Hadoop Distributed File System

By
Mr.D.B.Shanmugam
Associate Professor & HOD

Overview

- Distributed File System
- History of HDFS
- What is HDFS
- HDFS Architecture
- File commands
- Demonstration

Distributed File System

- Hold a large amount of data
- Clients distributed across a network
- Network File System(NFS)
  - Straightforward design
  - remote access- single machine
  - Constraints

History

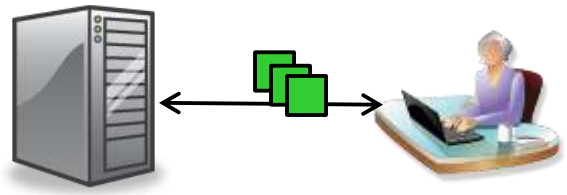
History

- Apache Nutch – open source web engine-2002
- Scaling issue
- Publication of GFS paper in 2003-addressed Nutch's scaling issues
- 2004 – Nutch distributed File System
- 2006 – Apache Hadoop – MapReduce and HDFS

HDFS

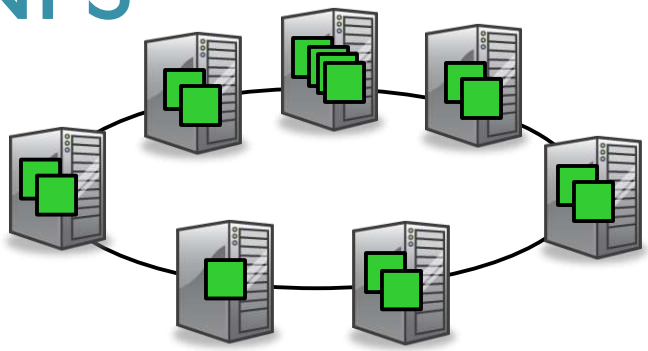
- Terabytes or Petabytes of data
- Larger files than NFS
- Reliable
- Fast, Scalable access
- Integrate well with Map Reduce
- Restricted to a class of applications

HDFS versus NFS



Network File System (NFS)

- Single machine makes part of its file system available to other machines
- Sequential or random access
- PRO: Simplicity, generality, transparency
- CON: Storage capacity and throughput limited by single server



Hadoop Distributed File System (HDFS)

- Single virtual file system spread over many machines
- Optimized for sequential read and local accesses
- PRO: High throughput, high capacity
- "CON": Specialized for particular types of applications

HDFS

Basics

- Distributed File System of Hadoop
- Runs on commodity hardware
- Stream data at high bandwidth
- Challenge –tolerate node failure without data loss
- Simple Coherency model
- Computation is near the data
- Portability – built using Java

Basics

- Interface patterned after UNIX file system
- File system metadata and application data stored separately
- Metadata is on dedicated server called Namenode
- Application data on data nodes

Basics

HDFS is good for

- Very large files
- Streaming data access
- Commodity hardware

Basics

HDFS is not good for

- Low-latency data access
- Lots of small files
- Multiple writers, arbitrary file modifications

Differences from GFS

- Only Single writer per file
- Open Source

HDFS Architecture

HDFS Concepts

- Namespace
- Blocks
- Namenodes and Datanodes
- Secondary Namenode

HDFS Namespace

- Hierarchy of files and directories
- In RAM
- Represented on Namenode by inodes
- Attributes- permissions, modification and access times, namespace and disk space quotas

Blocks

- HDFS blocks are either 64MB or 128MB
- Large blocks-minimize the cost of seeks
- Benefits-can take advantage of any disks in the cluster
- Simplifies the storage subsystem-amount of metadata storage per file is reduced
- Fit well with replication

Namenodes and Datanodes

- Master-worker pattern
- Single Namenode-master server
- Number of Datanodes-usually one per node in the cluster

Namenode

- Master
- Manages filesystem namespace
- Maintains filesystem tree and metadata-persistently on two files-namespace image and editlog
- Stores locations of blocks-but not persistently
- Metadata – inode data and the list of blocks of each file

Datanodes

- Workhorses of the filesystem
- Store and retrieve blocks
- Send blockreports to Namenode
- Do not use data protection mechanisms like RAID…use replication

Datanodes

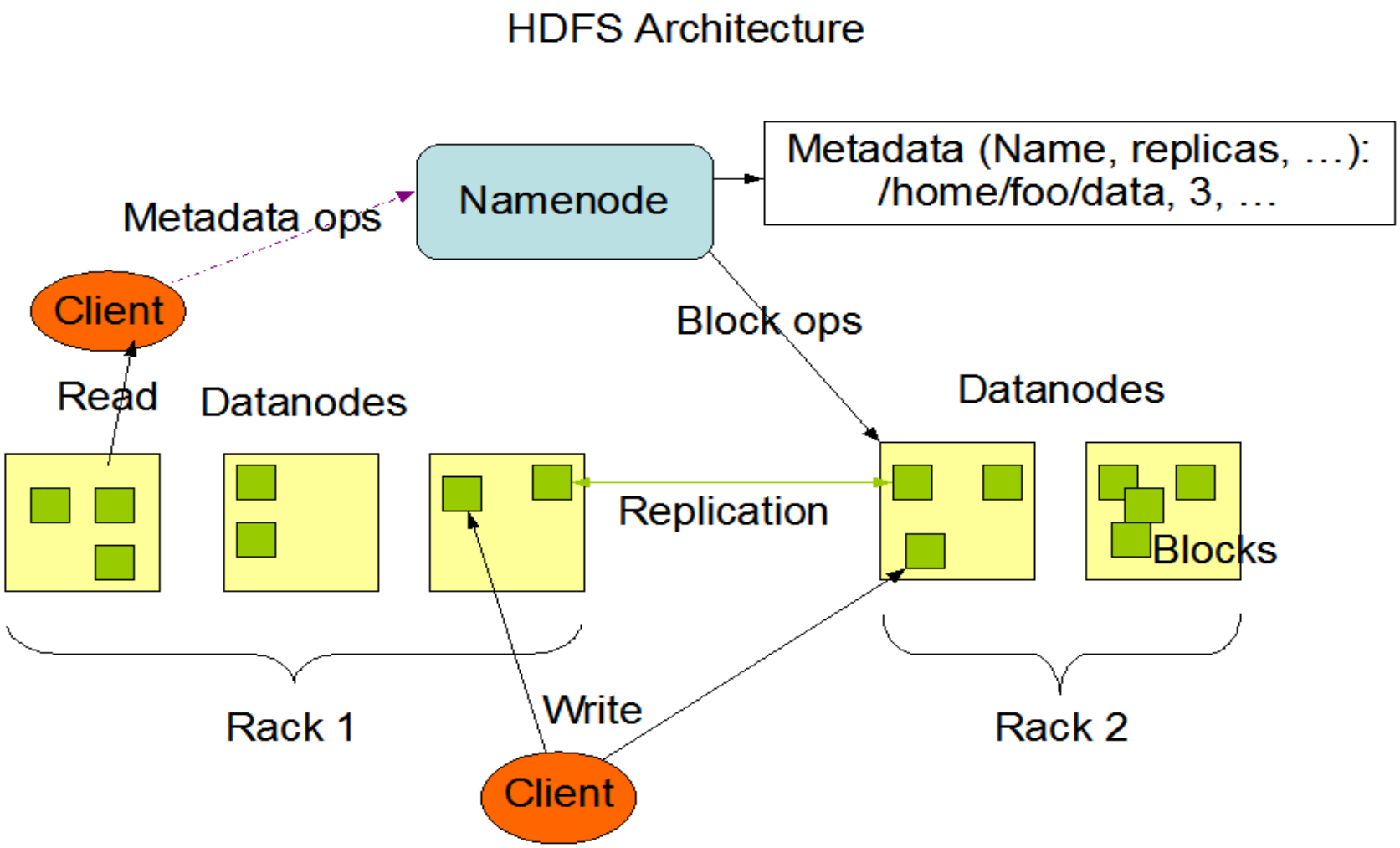
- Two files-one for data, other for block's metadata including checksums and generation stamp
- Size of data file equals actual length of block

DataNodes

- Startup-handshake:
  - Namespace ID
  - Software version

Datanodes

- After handshake:
  - Registration
  - Storage ID
  - Block Report
  - Heartbeats


HDFS Architecture
Namenode
Metadata (Name, replicas, …):
/home/foo/data, 3, …
Metadata ops
Client
Block ops
Read
Datanodes
Datanodes
Replication
Blocks
Rack 1
Write
Rack 2
Client

Secondary Namenode

- If namenode fails, the filesystem cannot be used
- Two ways to make it resilient to failure:
  - Backup of files
  - Secondary Namenode

Secondary Namenode

- Periodically merge namespace image with editlog
- Runs on separate physical machine
- Has a copy of metadata, which can be used to reconstruct state of the namenode
- Disadvantage: state lags that of the primary namenode
- Renamed as CheckpointNode (CN) in 0.21 release[1]
- Periodic and is not continuous
- If the NameNode dies, it does not take over the responsibilities of the NN

HDFS Client

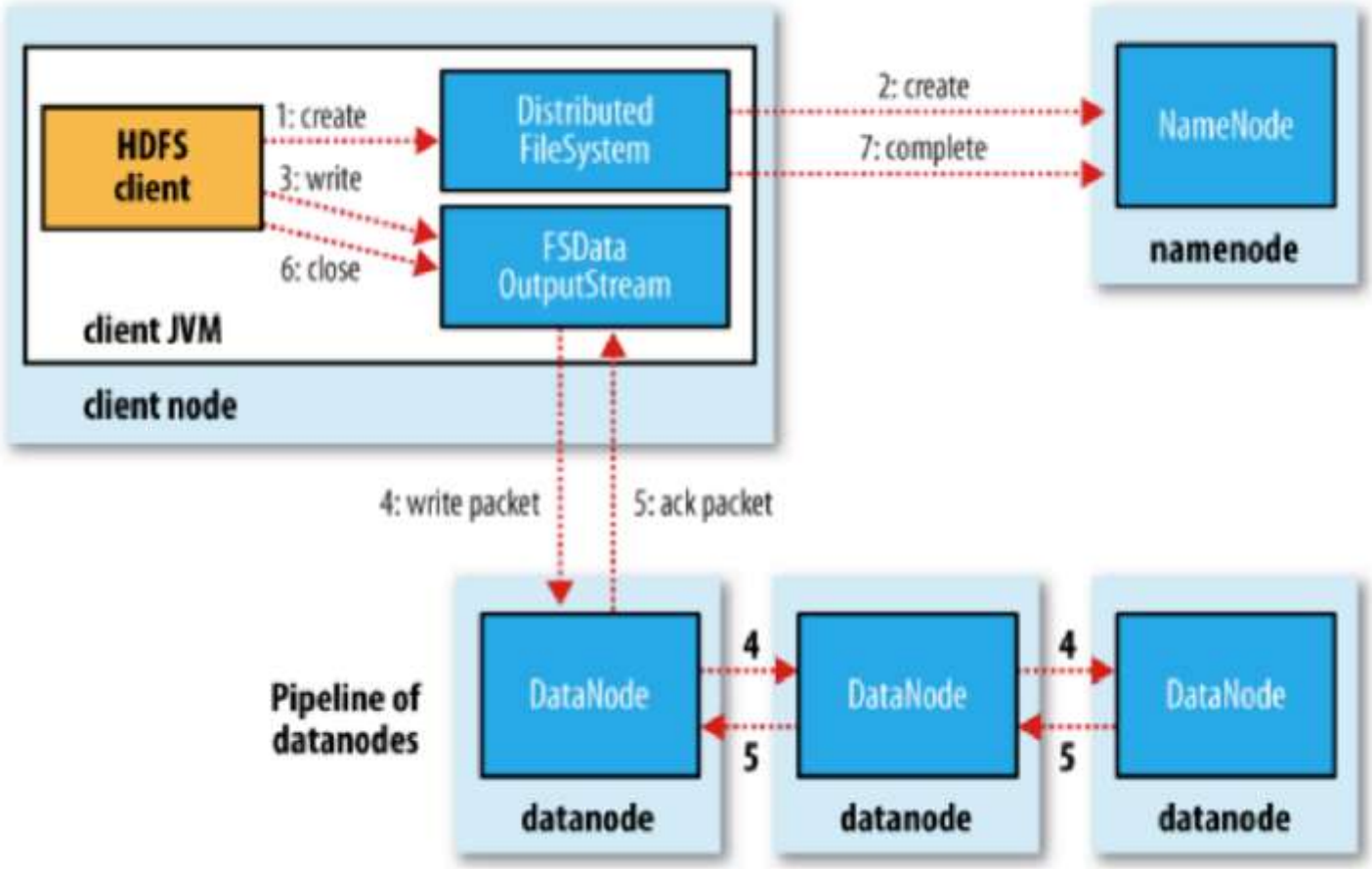
- Code library that exports the HDFS file system interface
- Allows user applications to access the file system

File I/O Operations

Write Operation

- Once written, cannot be altered, only append
- HDFS Client-lease for the file
- Renewal of lease
- Lease – soft limit, hard limit
- Single-writer multiple-reader model

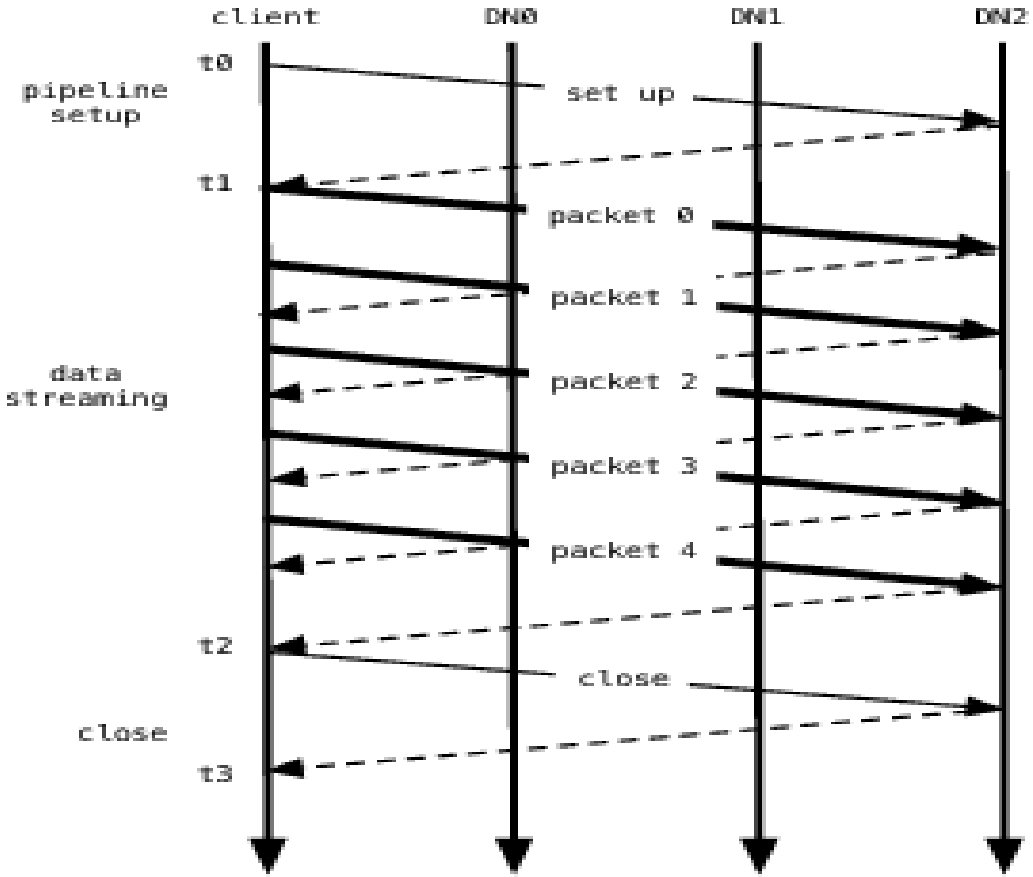
HDFS Write

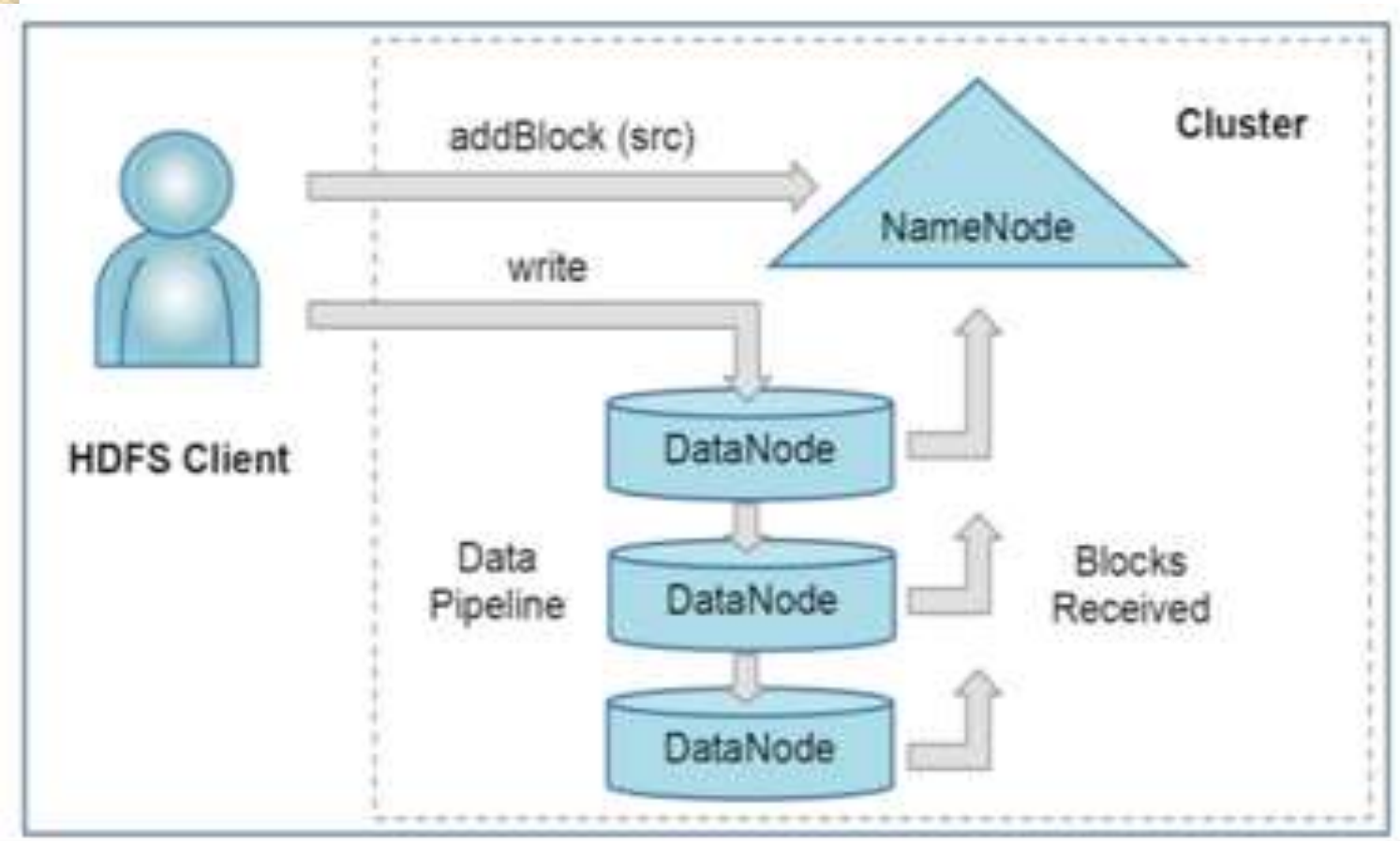

HDFS client
1: create
3: write
6: close
Distributed FileSystem
FSData OutputStream
client JVM
client node
2: create
7: complete
NameNode
namenode
4: write packet
5: ack packet
Pipeline of datanodes
DataNode
DataNode
DataNode
4
5
4
5
datanode
datanode
datanode

Write Operation

- Block allocation
- Hflush operation
- Renewal of lease
- Lease – soft limit, hard limit
- Single-writer multiple-reader model

Data pipeline during block construction



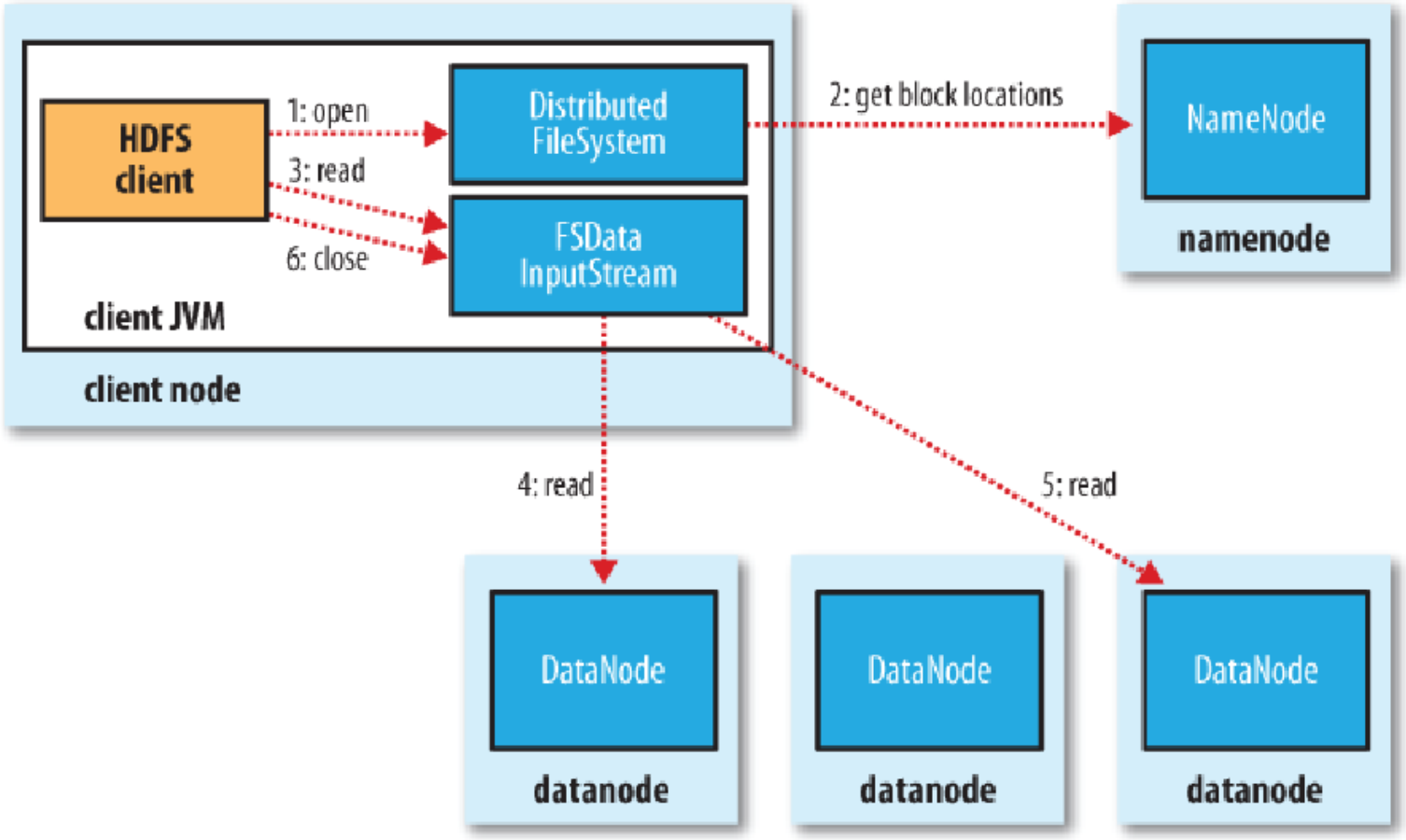


Creation of new file

Read Operation

- Checksums
- Verification

HDFS Read


HDFS client
1: open
3: read
6: close
Distributed FileSystem
FSData InputStream
client JVM
client node
2: get block locations
NameNode
namenode
4: read
5: read
DataNode
datanode
DataNode
datanode
DataNode
datanode

Replication

- Multiple nodes for reliability
- Additionally, data transfer bandwidth is multiplied
- Computation is near the data
- Replication factor

Image and Journal

State is stored in two files:

- fsimage: Snapshot of file system metadata
- editlog: Changes since last snapshot

Normal Operation:

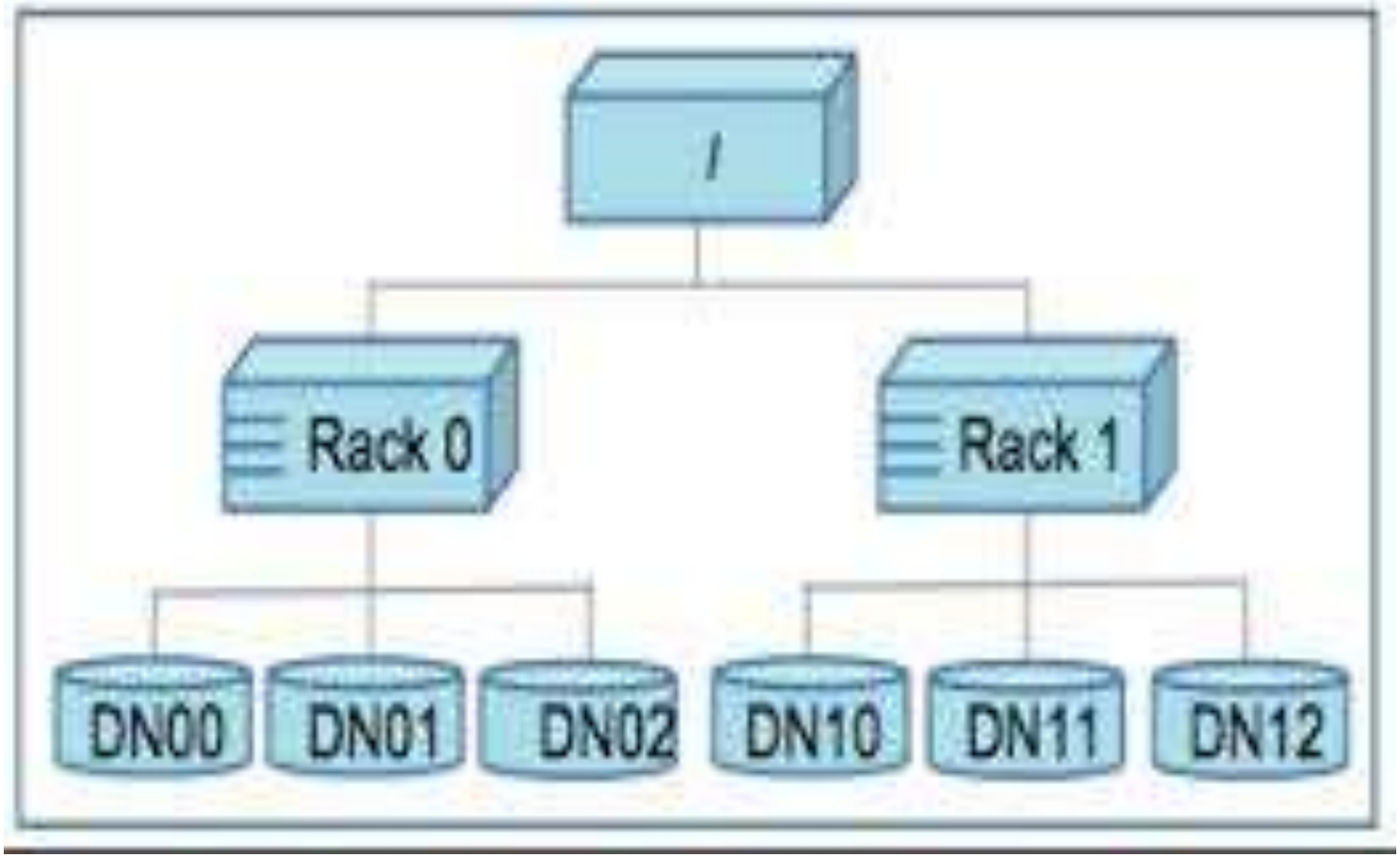
When namenode starts, it reads fsimage and then applies all the changes from edits sequentially

Snapshots

- Persistently save current state
- Instruction during handshake

Block Placement

- Nodes spread across multiple racks
- Nodes of rack share a switch
- Placement of replicas critical for reliability


/
Rack 0
Rack 1
DN00
DN01
DN02
DN10
DN11
DN12

Replication Management

- Replication factor
- Under-replication
- Over-replication

Balancer

- Balance disk space usage
- Optimize by minimizing the inter-rack data copying

Block Scanner

- Periodically scan and verify checksums
- Verification succeeded?
- Corrupt block?

Decommisioning

- Removal of nodes without data loss
- Retired on a schedule
- No blocks are entirely replicated

HDFS –What does it choose in CAP

- Partition Tolerance – can handle loosing data nodes
- Consistency

Steps towards Availability: Backup Node

Backup Node

- NameNode streams transaction log to BackupNode
- BackupNode applies log to in-memory and disk image
- Always commit to disk before success to NameNode
- If it restarts, it has to catch up with NameNode
- Available in HDFS 0.21 release
- Limitations:
  - Maximum of one per Namenode
  - Namenode does not forward Block Reports
  - Time to restart from 2 GB image, 20M files + 40 M blocks
    - 3 – 5 minutes to read the image from disk
    - 30 min to process block reports
    - BackupNode will still take 30 minutes to failover!

Files in HDFS

File Permissions

- Three types:
  - Read permission (r)
  - Write permission (w)
  - Execute Permission (x)
- Owner
- Group
- Mode

Command Line Interface

- hadoop fs –help
- hadoop fs –ls : List a directory
- hadoop fs mkdir : makes a directory in HDFS
- copyFromLocal : Copies data to HDFS from local filesystem
- copyToLocal : Copies data to local filesystem
- hadoop fs –rm : Deletes a file in HDFS

- More:

https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-common/FileSystemShell.html

Accessing HDFS directly from JAVA

- Programs can read or write HDFS files directly
- Files are represented as URIs
- Access is via the FileSystem API
  - To get access to the file: FileSystem.get()
  - For reading, call open() -- returns InputStream
  - For writing, call create() -- returns OutputStream

Interfaces

Getting data in and out of HDFS through the command-line interface is a bit cumbersome

Alternatives:

- FUSE file system: Allows HDFS to be mounted under Unix
- WebDAV Share: Can be mounted as filesystem on many OSes
- HTTP: Read access through namenode's embedded web svr
- FTP: Standard FTP interface

Demonstration

Questions?

Thankyou