HDFS

Hadoop Distributed File System

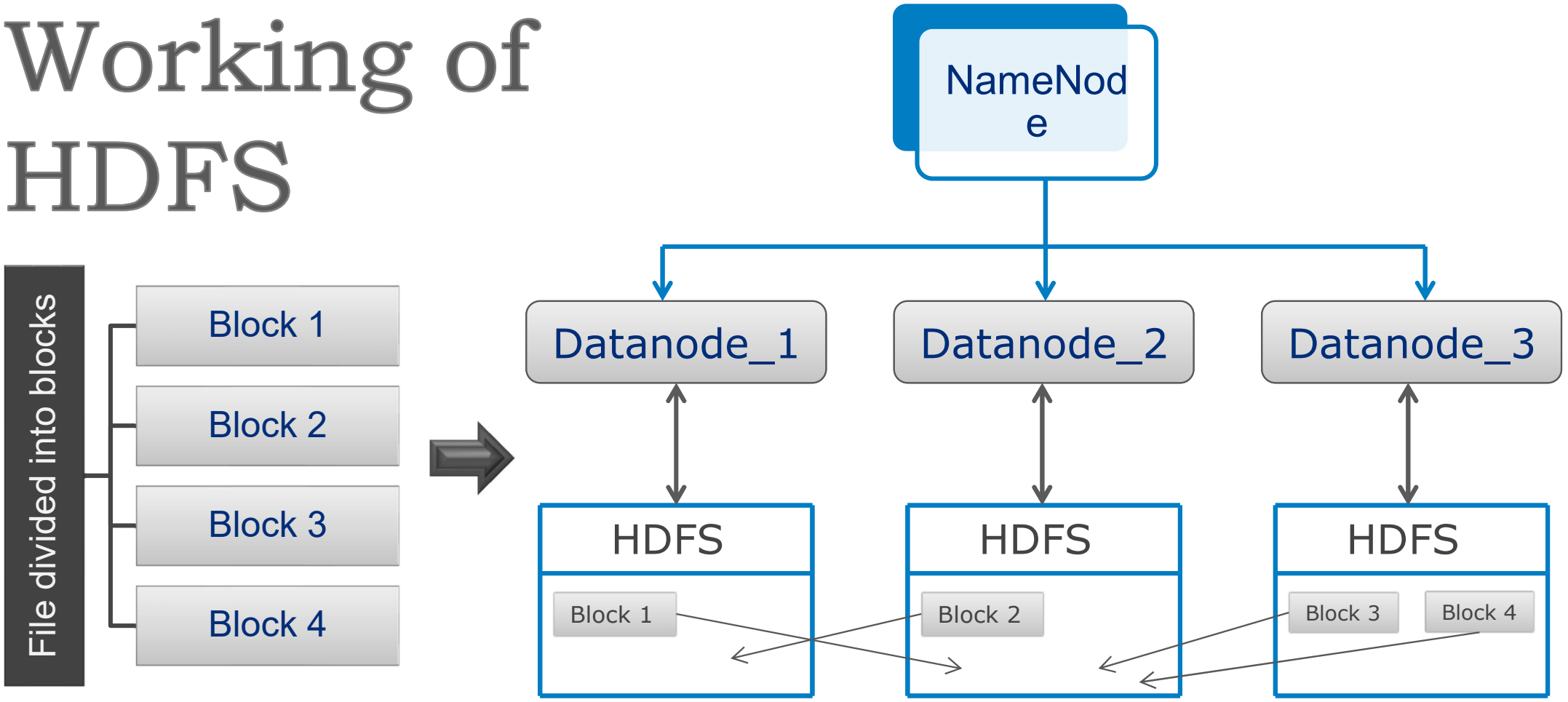
Topics Covered

- Design Goals
- Hadoop Blocks
- Rack Awareness, Replica Placement & Selection
- Permissions Model
- Anatomy of a File Write / Read on HDFS
- FileSystem Image and Edit Logs
- HDFS Check Pointing Process
- Directory Structure - NameNode, Secondary NameNode , DataNode
- Safe Mode, Trash, Name and Space Quota

HDFS Design Goals

- ***Hardware Failure -*** Detection of faults and quick, automatic recovery
- ***Streaming Data Access -*** High throughput of data access (Batch Processing of data)
- ***Large Data Sets -*** Gigabytes to terabytes in size.
- ***Simple Coherency Model –*** *Write once read many* access model for files
- *Moving computation is cheaper than moving data*

Working of HDFS



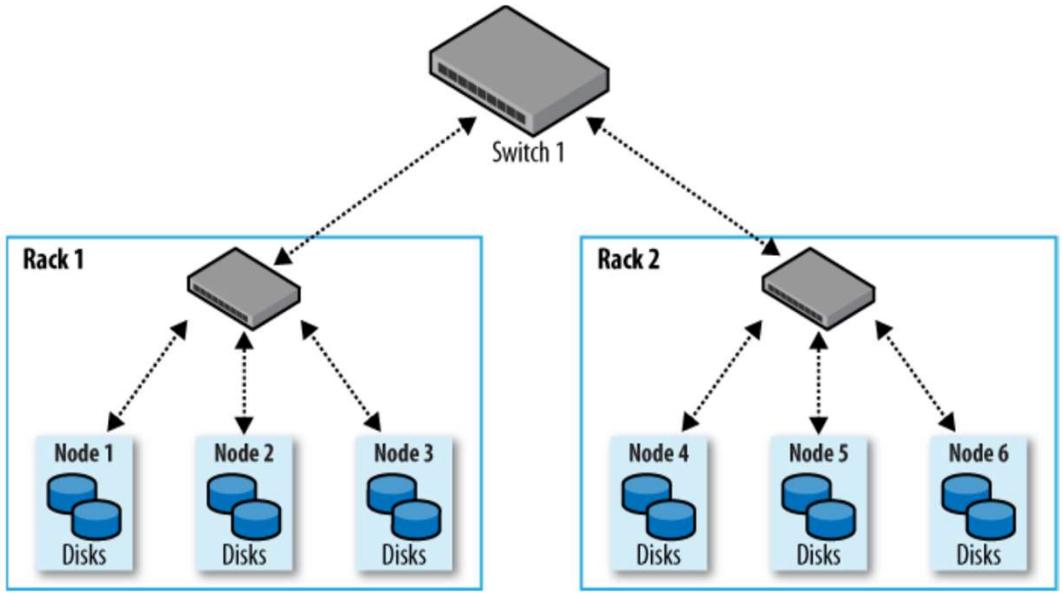
Storage & Replication of Blocks in HDFS

Blocks

- Minimum amount of data that can be read or write - 64 MB by default
- Minimize the cost of seeks
- A file can be larger than any single disk in the network
- Simplifies the storage subsystem – Same size & eliminating metadata concerns
- Provides fault tolerance and availability

Hadoop Rack Awareness

- Get maximum performance out of Hadoop cluster
- To provide reliable storage when dealing with DataNode failure issues.
- Resolution of the slave's DNS name (also IP address) to a rack id.
- Interface provided in Hadoop DNSToSwitchMapping, Default implementation is ScriptBasedMapping



Rack Topology - /rack1 & /rack2

Rack Awareness Configuration

File : hdfs-site.xml
Property : topology.node.switch.mapping.impl
Default Value : ScriptBasedMapping class

Property : topology.script.file.name
Value : <Absolute path to script file>

Sample Data (topology.data)

192.168.56.101 /dc1/rack1
192.168.56.102 /dc1/rack2
192.168.56.103 /dc1/rack2

Sample Command : ./topology.sh 192.168.56.101

Output : /dc1/rack1

```
HADOOP_CONF={${HADOOP_HOME}/conf
while [ $# -gt 0 ] ; do
 nodeArg=$1
 exec< ${HADOOP_CONF}/topology.data
 result=""
 while read line ; do
  ar=( $line )
  if [ "${ar[0]}" = "$nodeArg" ] ; then
   result="${ar[1]}"
  fi
 done
 shift
 if [ -z "$result" ] ; then
  echo -n "/default/rack "
 else
  echo -n "$result "
 fi
done
```

Ref Hadoop Wiki
http://wiki.apache.org/hadoop/topology_rack_awareness_scripts

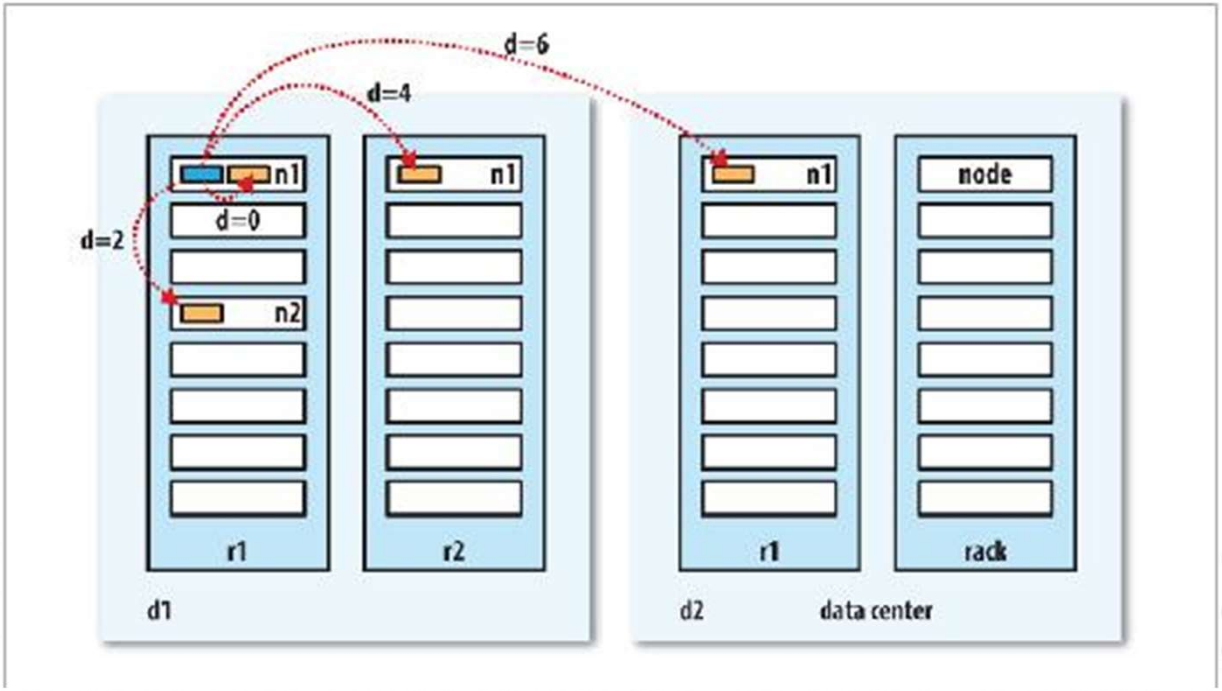
Sample Script

Replica Placement

Distance b/w Hadoop Client and DataNode

Same Node	:	d=0
Same Rack	:	d=2
Same Data Centre different rack	:	d=4
Different Data Centre	:	d=6

- Critical to HDFS reliability and performance
- Improve data reliability, availability, and network bandwidth utilization



Replica Placement cont..

Default Strategy :

a) First replica on the same node as the client.

b) Second replica is placed on a different rack from the first (off-rack) chosen at random

c) Third replica is placed on the same rack as the second, but on a different node chosen at random.

d) Further replicas are placed on random nodes on the cluster

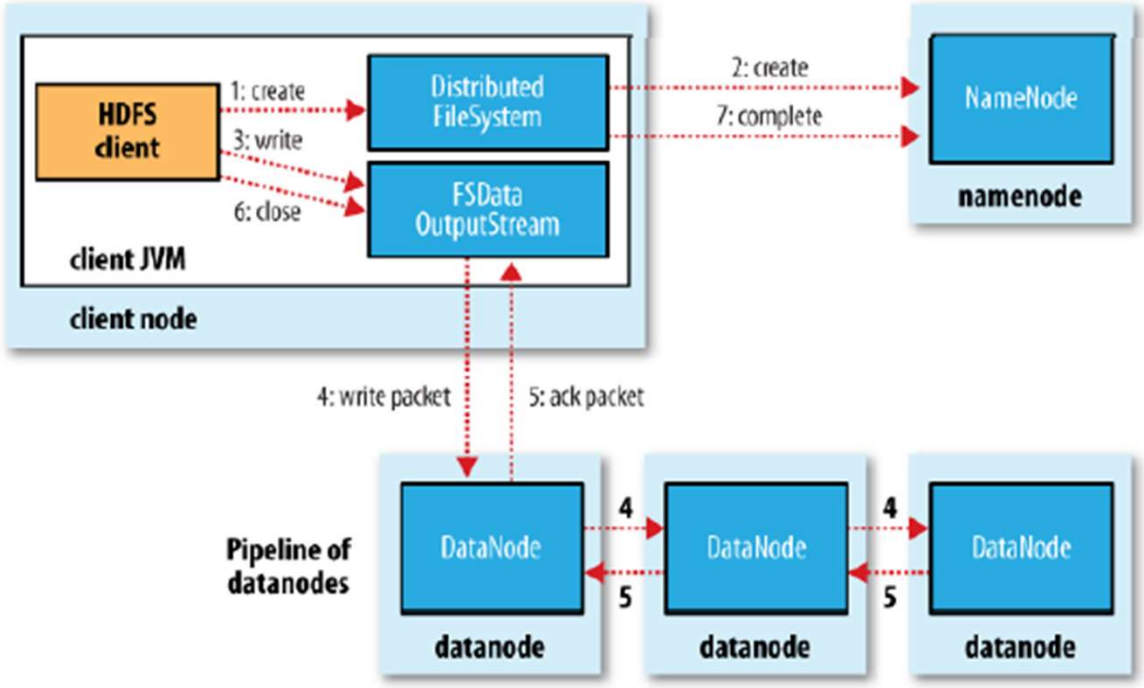
Replica Selection - HDFS tries to satisfy a read request from a replica that is closest to the reader.

Permissions Model

- Directory or file flag, permissions, replication, owner, group, file size, modification date, and full path.
- User Name : 'whoami'
- Group List : 'bash -c groups'
- Super-User & Web Server

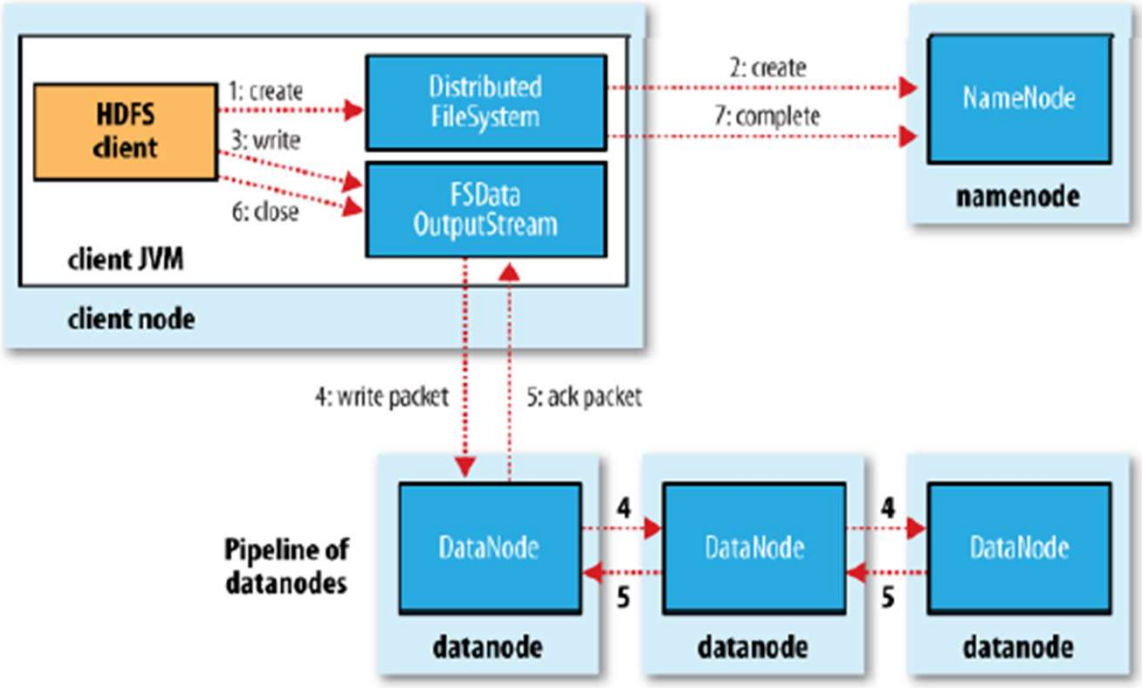
Anatomy of a File Write

- Client creates the file by calling create() method
- NameNode validates & processes the create request
- Spilt file into packets (DataQueue)
- DataStreamer asks NameNode for block / node mapping & pipelines are created among nodes.



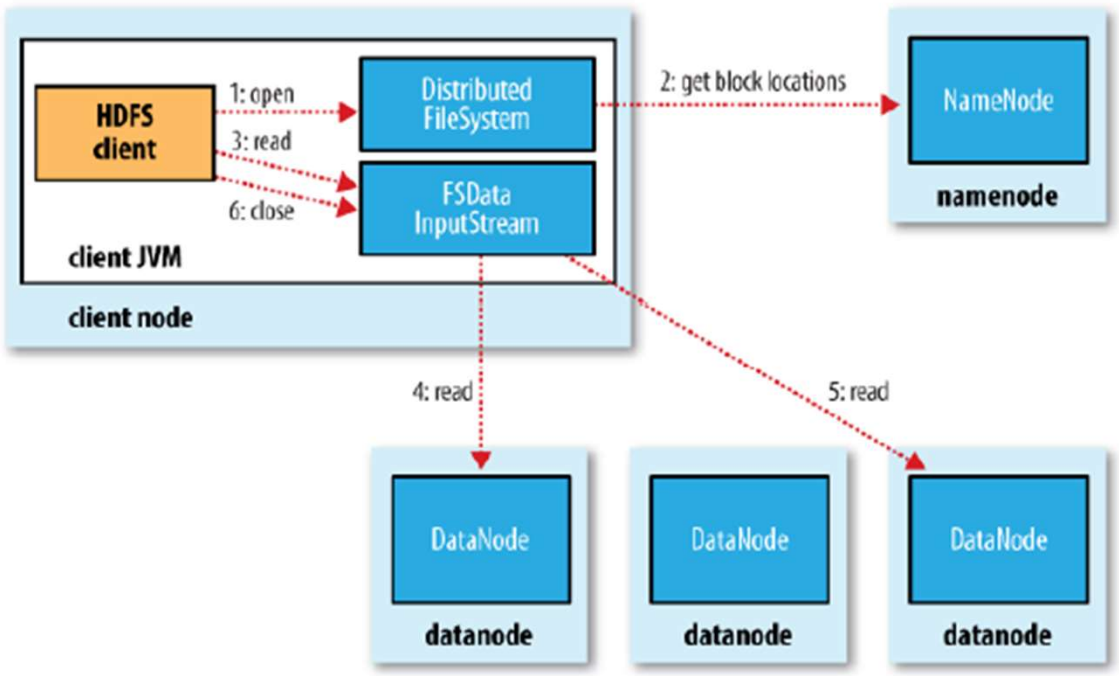
Anatomy of a File Write (cont..)

- DataStreamer streams the packets to the first DataNode
- DataNode forwards the copied packet to the next DataNode in the pipeline
- DFSOutputStream also maintains the ack queue and removes the packets after receiving acknowledgement from the DataNodes
- Client calls close() on the stream



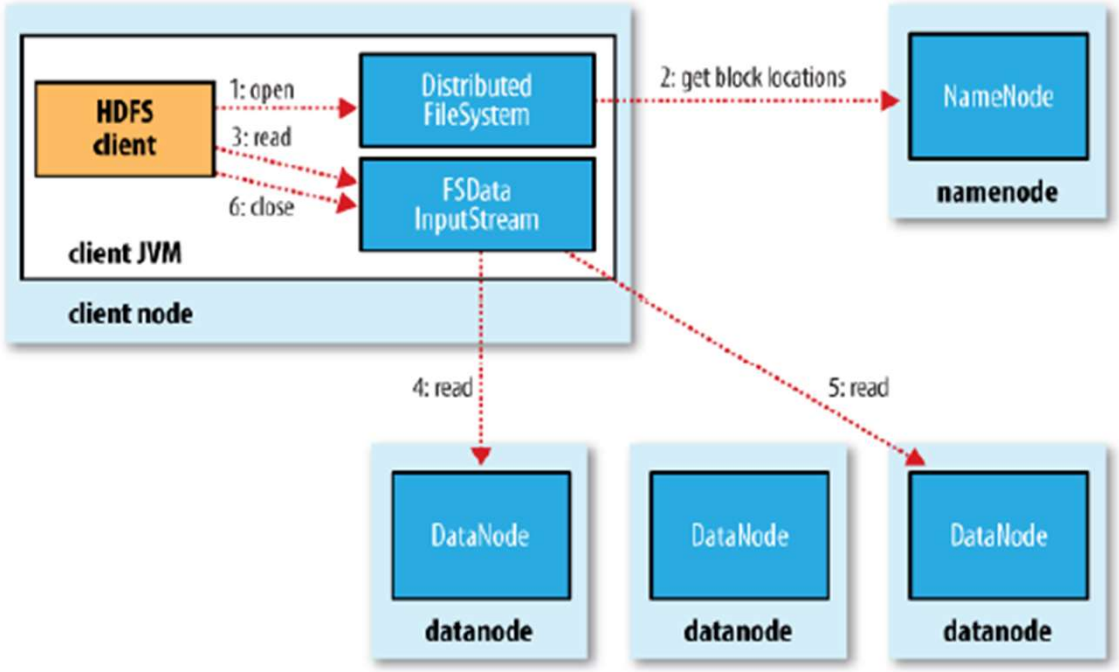
Anatomy of a File Read

- Client calls open() on the FileSystem object
- DistributedFileSystem calls the NameNode to determine the locations of the blocks
- NameNode validates request & for each block returns the list of DataNodes.
- DistributedFileSystem returns an input stream that supports file seeks to the client



Anatomy of a File Read (cont..)

- Client calls read() on the stream
- When the end of the block is reached, DFSInputStream will close the connection to the DataNode, then find the best DataNode for the next block.
- Client calls close() on the stream

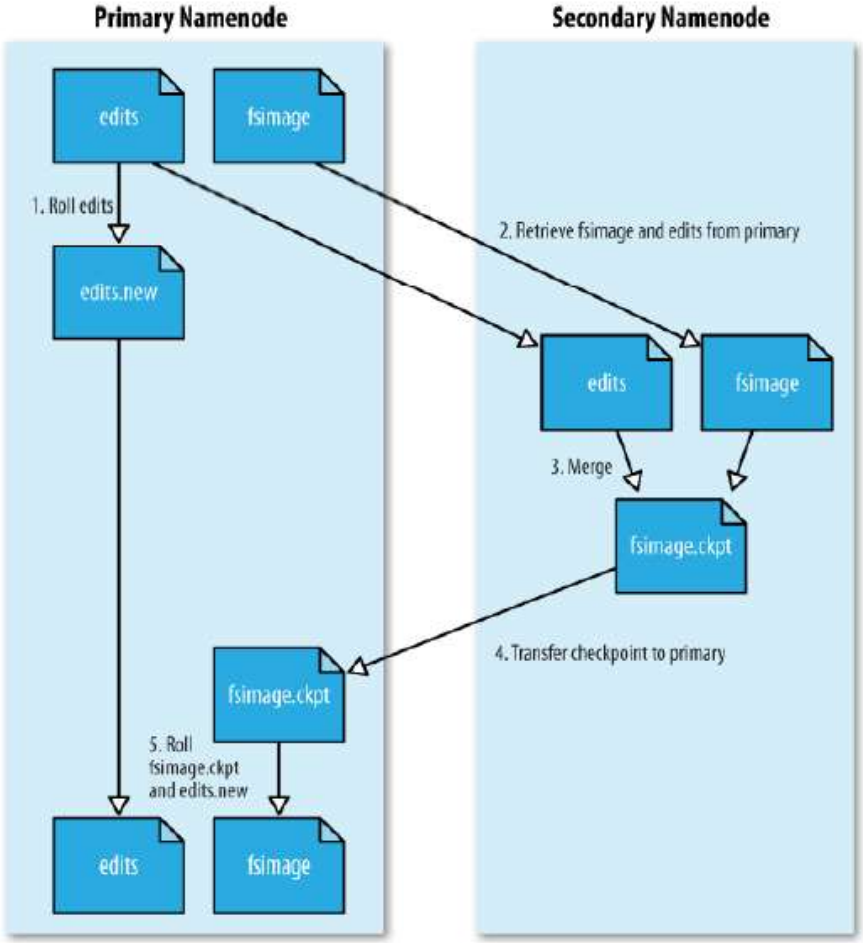


FileSystem Image and Edit Logs

- **fsimage** file is a persistent checkpoint of the file-system metadata
- When a client performs a write operation, it is first recorded in the edit log.
- The NameNode also has an in-memory representation of the files-ystem metadata, which it updates after the edit log has been modified
- Secondary NameNode is used to produce checkpoints of the primary's in-memory files-ystem metadata

Check Pointing Process

- Secondary NameNode asks the primary to roll its edits file, so new edits go to a new file.
- NameNode sends the fsimage and edits (using HTTP GET).
- Secondary NameNode loads fsimage into memory, applies each operation from edits, then creates a new consolidated fsimage file.
- Secondary NameNode sends the new fsimage back to the primary (using HTTP POST).
- Primary replaces the old fsimage with the new one. Updates the fstime file to record the time for checkpoint.



Directory Structure

NameNode (On NameNode only)
${dfs.name.dir}/current/VERSION
/edits
/fsimage
/fstime

Secondary NameNode (On SecNameNode Only)
${fs.checkpoint.dir}/current/VERSION
/edits
/fsimage
/fstime
/previous.checkpoint/VERSION
/edits
/fsimage
/fstime

DataNode (On all DataNodes)
${dfs.data.dir}/current/VERSION
/blk_<id_1>
/blk_<id_1>.meta
/blk_<id_2>
/blk_<id_2>.meta
/…
/blk_<id_64>
/blk_<id_64>.meta
/subdir0/
/subdir1/
/…

Block Count for a directory
dfs.datanode.numblocks property

Safe Mode

- On start-up, NameNode loads its image file (fsimage) into memory and applies the edits from the edit log (edits).
- It does the check pointing process itself. without recourse to the Secondary NameNode.
- Namenode is running in safe mode (offers only a read-only view to clients)
- The locations of blocks in the system are not persisted by the NameNode - this information resides with the DataNodes, in the form of a list of the blocks it is storing.
- Safe mode is needed to give the DataNodes time to check in to the NameNode with their block lists
- Safe mode is exited when the minimal replication condition is reached, plus an extension time of 30 seconds.

HDFS Safe Mode

SafeMode Properties – Configure safe mode as per cluster recommendations.

dfs.replication.min	*Default - 1*	Minimum no. of replicas for file wrtie success
dfs.safemode.threshold.pct	*Default – 0.999*	Min. portion of blocks statisfying the minimum replication to leave safe mode. Value 0 – Forces NN not to start in safemode Value 1 – Ensurs NN never leave safemode
dfs.safemode.extension	*Default – 30,000*	Extension time in milliseconds before NN leaves safemode

SafeMode Admin Commands – A command option for dfsadmin command: *hadoop dfsadmin -safemode*

get : Get the NameNode safemode status	*enter :* NameNode enters safemode
Wait : Wait for NameNode to exit safemode	*leave : NameNode leaves the safemode*

HDFS Trash – Recycle Bin

When a file is deleted by a user, it is not immediately removed from HDFS. HDFS moves it to a file in the /trash directory.

File	:	core-site.xml
Property	:	fs.trash.interval
Description	:	Number of minutes after which the checkpoint gets deleted.

A file remains in /trash for a configurable amount of time. After the expiry of its life in /trash, the NameNode deletes the file from the HDFS namespace.

File	:	core-site.xml
Property	:	fs.trash.checkpoint.interval
Description	:	Number of minutes between trash checkpoints. Should be smaller or equal to fs.trash.interval.

Undelete a file: User needs to navigate the /trash directory and retrieve the file by using mv command.

HDFS Quotas

Name Quota - a hard limit on the number of file and directory names in the tree rooted at that directory.

dfsadmin -setQuota <N> <directory>...	Set the name quota to be N for each directory.
dfsadmin -clrQuota <directory>...	Remove any name quota for each directory.

Space Quota - a hard limit on the number of bytes used by files in the tree rooted at that directory.

dfsadmin -setSpaceQuota <N> directory>..	Set the space quota to be N bytes for each directory.
dfsadmin -clrSpaceQuota <directory>...	Remove any spce quota for each directory.

Reporting Quota - count command of the HDFS shell reports quota values and the current count of names and bytes in use. With the -q option, also report the name quota value set for each directory, the available name quota remaining, the space quota value set, and the available space quota remaining.

fs -count -q <directory>..

FS Shell – Some Basic Commands

chgrp - Change group association of files.
Usage: hdfs dfs -chgrp [-R] GROUP URI [URI …]

chmod - Change the permissions of files.
Usage: hdfs dfs -chmod [-R] <MODE[,MODE]... | OCTALMODE> URI [URI …]

chown - Change the owner of files.
Usage: hdfs dfs -chown [-R] [OWNER][:[GROUP]] URI [URI]

du - Displays sizes of files and directories
Usage: hdfs dfs -du [-s] [-h] URI [URI …]
The -s option will result in an aggregate summary of file lengths being displayed, rather than the individual files.
The -h option will format file sizes in a "human-readable" fashion.

dus - Displays a summary of file lengths
Usage: hdfs dfs -dus <args>

http://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-common/FileSystemShell.html
http://hadoop.apache.org/docs/r1.2.1/file_system_shell.html

DfsAdmin Command

- bin/hadoop dfsadmin [Generic Options] [Command Options]

-safemode enter / leave / get / wait	Safe mode maintenance command. Safe mode can also be entered manually, but then it can only be turned off manually as well.
-report	Reports basic filesystem information and statistics.
-refreshNodes	Re-read the hosts and exclude files to update the set of Datanodes that are allowed to connect to the Namenode and those that should be decommissioned or recommissioned.
-metasave filename	Save Namenode's primary data structures to filename in the directory specified by hadoop.log.dir property. filename is overwritten if it exists. filename will contain one line for each of the following 1. Datanodes heart beating with Namenode 2. Blocks waiting to be replicated 3. Blocks currrently being replicated 4. Blocks waiting to be deleted

References

- *Hadoop Apache Website – http://hadoop.apache.org/*
- *Hadoop The Definitive Guide 3rd Edition By Oreilly*

Thank You..!!