**HDFS**

# Topics

* HDFS
* HDFS Architecture
* NameNodes and DataNodes
* NameNode in Hadoop-2
* Replication

HDFS

Hadoop Distributed File System (HDFS) is a Java-based file system that provides scalable and reliable data storage that is designed to span large clusters of commodity servers.

HDFS was designed to be a scalable, fault-tolerant, distributed storage system that works closely with MapReduce. HDFS will “just work” under a variety of physical and systemic circumstances. By distributing storage and computation across many servers, the combined storage resource can grow with demand while remaining economical at every size.

These specific features ensure that the Hadoop clusters are highly functional and highly available:

* **Rack awareness** allows consideration of a node’s physical location, when allocating storage and scheduling tasks.
* **Minimal data motion**. MapReduce moves compute processes to the data on HDFS and not the other way around. Processing tasks can occur on the physical node where the data resides. This significantly reduces the network I/O patterns and keeps most of the I/O on the local disk or within the same rack and provides very high aggregate read/write bandwidth.
* **Utilities** diagnose the health of the files system and can rebalance the data on different nodes
* **Rollback** allows system operators to bring back the previous version of HDFS after an upgrade, in case of human or system errors
* **Standby NameNode** provides redundancy and supports high availability
* **Highly operable.** Hadoop handles different types of cluster that might otherwise require operator intervention. This design allows a single operator to maintain a cluster of 1000s of nodes.

HDFS Architecture

An HDFS cluster is comprised of a NameNode which manages the cluster metadata and DataNodes that store the data. Files and directories are represented on the NameNode by inodes. Inodes record attributes like permissions, modification and access times, or namespace and disk space quotas.

The file content is split into large blocks (typically 128 megabytes), and each block of the file is independently replicated at multiple DataNodes. The blocks are stored on the local file system on the datanodes. The Namenode actively monitors the number of replicas of a block. When a replica of a block is lost due to a DataNode failure or disk failure, the NameNode creates another replica of the block. The NameNode maintains the namespace tree and the mapping of blocks to DataNodes, holding the entire namespace image in RAM.

The NameNode does not directly send requests to DataNodes. It sends instructions to the DataNodes by replying to heartbeats sent by those DataNodes. The instructions include commands to: replicate blocks to other nodes, remove local block replicas, re-register and send an immediate block report, or shut down the node.

According to 'Hadoop The definitive guide' - "The namenode manages the filesystem namespace. It maintains the filesystem tree and the metadata for all the files and directories in the tree."

Essentially, Namespace means a container. In this context is means the file name grouping or hierarchy structure.

Metadata contains things like the owners of files, permission bits, block location, size etc



NameNodes and DataNodes

**New File Creation**

**HDFS**

**Client**

**NameNode**

**AddBlock Request**

**DataNode Details**

**DataNode**

**DataNode**

**Write**

**DataNode**

**Blocks**

**Received**

**File Writing**

**Blocks**

**Received**

**HDFS**

**Client**

**NameNode**

**AddBlock Request**

**DataNode Details**

**DataNode**

**DataNode**

**DataNode**

**Write**

**Acknowledgement**

**File Reading**

**HDFS**

**Client**

**NameNode**

**Get Block Location**

**DataNode**

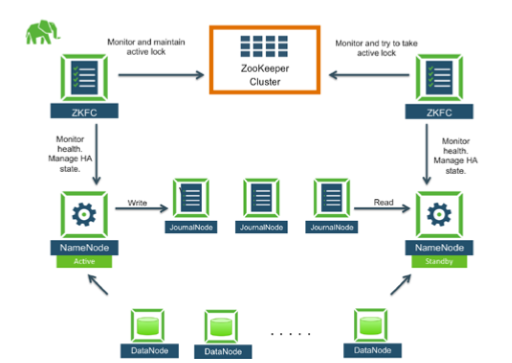
**DataNode**

**DataNode**

**Read**

**DataNode Details**

NameNode in Hadoop-2



Hadoop cluster’s storage resources were previously available only to HDFS. The new storage architecture generalizes the block storage layer so that it can be used not only by HDFS but also other storage services. The first use of this feature is HDFS federation, which allows multiple instances of HDFS namespaces to share the underlying storage. In future versions of Hadoop, other storage services (such as key-value storage) will use the same storage layer.

Replication

The NameNode endeavors to ensure that each block always has the intended number of replicas. The NameNode detects that a block has become under- or over-replicated when a block report from a DataNode arrives. When a block becomes over replicated, the NameNode chooses a replica to remove. The NameNode will prefer not to reduce the number of racks that host replicas, and secondly prefer to remove a replica from the DataNode with the least amount of available disk space. The goal is to balance storage utilization across DataNodes without reducing the block's availability.

When a block becomes under-replicated, it is put in the replication priority queue. A block with only one replica has the highest priority, while a block with a number of replicas that is greater than two thirds of its replication factor has the lowest priority. A background thread periodically scans the head of the replication queue to decide where to place new replicas. Block replication follows a similar policy as that of new block placement. If the number of existing replicas is one, HDFS places the next replica on a different rack. In case that the block has two existing replicas, if the two existing replicas are on the same rack, the third replica is placed on a different rack; otherwise, the third replica is placed on a different node in the same rack as an existing replica. Here the goal is to reduce the cost of creating new replicas.

The NameNode also makes sure that not all replicas of a block are located on one rack. If the NameNode detects that a block's replicas end up at one rack, the NameNode treats the block as mis-replicated and replicates the block to a different rack using the same block placement policy described above. After the NameNode receives the notification that the replica is created, the block becomes over-replicated. The NameNode then will decides to remove an old replica because the over-replication policy prefers not to reduce the number of racks.