02 HDFS and Zookeeper

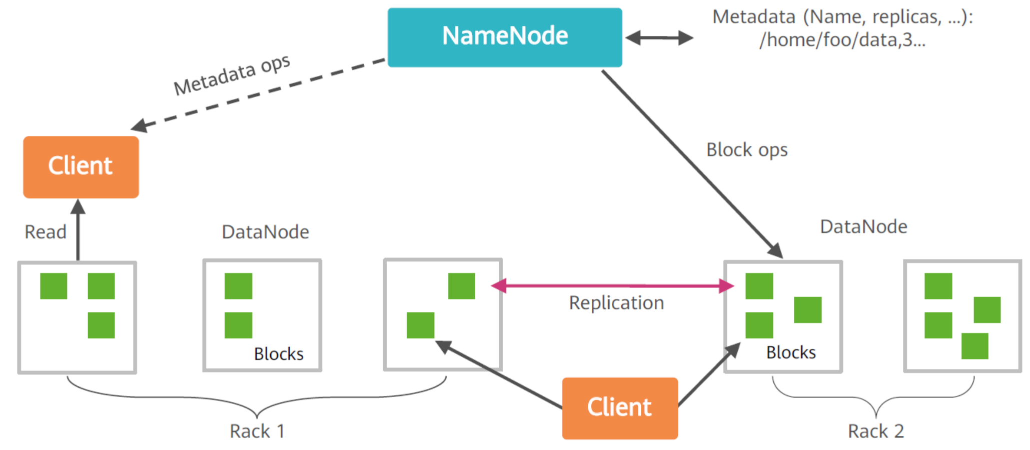
What the distributed file system?

- In a distributed file system files are stored on multiple computer nodes, thousands of computer nodes from a computer cluster
- Distributed file system is a system that allows files to be stored across multiple machines in a network but accessed and managed as if they were stored on a single local machine.
- Key features of the distributed file system
    - Data distribution
    - Scalability
    - Fault tolerance
    - Transparence
    - High availability

HDFS Overview

- Hadoop distributed file system is a distributed file system designed to run on commodity hardware
- HDFS is a part of the Apache Hadoop core project
- HDFS is a high fault tolerance and is deployed on cost-effective hardware
- HDFS provide high-throughput access to application data and is suitable for applications with large scale datasets

Basic HDFS System Architecture



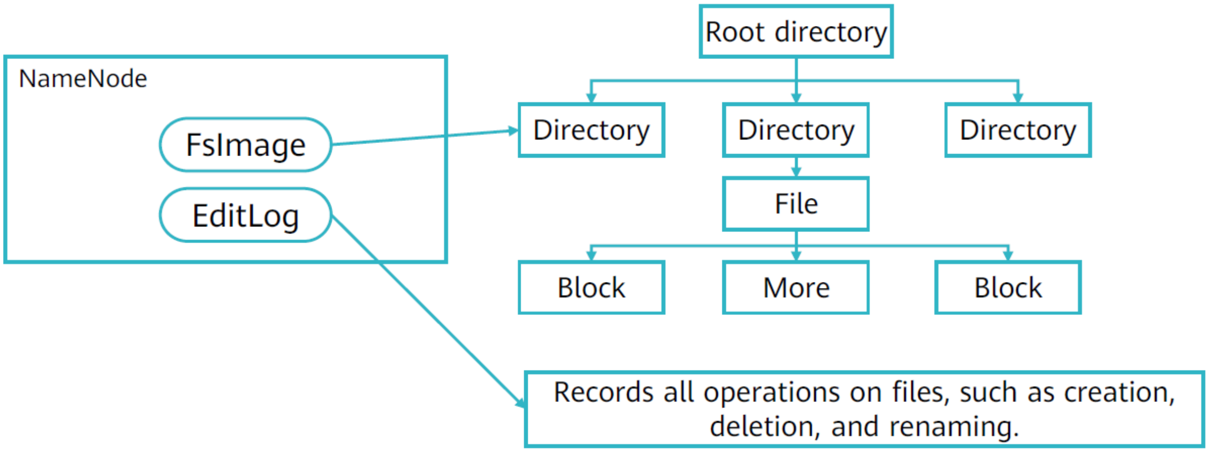
Basic HDFS system Architecture

1- Client

- Clients are the most common way of using HDFS.
- Is a library that contains HDFS interface that hide most of the complexity in the HDFS implementation
- Client is not a part of the HDFS
- It supports common operations such as opening, reading, and writing, and providing a shell-like command line mode to access data in HDFS
- HDFS also provides java APIs that serves as client programming interfaces for applications to access the file system

2- NameNode

- NameNode Manages the namespace of the distributed file system and stores two core data structures
  1- **FsImage**: maintain the metadata of the file system tree and all files and folders in that file tree
  2- **EditLog**: records all operations on file, such as creation, deletion, and renaming



3- DataNode

- DataNode is the worker node of the HDFS.
- It stores and retrieves data based-on the scheduling of clients or NameNode
- Data on each DataNode is stored in the local linux system of the node

Difference Between NameNode and DataNode

NameNode	DataNode
Stores metadata	Stores file content
Stores metadata in memory	Stores file content in the disk
Stores the mapping between files, blocks, and DataNodes	Maintains the mapping between blocks IDs and local files on DataNode

Block

- The default size of block is 128 MB. A file is divided into multiple blocks. A block is the storage unit
- The block has the following benefits
  - Large scale file storage
  - Simplified system design
  - Data backup

Metadata

- Metadata = information about the data
- It provides information that describes, explains, or gives content to another data, helping systems and users understand or manage the data
- Metadata helps in searching and organizing data
- Metadata includes
  - File permissions
  - Where each file is stored
  - Size and creation date
  - Which nodes have the pieces of the file

Namespace

- Namespace is a container or logical space that holds unique names or identifiers
- A **namespace** is a way to **organize and separate names** so that **identifiers (like file names, variables, or users)** can be reused **without conflict** in different contexts.

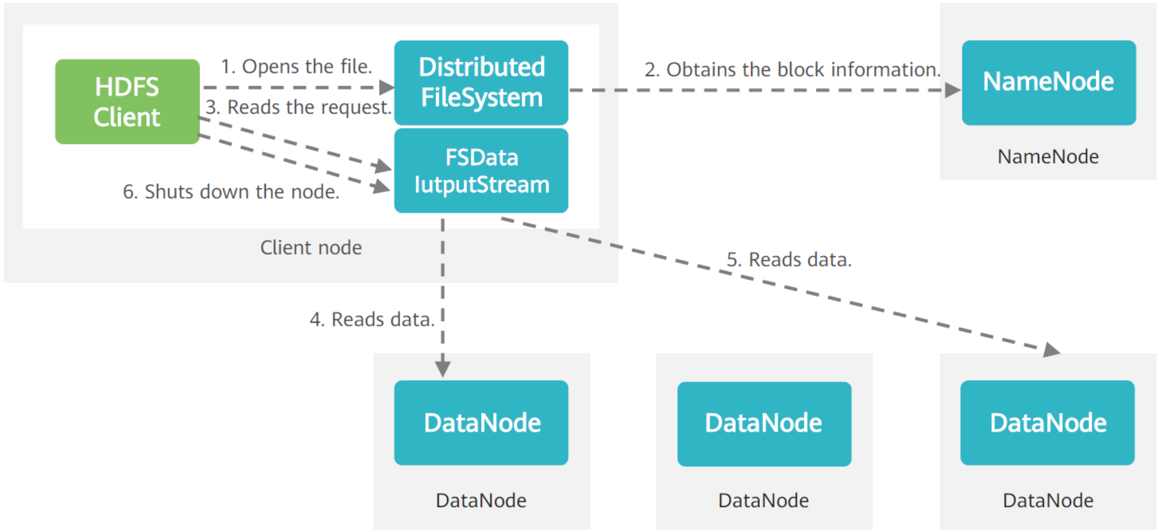
Communication Protocol

- HDFS is a distributed file system deployed on a cluster, therefore a large amount of data needs to be transmitted over the network
    - All HDFS communication protocols are based on TCP/IP
    - The client initiates a TCP connection to the NameNodes through a configurable port and uses the client protocol to interact with the NameNode
    - The NameNode and the DataNodes interacts with each other through the DataNode protocol

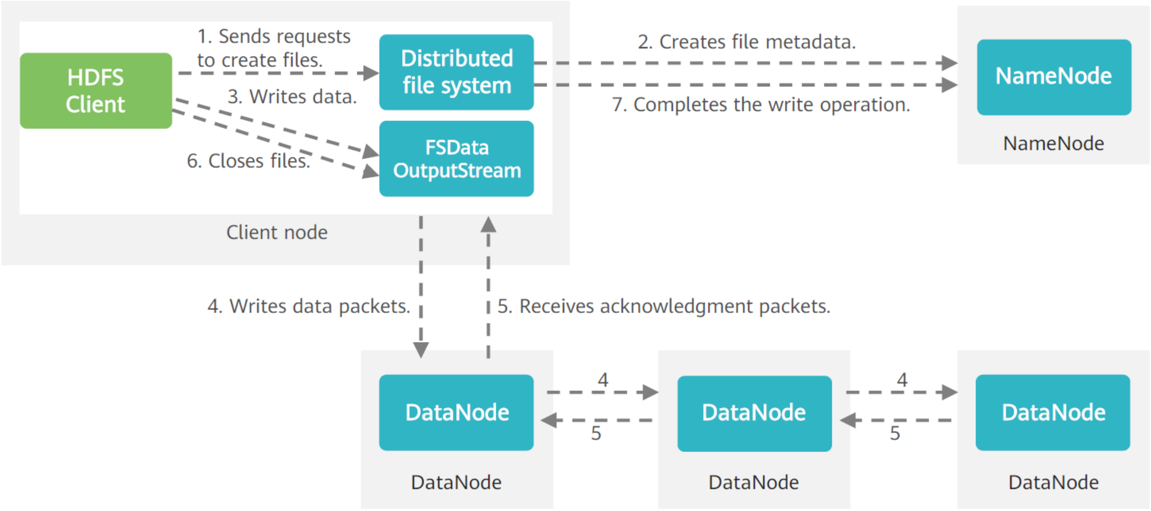
Disadvantages of the single NameNode Architecture

- **Namespace limitation**
    - NameNodes are stored in the memory, therefore the number of objects [files, and blocks] that can be contained in NameNode is limited by the memory size
- **Performance Bottleneck**
    - The throughput of the entire distributed file system is limited by the throughput of the single NameNode
- **Isolation**
    - Because there is only one NameNode and one namespace in the cluster, different applications can't be isolated
- **Cluster availability**
    - Once the only NameNode is faulty, the entire cluster becomes unavailable

HDFS Data Read Process

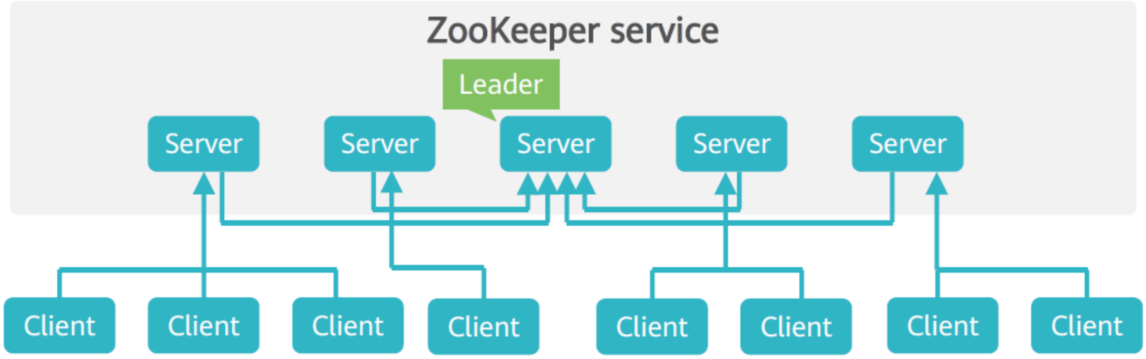


HDFS Data Write Process



Zookeeper Overview

- The zookeeper distributed service framework is used to solve some data management problems.
- Zookeeper is widely used and depended upon by upper layer components, such as Kafka, HDFS, HBase, and Storm
- It provides functions such as configuration management, naming service, distributed lock, and cluster management
- Zookeeper cluster consists of a group of servers. In this group there is only one leader node, with the other nodes
    - the leader is elected during startup
    - zookeeper uses the custom atomic message protocol to ensure data consistency among nodes in the entire system



Key Features of Zookeeper

- **eventual consistency**
  - all servers are displayed in the same view
- **real-time**
  - clients can obtain server updates and failures within a specified period of time
- **reliability**
  - a massage will be received by all servers
- **wait-free**
  - slow or faulty clients can’t intervene in the requests of rapid clients so that the requests of each client can’t be processed effectively
- **atomicity**
  - data transfer either succeeds or fails, but no transaction partial
- **sequence consistency**
  - updates sent by the client are applied in the sequence in which they are sent