another Hadoop distro is running, see Section Known Limit.

If IBM BigInsights IOP is running, core-site.xml and hdfs-site.xml are located in /etc/hadoop/conf/.

Step6: To ensure that the federated file system is functioning correctly, run the
Command `"hadoop fs –ls /"`

8.5.2. Federating two IBM Spectrum Scale file systems

You can federate two IBM Spectrum Scale file systems from different clusters or from the same cluster.

Irrespective of the mode that you select, configure one HDFS transparency cluster for each IBM Spectrum Scale file system (refer the Section 3 and Section 4), and then federate the two HDFS transparency clusters together.

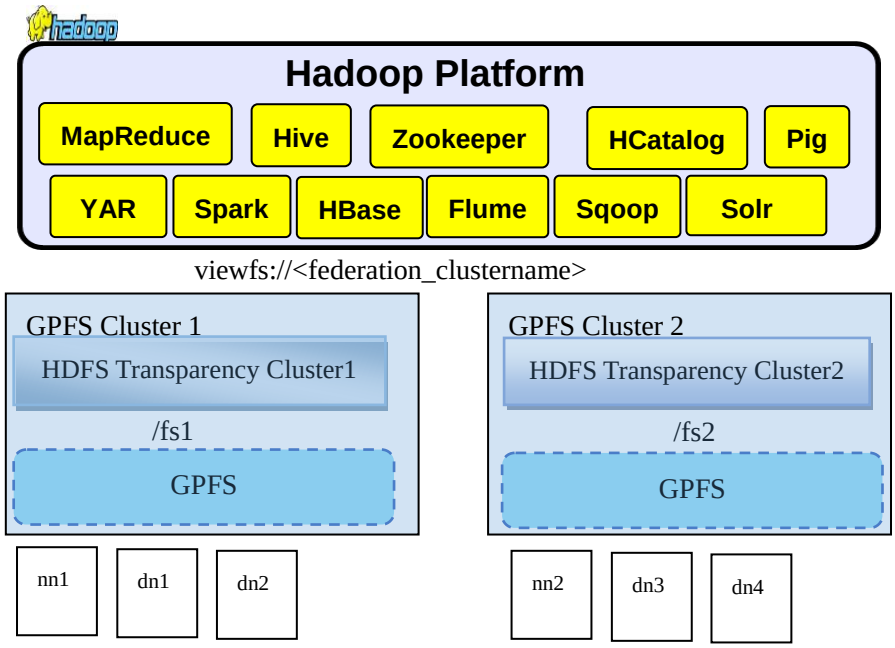


Figure 8.5-2: Federate two IBM Spectrum Scale file systems from different clusters

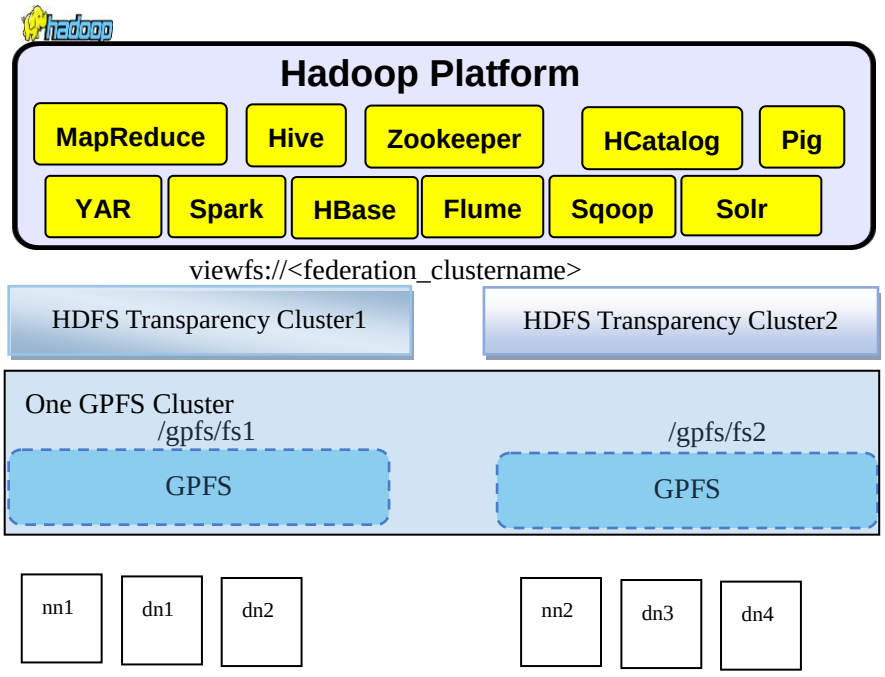


Figure 8.5-3: Federate two IBM Spectrum Scale file systems from the same cluster

To federate two file systems from the same cluster (shown in ***Figure 8.5-3)*** select the nodes to provide

HDFS transparency services for the first file system and the second file system.

Before configuring the federation, see Section 3 and Section 4 to configure HDFS transparency cluster1 and HDFS transparency cluster2 for each file system. After that, perform the following steps to configure the federation:

Step 1: To stop the HDFS Transparency services, run `mmhadoopctl connector stop` on both HDFS transparency clusters.

Step 2: On the host *nn1_host*, add the following configuration settings in `/usr/lpp/mmfs/hadoop/etc/hadoop/core-site.xml`:

```
<configuration>
<property>
   <name>fs.defaultFS</name>
   <value>viewfs://<federation_clustername> </value>
   <description>The name of the federation file system</description>
</property>

<property>

<name>fs.viewfs.mounttable.<federation_clustername>.link./<mount_dir></name>
    <value>hdfs://nn1:8020/<mount_dir></value>
    <description>The name of the gpfs file system</description>
</property>

<property>

<name>fs.viewfs.mounttable.<federation_clustername>.link./<mount_dir></name>
   <value>hdfs://nn2:8020/<mount_dir></value>
   <description>The name of the hdfs file system</description>
</property>
</configuration>
```

Note: Change ***<federation_clustername>*** and ***<mount_dir>*** according to your cluster configuration: *nn1_host* and *nn2_host*.

Step 3: On *nn1_host*, add the following configuration settings in `hdfs-site.xml`:

```
<configuration>
<property>
   <name>dfs.nameservices</name>
   <value>nn1,nn2</value>
</property>
<property>
   <name>dfs.namenode.rpc-address.nn1</name>
```

```
    <value>nn1:8020</value>
</property>
<property>
    <name>dfs.namenode.rpc-address.nn2</name>
    <value>nn2:8020</value>
</property>
<property>
    <name> dfs.namenode.http-address.nn1</name>
    <value>nn1:50070</value>
</property>
<property>
    <name>dfs.namenode.http-address.nn2</name>
    <value>nn2:50070</value>
</property>
</configuration>
```

Step 4: On *nn1*, synchronize the configuration change to other HDFS Transparency nodes.
Note: You cannot take mmhadoopctl connector syncconf /usr/lpp/mmfs/hadoop/etc/hadoop/ to synchronize the updated configurations because this might overwrite the configurations on the NameNode nn2-host in another cluster.

Take the following commands to synchronize the updated configurations from HDFS Transparency nn1-host with all other nodes in the same HDFS Transparency cluster:

```
#login the HDFS Transparency Cluster1 NameNode nn1 as root:
cd /usr/lpp/mmfs/Hadoop/etc/hadoop
scp * <hostX>:/usr/lpp/mmfs/Hadoop/etc/hadoop/
```

Note: The above must be done for each node in the HDFS Transparency Cluster1. For example, change the hostX accordingly and run it for each node in the HDFS Transparency Cluster1.

Step 5: On *nn2*, perform Step1 through Step 4.

Note: If you only want to federate HDFS Transparency Cluster2 into HDFS Transparency Cluster1, Step 5 is not needed.

Step 6: On *nn1*, start the HDFS Transparency cluster:

Step 6.1) On nn1, run the following command as root to start HDFS Transparency Cluster1 NameNode:

```
#cd /usr/lpp/mmfs/hadoop; /usr/lpp/mmfs/hadoop/sbin/hadoop-daemon.sh --config /usr/lpp/mmfs/hadoop/etc/hadoop --script /usr/lpp/mmfs/hadoop/sbin/gpfs start namenode
```

Step 6.2) On nn1, run the following command as root to start HDFS Transparency Cluster1 DataNode:

```
cd /usr/lpp/mmfs/hadoop; /usr/lpp/mmfs/hadoop/sbin/hadoop-daemons.sh --config
```

```
/usr/lpp/mmfs/hadoop/etc/hadoop --script /usr/lpp/mmfs/hadoop/sbin/gpfs start datanode
```

Warning: If you deployed IBM BigInsights IOP, IBM Spectrum Scale Ambari integration package gpfs.hdfs-transparency.ambari-iop_4.1-0 does not support federation configuration on Ambari. Therefore, starting the HDFS Transparency service will re-generate the core-site.xml and hdfs-site.xml from the Ambari database and overwrite the changes you made from Step1 to Step4. Repeat the Step 6.1 and Step 6.2 to start HDFS Transparency in the command mode.

Step 7: On *nn2*, start the other HDFS Transparency cluster:

Step 7.1) On nn2, run the following command as root to start the HDFS Transparency Cluster2 NameNode:

```
#cd /usr/lpp/mmfs/hadoop; /usr/lpp/mmfs/hadoop/sbin/hadoop-daemon.sh --config /usr/lpp/mmfs/hadoop/etc/hadoop --script /usr/lpp/mmfs/hadoop/sbin/gpfs start namenode
```

Step 7.2) On nn2, run the following command as root to start the HDFS Transparency Cluster2 DataNode:

```
cd /usr/lpp/mmfs/hadoop; /usr/lpp/mmfs/hadoop/sbin/hadoop-daemons.sh --config /usr/lpp/mmfs/hadoop/etc/hadoop --script /usr/lpp/mmfs/hadoop/sbin/gpfs start datanode
```

Warning: If you have deployed IBM BigInsights IOP, the IBM Spectrum Scale Ambari integration package gpfs.hdfs-transparency.ambari-iop_4.1-0 does not support federation configuration on Ambari. Therefore, starting the HDFS Transparency service will re-generate the core-site.xml and hdfs-site.xml from Ambari database and overwrite the changes made from step1 to step4. Repeat the Step7.1 and Step 7.2 to start HDFS Transparency in the command mode.

Step 8: Update `core-site.xml` and `hdfs-site.xml` for the Hadoop clients on which the Hadoop applications will run over federation.

If running with open source Apache Hadoop, the location of `core-site.xml` and `hdfs-site.xml` is `$YOUR_HADOOP_PREFIX/etc/hadoop/`. The `$YOUR_HADOOP_PREFIX` is the location of the Hadoop package. If running with another Hadoop distribution, see Section Known Limit.

Step 9: Restart the Hadoop applications on both clusters.

Note: You should always keep the native HDFS service non-functional if you select HDFS Transparency.

Step 10: To ensure that the federated file system feature is functioning correctly, run the `hadoop fs -ls /` command.

8.5.3. Known limits

1. All the changes in `/usr/lpp/mmfs/hadoop/etc/hadoop/core-site.xml` and `/usr/lpp/mmfs/hadoop/etc/hadoop/hdfs-site.xml` must be updated in the configuration files used by the Hadoop distributions. However, **Hadoop distributions manage their configuration and the management interface might not support the key used for federation, such as IBM BigInsights IOP takes Ambari and Ambari GUI does not support some property names** (see Ambari-15455).

   If you want to set up federation for IBM BigInsights IOP, send an email to scale@us.ibm.com.

2. The native HDFS and HDFS transparency cannot be run over the same node because of the network port number conflict.

3. If you select to federate both native HDFS and HDFS transparency, configure the native HDFS cluster and make the native HDFS service function. Configure the federation for native HDFS and HDFS transparency.

   For a new native HDFS cluster, while starting the service for the first time, DataNode registers itself with the NameNode. The HDFS Transparency NameNode does not accept any registration from the native HDFS DataNode. Therefore, an exception occurs if you configure a new native HDFS cluster, federate it with HDFS transparency, and then try to make both clusters (one native HDFS cluster and another HDFS Transparency cluster) function at the same time.

4. Start and stop the native HDFS cluster or the HDFS Transparency cluster separately if you want to maintain both of them.

8.6. Hadoop `distcp` support

Hadoop `distcp` can be used for data migration from HDFS to the IBM Spectrum Scale file system and between two IBM Spectrum Scale file systems.

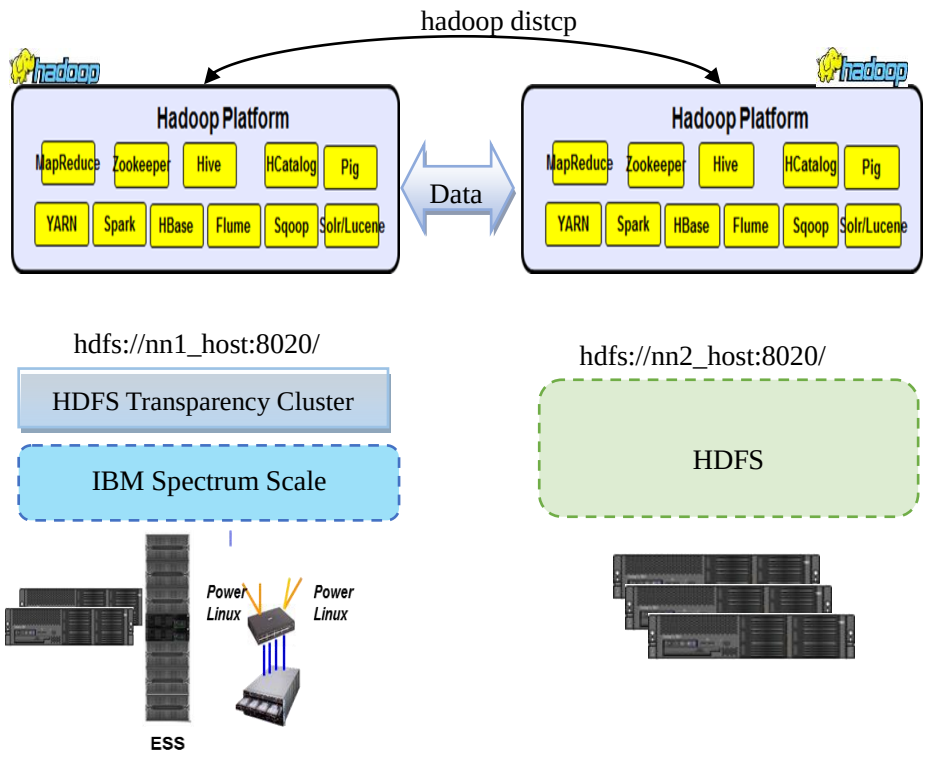


Figure 8.6: distcp-based data migration between HDFS and IBM Spectrum Scale

There are no additional configuration changes. The `hadoop distcp` command is supported in HDFS transparency 2.7.0-2 (gpfs.hdfs-protocol-2.7.0-2) and later.

Example:

```
# hadoop distcp hdfs://nn1_host:8020/source/dir
hdfs://nn2_host.:8020/target/dir
```

Known issues and workarounds:

Issue 1: Permission is denied when running the Hadoop distcp command with the root credential.

The super user *root* in Linux is not a super user for Hadoop automatically. If you do not add the super user *root* to the gpfs.supergroup parameter, the system displays the following error message:

org.apache.hadoop.security.AccessControlException: Permission denied: user=root, access=WRITE, inode="/user/root/.staging":hdfs:hdfs:drwxr-xr-x

at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(FSPermissionChecker.java:319)

Workaround: Configure *root* as a super user. Add *root* to the gpfs.supergroup parameter in gpfs-site.xml or run the related Hadoop distcp command as super user.

Issue 2: Access time exception while copying files from IBM Spectrum Scale to HDFS with the -p option.

```
[hdfs@c8f2n03 conf]$ hadoop distcp -overwrite -p
hdfs://c16f1n03.gpfs.net:8020/testc16f1n03/
hdfs://c8f2n03.gpfs.net:8020/testc8f2n03
```

Error: org.apache.hadoop.ipc.RemoteException(java.io.IOException): Access time for hdfs is not configured. Set the dfs.namenode.accesstime.precision configuration parameter.

at org.apache.hadoop.hdfs.server.namenode.FSDirAttrOp.setTimes(FSDirAttrOp.java:101)

Workaround: Change the `dfs.namenode.accesstime.precision` value from *0* to a value such as *3600000* (1 hour) in `hdfs-site.xml` for the HDFS cluster.

Issue 3: The distcp command fails when the src director is root.

```
[hdfs@c16f1n03 root]$ hadoop distcp hdfs://c16f1n03.gpfs.net:8020/
hdfs://c8f2n03.gpfs.net:8020/test5
```

16/03/03 22:27:34 ERROR tools.DistCp: Exception encountered

java.lang.NullPointerException

at org.apache.hadoop.tools.util.DistCpUtils.getRelativePath(DistCpUtils.java:144)

at org.apache.hadoop.tools.SimpleCopyListing.writeToFileListing(SimpleCopyListing.java:353)

Workaround: Specify at least one directory or file at the source directory.

Issue 4: The distcp command throws NullPointerException when the target directory is root in federation configuration but the job is completed.

This is not a real issue. See https://issues.apache.org/jira/browse/HADOOP-11724 for more detail. Note: This will not impact your data copy.

8.7. Automatic configuration refresh

This feature is available in gpfs.hdfs-protocol 2.7.0-2 and later. After changing configuration parameters under `/usr/lpp/mmfs/hadoop/etc/hadoop` or changing parameters for the IBM Spectrum Scale file system, such as Maximum number of Replicas and NSD server, the following command can be run to have these changes take effect without restarting the HDFS Transparency services:

```
# /usr/lpp/mmfs/hadoop/bin/gpfs dfsadmin -refresh
<namenode_hostname>:<port > refreshGPFSConfig
```

This command can be run from any of the HDFS Transparency nodes by changing the `<namenode_hostname>:<port>` according to the HDFS Transparency configuration. For example, if the `fs.defaultFS` is *hdfs://c8f2n03.gpfs.net:8020* in `/usr/lpp/mmfs/hadoop/etc/hadoop/core-site.xml`, `<namenode_hostname>` must be replaced with *c8f2n03.gpfs.net* and `<port>` with *8020*. HDFS Transparency synchronizes these configuration changes to HDFS Transparency services running on all the HDFS Transparency nodes and makes it immediately effective.

8.8. Ranger support

HDFS authorization can use POSIX style permissions (aka. HDFS ACLs) or use Apache Ranger. Apache Ranger (http://hortonworks.com/hadoop/ranger/) is a centralized security administration solution for Hadoop that enables administrators to create and enforce security policies for HDFS and other Hadoop platform components.

Ranger requires to be installed in native HDFS then configure for HDFS Transparency.

8.8.1. Installing Ranger in native HDFS

There are three steps to install Ranger using IOP 4.2 and Ambari 2.2 User Interface (UI) :

- Configuring MySQL for Ranger (installation prerequisites)
- Installing Ranger
- Enabling the Ranger HDFS plugin

8.8.1.1 Configuring MySQL for Ranger

Before proceeding with the installation, prepare your environment by performing the following steps:

Step1) Create an IOP 4.2 Hadoop cluster, run service check to ensure that the environment is running properly.

Step2) Configure MySQL for Ranger:

1) Create a non-root user to create the Ranger databases. In this example, the username rangerdba with password rangerdba will be used.

a. Log in as the root user to the **DB host node.** This is the node that has MySQL installed which is usually the Hive server node. Use the following commands to create the rangerdba user and grant the user adequate privileges.

```
CREATE USER 'rangerdba'@'localhost' IDENTIFIED BY 'rangerdba';

GRANT ALL PRIVILEGES ON *.* TO 'rangerdba'@'localhost';

CREATE USER 'rangerdba'@'%' IDENTIFIED BY 'rangerdba';

GRANT ALL PRIVILEGES ON *.* TO 'rangerdba'@'%';

GRANT ALL PRIVILEGES ON *.* TO 'rangerdba'@'localhost' WITH GRANT OPTION;

GRANT ALL PRIVILEGES ON *.* TO 'rangerdba'@'%' WITH GRANT OPTION;

FLUSH PRIVILEGES;
```

b. Use the exit command to exit MySQL.

c. Reconnect to the database as user rangerdba by using the following command:

```
mysql -u rangerdba -prangerdba
```

After testing the rangerdba login, use the exit command to exit MySQL.

2) Run the following command to confirm that the mysql-connector-java.jar file is in the Java share directory. **This command must be run on the Ambari server node.**

```
ls /usr/share/java/mysql-connector-java.jar
```

3) Use the following command to set the jdbc/driver/path based on the location of the MySQL JDBC driver .jar file. **This command must be run on the Ambari server node.**

```
ambari-server setup --jdbc-db={database-type}
--jdbc-driver={/jdbc/driver/path}
```

For example:
ambari-server setup --jdbc-db=mysql --jdbc-driver=/usr/share/java/mysql-connector-java.jar

Step3) Configure Audit for Ranger:

a. Log in as the root user to the **DB host node.** This is the node that has MySQL installed which is usually the Hive server node. Use the following commands to create the rangerlogger user with password YES and grant the user adequate privileges.

```
CREATE USER 'rangerlogger'@'localhost' IDENTIFIED BY 'YES';

GRANT ALL PRIVILEGES ON *.* TO 'rangerlogger'@'localhost';

CREATE USER 'rangerlogger'@'%' IDENTIFIED BY 'YES';

GRANT ALL PRIVILEGES ON *.* TO 'rangerlogger'@'%';
```

```
GRANT ALL PRIVILEGES ON *.* TO 'rangerlogger'@'localhost' WITH GRANT
OPTION;

GRANT ALL PRIVILEGES ON *.* TO 'rangerlogger'@'%' WITH GRANT OPTION;

FLUSH PRIVILEGES;
```

b. Use the exit command to exit MySQL.

c. Reconnect to the database as user rangerdba by using the following command:

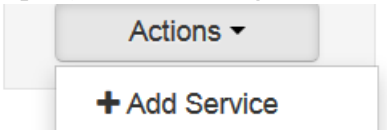
```
mysql -u rangerlogger -pYES
```

8.8.1.2 Installing Ranger

Step1) Start the installation:

Step1.1) Log in to Ambari UI.

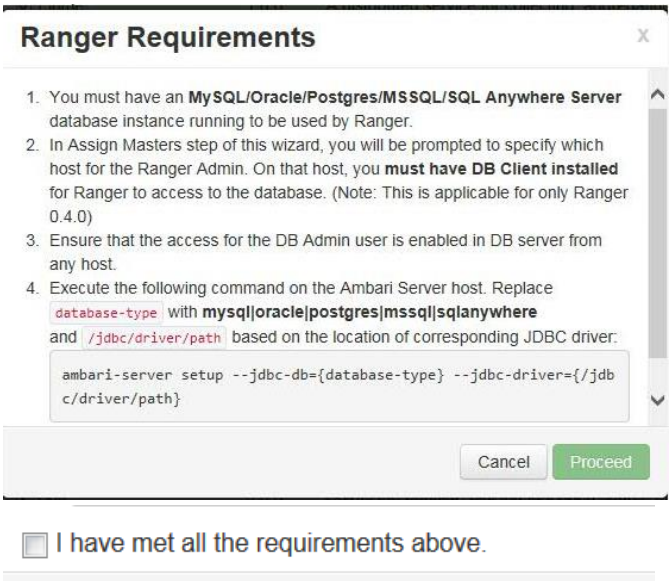
Step1.2) In the left navigation menu, click Actions, then select Add Service.



Step1.3) On the Choose Services page, select Ranger.

The system displays the Ranger Requirements page.

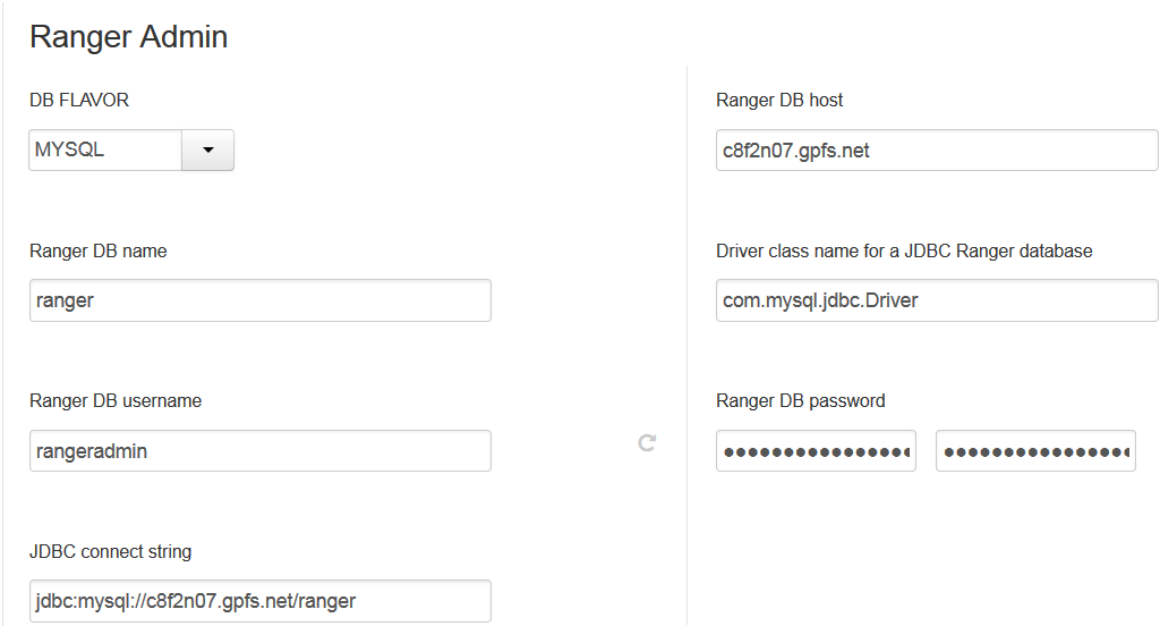
Step1.4) Ensure that you have met all the installation requirements as stated in the Ranger Requirement pop up window, then check the box to confirm the statement "I have met all the requirements above" before clicking Proceed.



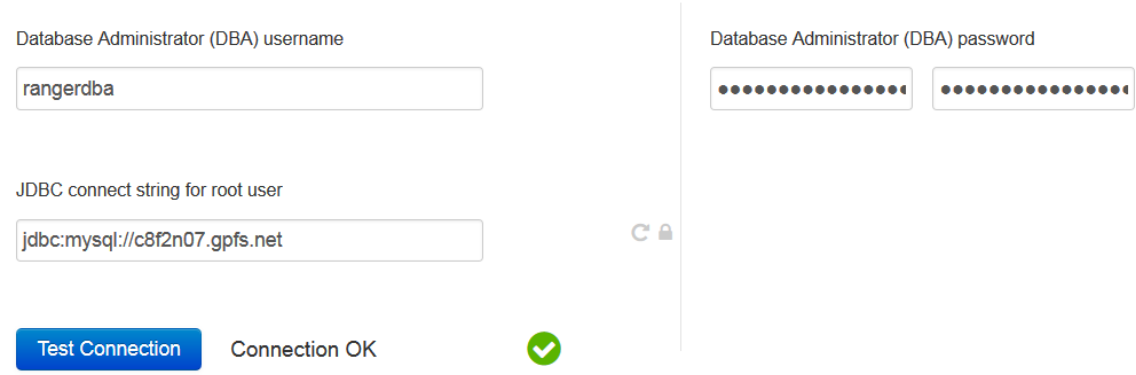
Step2) Customize the services.

In the Ranger Admin dashboard, configure the following:

- Under “DB Flavor”, select MYSQL.
- For the Ranger DB host, the hostname must be the location of MYSQL.
- For Ranger DB username, set the value to rangeradmin.
- For Ranger DB password, set the value to rangeradmin.

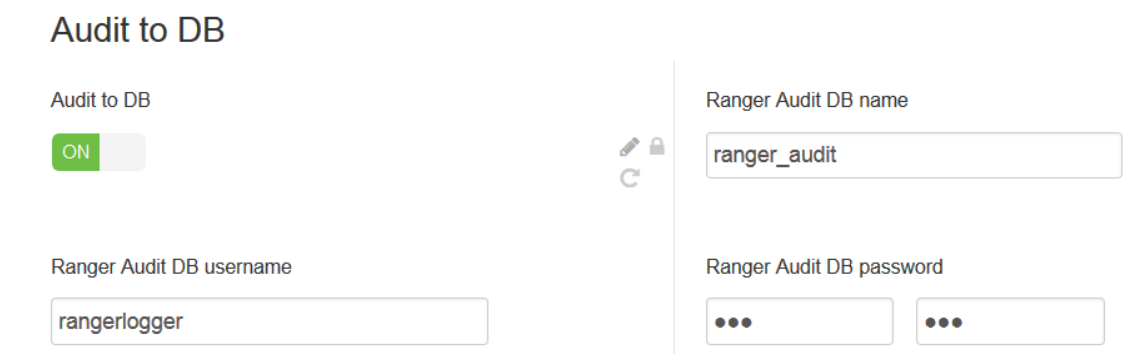


- For the Database Administrator (DBA) username, set the value to rangerdba.
- For the Database Administrator (DBA) password, set the value to rangerdba.
- Click on the Test Connection button and ensure that the connection result is OK.



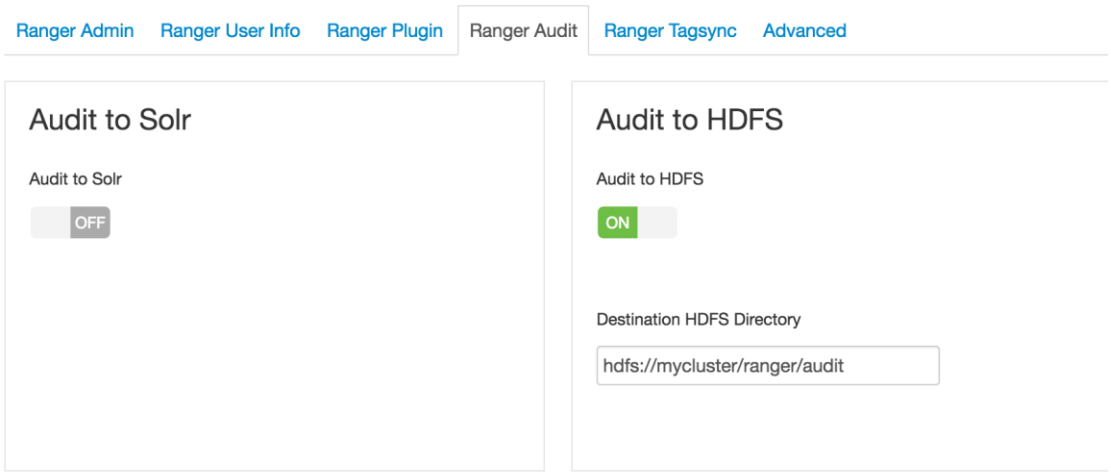
For IOP4.1/4.2:

- For the Ranger Audit DB username, set the value to rangerlogger.
- For the Ranger Audit DB password, set the value to YES.



For IOP4.2.5/HortonWorks2.6:

- Ranger Audit DB username: rangerlogger Ranger Audit DB password:YES



In the Ranger Audit tab, make sure that the Audit to Solr option is disabled

In the Advanced tab of the Ranger Configuration, in the Advanced ranger-admin-site section, set the value of ranger.audit.source.type to db

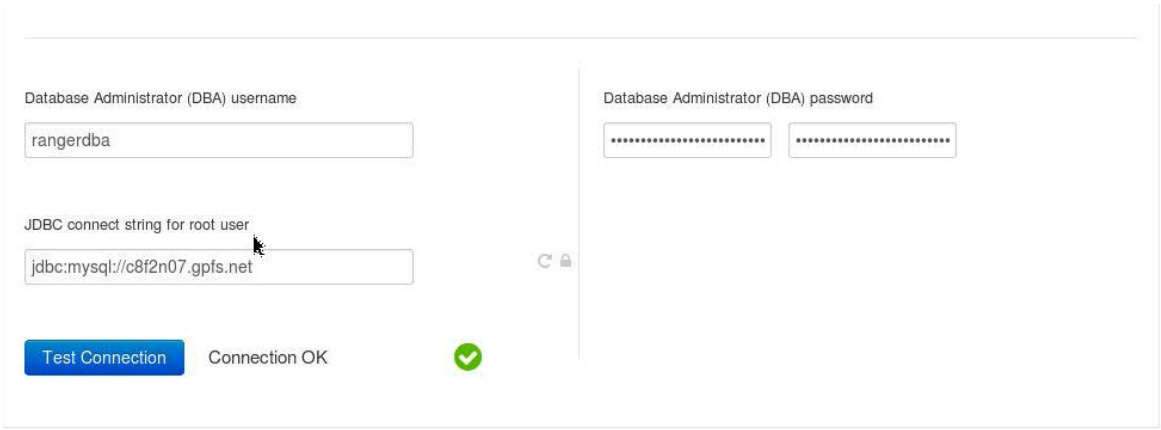
Step3) Deploy and complete the installation.

Assign the Ranger server to be on the same node as the HDFS Transparency NameNode for better performance.

- Select Next > Next > Deploy

Step4) Test the connection.

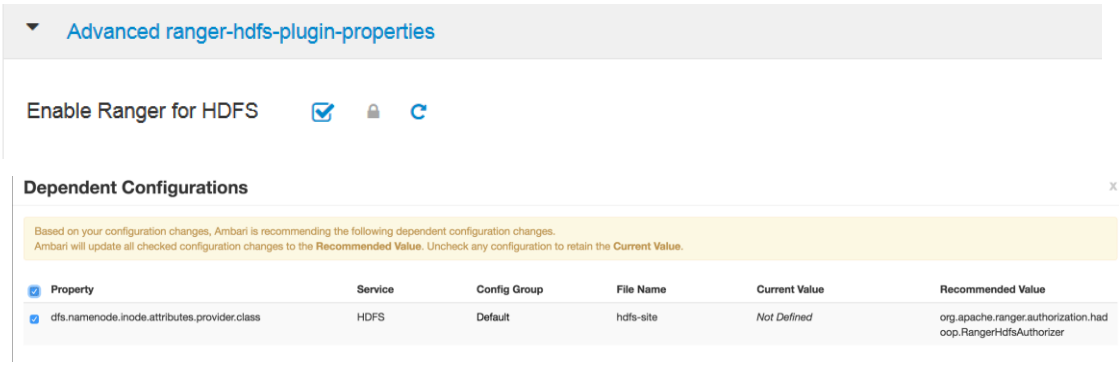
- On the Ranger dashboard, go to Configs > Ranger Admin > Test connection.



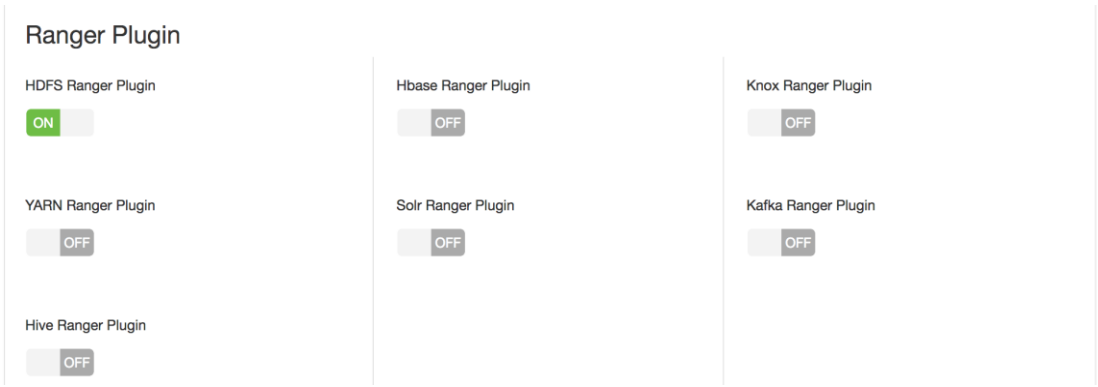
Note: After you install ranger, please enable the ranger hdfs plugin (refer to 8.8.1.3), lastly restart HDFS.

8.8.1.3 Enabling the Ranger HDFS plugin

1 HDFS—Configs tab—Advanced tab—>Advanced ranger-hdfs-plugin-properties. Select the Enable Ranger for HDFS check box



Ranger – Configs tab --- Ranger Plugin – Turn HDFS Ranger Plugin on

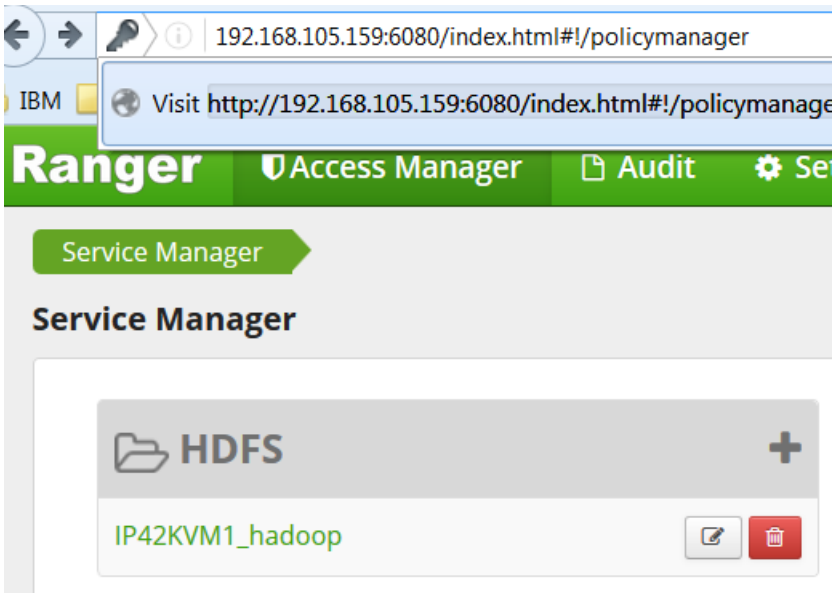


2. Save the configuration and the Restart Required message will be displayed at the top of the page. Click Restart and select Restart All Affected to restart the HDFS service and load the new configuration.

3. After HDFS restarts, the Ranger plugin for HDFS is enabled.

8.8.1.4 Log into the Ranger UI

After completing the above procedures, ensure that you can successfully log in to the Ranger UI: http://<gateway>:6080 (admin/admin).



8.8.2. Configuring Ranger with HDFS Transparency

Ranger configuration is based on the installation and configuration of HDFS Transparency. Therefore, HDFS transparency must be installed before configuring Ranger. To install HDFS Transparency, see Section 4 Installation and configuration.

1. Configuring Ranger

Step 1: Check that /etc/hadoop/conf/hdfs-site.xml contains the value `org.apache.ranger.authorization.hadoop.RangerHdfsAuthorizer` for the dfs.namenode.inode.attributes.provider.class.

/etc/hadoop/conf/hdfs-site.xml

```
      <property>
        <name>dfs.namenode.inode.attributes.provider.class</name>

   <value>org.apache.ranger.authorization.hadoop.RangerHdfsAuthorize
   r</value>
      </property>
```

Synchronize /usr/lpp/mmfs/hadoop/etc/hadoop/hdfs-site.xml to all the NameNodes and DataNodes.

```
mmhadoopctl connector syncconf /etc/hadoop/conf/hdfs-site.xml
```

Step 2: Copy the following four files to /usr/lpp/mmfs/hadoop/etc/hadoop on all the NameNode and DataNodes: ranger-hdfs-audit.xml, ranger-hdfs-security.xml, ranger-policymgr-ssl.xml, ranger-security.xml from the path /etc/hadoop/conf.

Step 3: Edit the /usr/lpp/mmfs/hadoop/etc/hadoop/hadoop-env.sh on the NameNode and add these two classes to CLASSPATH:

For IOP 4.2:

/usr/iop/4.2.0.0/ranger-hdfs-plugin/lib/*.jar

/usr/share/java/mysql-connector-java.jar

```
for f in /usr/iop/4.2.0.0/ranger-hdfs-plugin/lib/*.jar; do
  export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$f
done

for f in /usr/share/java/*.jar; do
  export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$f
done
```

For IOP4.2.5:

Change the above version string 4.2.0.0 into "4.2.5.0-0000".

For HortonWorks 2.6:

/usr/hdp/4.2.0.0/ranger-hdfs-plugin/lib/*.jar

/usr/share/java/mysql-connector-java.jar

```
for f in /usr/hdp//ranger-hdfs-plugin/lib/*.jar; do
  export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$f
done

for f in /usr/share/java/*.jar; do
  export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$f
done
```

Step 4: Ensure that the DB service is running on the DB host node. Run the command service mariadb restart or service mysqld restart if the database service is not running.

[root@c8f2n03kvm2 lib]# service mysqld status

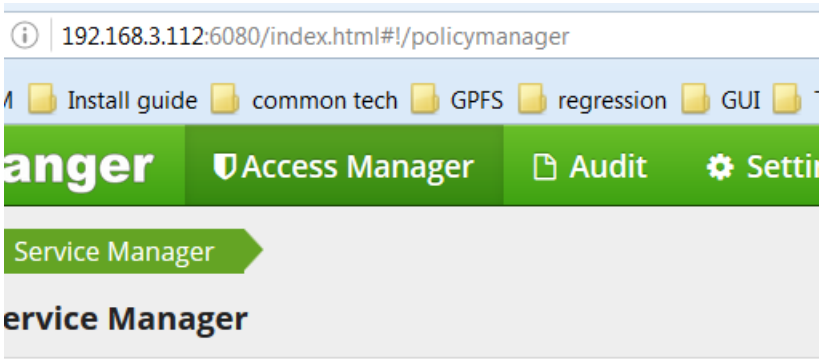
mysqld (pid 2774) is running...

On the Ranger DB Host node, ensure that the rangerlogger user exists.

```
mysql -u rangerlogger –pYES
```

2. Testing the Ranger policy for HDFS Transparency

1. Log in to the Ranger UI http://<gateway>:6080 (admin/admin).



2. Go to Service Manager > iop42_cluster_hadoop > Add New Policy.

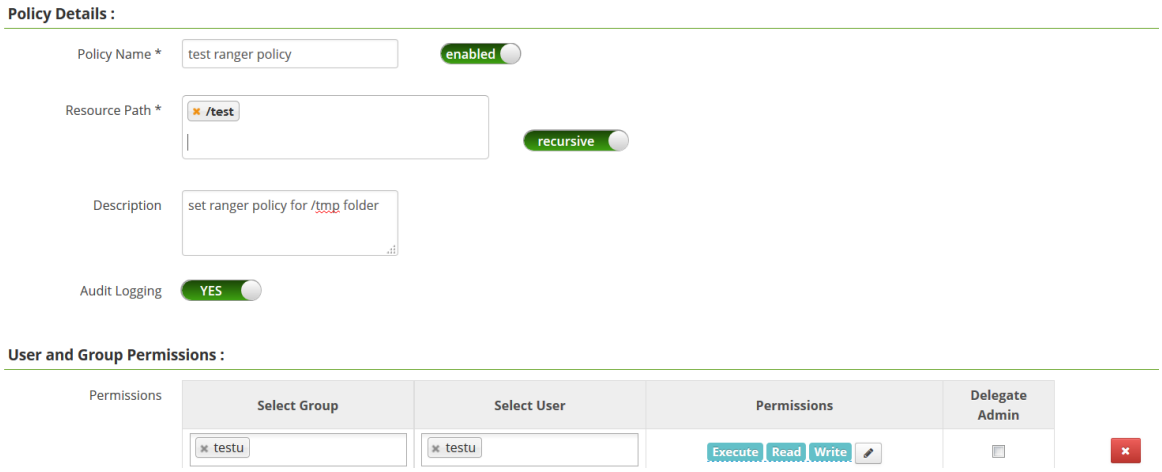
3. Type the following values in the fields displayed on the Add New Policy page.:

Ranger Policy Definition

Label	Description
Policy Name	Type the policy name. This name is cannot be duplicated for the same Service type (HDFS). This field is mandatory.
Resource path	Define the resource path for folder/file. You can add wildcard characters like /home* to avoid writing the full path as well as to enable the policy for all sub folders and files.
Description	Type the description for the policy you are creating.
Recursive	Indicate if all files or folders within the existing folder are valid for the policy. Can be used instead of wildcard characters.
Audit Logging	Indicate if this policy will be audited.
Group Permissions	From a user group list, pick a particular group and choose permissions for that group.
Enable/disable	By default, the policy is enabled. You can disable a policy to restrict user/group access for that policy

User Permissions	From a user list, pick a particular user and choose permissions for that user.
Delegate Admin	When a policy is assigned to a user or a group of users, those users become the delegated admin. The delegated admin can update and delete the policies. It can also create child policies based on the original policy (base policy).

3. Set the policy.



4. Test if the user *testu* has the RWX access for path or test.

8.8.3. Using Ranger to secure HDFS

Apache Ranger offers a federated authorization model for HDFS:

1. The Ranger plugin for HDFS checks for Ranger policies. If a policy exists, access is granted to the user.

2. If a policy does not exist in Ranger, Ranger defaults to the native permissions model in HDFS (POSIX or HDFS ACL).

After Apache Ranger and Hadoop have been installed, administers must perform the following steps:

1. Change HDFS umask to 077 from 022. This will prevent any new files or folders to be accessed by anyone other than the owner. To change the umask, from the HDFS dashboard > Configs tab > search for umask and change the value from 022 to 077.

2．Know which directory is managed by Ranger and which directory is managed by POSIX/HDFS/ACL. Let HDFS manage the permissions for the /tmp and the /user folders.

3．Do not configure a file to be controlled by both Ranger and POSIX/HDFS/ACL permissions. This creates confusion in permission control.

4．Do not deny permission to the owner if the file is controlled by Ranger.

Note:

1．Root is super user in GPFS mode.

2．If you want to do some operation (such as delete) to one file, please make sure you have the corresponding access (wx) to its parent dir.

3．When one user (such as u) delete one file or file folder, please make sure /user/u exists.

drwx------ - u u 0 2016-09-21 04:05 /user/u

If not, you can create the /user/u manually and chown u:u /user/u.

Example:

Root user create one file, common user u wants to delete this file but without the w access, we can add this w access through adding a policy through the Ranger UI.

1．For /fycheng/hosts, u user just has r-x access and have no w access, and cannot delete the hosts.

```
[root@c8f2n04 hadoop]# hdfs dfs -ls -d /fycheng
drwxr-xr-x - root root 0 2016-10-12 23:29 /fycheng

[root@c8f2n04 hadoop]# hdfs dfs -ls /fycheng
Found 1 items
-rw-r--r-- 2 root root 158 2016-10-12 23:29 /fycheng/hosts
```

2．The u user wants to delete the /fycheng/hosts. In order to do this, follow these steps:

- Make sure that the /user/u exists.
- Check that u has the rwx access to /fycheng.
- Give correct policy access to /fycheng.
- As u user, do the delete operation to the files under /fycheng.

Here is a list of sequence based on the steps above:

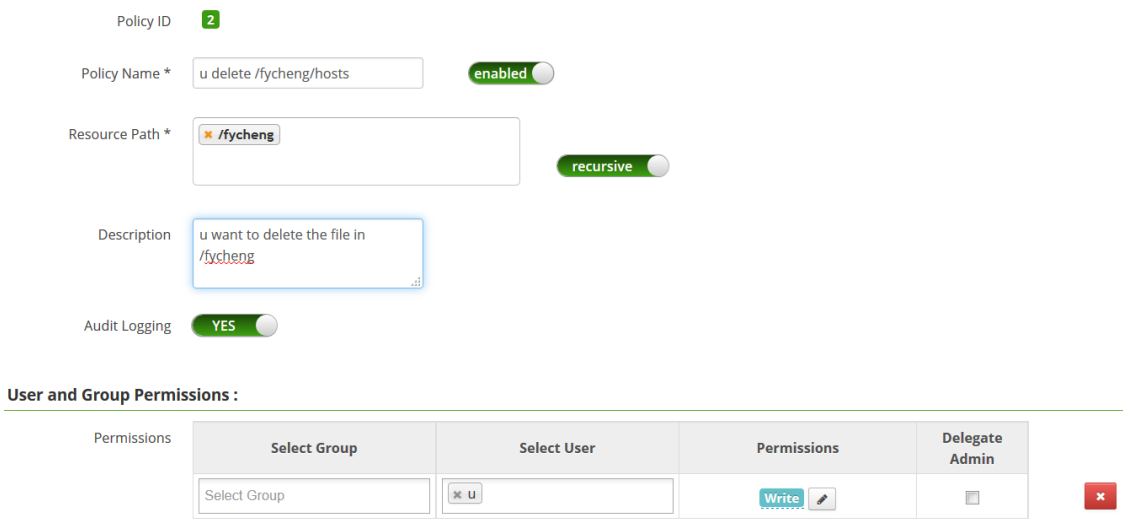
```
# Check if /user/u exists
[root@c8f2n04 hadoop]# su - u
Last login: Wed Oct 12 23:06:42 EDT 2016 on pts/1

[root@c8f2n04 hadoop]# hdfs dfs -ls /user/u
ls: `/user/u': No such file or directory

# Delete operation to files under /fycheng. Note this command will fail.
[u@c8f2n04 ~]$ hdfs dfs -rmr /fycheng
```

rmr: DEPRECATED: Please use 'rm -r' instead.
16/10/12 23:07:22 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 360 minutes, Emptier interval = 0 minutes.
16/10/12 23:07:22 WARN fs.TrashPolicyDefault: **Can't create trash directory: hdfs://c8f2n04.gpfs.net:8020/user/u/.Trash/Current**
org.apache.hadoop.security.AccessControlException: Permission denied: user=u, access=WRITE, inode="/user/u/.Trash/Current":hdfs:hadoop:drwxr-xr-x

Create the /user/u manually, and chown u:u
[root@c8f2n04 hadoop]# hdfs dfs -ls -d /user/u /user/u
drwxr-xr-x - u u 0 2016-10-12 23:36 /user/u

Delete operation to the files under /fycheng will fail due to permission
[u@c8f2n04 ~]$ hdfs dfs -rmr /fycheng/hosts
rmr: DEPRECATED: Please use 'rm -r' instead.
16/10/12 23:38:02 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 360 minutes, Emptier interval = 0 minutes.
Rmr: Failed to move to trash: hdfs://c8f2n04.gpfs.net:8020/fycheng/hosts: **Permission denied: user=u, access=WRITE, inode="/fycheng/hosts":root:root:drwxr-xr-x**

Give correct policy access to /fycheng
In Ranger UI, add policy w access for u to /fycheng



Delete operation to files under /fycheng. Now the command will succeed.
[u@c8f2n04 ~]$ hdfs dfs -rmr /fycheng/hosts
rmr: DEPRECATED: Please use 'rm -r' instead.
16/10/12 23:42:48 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 360 minutes, Emptier interval = 0 minutes.
Moved: 'hdfs://c8f2n04.gpfs.net:8020/fycheng/hosts' to trash at: hdfs://c8f2n04.gpfs.net:8020/user/u/.Trash/Current

8.8.4. Enabling Ranger auditing

Refer the Enable Ranger Auditing to enable and configure Ranger auditing,

Note:
1 In order to enable Audit to Solr for the Ranger plugins, you need to manually set the xasecure.audit.destination.solr.zookeepers = <host>:2181/solr
2 if you get an Unable to connect to Audit store!! message on the Ranger UI, the HDFS cores have write locks on them and you can remove the write lock found in HDFS in your /apps/solr/data/ranger_audits/core_node1/data/index/write.lock, and then restart HDFS and Solr.

8.8.5. Disable Ranger Support

Ranger is supported by default since HDFS Transparency version 2.7.2-X. From HDFS Transparency version 2.7.2-1 onwards, Ranger support could be disabled by configuring the gpfs.ranger.enabled property field in the /usr/lpp/mmfs/hadoop/etc/hadoop/gpfs-site.xml.

To disable Ranger support, modify the /usr/lpp/mmfs/hadoop/etc/hadoop/gpfs-site.xml file on one of the HDFS transparency nodes to false:

```
  <property>
    <name>gpfs.ranger.enabled</name>
    <value>false</value>
    <description>Set false to disable Ranger mechanism</description>
  </property>
```

Then synchronize the modified gpfs-site.xml into all the other HDFS Transparency nodes and restart the HDFS Transparency. When Ranger support is in disabled mode, Ranger will not work over HDFS Transparency.

If you did not install or would not be using the Apache Ranger over HDFS Transparency version 2.7.2-1+, then set the gpfs.ranger.enabled field value to false to get better performance over HDFS Transparency.

8.8.6. Known issues

Directory permission issues might be hit when Ranger is enabled if the 8.8.3 Using Ranger to Secure

HDFS section was not followed during Ranger setup.

Ranger directory permission issues

User name	Directory Permission in Ranger	directory permission on native HDFS	Results
fvtuser (not super user, not the owner of the directory)	r--	--x	Expectation for user fvtuser is to have r and x permission. However, the user has no permission to read the directory. To correct this issue, grant r-x access through the ranger UI or in HDFS.
fvtuser (not super user, not the owner of the directory)	--x	-w-	Expectation for user fvtuser is to have x and w permission. However, the user has no permission to create file under the directory. To correct this issue, grant -wx access in the ranger UI or in HDFS.
fvtuser (not super user, not the owner of the directory)	--x	r--	Expectation for user fvtuser is to have x and r permission. However, the user has no permission to read the directory. To correct this issue, grant r-x access in the ranger UI or in HDFS.

Note: The issues in the table above are for both native HDFS and HDFS Transparency.

Restriction:
If the uid or gid value is larger than 8388607, Hadoop will report that the uid or gid is too large during the permission checking in Ranger. Therefore, the recommendation is to have uid and gid values to be less than 8M.

8.9. Rack locality support for shared storage

HDFS Transparency 2.7.2-0, rack locality is supported for shared storage including IBM ESS. If your cluster meets the following conditions, you can enable this feature:

- There is more than one rack in the IBM Spectrum Scale cluster.
- Each rack has its own ToR (Top of Rack) Ethernet switch and there are rack-to-rack switches between the two racks.

Otherwise, enabling this feature will not benefit your Hadoop applications. The key advantage of the feature is to reduce the network traffic over the rack-to-rack Ethernet switch and make as many map/reduce tasks as possible to read data from the local rack.

The typical topology is shown by the following figure:

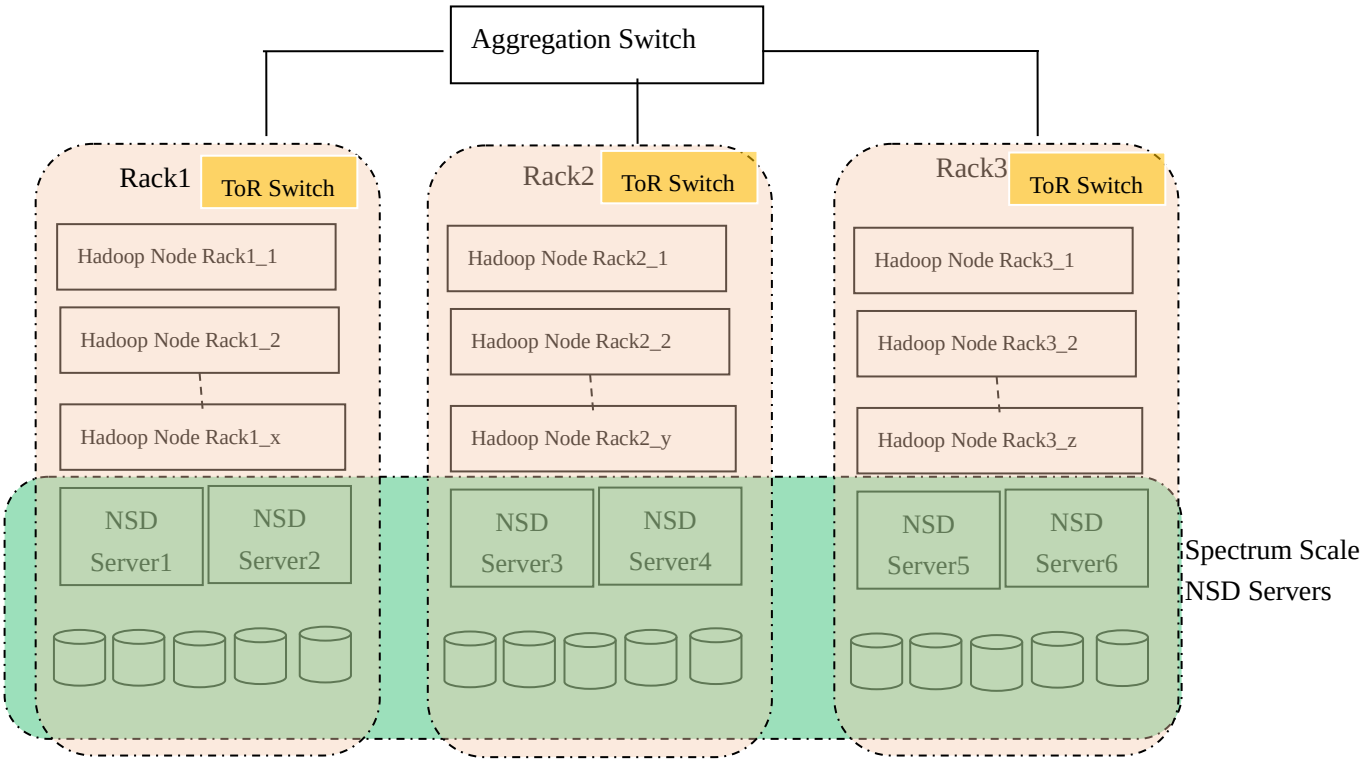


Figure 8.9.1 Topology of rack awareness locality for shared storage

For IBM Spectrum Scale over shared storage or IBM ESS, there is no data locality in the file system. The maximal file system block size from IBM Spectrum Scale file system is 16M bytes. However, on the Hadoop level, the dfs.blocksize is 128M bytes by default. The dfs.blocksize on the Hadoop level will be split into multiple 16MB blocks stored on the IBM Spectrum Scale file system. After enabling this feature, HDFS Transparency will consider the location of 8 blocks (16Mbytes * 8 = 128M bytes) including replica (if you take replica 2 for your file system) and will return the hostname with most of the data from the blocks to the applications so that the application can read most of the data from the local rack to reduce the rack-to-rack switch traffic. If there are more than one HDFS Transparency DataNodes in the selected rack, HDFS Transparency randomly returns one of them as the DataNode of the block location for that replica.

Enabling rack-awareness locality for shared storage

Step1: Select the HDFS Transparency nodes from the Hadoop node in Figure 8.9.1. You can select all of the Hadoop nodes as the HDFS Transparency nodes, or part of them as the HDFS Transparency nodes.

All of the selected HDFS Transparency nodes must be installed with IBM Spectrum Scale and can mount the file system locally. Select at least one of the Hadoop node from each of the rack for HDFS Transparency.

Select all Hadoop Yarn Node Managers as the HDFS Transparency nodes to avoid data transfer delays from the HDFS Transparency node to the Yarn Node Manager node for Map/Reduce jobs.

Step2: On the HDFS Transparency NameNode, modify the /usr/lpp/mmfs/hadoop/etc/hadoop/core-site.xml:

```
<property>
    <name>net.topology.table.file.name</name>
    <value>/usr/lpp/mmfs/hadoop/etc/hadoop/topology.data</value>
  </property>
  <property>
    <name>net.topology.node.switch.mapping.impl</name>
    <value>org.apache.hadoop.net.TableMapping</value>
  </property>
```

Step3: On the HDFS Transparency NameNode, create the topology in /usr/lpp/mmfs/hadoop/etc/hadoop/topology.data:

```
# vim topology.data

192.168.200.57          /dc1/rack1
192.168.200.58          /dc1/rack1
192.168.80.129           /dc1/rack1
192.168.80.130           /dc1/rack1
192.168.172.6            /dc1/rack2
192.168.172.7            /dc1/rack2
192.168.172.8            /dc1/rack2
192.168.172.15           /dc1/rack2
```

Note: The toplogy.data file uses IP addresses. To configure two IP addresses, see the Dual Network Interface section. The IP addresses here must be the IP addresses used for Yarn services and the IBM Spectrum Scale NSD server.

Also, it is required to specify the IP addresses for the IBM Spectrum scale NSD servers. For figure 8.9.1, specify the IP and corresponding rack information for NSD Server1/2/3/4/5/6.

Step4: On the HDFS Transparency NameNode, modify the /usr/lpp/mmfs/hadoop/etc/hadoop/gpfs-site.xml:

```
  <property>
    <name>gpfs.storage.type</name>
    <value>rackaware</value>
  </property>
```

Step5: On the HDFS Transparency NameNode, run the mmhadoopctl connector syncconf /usr/lpp/mmfs/hadoop/etc/hadoop command to synchronize the configurations to all the HDFS

Transparency nodes.

Step6 (optional): Starting the HDFS Transparency service (version 2.7.2-x) for the first time, the HDFS Transparency service will execute the ***/usr/lpp/mmfs/hadoop/sbin/initmap.sh <fsName> diskmap nodemap clusterinfo*** script to generate the ***/var/mmfs/etc/diskid2hostname***, ***/var/mmfs/etc/nodeid2hostname*** and ***/var/mmfs/etc/clusterinfo4hdfs*** files.

HDFS Transparency does not rerun the ***initmap.sh*** scripts if it detects that all three files exist. Note: If any one of these three files is missing, HDFS transparency re-runs the script and takes more time to start up.

Note: If new disks are added into the file system, or the file systems are recreated, then the ***/usr/lpp/mmfs/hadoop/sbin/initmap.sh <fsName> diskmap nodemap clusterinfo*** script must be rerun manually on the HDFS Transparency NameNode so that the ***/var/mmfs/etc/diskid2hostname***, ***/var/mmfs/etc/nodeid2hostname,*** and ***/var/mmfs/etc/clusterinfo4hdfs*** files are updated with the new information. Otherwise, unexpected exceptions in the HDFS Transparency logs might be seen.

If you configure multi-cluster between IBM Spectrum Scale NSD servers and anIBM Spectrum Scale HDFS Transparency cluster, to enable this feature, configure root passwordless access from the HDFS Transparency NameNode to at least one of the contact nodes from the remote cluster. If the passwordless access configuration cannot be set up, another method is to run the initmap.sh script manually on the remote cluster node and then copy the, ***/var/mmfs/etc/diskid2hostname***, ***/var/mmfs/etc/nodeid2hostname*** and ***/var/mmfs/etc/clusterinfo4hdfs*** files to HDFS Transparency NameNode.

8.10. Accumulo support

Apache Accumulo is fully tested over HDFS Transparency. See the Installing Apache Accumulo link for Accumulo configuration information.

By default, the property ***tserver.wal.blocksize*** is not configured and its default value is 0. Accumulo will calculate the block size accordingly and set the block size of the file in the distributed file system. For Spectrum Scale, the valid block size could only be integral multiple of 64KB, 128KB, 256KB, 512KB, 1MB, 2MB, 4MB, 8MB and 16MB. Otherwise, HDFS Transparency will throw an exception.

To avoid this exception, configure tserver.wal.blocksize as the file system data block size. Use the "mmlspool <fs-name> all -L" command to check the value.

9. Hadoop distribution support

Only the open source Apache packages and the IBM BigInsights IOP 4.0/4.1/4.2 are officially

supported. Hortonworks® and Cloudera® are not officially supported yet. Contact scale@us.ibm.com for more information.

9.1. Replacing native HDFS service with HDFS transparency

Step 1. Install Hadoop distribution over native HDFS.
If you are using host1/NameNode, all other hosts are DataNodes.

Step 2. Configure Hadoop configuration from your Hadoop distribution GUI.

Ensure that `dfs.client.read.shortcircuit` in HDFS service is disabled if you are using Hadoop 2.7.1/2.7.2. By default, it is disabled by most Hadoop distributions.

Step 3. Set up an IBM Spectrum Scale cluster.

Note: Host1 used for the HDFS NameNode service must be added to the IBM Spectrum Scale cluster.

Step 4. Download and install the HDFS Transparency cluster according to the link.

All nodes in HDFS transparency cluster must be installed with the GPFS packages and must be able to mount the GPFS file system.

Configure the native HDFS NameNode as the default NameNode for HDFS Transparency. This will make the configuration changes simpler.

Step 5. Set up the HDFS Transparency cluster

- ✓ Copy the `/etc/hadoop/conf/core-site.xml, hdfs-site.xml` from the Hadoop distro `host1/NameNode` into `host1:/usr/lpp/mmfs/hadoop/etc/hadoop/.`
- ✓ Modify the `host1:/usr/lpp/mmfs/hadoop/etc/hadoop/slaves` to add the DataNode service for HDFS Transparency.
- ✓ Modify the `host1:/usr/lpp/mmfs/hadoop/etc/hadoop/gpfs-site.xml` according to the comments for each xml property.
- ✓ On host1, run the command "`/usr/lpp/mmfs/bin/mmhadoopctl connector syncconf /usr/lpp/mmfs/hadoop/etc/hadoop`"

Step 6. On host1, run the `/usr/lpp/mmfs/bin/mmhadoopctl connector start` command to start the HDFS Transparency services.

Step7: Run `hadoop dfs -ls /` to ensure that the system displays the correct output.

Step 8: Start the Hadoop distro services, hbase, yarn, and the other services. Note: The native HDFS service must not be functioning in the Hadoop distribution GUI.

9.2. IBM BigInsights IOP support

IBM BigInsights IOP 4.1.0.2 is verified over HDFS Transparency 2.7.0-x. Short Circuit Read is enabled by default and you must disable it on the Ambari GUI.

IBM BigInsights IOP 4.2 is verified over HDFS Transparency 2.7.2-x. Short Circuit Read can be enabled.

IBM BigInsights IOP 4.2.5 is verified over HDFS Transparency 2.7.3-x. Short Circuit Read can be enabled.

In IBM BigInsights IOP 4.2/4.2.5, some services, such as Hive, will take the user name "anonymous"(whose group name is "anonymous") to do some service check. Therefore, we need to create the group and user in advance if they are not existing. The gid/uid of "anonymous" on all nodes should be of the same, e.g.

```
# mmdsh -N all id anonymous
c35f1m4n15.gpfs.net:   uid=2825(anonymous) gid=2825(anonymous) groups=2825(anonymous)
c35f1m4n14.gpfs.net:   uid=2825(anonymous) gid=2825(anonymous) groups=2825(anonymous)
c35f1m4n13.gpfs.net:   uid=2825(anonymous) gid=2825(anonymous) groups=2825(anonymous)
c35f1m4n16.gpfs.net:   uid=2825(anonymous) gid=2825(anonymous) groups=2825(anonymous)
```

10. Limitations and differences from native HDFS

The configuration that differ from HDFS in IBM SPECTRUM SCALE

Property name	Value	New definition or limitation
`dfs.permissions.enabled`	True/false	For HDFS protocol, permission check is always done.
`dfs.namenode.acls.enabled`	True/false	For native HDFS, the NameNode manages all meta data including the ACL information. HDFS can use this information to turn on or off the ACL checking. However, for IBM

		Spectrum Scale, HDFS protocol will not save the meta data. When ACL checking is on, the ACL will be set and stored in the IBM Spectrum Scale file system. If the admin turns ACL checking off, the ACL entries set before are still stored in IBM Spectrum Scale and remain effective. This will be improved in the next release.
`dfs.blocksize`	Long digital	Must be an integer multiple of the IBM Spectrum Scale file system blocksize (mmlsfs -B), the maximal value is 1024 * file-system-data-block-size (mmlsfs –B)
`dfs.namenode.fs-limits.max-xattrs-per-inode`	INT	Does not apply to HDFS Transparency.
`dfs.namenode.fs-limits.max-xattr-size`	INT	Does not apply to HDFS Transparency.
dfs.namenode.fs-limits.max-component-length	Not checked	Doesn't apply to HDFS Transparency; the file name length is controlled by Spectrum Scale. Refer Spectrum Scale FAQ for file name length limit(255 unicode-8 chars)
Native HDFS encrytion	Not supported	Customers should take native Spectrum Scale encryption.
Native HDFS caching	Not supported	Spectrum Scale
NFS Gateway	Not supported	Spectrum Scale provides POSIX interface and taking Spectrum Scale protocol could give your better performance and scaling

Functional limitations

- The maximum number of EA is limited by IBM Spectrum Scale, the total size of EA key and value must be less than the metadata block size in IBM Spectrum Scale.
- EA operation on snapshots is not supported now.
- Raw namespace is not implemented because it is not used internally.
- If gpfs.replica.enforced is configured as gpfs, the Hadoop shell command "hadoop dfs -setrep" will not take effect. Also, "hadoop dfs -setrep -w" will stop functioning and not

exit.

- HDFS Transparency namenode doesn't provide "safemode" because it's stateless.
- HDFS Transparency namenode doesn't need the second namenode as native HDFS does
- Maximal replica for Spectrum Scale is 3.
- "hdfs fsck" doesn't work against HDFS Transparency, please take mmfsck.
- Spectrum Scale has no ACL entry number limit(maximal entry number is limited by Int32)
- "distcp --diff" is not supported over snapshot.

11. Problem determination

Refer the link for known problem determination

12. Revision history

#	Date	Comments
0.1	2016-1-4	Yong(zhengyzy@cn.ibm.com) initialized the first version from GPFS Advanced PDF.
0.2	2016-1-5	Merge the draft from Tian(ftbj@cn.ibm.com) about shared storage support.
0.3	2016-1-5	Yong(zhengyzy@cn.ibm.com) re-constructed the sections for installation, configuration and upgrade.
0.4	2016-1-13	Tian(ftbj@cn.ibm.com) added the section "**How to update environment variables for hdfs Transparency service.**"
0.5	2016-1-22	Merge the comments from Akash(avagrawa@in.ibm.com)
0.6	2016-2-17	Yong(zhengyzy@cn.ibm.com) corrected the guide for HA configuration.
0.6.1	2016-2-27	Updated the section
0.6.2	2016-2-29	Merged the comments from ID team members Alifiya/Lata
0.7	2016-3-2	Yong reconstructed the guide and added the section for Docker support.
0.8	2016-3-18	Li Xia added the section for federation and distcp
0.8.1	2016-4-4	Merged the comments from ID team member Lata
0.8.2	2016-4-8	Merged the comments from customers
0.8.3	2016-4-11	Merged the comments from Power Solution team
0.9.1	2016-4-13	Copied the section 8.1.1 into section 8.1.2 and update several steps about Auto NameNode HA

		from Li xia
0.9.2	2016-4-20	Merged the comments from Linda Cham
0.9.3	2016-5-6	Merged the comments from PC(Piyush Chaudhary)
0.9.4	2016-5-9	Merged the minor changes from PC and Linda
0.9.5/0.9.6	2016-8-10	Yong corrected some steps for Federation.
0.9.7	2016-8-30	Fang Yuan added the section for short circuilt and Ranger Support
0.9.8	2016-8-31	Yong modified Fang Yuan's sections and merged comments from Linda;
0.9.9	2016-9-5	Yong updated the rack locality section
1.0.0	2016-9-13	Yong corrected some description about super group.
1.0.1	2016-9-14	Yong merged the comments from Linda and added the limit for "Hadoop dfs -setrep -w" under gpfs.replica.enforce=gpfs.
1.0.2	2016-9-23	Merged the comment from ID member
1.0.6	2016-10-25	Added Ranger support
1.0.9	2016-12-14	Fang Yuan, Linda and Yong updated the configuration for gpfs.ranger.disable.
1.1.0	2017-1-17	Updated the section 8.8.5 about current Ranger issues on native HDFS, 8.10 Accumulo support and update the description for shortcircuit in section 8.2.2
1.1.1	2017-1-20	Fang Yuan added some limits about Ranger
1.1.2	2017-3-3	Yong corrected the Auto HA zkfc format command.
1.1.3	2017-3-3	Merged comments from Linda
1.1.4	2017-6-23	Updated the section 3.2, 4.2.2, 8.8.1.2,8.8.1.3, 8.8.2, (1.configuring Ranger), 8.8.4 and 11.
1.1.5	2017-6-23	Merged the comments from Linda.