DELL EMC ISILON BEST PRACTICES GUIDE FOR HADOOP DATA STORAGE

ABSTRACT

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

May 2018

The information in this publication is provided “as is.” DELL EMC Corporation makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

DELL EMC believes the information in this document is accurate as of its publication date. The information is subject to change without notice.

DELL EMC is now part of the Dell Technologies.

Contents

Introduction

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

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

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

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

- Store data in a POSIX-compliant file system with SMB and NFS workflows and then access it through HDFS
- Scale storage independently of compute as your data setsgrow
- Protect data more reliably and efficiently instead of replicatingit with HDFS 3X mirror replication
- Eliminate HDFS copy operations to ingest data and Hadoop file system commands to manage data
- Implement distributed fault-tolerant NameNodeservices
- Manage data with enterprise storage features such as deduplication and snapshots

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

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

Overview of Isilon for Big Data

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

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

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

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

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

An Isilon cluster also includes enterprise features to back up your data and to provide high availability. For example, in managing your DataNode data, a best practice with a traditional Hadoop system is to back up your data to another system—an operation that must be performed with brute force by using a tool like Distcp. OneFS includes clones, NDMP backups, synchronization, geo- replication, snapshots, file system journal, virtual hot spare, antivirus, IntegrityScan, dynamic sector repair (DSR), and accelerated drive rebuilds. For complete information about the data availability features of OneFS, see the white paper titled: HIGH AVAILABILITY AND DATA PROTECTION WITH DELL EMC ISILON SCALE-OUT NAS

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

For security, OneFS can authenticate HDFS connections with Kerberos and LDAP providers to provide central authentication and identity management.

How Hadoop works with Scale-Out Isilon NAS

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

OneFS implements the server-side operations of the HDFS protocol on every node, and each node functions as both a NameNode and a DataNode. An Isilon node, however, does not act as a resource manager or a node manager; those functions remain the purview of Hadoop clients. OneFS contains no concept of a secondary NameNode: Since every Isilon node functions as a NameNode, the function of the secondary NameNode—checking pointing the internal NameNode transaction log—is unnecessary.

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

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

Client computers can access any node in the cluster through dual 10 GigE or 40GigE on newer node network interfaces. A SmartConnect license adds additional network resilience with IP address pools that support multiple DNS zones in a subnet as well as IP failover. In OneFS, an IP address pool appears as a: `groupnet:subnet:poolname`. A best practice, which is discussed later in this paper, networking topologies and implementations are discussed later in this paper.

NameNode Redundancy

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

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

The HDFS Architecture of OneFS

When HDFS client connects to a NameNode to query or modify data or metadata in the OneFS file system, each node has access to this data and metadata. The metadata includes the logical location of data for the file stream—that is, the address of the Isilon nodes on which the blocks resides. An HDFS client can modify the metadata through the NameNode's RPC interface. OneFS protects HDFS metadata at the same protection level as HDFS file data. In fact, OneFS handles all the metadata; you do not need to worry about managing it or backing it up.

With Isilon, the NameNode daemon translates HDFS semantics and data layout into OneFS semantics and file layout. For example, the NameNode translates a file's path, offset, and LEN into lists of block IDs and generation stamps.

A DataNode stores blocks of files. More specifically, the DataNode maps blocks to block data. With HDFS, a block is an inode-offset pair that refers to a part of a file. With OneFS, you can set the size of an HDFS block to optimize performance. A Hadoop client can connect to a DataNode to read or write a block, but the client may not write a block twice or delete a block. To transfer a block to a DataNode, the HDFS client encapsulates the block in packets and sends them over a TCP/IP connection. The Isilon HDFS daemon performs zero-copy system calls to read and write blocks to the file system. On OneFS, the DataNode reads packets from and writes packets to disk.

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

HDFS Integration

This section describes the best practices that can help set up an Isilon cluster to work with Hadoop distributions to solve real-world problems in data analysis. This section does not, however, cover how to perform many of the tasks, such as creating an Access Zone or SmartConnect Zone configurations, that it refers to; those instructions can be found in the referenced implementation guides and Isilon documentation.

Supported Hadoop Distributions and Projects

A Dell EMC Isilon cluster is platform agnostic for compute, and there is no vendor lock in: You can run most of the common Hadoop distributions with an Isilon cluster, including Apache Hadoop, Hortonworks HDP, Cloudera CDH , IBM BigInsights, and others.

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

Since the pace of development and release of Hadoop distributions and projects is rapid the most up to date list of distributions and applications supported by OneFS can be found here: https://community.emc.com/docs/DOC-37101.

Isilon Cluster Integration with Hadoop

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

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

Best Practice: Review the supportability of release compatibility with OneFS version: https://community.emc.com/docs/DOC-37101

Isilon OneFS Version

Best Practice: It is recommended to implement at least OneFS version 8.0.1.x or greater for pure HDFS workflows due to significant feature enhancement made in this version of OneFS; mainly DataNode Load Balancing and Pipeline Write recovery, which will be discussed later in this paper

Additional requirements and workflows may dictate a different version of OneFS is used but it should be noted version prior to 8.0.1.0 do not contain these major feature enhancements.

We recommend that you always follow the following Isilon Hadoop implementation guides for installation and configuration depending on your distribution:

Ambari and Hortonworks HDP

- EMC Isilon OneFS with Hadoop and Hortonworks Installation Guide

- EMC Isilon OneFS with Hadoop and Hortonworks for Kerberos Installation Guide
- Isilon OneFS with Ambari Multitenant Active Directory Guide

Cloudera Manager CDH

- EMC Isilon OneFS with Hadoop and Cloudera Installation Guide
- EMC Isilon OneFS with Hadoop and Cloudera for Kerberos Installation Guide

Access Zones

It is recommended to leverage a dedicated HDFS Access Zone for the HDFS root directory on OneFS, this provides isolation of User and Group management, isolation of identity management and the ability to control connections to this data. Each access zone has several separate parameters that define them.

1. Access Zone Root directory – a subdirectory of /ifs

   Use a directory with the clustername to allow for SyncIQ uniqueness in the future, and then the name of the access zone

   Example: /ifs/cls1/prod-hdp

2. Access Zone specific local Authentication Provider – Zone specific local users and groups for identity management of the users in the Hadoop cluster. Add AD, Kerberos and LDAP providers as needed to each zone

3. SmartConnect Zone Name – FQDN of the network pool assigned to this Access Zone. This will also be the NameNode for the Hadoop clients

   Use an FQDN that closely associates it with the Access Zone name.

   Example: prod-hdp.domain.com

4. HDFS root directory – separate from the Access Zone root directory, create one level below the Access Zone and name 'hadoop'

   Example: /ifs/cluster1/prod-hdp/hadoop

5. Quotas – To limit HDFS storage per Access Zone if needed.

   a. Use a directory quota, exclude /tmp and report the quota as the logical size of the cluster.

Best Practice: Define and utilize a dedicated Access Zone for the HDFS root directory, do not use the OneFS system Zone. Additionally, define separate Access Zones for Test, Development and Hadoop Tiered Storage clusters, also use separate Access zones for different business units that own Hadoop clusters.

Isilon OneFS and HDFS Requirements

The following section outlines core requirements that are needed when Hadoop clusters and specific configurations are used with Isilon and Hadoop integrations.

Kerberos and DNS Reverse Lookup

When using Kerberized Hadoop clusters, accurate and available DNS is required to enable fully functional Kerberos.

Required: Full forward and reverse name resolution is required to use Isilon and Hadoop Kerberos, this includes SmartConnect based forward name resolution and reverse PTR records on all IP's in the pool assigned to the SmartConnect Zone used by HDFS.

Using Active Directory with RFC2307 - Replication and Indexing

When using the Active Directory provider with rfc2307 enabled to provide unix attributes for identity management. The following Active Directory configuration is needed.

Required: Replication and Indexing of the attributes is required, see the following KB: https://support.emc.com/kb/335338

Working with Hadoop Distributions

For all the distributions, the following considerations outline the recommended approaches and Best Practices where possible.

HDFS Directories and Permissions

Since the deployment of a Hadoop distribution does not automatically create the HDFS data directories on OneFS, this must be done manually prior to integration with the compute cluster. OneFS controls access to directories and files with POSIX mode bits or access control lists (ACL) using the standard OneFS unified access model. When you set up your directories, files, accounts, and permissions for Hadoop, you must make sure that they have the correct permissions so that users and applications can access their directories and files.

Best Practice: Create the required base directory structure in the assigned Hadoop root in your OneFS Access Zone and implement the required permissions as needed by the Hadoop distribution being deployed. It is recommended to always use the Isilon Hadoop tools to create and permission the base Hadoop directory structure.

Permissions - POSIX on Core Hadoop System Directories but OneFS ACLs can be used on Data

In many cases, specific permission must exist on certain 'system' type directory that other Hadoop services leverage; yarn, hive etc. It is recommended to implement posix only permission on these directories examples include /tmp, /ats, apps, app-logs, hbase etc.

Best Practice: Implement posix only permissions on the base system directories, Hadoop distribution expect to see specific permissions and ownerships of these directories. User defined data can be permissioned using full OneFS permissions if needed, ACL's can be used on these data directories to implement granular data access control.

UID and GID Parity

Historically HDFS file systems and compute clusters did not enforce a strong security model, in which only user and group names mattered for file access. With OneFS presenting a unified permission model with true multiprotocol access, not only do names matter but so does the underlying identity of that user: UID, GID or SID. In order to facilitate multiprotocol and consistent access, it is critical to ensure all users and groups on the compute cluster and Isilon cluster must have identity parity. This may involve additional consideration during setup to ensure all users created have the same identities, but this will allow OneFS to managing access correctly. This is implemented by creating all users—on Isilon or Hadoop compute nodes—with the same UID and GID, or leveraging a central Directory Service to supply them, for example:

User: hdfs on compute: UID 501 = hdfs on Isilon: UID 501

User: yarn on compute: UID 502 = yarn on Isilon: UID 502

Group: Hadoop on compute: GID 500 = Hadoop on Isilon: GID 500

Group: hive on compute: GID 504 = hive on Isilon: GID 504

Setting up and leveraging UID and GID parity will require additional planning and implementation and should be implemented to support your requirements.

Best Practice: Implement UID and GID parity between Isilon OneFS Users and Compute clients, or leverage Directory Services—AD or LDAP—to provide parity for all data written to Isilon.

Isilon Hadoop Tools

Scripts to facilitate the creation of users, groups, and base directories are available to help the implementation of Isilon with Hadoop clusters. They can be found at the following GitHub: https://github.com/Isilon/isilon_Hadoop_tools. Using these scripts will create all the required users, groups, and directories required for integration with the specific Hadoop distribution. Once they are created on Isilon, all the required local users and groups can be pre-created on your compute clients to ensure UID and GID parity.

Best Practice: Use the Isilon User and Group, and Directory create scripts for your Hadoop distribution. This will create and apply a standard set of user and group permissions to a well-defined base set of Hadoop directories.

Virtual Rack Implementations – Rack Configurations

A virtual rack attempts to mimic the implemented data locality capabilities found within Direct Attached Storage (DAS) Hadoop Clusters. The rack associates the compute clients' IP addresses with a specific pool of Isilon nodes such that when a client connects to the cluster, OneFS assigns the data connection to one of the Isilon nodes in the racks pool, which is usually referred to in the following format `groupnet:subnet:poolname`. In this way, a virtual rack can, for example, route a client's connection through its optimal network path of a single switch shared between the compute node and Isilon node.

The primary purpose of racks is to provide location-aware data access for compute nodes to Isilon nodes. If no colocation or separation of compute nodes and Isilon exist, then racks do not provide any benefit. In older versions of OneFS; OneFS 7.2.x, we recommended that you use a default rack to separate NameNode and DataNode access. This is no longer suggested due to improvements with OneFS.

In the implementation seen in Figure 1, all the compute is collocated with all the Isilon or the Isilon is completely separated from the compute infrastructure racks. In this configuration, HDFS racks are not suggested, as they do not provide any benefit of data locality between compute and disk.

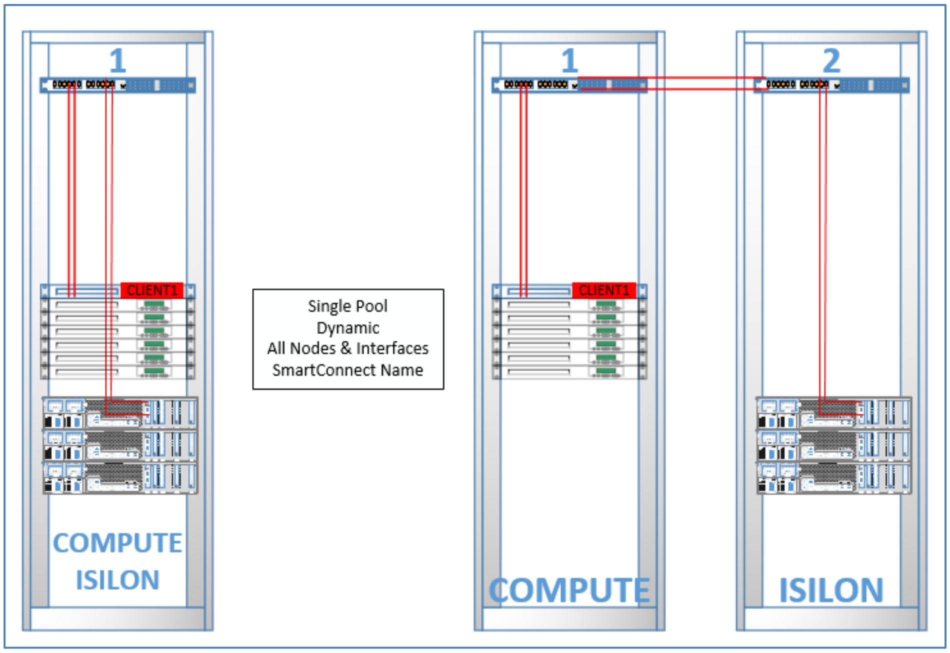


Figure 1 Compute and Isilon layout not requiring data locality

When compute and Isilon nodes are split across racks as seen in Figure 2, then HDFS racks can provide benefit by providing data locality.

- All Isilon OneFS nodes act as NameNodes through a SmartConnect pool with all Isilon nodes
- Compute nodes in Rack1 are associated with Isilon nodes in Rack1 through virtual Rack1
- Compute nodes in Rack2 are associated with Isilon nodes in Rack2 through virtual Rack2

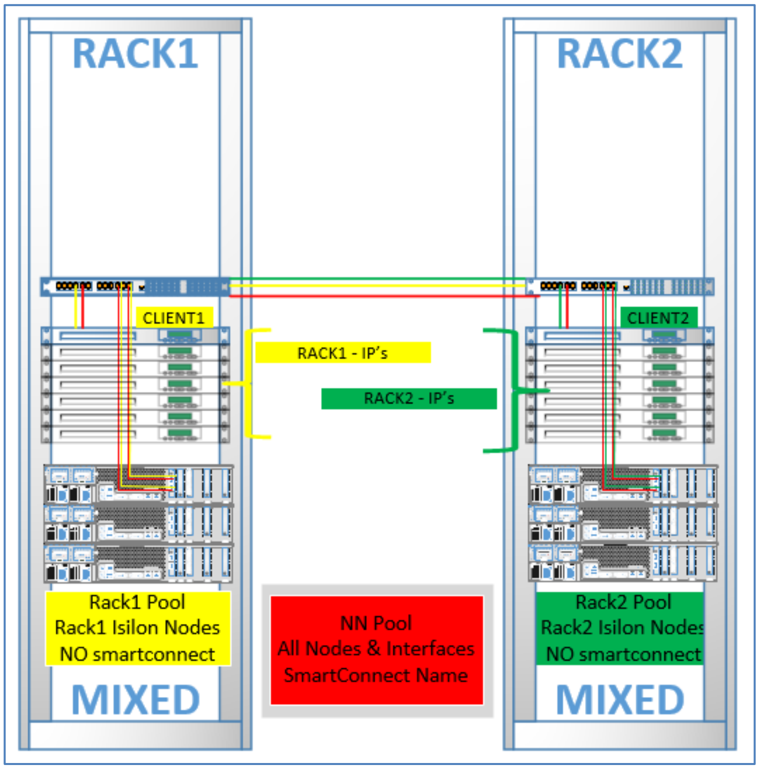


Figure 2 Compute and Isilon layout benefitting for data locality with racks

When a NameNode returns three IP addresses to a Hadoop client's request to read data, the NameNode selects the DataNode. The OneFS HDFS service checks which client is connecting and then returns the IP addresses for the DataNode, the secondary DataNode from one rack, and an IP address for the third DataNode from another rack.

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

A Hadoop client connects over HDFS to the DataNodes with interfaces that are assigned to the pool. For more information, see the https://community.emc.com/docs/DOC-44304 or the OneFS Administration Guide for your version of OneFS.

Additional information on racks can be found on the following EMC community network blog: SmartConnect, Network Pools, and HDFS Racks for Hadoop.

Best Practice: Only Implement HDFS racks to provide data locality to collocated compute and Isilon nodes to provide location-aware data access and best leverage the network architecture provided by top of rack switching when separation of network interfaces is implemented.

In all networking configurations, we recommend that you implement High Availability networking architecture to prevent single points of failure.

NameNode and DataNode Requests

Before we discuss recommended approaches and best practices around network connectivity for a Hadoop cluster to an Isilon cluster, let's review how the cluster manages connections, and the behavior of the HDFS protocol calls from the client.

1. The resource manager/ node manager on a compute client resolves the default name of the file system to an IP address by using DNS based on the SmartConnect DNS name defined in the Hadoop core-site.xml file.
2. The resource manager/ node manager then makes a NameNode request to the IP address that DNS returned and initiates either a read or a write sequence.

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

For writes, the resource manager/ node manager requests a location to write its chunk of data. OneFS 8.0.1 and later also returns a list of three IP addresses that the resource manager/ node manager can provide to tasks for writing data. (Prior to 8.0.1.x Isilon would only return a single IP for Write operations, this could lead to job failures if that IP became unavailable)

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

DataNode Load Balancing

With the introduction of OneFS 8.0.1, HDFS connection load balancing changed significantly on OneFS. Prior to this version of OneFS, the load balancing of DataNode connection requests was entirely based on a round robin methodology where the service allocated the next IPs in sequence without any awareness of the connections on that node. This could potentially lead to imbalances and overloading of nodes just through a standard round robin allocation. The introduction of DataNode load balancing in 8.0.1.0 gave additional intelligence to the DataNode IP allocation mechanism, so that after a NameNode request is made the returned DataNode IP will be the node with the lowest connection count. This mechanism will lead to better load balancing of connections across the Isilon cluster and not overloading of node interfaces. DataNode load balancing occurs after initial NameNode connections are made and is independent of any load balancing provided by SmartConnect.

Pipeline Write Recovery

Also with the introduction of 8.0.1.0 Isilon introduced the data Pipeline Write Recovery feature, prior to this release of OneFS, a NameNode request would respond with 3 DataNode IP on read requests from clients, but it would only respond with 1 DataNode IP's on client write requests This could potential lead to job failures through connectivity issue during write operations from failure associated with connectivity to that DataNode. The data Pipeline Recovery feature now supports OneFS responding with 3 IP's on write requests, this allows clients to retry different node IP's on an accounted issues, basically the client will now retry a different IP on any accounted issues. This will provide increased resiliency and stability with write requests by providing an automatic retry mechanism on reads and now on writes providing parity to traditional NameNode based DAS.

SmartConnect

HDFS access redundancy is provided by the compute cluster leveraging a SmartConnect zone in OneFS. SmartConnect requires that you add a new name server (NS) record as a delegated domain to the authoritative DNS zone that contains the cluster. For instructions on how to set up a SmartConnect zone, see the https://community.emc.com/docs/DOC-44304 for your version of OneFS.

Note: By default, the OneFS SmartConnect module balances connections among nodes by using a round-robin policy. A SmartConnect Advanced license adds advanced balancing policies and lets you define IP address pools to support multiple IP pools within a subnet. The licensed version supports dynamic IP failover to provide continuous access to the same IP address. The rest of this section assumes that you have the licensed version of SmartConnect Advanced in place.

If a node or one of its interfaces fails, SmartConnect can manage the behavior and reassignment of the IP addresses to the remaining interfaces in the pool depending on the configuration deployed. OneFS then routes connections to the appropriate member interface without interrupting access. When the failed interface or the failed node comes back online, OneFS updates the list of online pool members and can manage the operation of bringing the additional IP back into service.

SmartConnect offers several connection-balancing strategies:

- **Zoning:** You can optimize storage performance by designating zones to support workloads or clients. For large clusters, you can use zoning to partition the cluster's networking resources and allocate bandwidth to each workload, which minimizes the likelihood that heavy traffic from one workload will affect network throughput for another. Zoning can help you segment a cluster for two disparate Hadoop workflows.

- **Inclusion and exclusion:** You can add node interfaces and interface types to SmartConnect pools. For example, in an inclusive approach, a pool includes all the 10 GbE interfaces in the cluster, regardless of node type. An exclusive strategy, by contrast, limits membership to only one interface type from one node type—for example, only 40 GbE interfaces on a H600 Genration6 nodes. As a result, clients connecting to a pool receive similar performance from any node in the pool.

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

- **Round robin:** This policy, which is the default, rotates connections among IP addresses in a list. Round robin is the best balancing option for most workflows.

- **CPU usage:** This policy examines average CPU usage in each node and then attempts to distribute the connections to balance the workload evenly across all the nodes.

- **Network throughput:** This policy evaluates the overall average network throughput per node and assigns new connections to the nodes with the lowest relative throughput in the pool.

- **Connection count:** This policy uses the number of open TCP connections on each node in the pool to determine which node a client will connect to.

These policies only apply to the initial NameNode connection, DataNode Load Balancing of client connections is handled by HDFS internally.

IP Pools

When defining network IP pools on OneFS we have 2 options; static or dynamic allocation, additional information can be found in the OneFS Web Administration Guides Each type of allocation has benefits when used with the appropriate workload. The choice of pool may depend on a number of factors. HDFS is unique in that it can leverage multiple pools if needed to manage NameNode and DataNode traffic independently. Ultimately pool allocations are dependent on failover capabilities of protocol and applications and the ability to meet the requirements of a pool; for additional information see the: Isilon External Network Connectivity Guide. The main difference between pools is highlighted below.

Static

- One IP per interface is assigned, will likely require less IP's to meet minimum requirements
- No Failover of IP's to other interfaces

Dynamic

- Multiple IP per interface is assigned, will require more IP's to meet minimum requirements
- Failover of IP's to other interfaces, Failback policies are needed

HDFS Pool Usage and Assignments

HDFS connections are unique in the fact that they are made up of two separate connections; a NameNode connection and DataNode connection. Due to this behavior of a client requesting data locality in the initial NameNode connection then being handed a DataNode IP and subsequently making a DataNode connection. It allows Isilon the option of using a single pool or using two or more pools to manage network connections based on requirements with regard to separating these connections.

SmartConnect pools, as previously discussed, defines which nodes and interfaces are used for client connectivity. It provides a lot of flexibility to manage connections to node interfaces. As OneFS HDFS integration has evolved, many of the best practices around SmartConnect pools have changed as the product has gained additional functionality. In order to determine the best strategy for SmartConnect Pool implementation, a number of factors must be assessed as follows:

- Use of OneFS racks if needed
- Node Pools – is the cluster a single heterogeneous node type or do different Node Pools exist
- Availability of IP addresses

The following recommendations for pool allocation are based on OneFS 8.0.1.x code and later. All recommendation should be assessed in context to all Isilon workflows the cluster will participate in.

Racks Not Required – No Data Locality in Use

Single Pool

When no Racks are implemented, the networking required can be simplified with current releases of OneFS (OneFS 8.x and later). The additional capabilities offered by DataNode Load Balancing and Data Pipeline Recovery allow us to leverage a simple single pool architecture with static IP allocation. This is likely a very common scenario and represents the majority of deployed clusters. It is the simplest and easiest configuration to deploy.

Best Practice: Implement a single static IP pool with a configured SmartConnect zone name

Racks Required – Data Locality to be Implemented

Multiple Pools

When racks are to be deployed to facilitate location-aware compute and Isilon data locality, we need to implement additional network pools as follows:

NameNode Pool: Static pool - All Isilon Nodes that will be part of HDFS services

DataNode pools: Static pools -An IP pool per defined Rack for allocation to a client IP range

- Rack Pool: A per client IP range defined rack associated with an IP Pool of specific Isilon Nodes

Additional information on configuring pools and racks can be found in the OneFS documentation. Previously, a recommended approach to using racks was to implement a default rack to create two pools: 1. NameNode pool and 2. DataNode pool. Due to enhancements in OneFS, this is no longer the recommended approach.

Best Practice: Implement a single static NameNode pool with a configured SmartConnect zone name and a per defined Rack static pool for DataNodes. No SmartConnect name is required on the DataNode pools.

Multi-Use Pools

Since Isilon supports multiprotocol data access to the file system it may be required to leverage IP pools and SmartConnect for other protocols. In this situation standalone pools can clearly be used but existing HDFS pools can be used also. The pools and allocation will depend on the additional protocols in use.

Protocols in Use	**Pool Allocation**
HDFS + NFS	Dynamic Pool
HDFS + SMB	Static Pool
HDFS + NFS + SMB	Dynamic or Static (requirement specific)

Node Allocation in SmartConnect Pools

The number of nodes and interfaces assigned to the IP pools is determined by your requirements and throughput needs. Allocate the nodes and interfaces to meet your base requirements. When nodes of a different class are mixed in a single pool this can create a disparity in the performance characteristics of node responsiveness, and this can have adverse effects on the behavior of jobs if there is a significant difference in the data availability based on node types. An example of mixing X410 and HD400 is an extreme example, however it illustrates the potential bottlenecking of data access from archive HD400 nodes versus high performance X410 nodes.

Best Practice: Assign only similar node types to the IP Pools to provide consistent performance

IP Address Allocation by Interface

This approach consists of binding multiple IP addresses to each node interface in a pool. The ideal number of IP addresses per interface depends on the size of the pool and the allocation methodology. Because static pools include no failover capabilities, a static pool requires only one IP address per interface. Pools should always be allocated with enough IP's to support growth of the pool. The following recommendations apply to dynamic pools only.

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

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

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

Related to failover, if the IP pool is small you can get an imbalance in IP distribution across nodes. See the EMC Isilon External Network Connectivity Guide for more information. The IP allocation policy 'static versus dynamic' is related to failover behaviors of protocols.

LACP and Bonding

The link aggregation control protocol (LACP) provides resiliency of network connection in the event of NIC or port loss. In general, HDFS workflows do not benefit from this, as automatic retry mechanisms exist within HDFS services and job execution.

Best Practice: Leverage all node interfaces in the pool as single interfaces that are not bonded to optimize network throughput

Jumbo Frames – MTU 9000

Network connectivity between the compute nodes and OneFS nodes can benefit from increased frame sizes when the entire network is configured to use a larger frame size. If any interfaces or switch ports are not configured correctly for large frame sizes then additional overhead and buffering may occur likely leading to additional delay and latency.

Best Practice: Implement Jumbo Frames only when all clients, switch ports, and Isilon interfaces are configured for MTU 9000

OneFS HDFS Settings

The following settings relate to Isilon OneFS specific HDFS configuration settings.

Checksum Type

This is a setting that was implemented to allow Isilon to work with older HDFS clients that required it. OneFS does not checksum in the traditional sense that the HDFS file system does. OneFS is not 3X mirroring the files but instead is storing them using OneFS's own protection scheme utilizing Forward Error Correction (FEC). OneFS's on disk representation of the file is different than a file on HDFS (3x mirrored), so any block-based checksums will always be different. This also explains why check summing is not supported with DistCP from HDFS DAS – Isilon, as the underlying protection of the file is different on an Isilon cluster and integrity is handled by OneFS. Any comparison of block checksum will fail as the source and target file systems are not implementing the same HDFS-based file system. This is a legacy setting for use with specific application that require to see that a specific type of checksum has occurred, in most cases it can be left at the default.

HDFS Blocksize

This setting does not manage the On-disk stored block size that is configurable on standard HDFS file systems. On an Isilon cluster, raising the HDFS block size from the default of 64 MB to 128 MB optimizes performance for most use cases. Boosting the block size lets Isilon nodes read and write HDFS data in larger blocks, which can decrease drive-seek operations and increase performance for MapReduce jobs.

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

The server-side block size determines how the OneFS HDFS daemon returns data to read requests. The HDFS clients must contain enough memory to accept the server-side block size. If your server-side block size, for example, is set to 512 MB, and if a client's maximum map JVM size is set to 512 MB, the client will be unable to pull files from the cluster. The server-side block size also controls the number of map tasks and the file input size of each map task.

Best Practice: Validate your workflow and set the appropriate HDFS blocksize on OneFS.

HDFS Threads

HDFS threads tuning is no longer tunable in OneFS 8.x code and above. This setting was depreciated and should not be tuned.

NANON with Ambari - Not All Nodes On Network

When integrating OneFS with a Hortonwork's Ambari cluster, the Isilon Ambari agent service will send a heartbeat and metrics to the Ambari management server. If the agent service on OneFS is located on a node that does not have network connectivity to the Ambari server, the Isilon cluster will appear offline. Since the location of the OneFS Ambari agent is managed by OneFS and not controllable, the agent can be located on any node in the cluster. The inability of the agent to not send heartbeat to Ambari does not affect the ability to run HDFS jobs as NameNode and DataNode connectivity is dependent only on the SmartConnect configuration and connectivity from compute nodes to Isilon nodes. However, to ensure consistent operation of OneFS with Ambari, we recommend that you do not have nodes without network connectivity to the Ambari host. OneFS does not run a Cloudera agent service, so this setting is not applicable when integrating with Cloudera Manager.

Best Practice: We recommend that you have all nodes on the network when using HDFS with OneFS, no NANON nodes..

Improving HDFS Write Throughput – Isilon Write Coalescer

If the amount of allocated space to the OneFS Write Coalescer is too small, and the amount of writes is high, it can be forced to flush to disk too frequently. Increasing disk space can allow more writes to occur before flushing is forced to occur. During this flush, the I/O thread needs to sleep, and thus cannot service the write. If we increase the write coalescer buffer sizes, then the system can buffer more in-flight data while the system can do a lazy flush in the background. This applies only to: Generation 5 nodes (S210, X410, NL410, HD400 and earlier) with OneFS 8.0 and earlier.

Identifying symptoms of frequent flushing:

Identifying when you may be in this situation requires using a sysctl to see if the number of forces that the write coalescer flushes increases at a significant rate. You also need to be running a heavy write-intensive work load, and you need to monitor the value.

Run the following sysctl command to identify symptoms of frequent flushing:

```
        sysctl efs.bam.coalescer_fstats | grep flush
```

Example output:

```
        flush.count=7655

        flush.bytes=432481554

        flush.full=107

        flush.reason_lock_loss=0

        flush.reason_age=5833

        flush.reason_lwm=1220

        flush.reason_frags=0

        flush.reason_explicit=340
        flush.reason_overlap=0
        flush.reason_forced=170
        flush.reason_forced_full=107
```

Run the sysctl command every 30 seconds or so and collect the output. What you look for to identify the issue is the flush.reason_forced_full increasing at a significant rate. In this example, the flush.reason_forced_full value was very high. This value is reset on reboot, and it is kept on a per node basis.

```
         flush.reason_forced_full=9084779
```

You can see from above that a normal cluster should have this value at a low value, while a cluster that can use an increase in the coalescer has a very high value. The value also increased at a rate of more than 1 per second. If you see the value being very high, and during write testing the value is increasing at a rate of 1/second of faster, then you have a high likelihood of being impacted.

The following sysctl change takes effect immediately and needs to be set on each node. You can do this using the `isi_sysctl_cluster` command to keep it persistent across reboots.

There are three values that you can attempt. Start with the smallest value and move up while benchmarking at each step. The major impact of increasing this value is to consume more memory for the write coalescer. If you have a node that has the minimum RAM for your model, then you may want to only try the first two values.

The sysctls you need to modify are as follows:

```
        efs.bam.coalescer.insert_hwm
        efs.bam.coalescer.insert_lwm
```

Check your default values first:

```
        # sysctl efs.bam.coalescer.insert_hwm
        efs.bam.coalescer.insert_hwm: 47185920

        # sysctl efs.bam.coalescer.insert_lwm
        efs.bam.coalescer.insert_lwm: 41943040
```

The values above are defaults for Gen 5. Change the values to one of the following:

Low

```
        efs.bam.coalescer.insert_hwm=251658240
        efs.bam.coalescer.insert_lwm=125829120
```

Medium

```
        efs.bam.coalescer.insert_hwm=402653184
        efs.bam.coalescer.insert_lwm=201326592
```

High

```
        efs.bam.coalescer.insert_hwm=524288000
        efs.bam.coalescer.insert_lwm=262144000
```

The values above are in bytes. So the low values are 120 MiB for the low water mark and 240 MiB for the high water mark. Medium is 192 MiB and 384 MiB. High is 256 MiB and 512 MiB. The values above are not any required sizes. The above values are just good base 2 sizes in MiB.

L1/L2 Cache Size

We recommend that you leverage Isilon nodes with as much RAM as possible for analytics workflows.

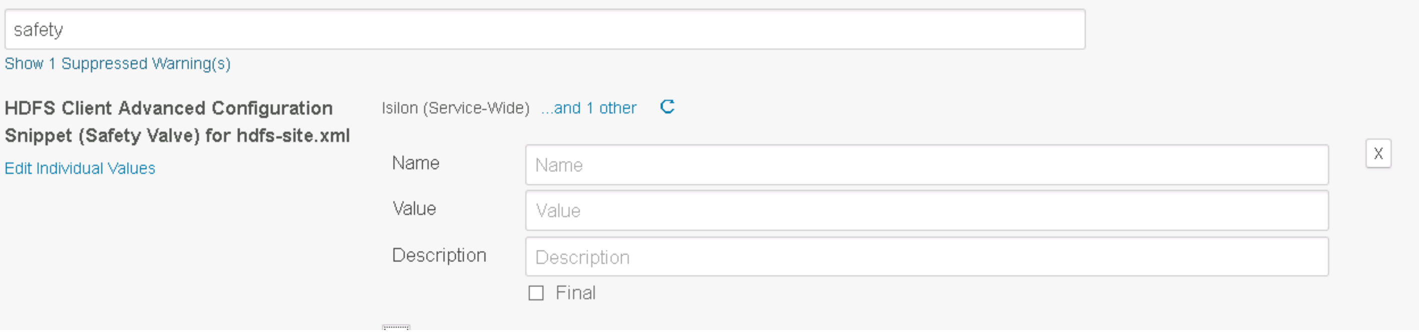
Global Namespace Enabled

Global Namespace Acceleration (GNA) has a specific use case; it is suggested when specific cluster nodes do not have SSD. The Isilon cluster as a whole must support GNA to use it also. You must have the appropriate amount of SSD and the correct number of nodes to enable it. GNA enables data stored on node pools without SSDs to access SSDs elsewhere in the cluster to store extra metadata mirrors. Extra metadata mirrors accelerate metadata read operations. Running analytics against node pools that require GNA is not recommended.

Hadoop Cluster Configuration

The following section highlights Hadoop cluster configuration parameters and how they interact with OneFS. It may be necessary to add custom properties in a Safety Valve or add a property to configure settings.

In Cloudera Manager locate the configuration of the service you wish to add a safety valve to.



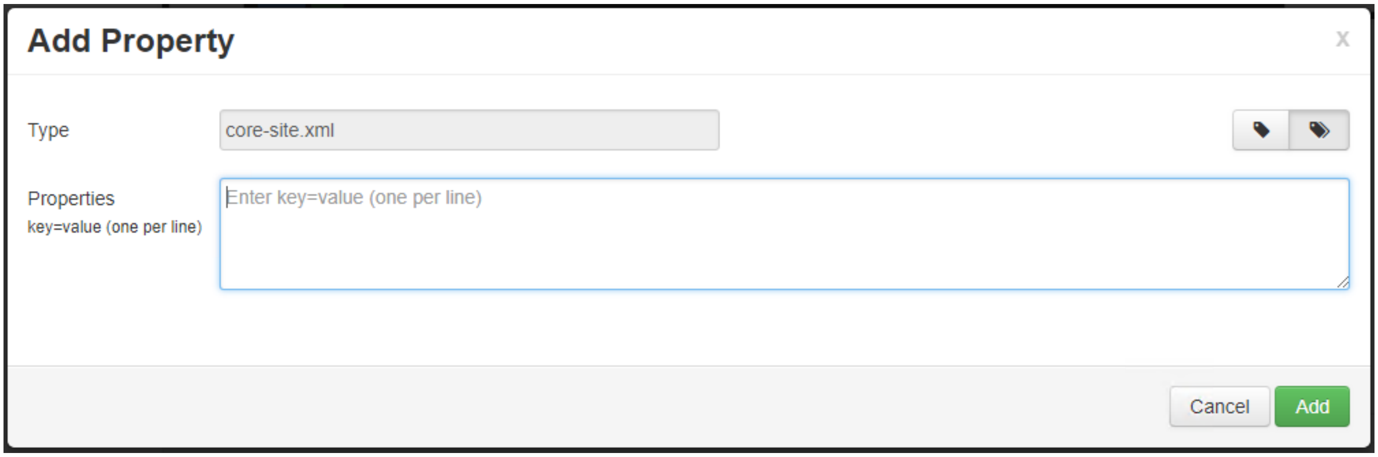
safety

HDFS Client Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml

Isilon (Service-Wide) ...and 1 other

Figure 3 Set a Safety Valve in Cloudera Manager to set a custom setting

Or in Ambari, navigate to **HDFS > Advanced > Custom core-site** or **hdfs-site** and in the **Add Properties** dialog box, create the key.



Add Property

Type: core-site.xml

Properties key=value (one per line): Enter key=value (one per line)

Figure 4 Add a custom property in Ambari

Hadoop Compute Settings

Review the following setting on your Hadoop cluster and determine if it makes sense to make changes for your environment.

mapred.local.dir

Local high-speed disk SSD is preferred for scratch space. A local direct attached disk will always be faster than a NFS-mounted file system for a scratch disk. This represents a standard analytics workflow: The shared data -- remote/HDFS; the scratch/shuffle space -- local to node for better performance. If sufficient local disk is not available, then we can use an NFS mount to provide additional disk space to a disk lite client through NFS from the same Isilon cluster, but it will always have performance implications, as you are placing this data across the wire on the same shared resource that HDFS is using. This setting would allow modification to the location of the mapred shuffle space if needed.

mapred.compress.map.output

This setting is a client-side configuration to compress the output of maps before sending them across the network. The default value is to leave this setting enabled.

dfs.replication

This setting is ignored by OneFS. OneFS will always protect the file at the requested protection level of the node pool. It can be changed on compute side, but it has no effect on Isilon resident data. The default value is three, this can be left as it is ignored by OneFS.

dfs.permissions

Since OneFS is not an HDFS based file system, this is ignored by OneFS on data access. Permission management is completely managed by OneFS, as the remote file system that OneFS implements has a unified access policy. Access tokens or file permissions must allow access to the underlying data to enforce a multiprotocol model. The permissions bypass mode is not able to be managed from Hadoop with this flag. The default value is to leave this enabled as the setting is ignored by OneFS.

dfs.client-write-packet-size

Hadoop transmits data in the form of packets (the default size is 64KB). This parameter can be tweaked with the **dfs.client-write-packet-size** option in the Hadoop **hdfs-site.xml** configuration.

Best Practice: Set the **dfs.client-write-packet-size** to 131072.

dfs.checksum.type

OneFS ignores the checksum. If it is left off, fewer packets are sent (resulting in performance improvements).

Best Practice: The **dfs.checksum.type** in the HDFS-site should be set to NULL.

Hortonwork's HDP - Scheduler

yarn.scheduler.capacity.node-locality-delay

On the **Customize Services** screen for the Yarn service, on the **Advanced** settings tab, set **yarn.scheduler.capacity.node-locality-delay** to 0.

Best Practice: Set the yarn.scheduler.capacity.node-locality-delay to 0.

Cloudera uses the org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler by default, no additional tunings have been identified here currently.

HDFS OPERATION OPTIMIZATIONS

This section describes strategies and options for tuning an Isilon cluster to improve performance for Hadoop data sets, workflows, and workloads. As with all tuning parameters or configuration changes, the effect of the tuning should be evaluated and assessed. We recommend that you measure the required operation, make the tuning adjustment and reevaluate the operation for a movement in performance. Hadoop configurations are extremely complex with many hundreds of tuning parameters available. The following represents some high-level configuration changes that can be implemented on OneFS to assess performance tunings.

Data Protection - OneFS Protection Levels

OneFS takes a more efficient approach to data protection than HDFS. The HDFS protocol, by default, replicates a block of data three times to protect it and to make it highly available (3X Mirroring). Instead of replicating the data, OneFS stripes the data across the cluster over its internal InfiniBand network and protects the data with forward error correction codes.

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

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

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

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

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

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

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

Best Practice: Leverage an OneFS protection policy that meets the requirements and suggested level for the Isilon cluster configuration. If additional nodes are added, the protection policy may need to be reevaluated.

Aligning Workloads with Data Access Patterns

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

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

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

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

ONEFS ACCESS PATTERN	DESCRIPTION
Concurrent(default)	Prefetches data at a medium level
Streaming	Prefetches data far in advance of requests
Random	Prefetches no data

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

Since Access patterns can be defined on a per directory basis, it is worthwhile testing and evaluating different access patterns on different workflow data against different tools; an hbase job may benefit from a Streaming pattern while a mapred job may benefit from a Concurrent access pattern.

Best Practice: Test and validate the Access Pattern against each data set and the jobs accessing it.

Note: It is likely that different HDFS workflows will benefit from different access patterns as their fundamental data access is different; mapred jobs may benefit from streaming while database based workflows Hive may benefit from concurrent, even read and write directories may benefit from different patterns.

Write Caching with SmartCache

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

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

SSD Usage

L3 Cache is the default SSD usage setting for OneFS 8.x. This is the recommend initial setting for HDFS node pools. Specific workflows may benefit from a different SSD Strategy, likely Metadata R/W, but this should be evaluated on a cluster specific basis for any benefits. L3 Caching was completely rewritten in OneFS 8.x and represents the recommended approach for optimizing SSDs.

L3

- Uses more of the SSDs effectively
- Allows for Read Caching

Metadata Read and Write

- Uses more of the SSDs
- Both Metadata Mirrors live on SSD
- Optimizes Metadata Write operations

Metadata Read

- Uses the least amount of SSDs
- Only one Metadata Mirror lives on SSDs
- Optimizes Metadata Read Operations

Best Practice: L3 is the recommended default settings. Specific workflows can be tested against alternate strategies for performance. However, since SSD is node pool-specific, the impact to other workflows (if they are present on those nodes) should be taken into consideration.

ALIGN DATASETS WITH STORAGE POOLS

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

Because you can combine Dell EMC Isilon nodes from the Generation 6 nodes; F, H and A and Generation 5 nodes; S, X, NL and HD nodes into a cluster, you can set up pools of nodes to accelerate access to important working sets while placing inactive data in more cost-effective storage.

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

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

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

SmartPools for Analytics Data

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

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

Guidelines for File Pool Management

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

- Plan file pool policies carefully, especially the sequence in which they are applied. Without proper planning and analysis, policies can conflict with or override one another.
- Note that directories with large amounts of stale data—for instance, data that has not been accessed for more than 60 days— can be migrated with a file pool policy to archive storage on an NL-Series pool. OneFS supplies a template for this policy.
- If directories take a long time to load in a file manager like Windows Explorer because there are a large number of objects— directories, files, or both—enable metadata acceleration for the data to improve load time.

For more information on SmartPools file pool policies, see Next Generation Storage Tiering with Dell EMC Isilon SmartPools and the OneFS Administration Guide for your version of OneFS.

Storage Pools – NodePools for Different Data Sets

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

Storing Intermediate Jobs on an Isilon Cluster

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

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

/ifs/access-zone/compute

/ifs/access-zone/compute/client1

/ifs/access-zone/compute/client2

/ifs/access-zone /compute/clientn

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

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

Dealing with Space Bottlenecks

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

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

Review Data Settings on Files and Directories

You can check the attributes of directories and files to see your current settings by running the following command and replacing `<data>` in the example with the name of a directory or file. The command’s output, which shows the properties of a directory named data, is abridged to remove some immaterial information.

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


        ******************************************
        NEW FILE ATTRIBUTES  ❺
        Access attributes:  active
        Write Cache:  on
        Access Pattern:     concurrency
```

```
      At_r: 0
      Protection attributes:     active
      Protection Policy:  Diskpool default
      Disk pools:  policy any pool group ID
      SSD Strategy: metadata
      *************************************************
```

Here is what some of these lines mean (for the numbers in circles in the output above)

1. This is a OneFS command to display the file system properties of a directory or file. For more information, see the [OneFS Command Reference]().
2. The directory's data access pattern is set to concurrency.
3. Write caching (SmartCache) is turned on.
4. The SSD strategy is set to metadata, which is the default.

Files that are added to the directory are governed by these settings, most of which can be changed by applying a file pool policy to the directory.

Hadoop and OneFS Kerberos Integration

OneFS integrates with Hadoop cluster Kerberization to provide secure and controlled data access. OneFS provides a unified permission model to facilitate unified access and secure permissioning of data. Kerberization of a Hadoop cluster with an Isilon cluster requires full understanding of the authentication model to be implemented and the additional steps that will be required to integrate the two systems. It is strongly recommended that you review the following Kerberos Installation Guides before starting this process. Also all Hadoop cluster Kerberos requirements must be met.

It is also worth reviewing the following OneFS permissioning model blog: [EMC Isilon Multiprotocol Concept Series]().

Implementation Approaches

It is important to recognize that Kerberos is just authentication, it validates 'you are who you say you are'. Isilon like most current Hadoop distributions support both standalone KDC based Kerberos and Microsoft's Active Directory for Kerberos Authentication. An additional challenge with Enterprise deployments is the addition of Directory Services based Identity Management (LDAP or AD). Additional consideration beyond the scope of this paper need to be considered when implementing LDAP or AD providers to ensure Isilon Identity Management is handled correctly. A number of post on this topic can be found here: [Using Hadoop with Isilon - Isilon Info Hub]()

Before implementing Kerberos on an Isilon integrated Hadoop cluster review and follow the steps in the following guides.

Ambari and Hortonworks HDP

It is highly recommended to consult the Isilon Kerberization documents prior to attempting to Kerberize the Ambari and Isilon clusters

- [EMC Isilon OneFS with Hadoop and Hortonworks for Kerberos Installation Guides]()
- [Isilon OneFS with Ambari Multitenant Active Directory Integration Guide]()

Cloudera Manager CDH

- [EMC Isilon OneFS with Hadoop and Cloudera for Kerberos Installation Guide]()

DNS – SmartConnect

Kerberos implementations are highly dependent on DNS for their operation, the following DNS requirements must be met before integrating Isilon with Hadoop for Kerberos.

- Forward and reverse DNS lookups must be enabled on all the hosts.
- All the compute hosts must have forward DNS lookup resolved correctly for all the hosts.

- Isilon SmartConnectZone Name lookups must resolve correctly.
- Reverse PTR records for all IP addresses in the SmartConnect pool must exist. All IP's in a pool must have PTR's, static pools will limit the number of IP's needing PTR records
- Isilon OneFS must be able to resolve all the hosts, KDCs, and Active Directory servers as needed.

Time

Kerberos implementations are highly dependent accurate time, all Hadoop nodes and OneFS must be configured to use a reliable accurate time source such as NTP. This should be a strata 3 or higher time source. Oftentimes an AD or DNS controller will also server out NTP requests on an enterprise network because of the high availability nature of these infrastructure services.

Isilon SPNs

Review the Kerberos installation guides for specific guidance on configuring Service Principal Names (SPN's) and user mappings when transitioning from local accounts to Directory Service based accounts.

- Validate the required SPN's are present for Isilon
- Correct versions of KDC SPN's keytbabs (KVNO) are present on Isilon

ID Management

Having completed Kerberization of the Hadoop and Isilon clusters validate that Hadoop Service Account Name resolution is still fully operational. Review the Isilon Hadoop Kerberos installation guides for additional details.

hadoop.security.token.service.use_ip

Since Isilon connectivity is handled by a SmartConnect service name which handles IP allocation of NameNode and DataNode IPs, the following setting must be configured to allow interaction between compute nodes and Isilon nodes to successfully handle Kerberos negotiations.

Best Practice: Add or modify the **hadoop.security.token.service.use_ip=false** and set its value to false in the core-site.xml.

Isilon Hadoop Proxy Users

When a Hadoop cluster and Isilon are Kerberized, Hadoop Proxy users are required on Isilon to allow impersonation of accounts. The required proxy users should be created and populated with the correct users and groups that require. If the cluster is not Kerberized then Proxy users are not required.

A Kerberos user: hdpuser3 tries to run a hive query, and no proxy user exists you will see the following errors and log entries.

```
 [hdpuser3@hdp4 ~]$ kinit
Password for hdpuser3@FOO.COM:

[hdpuser3@hdp4 ~]$ klist
Ticket cache: FILE:/tmp/krb5cc_49003
Default principal: hdpuser3@FOO.COM

Valid starting     Expires            Service principal
05/02/16 14:53:51  05/03/16 00:52:08  krbtgt/FOO.COM@FOO.COM
```

```
        renew until 05/09/16 14:53:51



[hdpuser3@hdp4 ~]$ hive

WARNING: Use "yarn jar" to launch YARN applications.


Logging initialized using configuration in file:/etc/hive/2.3.4.0-3485/0/hive-log4j.properties

hive> SHOW TABLES;

FAILED: SemanticException
MetaException(message:org.apache.hadoop.ipc.RemoteException(java.lang.SecurityException): User
hive/hdp6.foo.com cannot impersonate User hdpuser3@FOO.COM Reason: Proxyuser "hive" Members do not
intersect)

hive>
```

Isilon log:

```
2016-04-25T17:23:17-04:00 <30.6> rip1-1 hdfs[93963]: [hdfs] RPC V9  user: hdpuser3@FOO.COM exception:
java.lang.SecurityException cause: User hive/hdp6.foo.com cannot impersonate
User hdpuser3@FOO.COM Reason: Proxyuser "hive" Members do not intersect
```

Adding a Proxy User; hive and adding hdpuser3 to that proxy user will resolve this issue.

```
rip1-1# isi hdfs proxyusers members list --zone=horton2 hive

Type  Name

----------------------

user  hdpuser3

----------------------

Total: 1



hive> SHOW TABLES;

OK

passwords

Time taken: 1.612 seconds, Fetched: 1 row(s)
```

Best Practice: Create all the required Access Zone specific Proxy Users before Kerberization, It is recommended to always use the Isilon Hadoop tools to create Users and Groups in the local provider, it will also create all the required Proxy users and add initial members. All additional Users and Groups will need to be added manually.

Monitoring and Performance with InsightIQ

We recommend that you set up and configure InsightIQ to monitor the OneFS cluster for Hadoop workflows. The complex nature of clustered workflows against multiple nodes on OneFS makes InsightIQ an ideal tool to use to review and validate the operational status and performance of the OneFS cluster. InsightIQ provides:

Powerful cluster monitoring and reporting

- Live performance statistics for cluster, node, protocol, client, and more
- View and drill into throughput, operations, CPU utilization, and so on, to plan optimization efforts

Flexible reporting on file system characteristics

- View "top N" and drill into directory and file sizes, ages, access time, and so on
- Track health and performance of the cluster

Capacity forecasting

- Estimate capacity utilization based on selected time frames
- Plan for cluster expansion or file system cleanup activities

Hadoop Tiered Storage

The introduction of Hadoop Tiered Storage, utilizing Isilon and a DAS cluster to create multiple storage tiers with persistent data access to all the data introduces increased flexibility and a new operational model for integrating and managing HDFS data. The following configurations should be implemented to support Isilon Hadoop Tiered Storage.

Additional information on Isilon Hadoop Tiered Storage can be found here:

- Hortonworks and Isilon Hadoop Tiered Storage Solution Guide
- Cloudera and Isilon Hadoop Tiered Storage Solution Guide (work in progress, June 2018)

Unique HDFS Roots

Best Practice: Multiple Hadoop clusters should not share or write to a single Isilon Hadoop root zone

Kerberos – Kerberos Only

Best Practice: Security parity should be maintained across Hadoop HTS integrated clusters; Only Simple(DAS) – Simple(Isilon) or Kerberos(DAS) – Kerberos(Isilon) should be used for HTS.

Additional Best Practices, updated May 2018

The additional Best Practice have been identified and should be evaluated depending on the deployment and implementation approaches.

fs.trash Setting

If your Hadoop cluster is set to use the fs.trash.interval with HDFS on Isilon, any files deleted without using the -skipTrash switch will be moved to the .Trash directory under the user executing. Currently a known limitation is that these files will not be checkpointed and automatically deleted according to the fs.trash.interval. So these files will not get deleted over time and may consume space on the cluster.

The fs.trash.interval is found in the core-site.xml:

```
    <property>
<name>fs.trash.interval</name>
      <value>1440</value>
    </property>
```

With trash enabled, you see the following on removal of the file:

```
[hdfs@cdhcm2 hadoop-mapreduce]$ hadoop fs -rm /user/russ/test.txt
15/02/10 16:28:02 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 1440
minutes, Emptier interval = 0 minutes.
Moved: 'hdfs://cdhcm2.foo.com:8020/user/russ/test.txt' to trash at:
hdfs://cdhcm2.foo.com:8020/user/russ/.Trash/Current
```

The removed file will then sit in the .Trash folder until manually deleted on Isilon.

While with trash disabled you will see the following:

```
[yarn@hdp4 conf]$ hadoop fs -rm /user/russ/test2.txt
15/02/10 10:19:53 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 0
minutes, Emptier interval = 0 minutes.
Deleted /user/russ/test2.txt
```

The file will be deleted and not moved to .Trash

The trash setting is often set to on as default, if your workflow is removing files without -skipTrash you may not be 'actually' getting files deleted and additional space may be used.

If you believe you need the ability to recover files, consider using SnapShotIQ on the original file as one method of obtaining data recoverability.

Best Practices:

1. Don't enable fs.trash.interval, set it to 0 in the core-site.xml
2. Always use -skipTrash when deleting files via hadoop commands if a trash interval is set
3. Delete data via another Isilon method to bypass .Trash - nfs/smb etc
4. Manually cleanup .Trash if in use
5. Monitor .Trash for undeleted files

Enablement of FSImage and iNotify on Isilon

OneFS 8.1.1.0 enabled the integration of Isilon OneFS with Cloudera Navigator. For additional details see: OneFS and Cloudera Navigator Support and OneFS 8.1.1.0 and Cloudera CDH 5.13+ Support for Cloudera Navigator The integration is implemented by enabling an FSImage and iNotify log on the HDFS access zone.

Best Practice: Enable FSImage and INotify prior to Cloudera deployment of Navigator

Best Practice: FSImage and iNotify should only be enabled on zones that require Navigator integration, it adds additional load to the zone and unless it is being leveraged it should not be enabled

Best Practice: Once enabled on a zone, There is no known need to adjust the duration of HDFS FSImage and INotify jobs unless instructed to by support

Best Practice: Once enabled on a zone for integration it should not be disabled unless zone integration is no longer required

distcp

Isilon implements the OneFS file system for data layout and protection and not a HDFS based file system with mirrored blocks, since multiple mirrors of blocks do not need to be checked during the copy as data integrity is handled natively via OneFS checksum on the blocks are not needed.

Best Practice: When using distcp to copy data to Isilon the -skipcrccheck option must be set, the source data must also be not active, or use a DAS snapshot as the source.

Job Engine

Best Practice: Validate the Isilon job engine schedule is not running maintenance jobs at the same time as critical Hadoop jobs, reschedule them to off hours if possible. All data protection jobs should be left to run automatically

Isilon Quotas - /tmp

Best Practice: Do not set Isilon quotas on temporary Hadoop job folder locations e.g.: /tmp /tmp/hive etc..

/tmp

Best Practice: Review and optimize the /tmp directory in the Hadoop root, potential optimizations include excluding from snapshot, reducing the protection to +2:1 and use a layout of on disk random

Hadoop Temporary Shuffle

Best Practice: Hadoop temporary shuffle space should not be located on Isilon for performance driven workflows. Some specific use cases VMWare based compute, Dev and Test environments can leverage NFS mount based shuffle to create support for deployments but there are significant performance consequences in co located shuffle and hdfs data on the same Isilon.

CONCLUSION

The complexity and sheer volume of configuration options within Hadoop clusters and its services make it extremely difficult to provide absolute Best Practice for many Hadoop services and workflows. We always recommend that you evaluate any Best Practice or tuning for applicability to your workflow and implementation. The Hadoop space is also changing rapidly—much quicker than traditional workflows—and it is always suggested to consult the latest version of documents for current approaches and new best practices outside of this document.

Additional and Latest Content

The primary source of updated information and content on OneFS and Hadoop integration is the Isilon Hadoop infohub. Check it regularly or subscribe for updates: https://community.emc.com/docs/DOC-3952

- **Isilon and Hadoop Cluster Install Guides**
- **Isilon OneFS with Hadoop Kerberos and Identify Management Approaches**
- **Hadoop Tiered Storage Solution Guide**
- **Prepare an Isilon for Hadoop Cheat Sheet**
- **OneFS with HDFS Reference Guide**
- **Isilon and Hadoop Multitenant Installation and Integration Guide**
- **Isilon OneFS with Ambari Multitenant Active Directory Guide**
- **Functional Deployment Architectures for VMware-based Hadoop Deployments with Isilon**
- **Isilon and Cloudera Backup and Disaster Recovery Integration**
- **Isilon and Cloudera Backup and Disaster Recovery Integration - Hive Metastore and Data Replication**
- **Isilon Customer Troubleshooting Guide - Troubleshoot HDFS and Ambari**
- **Isilon Customer Troubleshooting Guide - Troubleshoot HDFS and Ambari with Kerberos**
- **Isilon Customer Troubleshooting Guide - Troubleshoot HDFS and Cloudera**
- **Isilon Customer Troubleshooting Guide - Troubleshoot HDFS and Cloudera and Kerberos**
- **Hadoop Kerberization Checklist - Troubleshooting**

Contacting Dell EMC Isilon Technical Support

Online Support: https://support.emc.com/

Telephone Support:
United States: 800-782-4362 (800-SVC-4EMC)
Canada: 800-543-4782
Worldwide: +1-508-497-7901
Additional worldwide access numbers

Help with Online Support Tools:

For questions specific to HDFS, contact support@emc.com