Chapter 2 HDFS and ZooKeeper

Foreword

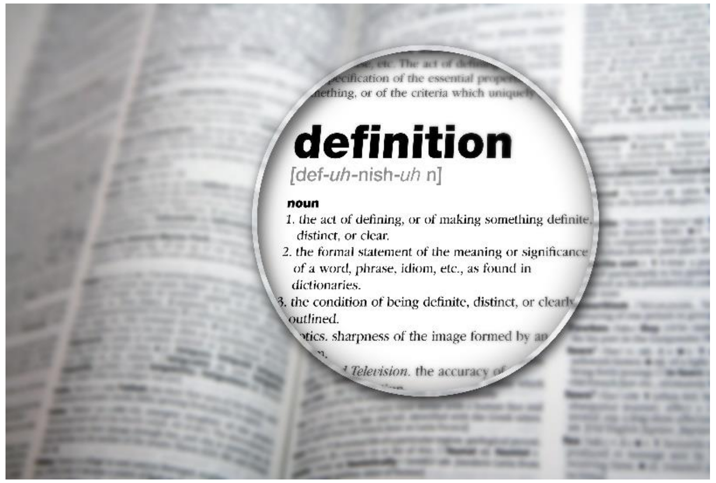
- This course describes the big data distributed storage system HDFS and the ZooKeeper distributed service framework that resolves some frequently-encountered data management problems in distributed services. This chapter lays a solid foundation for subsequent component learning.

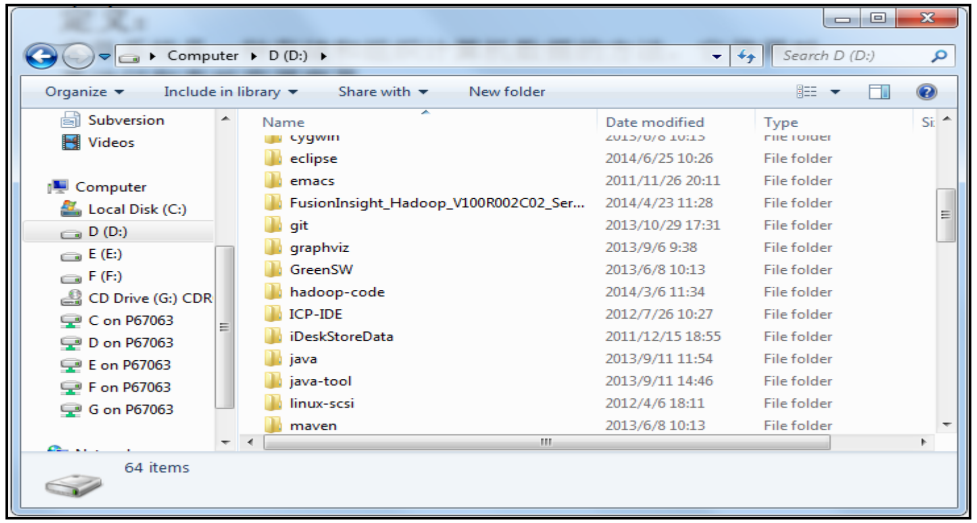
Objectives

- Upon completion of this course, you will be able to learn:
  - HDFS-related concepts
  - HDFS application scenarios
  - HDFS system architecture
  - HDFS Key features
  - ZooKeeper-related concepts
  - ZooKeeper usage

Contents

Dictionary and File System


definition
[def-uh-nish-uh n]
noun
1. the act of defining, or of making something definite, distinct, or clear.
2. the formal statement of the meaning or significance of a word, phrase, idiom, etc., as found in dictionaries.
3. the condition of being definite, distinct, or clearly outlined.



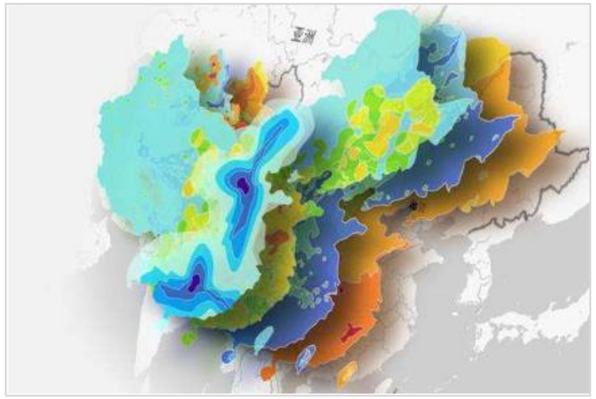
Name	Date modified	Type
cygwin	2013/6/8 10:13	File folder
eclipse	2014/6/25 10:26	File folder
emacs	2011/11/26 20:11	File folder
FusionInsight_Hadoop_V100R002C02_Ser...	2014/4/23 11:28	File folder
git	2013/10/29 17:31	File folder
graphviz	2013/9/6 9:38	File folder
GreenSW	2013/6/8 10:13	File folder
hadoop-code	2014/3/6 11:34	File folder
ICP-IDE	2012/7/26 10:27	File folder
iDeskStoreData	2011/12/15 18:55	File folder
java	2013/9/11 11:54	File folder
java-tool	2013/9/11 14:46	File folder
linux-scsi	2012/4/6 18:11	File folder
maven	2013/6/8 10:13	File folder

Dictionary	File System
Radical Index (1) Radical directory (2) Word index (3) Stroke index for Rare characters	File name Metadata
Dictionary body	Data block

HDFS Overview

- Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware.
- HDFS has a high fault tolerance capability and is deployed on cost-effective hardware.
- HDFS provides high-throughput access to application data and applies to applications with large data sets.
- HDFS looses some Potable Operating System Interface of UNIX (POSIX) requirements to implement streaming access to file system data.
- HDFS was originally built as the foundation for the Apache Nutch Web search engine project.
- HDFS is a part of the Apache Hadoop Core project.

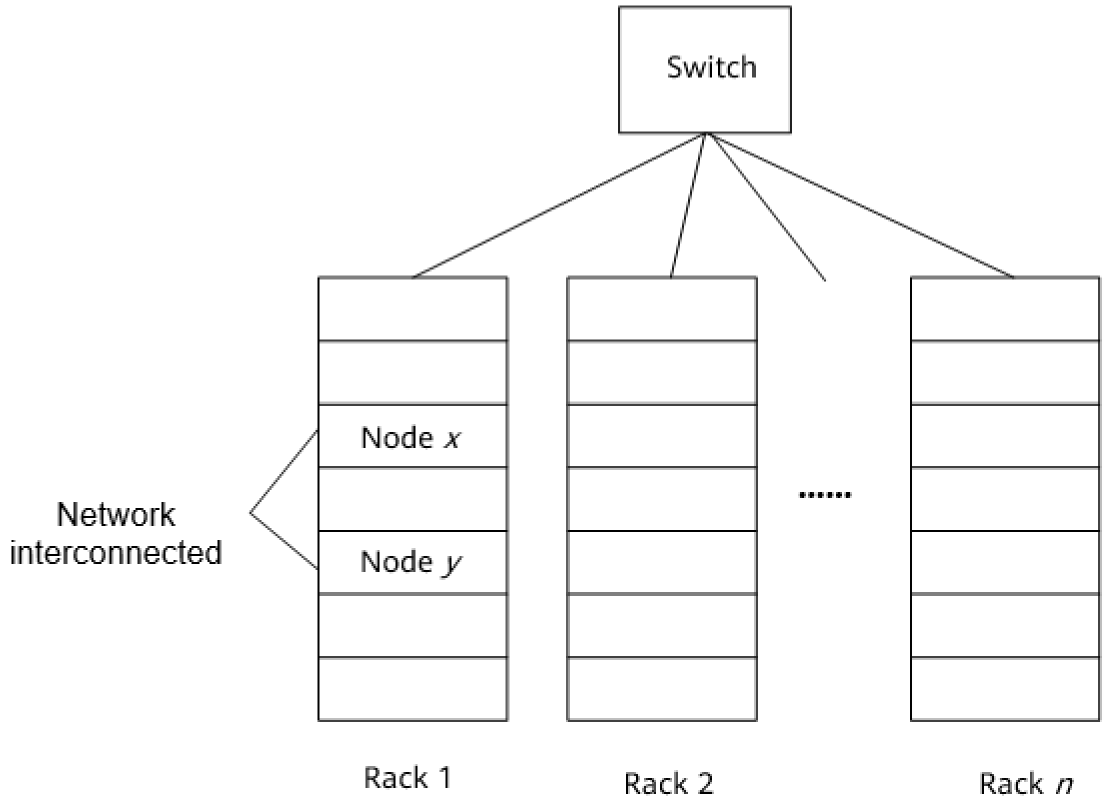
HDFS Application Scenario Example



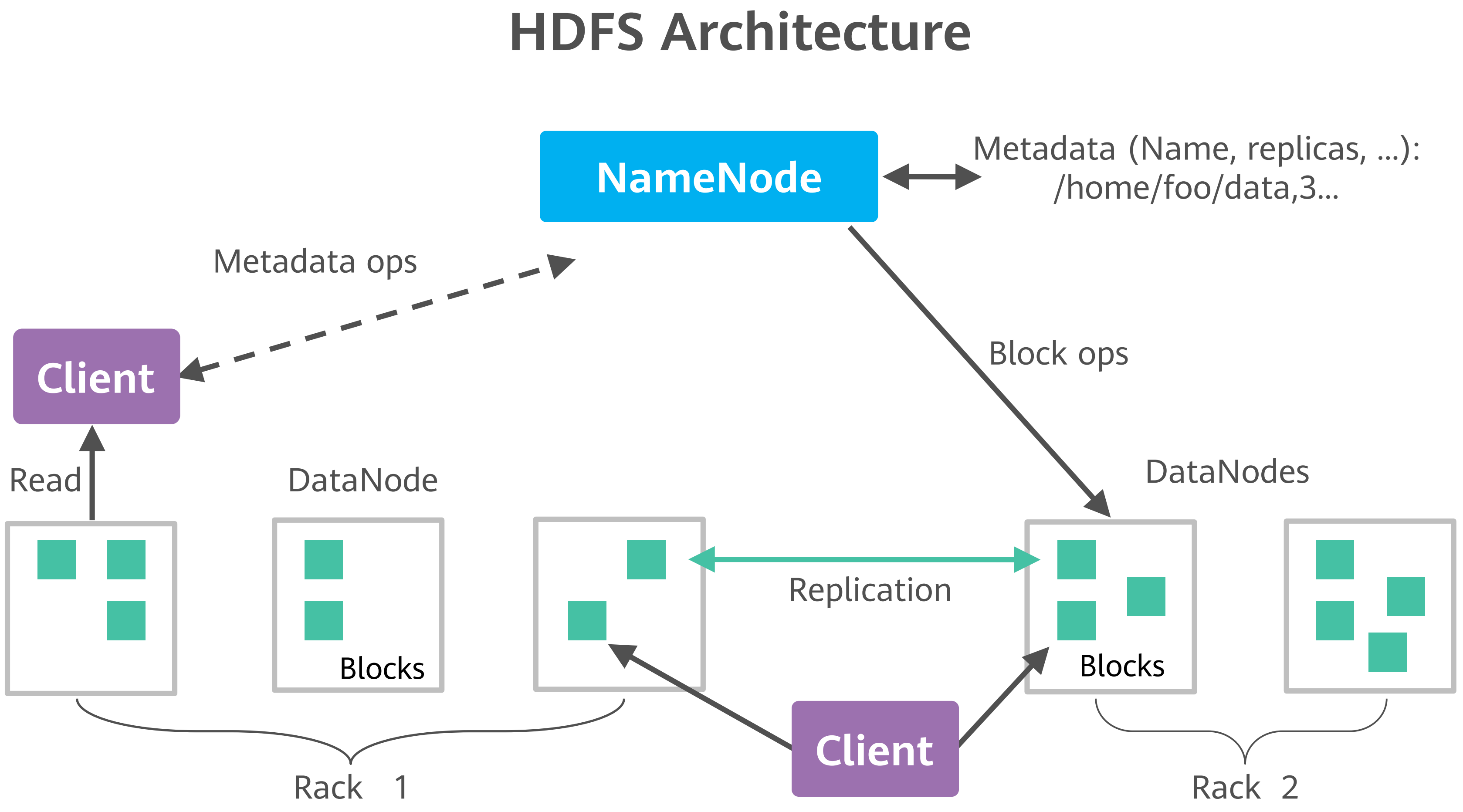
Contents

Computer Cluster Structure

- The distributed file system stores files on multiple computer nodes. Thousands of computer nodes form a computer cluster.
- Currently, the computer cluster used by the distributed file system consists of common hardware, which greatly reduces the hardware overhead.



Basic System Architecture



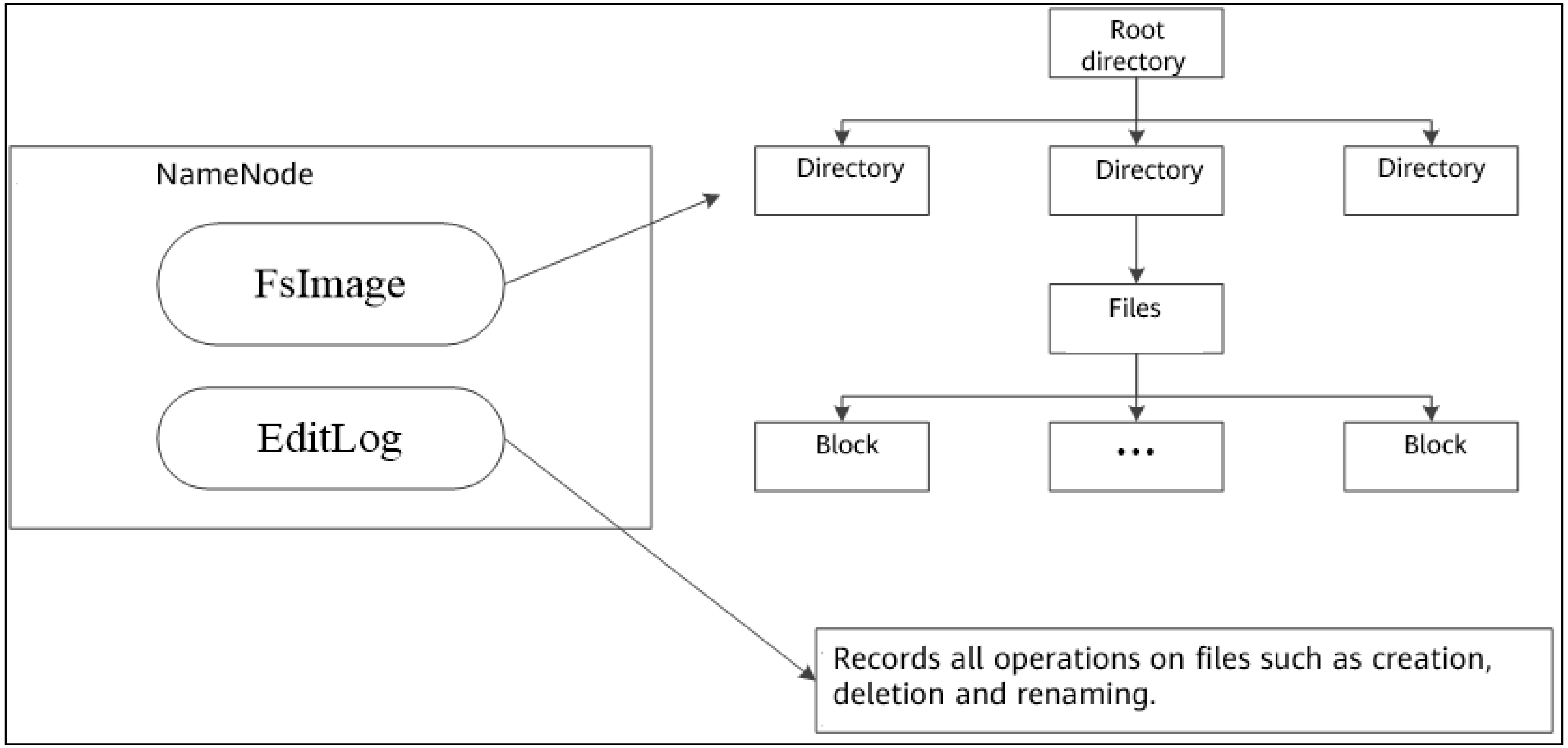
Block

- The default size of an HDFS block is 128 MB. A file is divided into multiple blocks, which are used as the storage unit.
- The block size is much larger than that of a common file system, minimizing the addressing overhead.
- The abstract block concept brings the following obvious benefits:
  - Supporting large-scale file storage
  - Simplifying system design
  - Applicable to data backup

NameNode and DataNode (1)

NameNode	DataNode
Stores metadata.	Stores file content.
Metadata is stored in the memory.	The file content is stored in the disk.
Saves the mapping between files, blocks, and DataNodes.	Maintains the mapping between block IDs and local files on DataNodes.

NameNode and DataNode (2)

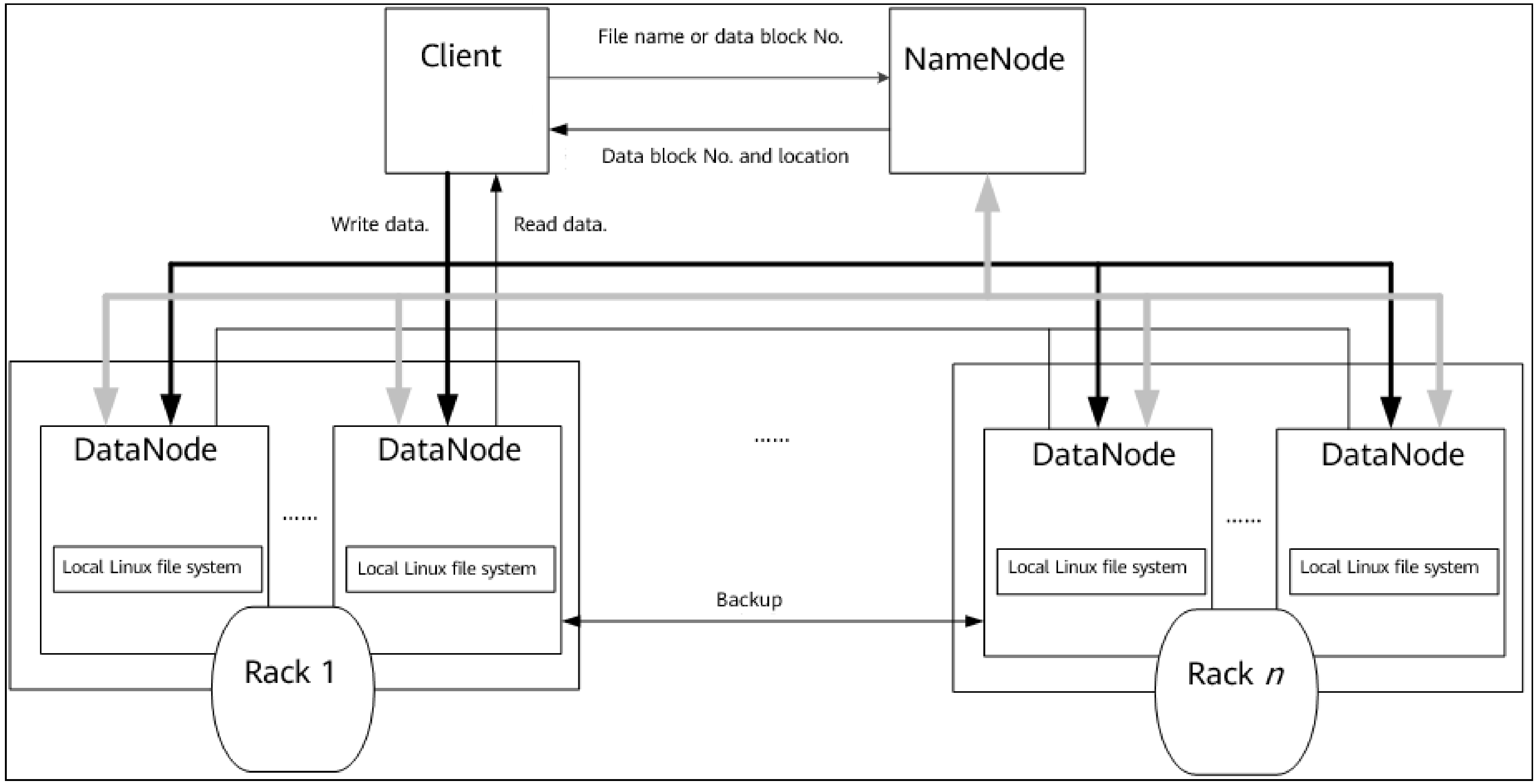


DataNodes

- DataNodes are working nodes that store and read data in HDFS. DataNodes store and retrieve data based on the scheduling of the clients or NameNodes, and periodically send the list of stored blocks to the NameNodes.
- Data on each DataNode is stored in the local Linux file system of the node.

Contents

HDFS Architecture Overview



HDFS Namespace Management

- The HDFS namespace contains directories, files, and blocks.
- HDFS uses the traditional hierarchical file system. Therefore, users can create and delete directories and files, move files between directories, and rename files in the same way as using a common file system.
- NameNode maintains the file system namespace. Any changes to the file system namespace or its properties are recorded by the NameNode.

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 the TCP/IP protocol.
  - The client initiates a TCP connection to the NameNode through a configurable port and uses the client protocol to interact with the NameNode.
  - The NameNode and the DataNode interact with each other by using the DataNode protocol.
  - The interaction between the client and the DataNode is implemented through the Remote Procedure Call (RPC). In design, the NameNode does not initiate an RPC request, but responds to RPC requests from the client and DataNode.

Client

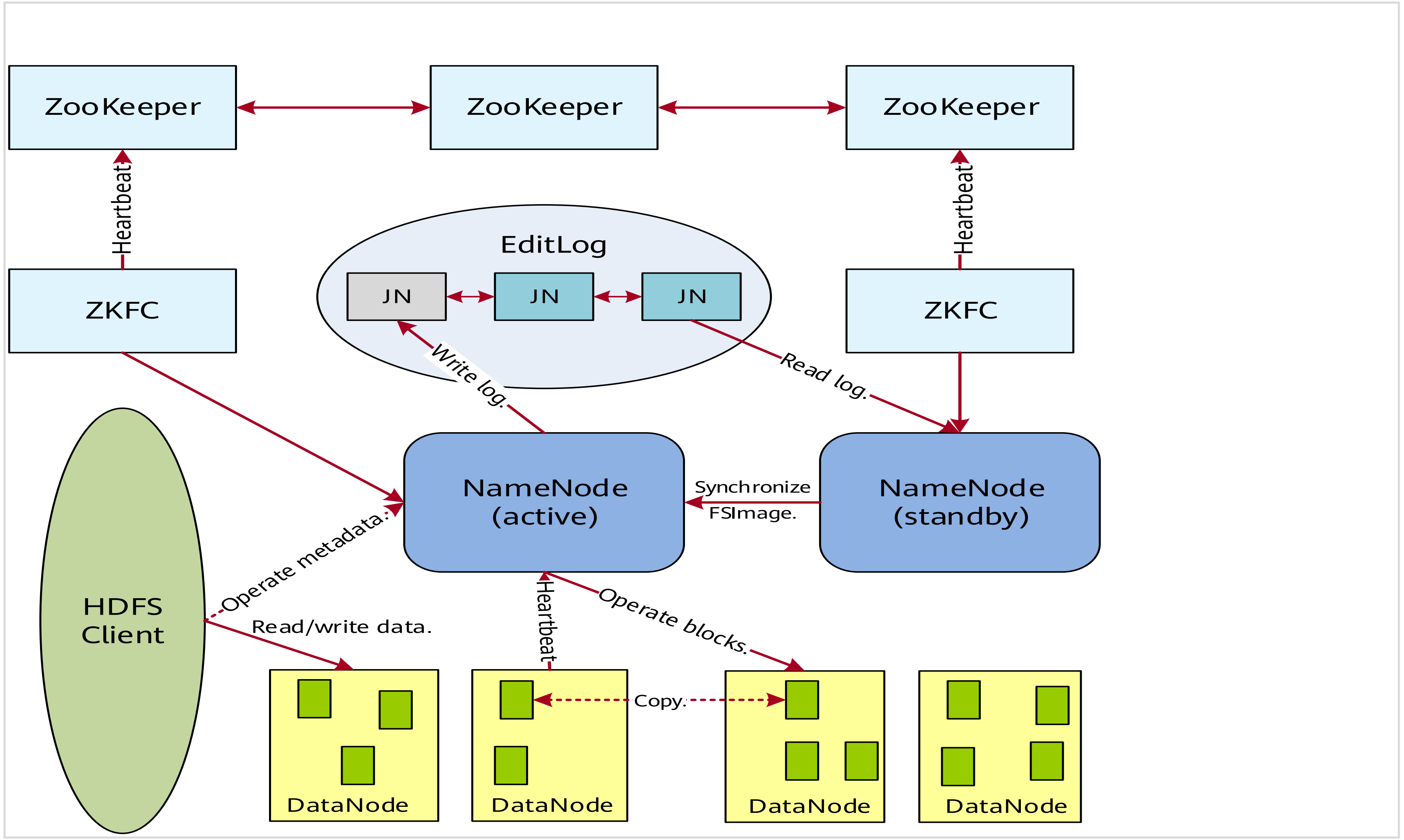
- The client is the most commonly used method for users to operate HDFS. HDFS provides a client during deployment.
- The HDFS client is a library that contains HDFS file system interfaces that hide most of the complexity of HDFS implementation.
- Strictly speaking, the client is not a part of HDFS.
- The client supports common operations such as opening, reading, and writing, and provides a command line mode similar to Shell to access data in HDFS.
- HDFS also provides Java APIs as client programming interfaces for applications to access the file system.

Disadvantages of the HDFS Single-NameNode Architecture

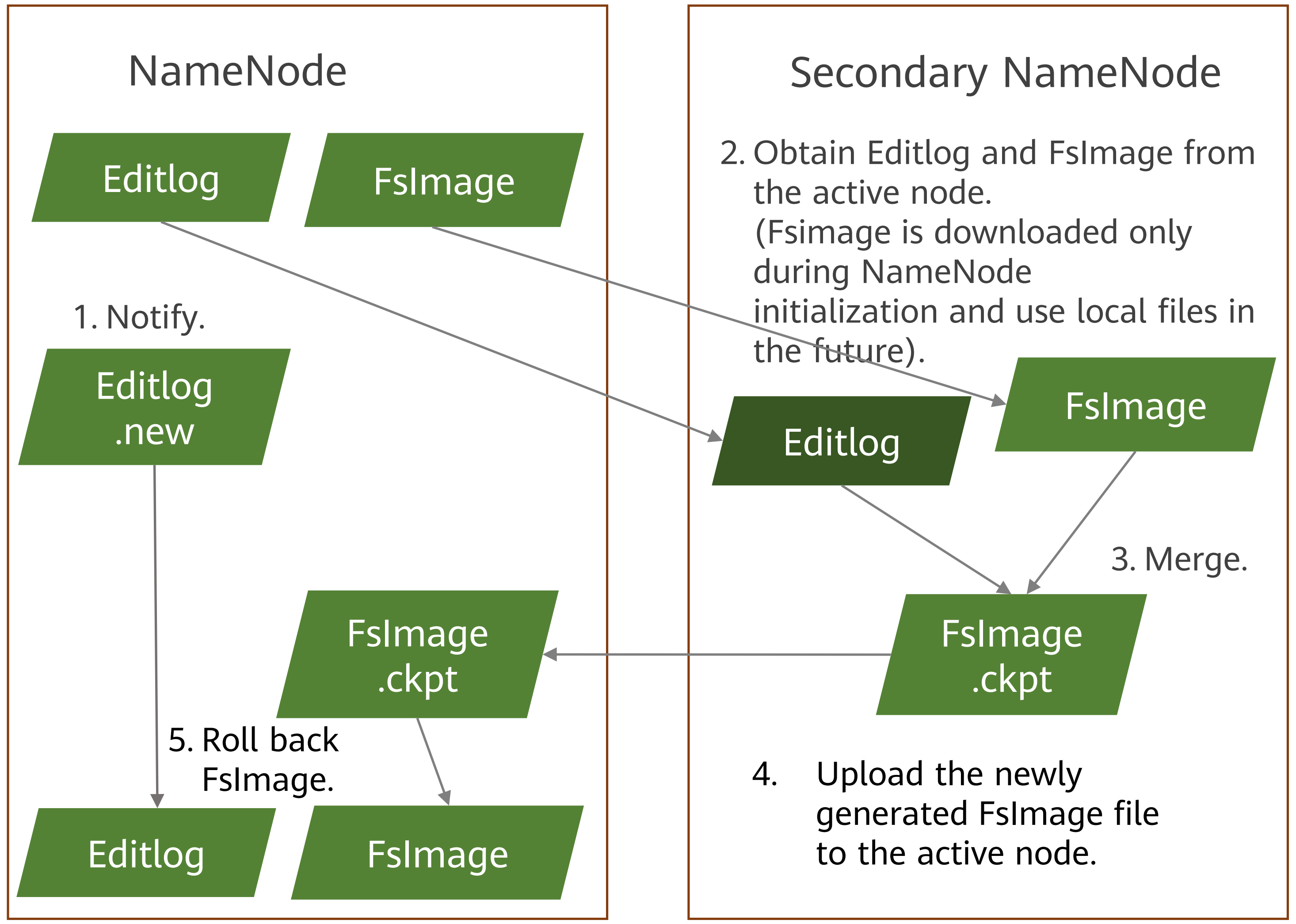
- Only one NameNode is set for HDFS, which greatly simplifies the system design but also brings some obvious limitations. The details are as follows:
  - **Namespace limitation**: NameNodes are stored in the memory. Therefore, the number of objects (files and blocks) that can be contained in a NameNode is limited by the memory size.
  - **Performance bottleneck**: The throughput of the entire distributed file system is limited by the throughput of a single NameNode.
  - **Isolation**: Because there is only one NameNode and one namespace in the cluster, different applications cannot be isolated.
  - **Cluster availability**: Once the only NameNode is faulty, the entire cluster becomes unavailable.

Contents

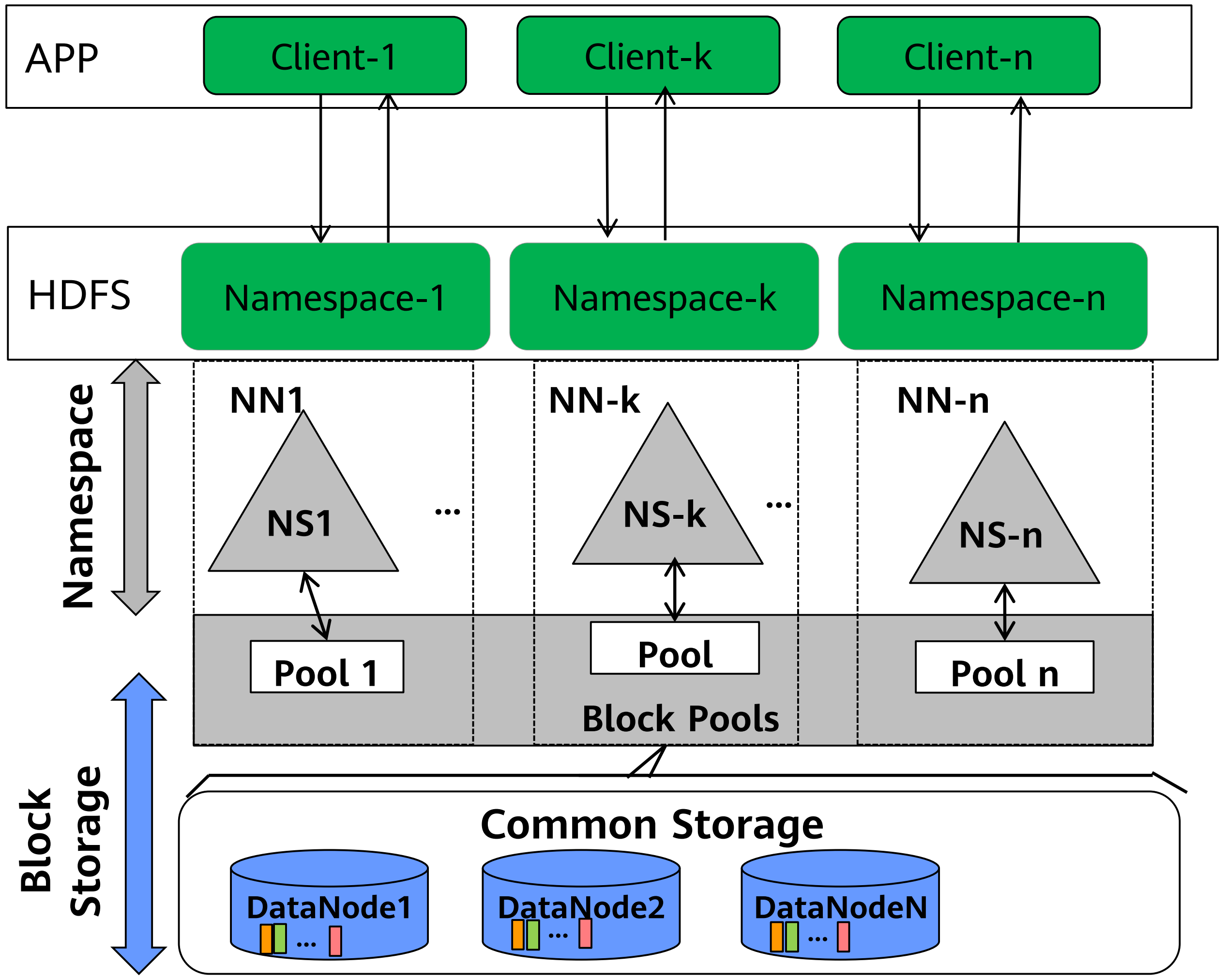
HDFS High Availability (HA)



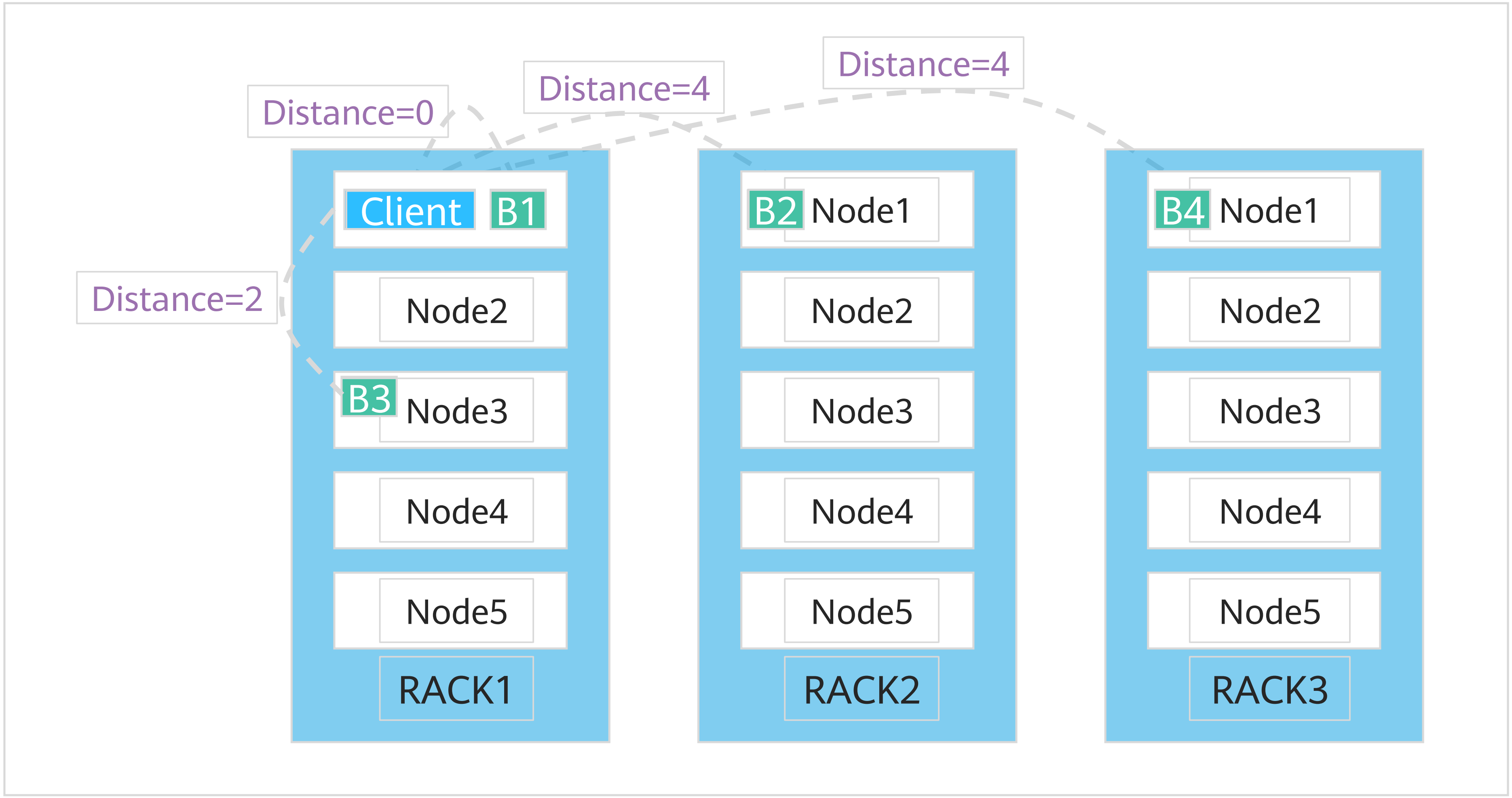
Metadata Persistence


NameNode
Editlog
FsImage
1. Notify.
Editlog .new
FsImage .ckpt
5. Roll back FsImage.
Editlog
FsImage
Secondary NameNode
2. Obtain Editlog and FsImage from the active node. (Fsimage is downloaded only during NameNode initialization and use local files in the future).
FsImage
Editlog
3. Merge.
FsImage .ckpt
4. Upload the newly generated FsImage file to the active node.

HDFS Federation



Data Replica Mechanism



HDFS Data Integrity Assurance

- HDFS aims to ensure the integrity of storage data and ensures the reliability of components.
- Rebuilding the replica data of failed data disks
  - When the DataNode fails to report data to the NameNode periodically, the NameNode initiates the replica rebuilding action to restore the lost replicas.
- Cluster data balancing:
  - The HDFS architecture designs the data balancing mechanism, which ensures that data is evenly distributed on each DataNode.
- Metadata reliability:
  - The log mechanism is used to operate metadata, and metadata is stored on the active and standby NameNodes.
  - The snapshot mechanism implements the common snapshot mechanism of file systems, ensuring that data can be restored in a timely manner in the case of mis-operations.
- Security mode:
  - HDFS provides a unique security mode mechanism to prevent faults from spreading when DataNodes or disks are faulty.

Other Key Design Points of the HDFS Architecture

- Space reclamation mechanism:
  - Supports the recycle bin mechanism and dynamic setting of the number of copies.
- Data organization:
  - Data is stored by data block in the HDFS of the operating system.
- Access mode:
  - Provides HDFS data accessing through Java APIs, HTTP or SHELL modes.

Common Shell Commands

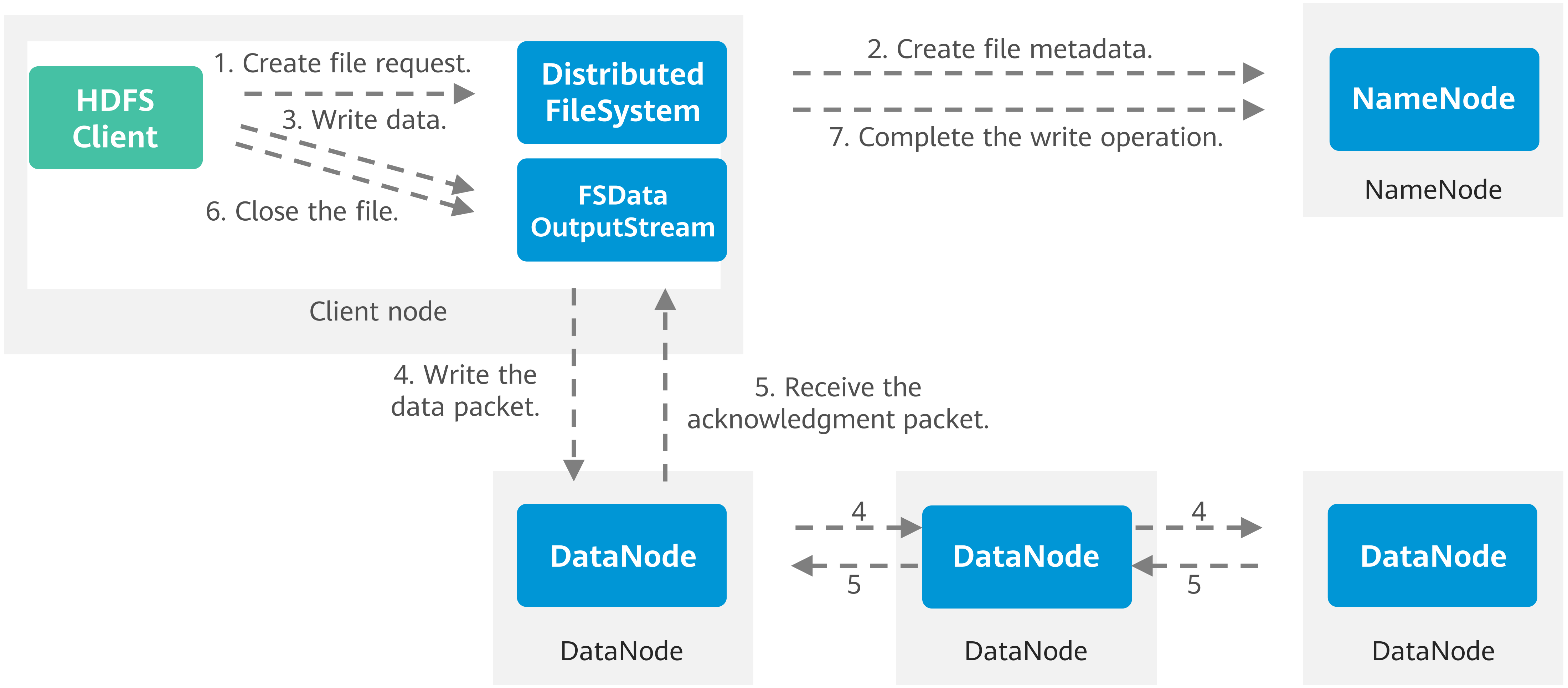
Type	Command	Description
dfs	-cat	Displays file content.
	-ls	Displays the directory list.
	-rm	Delete a file.
	-put	Uploads the directory or file to HDFS.
	-get	Downloads directories or files to the local host from HDFS.
	-mkdir	Creates a directory.
	-chmod/-chown	Changes the group of the file.
	...	...
dfsadmin	-safemode	Performs security mode operations.
	-report	Reports service status.

New Features of HDFS 3.0

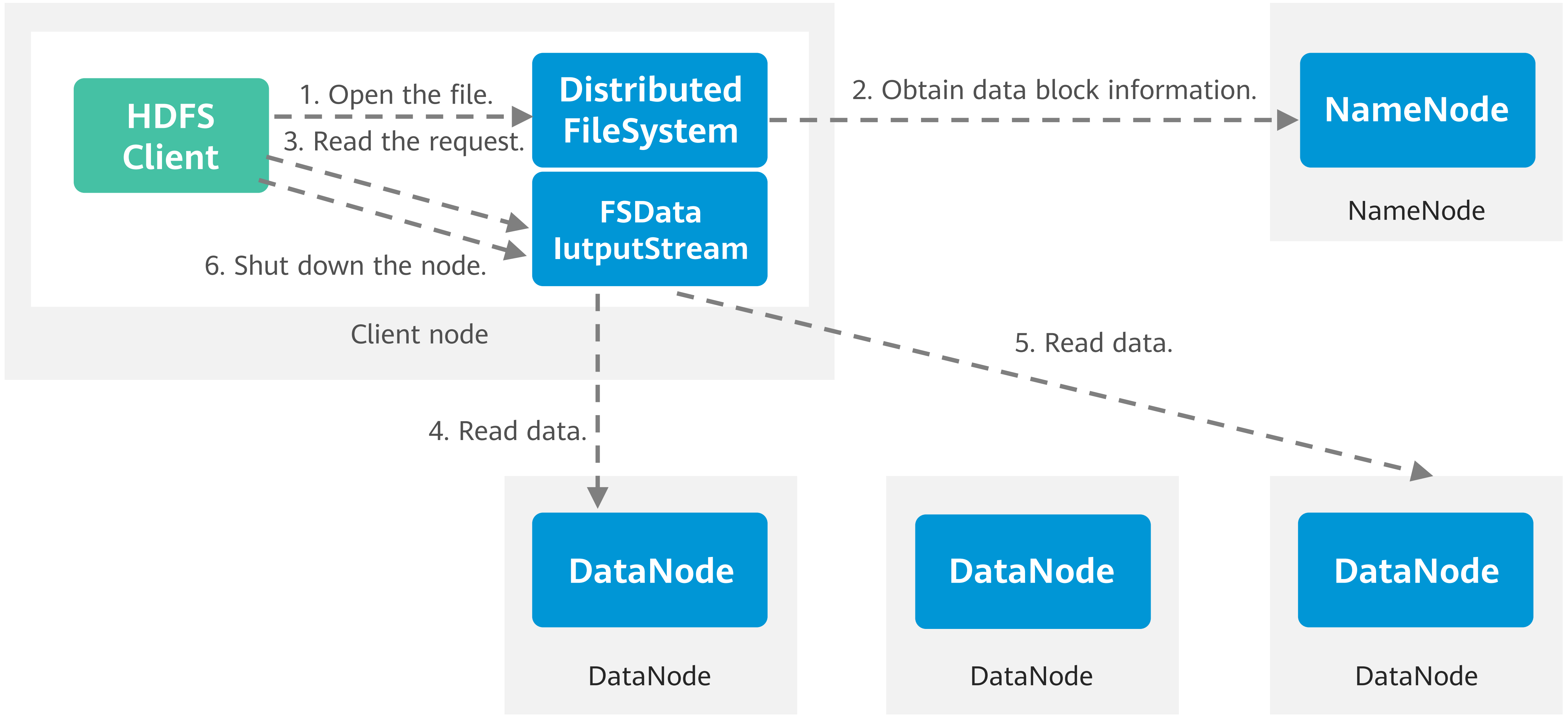
- Erasure Code (EC) in HDFS is supported.
- Union based on the HDFS router is supported.
- Multiple NameNodes are supported.
- Disk balancers are added to DataNodes for load balancing.

Contents

HDFS Data Write Process



HDFS Data Read Process



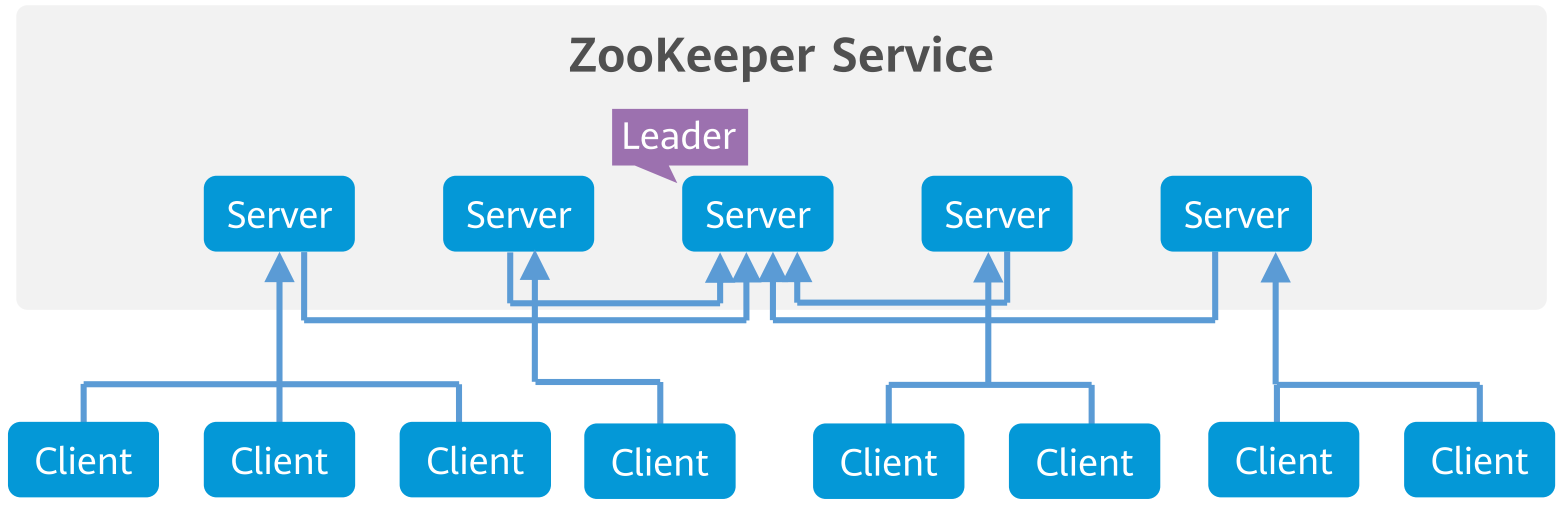
Contents

ZooKeeper Overview

- The ZooKeeper distributed service framework is used to solve some data management problems that are frequently encountered in distributed applications and provide distributed and highly available coordination service capabilities.
- In security mode, ZooKeeper depends on Kerberos and LdapServer for security authentication, but in non-security mode, ZooKeeper does not depend on them any more. As a bottom-layer component, ZooKeeper is widely used and depended by upper-layer components, such as Kafka, HDFS, HBase and Storm.

Contents

ZooKeeper Service Architecture - Model



- The ZooKeeper cluster consists of a group of server nodes. In this group, there is only one leader node, and other nodes are followers.
- The leader is elected during the startup.
- ZooKeeper uses the custom atomic message protocol to ensure data consistency among nodes in the entire system.
- After receiving a data change request, the leader node writes the data to the disk and then to the memory.

ZooKeeper Service Architecture - DR Capability

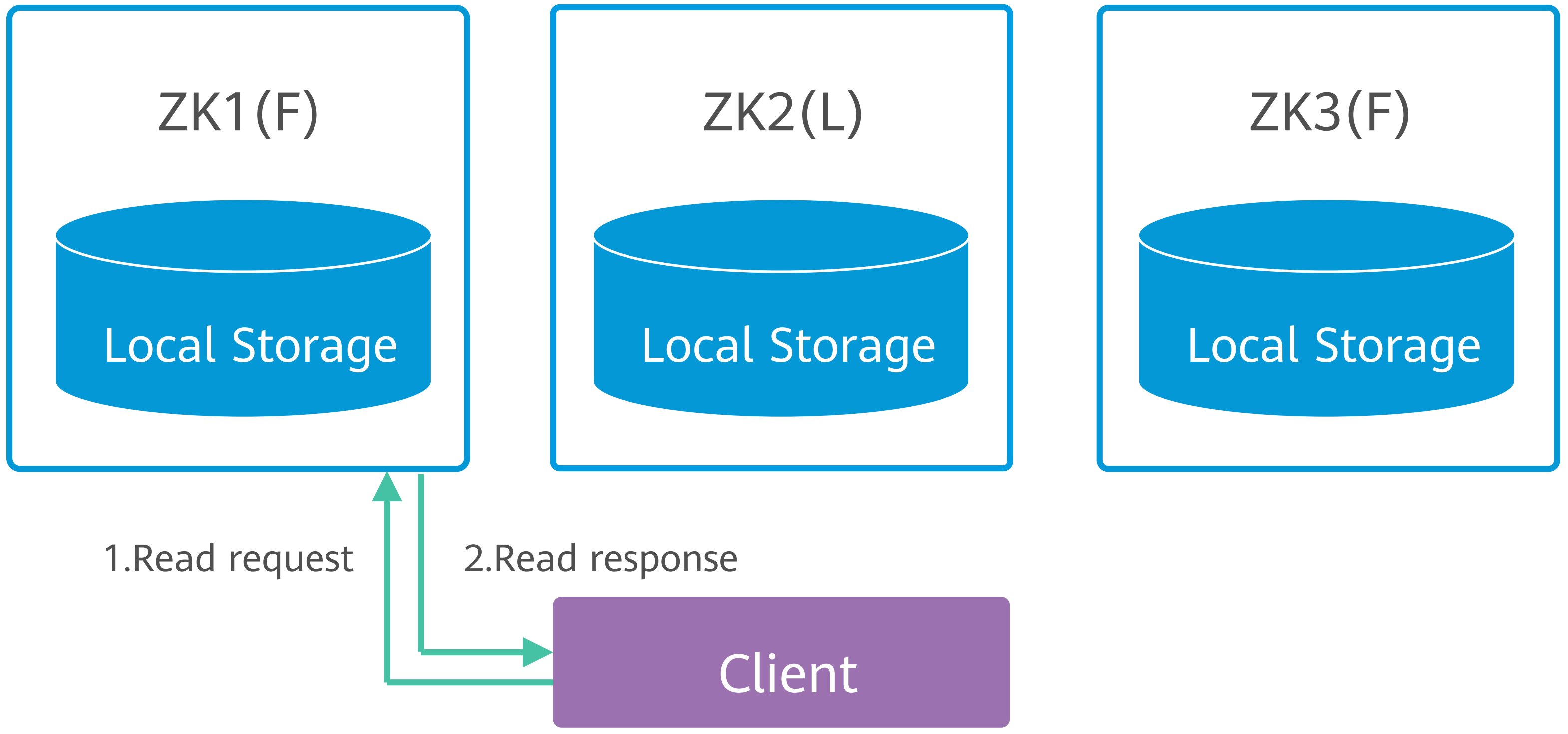
- If the ZooKeeper can complete the election, it can provide services for external systems.
  - During ZooKeeper election, if an instance obtains more than half of the votes, the instance becomes the leader.
- For a service with n instances, n may be an odd or even number.
  - When n is an odd number, assume: $n = 2x + 1$; then the node needs to obtain x+1 votes to become the leader, and the DR capability is x.
  - When n is an even number, that: $n = 2x + 2$; then the node needs to obtain x+2 (more than half) votes to become the leader, and the DR capability is x.

Key Features of ZooKeeper

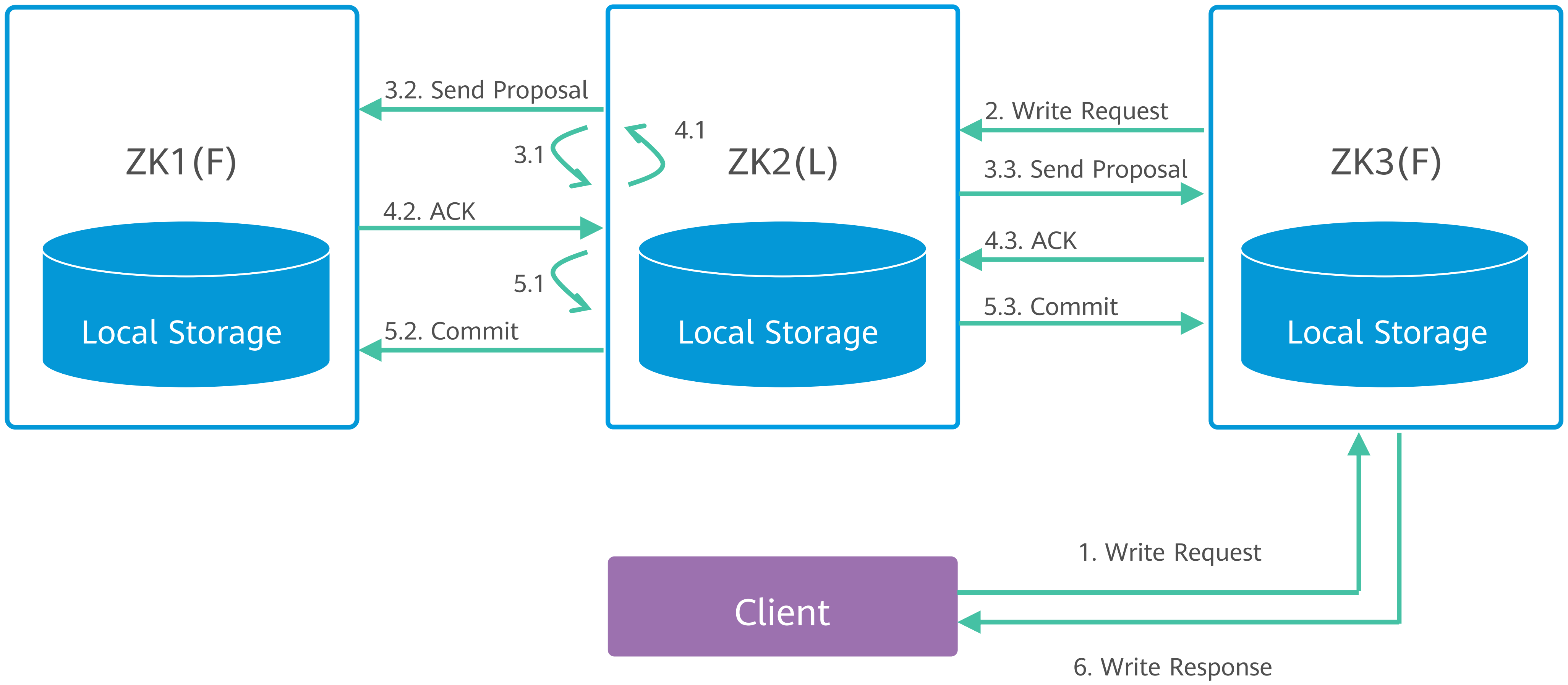
- Eventual consistency: All servers are displayed in the same view.
- Real-time capability: Clients can obtain server updates and failures within a specified period of time.
- Reliability: A message will be received by all servers.
- Wait-free: Slow or faulty clients cannot intervene the requests of rapid clients so that the requests of each client can be processed effectively.
- Atomicity: Data update either succeeds or fails. There are no intermediate states.
- Sequence consistency: Updates sent by the client are applied in the sequence in which they are sent.

Read Function of ZooKeeper

- According to the consistency of ZooKeeper, the client obtains the same view regardless of the server connected to the client. Therefore, read operations can be performed between the client and any node.



Write Function of ZooKeeper



Commands for ZooKeeper Clients

- To invoke a ZooKeeper client, run the following command:

```
zkCli.sh –server 172.16.0.1:24002
```

- To create a node: **create /node**
- To list subnodes: **ls /node**
- To create node data: **set /node data**
- To obtain node data: **get /node**
- To delete a node: **delete /node**
- To delete a node and all its subnodes: **deleteall /node**

Summary

- The distributed file system is an effective solution for large-scale data storage in the big data era. The open-source HDFS implements GFS and distributed storage of massive data by using a computer cluster formed by inexpensive hardware.
- HDFS is compatible with inexpensive hardware devices, stream data read and write, large data sets, simple file models, and powerful cross-platform compatibility. However, HDFS has its own limitations. For example, it is not suitable for low-latency data access, cannot efficiently store a large number of small files, and does not support multi-user write and arbitrary file modification.
- "Block" is the core concept of HDFS. A large file is split into multiple blocks. HDFS adopts the abstract block concept, supports large-scale file storage, simplifies system design, and is suitable for data backup.
- The ZooKeeper distributed service framework is used to solve some data management problems that are frequently encountered in distributed applications and provide distributed and highly available coordination service capabilities.

Quiz

1. Why is it recommended that the number of ZooKeepers be deployed in an odd number?
2. Why is the HDFS data block size larger than the disk block size?
3. Can HDFS data be read when it is written?

Recommendations

- Huawei Cloud Official Web Link:
  - https://www.huaweicloud.com/intl/en-us/
- Huawei MRS Documentation:
  - https://www.huaweicloud.com/intl/en-us/product/mrs.html
- Huawei TALENT ONLINE:
  - https://e.huawei.com/en/talent/#/

Thank you.

Bring digital to every person, home, and organization for a fully connected, intelligent world.

HUAWEI