# Deep Dive

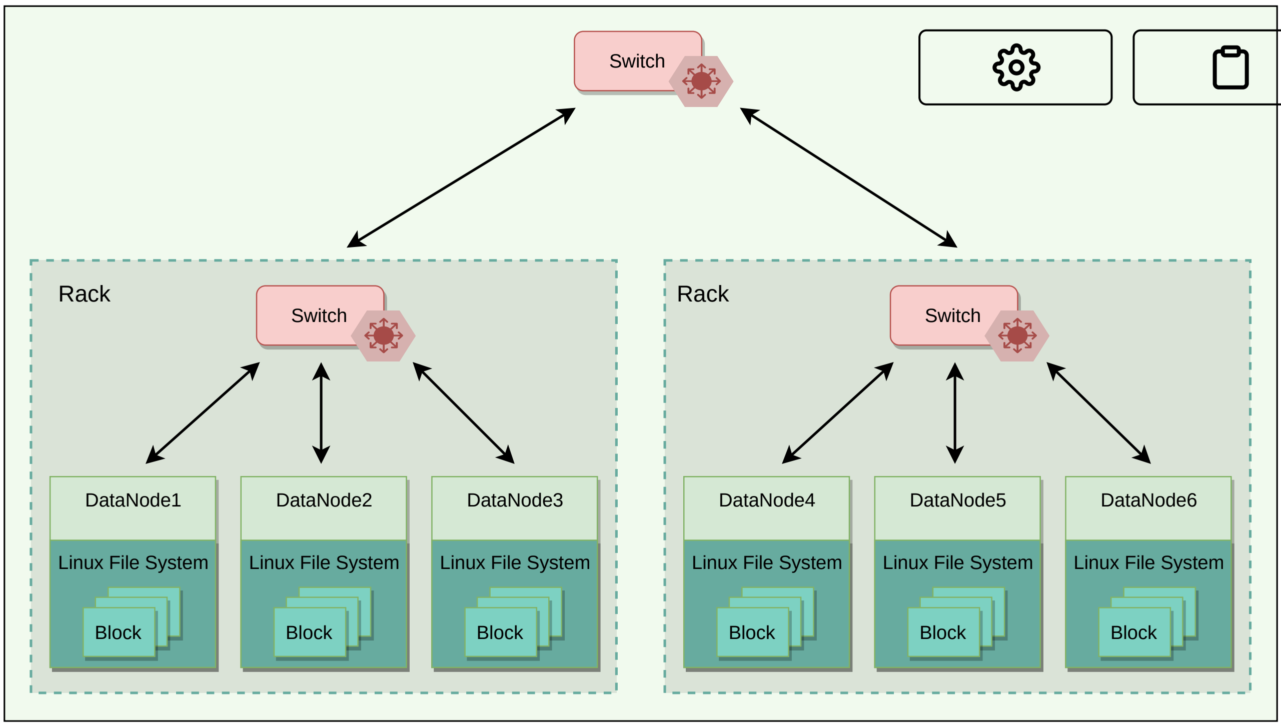
Let's explore some of HDFS's design components.

**We'll cover the following**

- Cluster topology
- Rack aware replication
- Synchronization semantics
- HDFS consistency model

## Cluster topology#

A typical data center contains many racks of servers connected using switches. A common configuration for Hadoop clusters is to have about 30 to 40 servers per rack. Each rack has a dedicated gigabit switch that connects all of its servers and an uplink to a core switch or router, whose bandwidth is shared by many racks in the data center, as shown in the following figure.



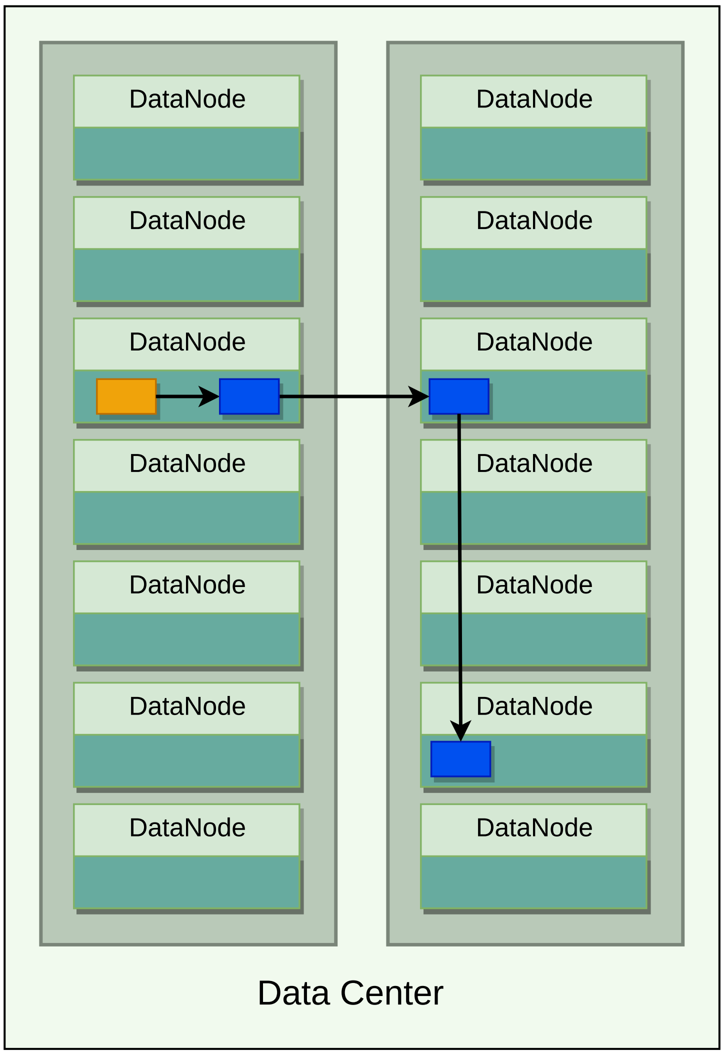
HDFS cluster topology

When HDFS is deployed on a cluster, each of its servers is configured and mapped to a particular rack. The network distance between servers is measured in hops, where one hop corresponds to one link in the topology. Hadoop assumes a tree-style topology, and the distance between two servers is the sum of their distances to their closest common ancestor.

In the above figure, the distance between Node 1 and itself is zero hops (the case when two processes are communicating on the same node). Node 1 and Node 2 are two hops away, while the distance between Node 3 and Node 4 is four hops.

## Rack aware replication#

The placement of replicas is critical to HDFS reliability and performance. HDFS employs a rack-aware replica placement policy to improve data reliability, availability, and network bandwidth utilization. If the replication factor is three, HDFS attempts to place the first replica on the same node as the client writing the block. In case a client process is not running in the HDFS cluster, a node is chosen at random. The second replica is written to a node on a different rack from the first (i.e., off-rack replica). The third replica of the block is then written to another random node on the same rack as the second. Additional replicas are written to random nodes in the cluster, but the system tries to avoid placing too many replicas on the same rack. The figure below illustrates the replica placement for a triple-replicated block in HDFS. The idea behind HDFS's replica placement is to be able to tolerate node and rack failures. For example, when an entire rack goes offline due to power or networking problems, the requested block can still be located at a different rack.



Rack-aware replication

The default HDFS replica placement policy can be summarized as follows:

1. No DataNode will contain more than one replica of any block.

2. If there are enough racks available, no rack will contain more than two replicas of the same block.

Following this rack-aware replication scheme slows the write operation as the data needs to be replicated onto different racks, but this is an intentional tradeoff between reliability and performance that HDFS made.

## Synchronization semantics#

Early versions of HDFS followed strict **immutable semantics**. Once a file was written, it could never again be re-opened for writes; files could still be deleted. However, current versions of HDFS support append. This is still quite limited in the sense that existing binary data once written to HDFS cannot be modified in place.

This design choice in HDFS was made because some of the most common MapReduce workloads follow the **write once, read many data-access** pattern. MapReduce is a restricted computational model with predefined stages. The reducers in MapReduce write independent files to HDFS as output. HDFS focuses on fast read access for multiple clients at a time.

## HDFS consistency model#

HDFS follows a **strong consistency model**. As stated above, each data block written to HDFS is replicated to multiple nodes. To ensure strong consistency, a write is declared successful only when all replicas have been written successfully. This way, all clients see the same (and consistent) view of the file. Since HDFS does not allow multiple concurrent writers to write to an HDFS file, implementing strong consistency becomes a relatively easy task.

← Back Next →

High-level Architecture

Anatomy of a Read Operation

Mark as Completed

Report an Issue