



HDFS OVERVIEW

Bin Jiang

02/11/2017

Objective

Participants will learn about

- Hadoop Distributed File System (HDFS)
- Ways to access HDFS

Hadoop Distributed File System

- The hadoop Distributed File System (HDFS) is a distributed, scalable, fault-tolerant and portable file system written in Java
- Its creation was influenced by the Google File System papers
- The architecture of HDFS is based on the master/slave design pattern
- An HDFS cluster consists of:
 1. A NameNode (metadata server) holding directory information about file fragments persisted on Data nodes on their internal hard drives
 2. A number of DataNode processes (slaves), usually one per machine
- HDFS design is "rack-aware" to minimize network latency

HDFS Concept

- Blocks
 - Why is a block in HDFS so large
- NameNode
 - Secondary NameNode
- DataNode
- Block Caching
 - off-heap
 - cach pool

HDFS Concept

- HDFS Federation
 - viewfs:// URIs
 - How Much Memory Does a Namenode Need
- HDFS Availability
 - filesystem and edit log
 - checkpoints
 - standby namenode
 - shared storage (NFS filer, QJM – quorum journal manager)
 - datanode reports to both namenodes
 - Client failover

HDFS Concept

- Failover
 - failover controller
 - failover implementation - zookeeper
- Fencing
 - ungraceful failover
 - slow network
 - kill namenode process via ssh
 - strong fencing (NFS filer)
 - why QJM is recommended
 - disable network port
 - client failover (HDFS url)

NameNode FsImage and Edit Logs

- Read FsImage

- `http://sandbox.hortonworks.com:50070/imagetransfer?getimage=1&txid=latest`
- `hdfs oiv -p Delimited -i /hadoop/hdfs/namenode/current/fsimage_0000000000000511883 -o /root/fsimage.txt`

- Read EditLog

- `http://sandbox.hortonworks.com:50070/imagetransfer?getedit=1&startTxId=0000000000000170874&endTxId=0000000000000176598slow network`
- `hdfs oev -p xml -i /hadoop/hdfs/namenode/current/edits_0000000000000170874-0000000000000176598 -o /root/editlog.txt`

Hadoop Filesystems

- Local
 - file
 - fs.LocalFileSystem
- HDFS
 - Hdfs
 - hdfs.DistributedFileSystem
- WebHDFS
 - webhdfs
 - hdfs.web.WebHdfsFileSystem

Hadoop Filesystems

- Local
 - file
 - fs.LocalFileSystem
- HDFS
 - Hdfs
 - hdfs.DistributedFileSystem
- WebHDFS
 - webhdfs
 - hdfs.web.WebHdfsFileSystem

Hadoop Filesystems

- SecureWebHDFS
 - `swebhdfs`
 - `hdfs.web.SWebHdfsFileSystem`
- HAR
 - `har`
 - `fs.HarFileSystem`
- View
 - `viewfs`
 - `viewfs.ViewFileSystem`

Hadoop Filesystems

- FTP
 - ftp
 - fs.ftp.FTPFileSystem
- S3
 - s3a
 - fs.s3a.S3AFileSystem
- Azure
 - wasb
 - fs.azure.NativeAzureFileSystem

Accessing HDFS

- HDFS is not a Portable Operating System Interface (POSIX) compliant file system
- It is modeled after traditional hierarchical file systems containing directories and files
- File-system commands (copy, move, delete etc.) against HDFS can be performed in a number of ways:
 1. Through the HDFS Command-Line Interface (CLI) which supports Unix-like commands: cat, chown, ls, mkdir, mv etc.
 2. Using Java API for HDFS
 3. Via the C-language wrapper around the Java API
 4. Using regular HTTP browser for file-system and file content viewing
 5. Using WebHDFS API

Accessing HDFS

- Interfaces

- HTTP

- C

- NFS

- FUSE

- Java

Command Line Interface

- `hdfs-site.xml`
 - `fs.defaultFS`
 - HDFS port: 8020
 - `dfs.replication: 3`
- Basic File Operation
 - `hadoop fs`

Examples of HDFS Commands

- The common way to invoke the HDFS CLI: `Hadoop fs {HDFS commands}`

- Ask for Help:

`hadoop fs -help`

`Hadoop fs -help rm`

- List the directory:

`hadoop fs -ls`

`Hadoop fs -ls /user/root`

`hadoop fs -ls -R`

- Copying a file from the local file system over to HDFS:

`Hadoop fs -help put`

`hadoop fs -put install.log`

`hadoop fs -put install.log install1.log`

`hadoop fs -put install.log /tmp`

Examples of HDFS Commands

- The common way to invoke the HDFS CLI: `Hadoop fs {HDFS commands}`

- Copying the file from HDFS to the local file system:

```
Hadoop fs -help get
```

```
hadoop fs -get install1.log
```

- Creating Folders:

```
hadoop fs -mkdir report
```

- Moving Files Around:

```
hadoop fs -mv install1.log /user/root/report/install.log
```

- Deleting Files:

```
hadoop fs -rm install1.log
```

```
hadoop fs -rm -R /user/root/report
```

```
Hadoop fs -rm -R -skipTrash /user/root/report
```

Examples of HDFS Commands

- Checking the health of files in HDFS:

```
hdfs fsck /root -files -blocks
```

WebHDFS

- WebHDFS is a REST API for accessing all of the HDFS file system interfaces
- Via WebHDFS you can use such common tools as curl and wget to interface with HDFS
- WebHDFS is embedded in HDFS's NameNode and DataNode
- You can use Kerberos (SPNEGO) and Hadoop delegation tokens for client authentication

Examples of WebHDFS Calls

- **List Directory**

<http://127.0.0.1:50070/webhdfs/v1/user/root?op=LISTSTATUS>

<http://127.0.0.1:50070/webhdfs/v1/user/root/business?op=LISTSTATUS>

- **Get Content Summary of a Directory**

<http://127.0.0.1:50070/webhdfs/v1/user/root?op=GETCONTENTSUMMARY>

- **Get Home Directory**

<http://127.0.0.1:50070/webhdfs/v1/user/root?op=GETHOMEDIRECTORY>

Examples of WebHDFS Calls

- Read File

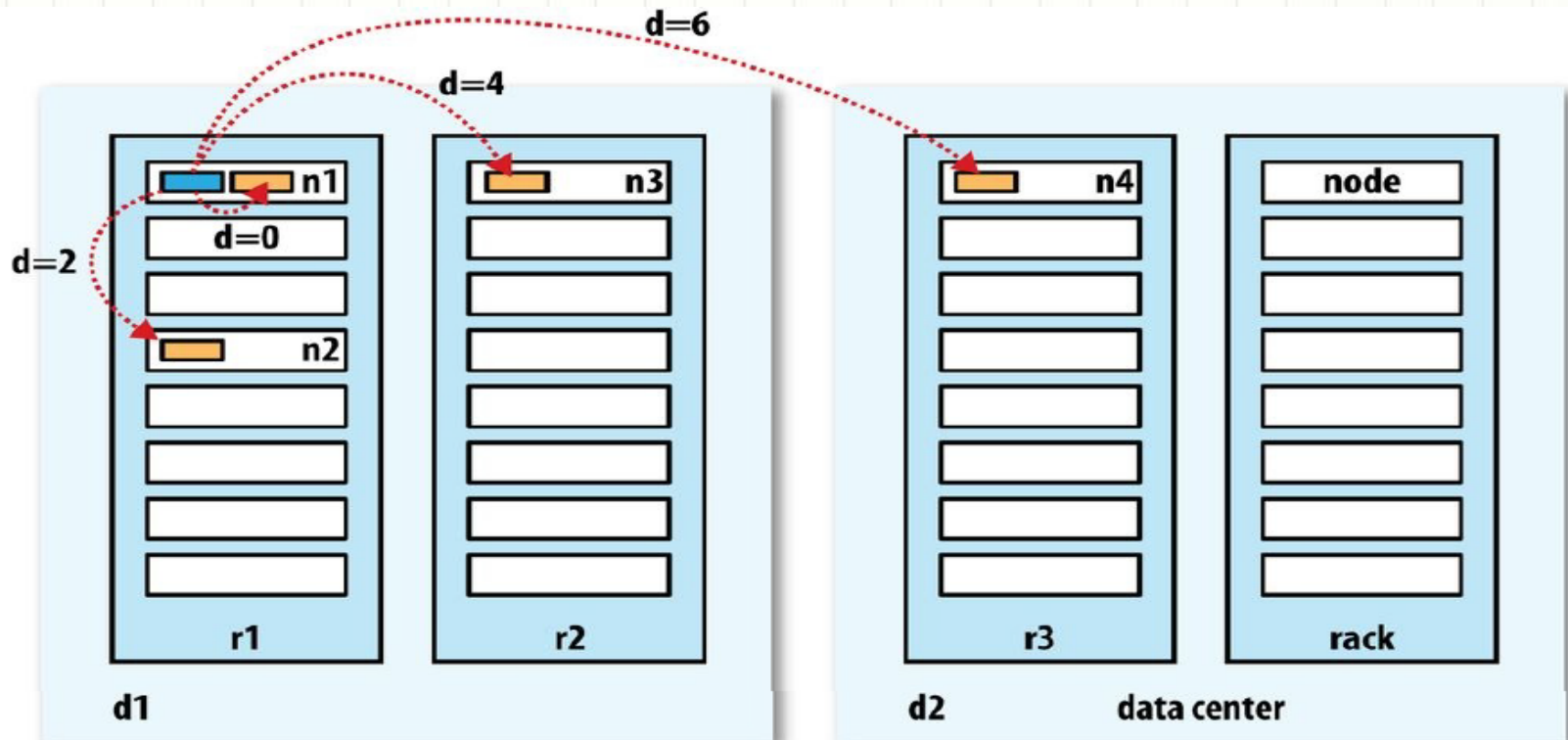
<http://127.0.0.1:50070/webhdfs/v1/user/root/business/part-m-00000?op=OPEN&namenoderpcaddress=sandbox.hortonworks.com:8020&offset=0>

- Get File Status

<http://127.0.0.1:50070/webhdfs/v1/user/root/business/part-m-00000?op=GETFILESTATUS&namenoderpcaddress=sandbox.hortonworks.com:8020&offset=0>

- Network Topology and Hadoop

- Bandwidth
- Distance between nodes

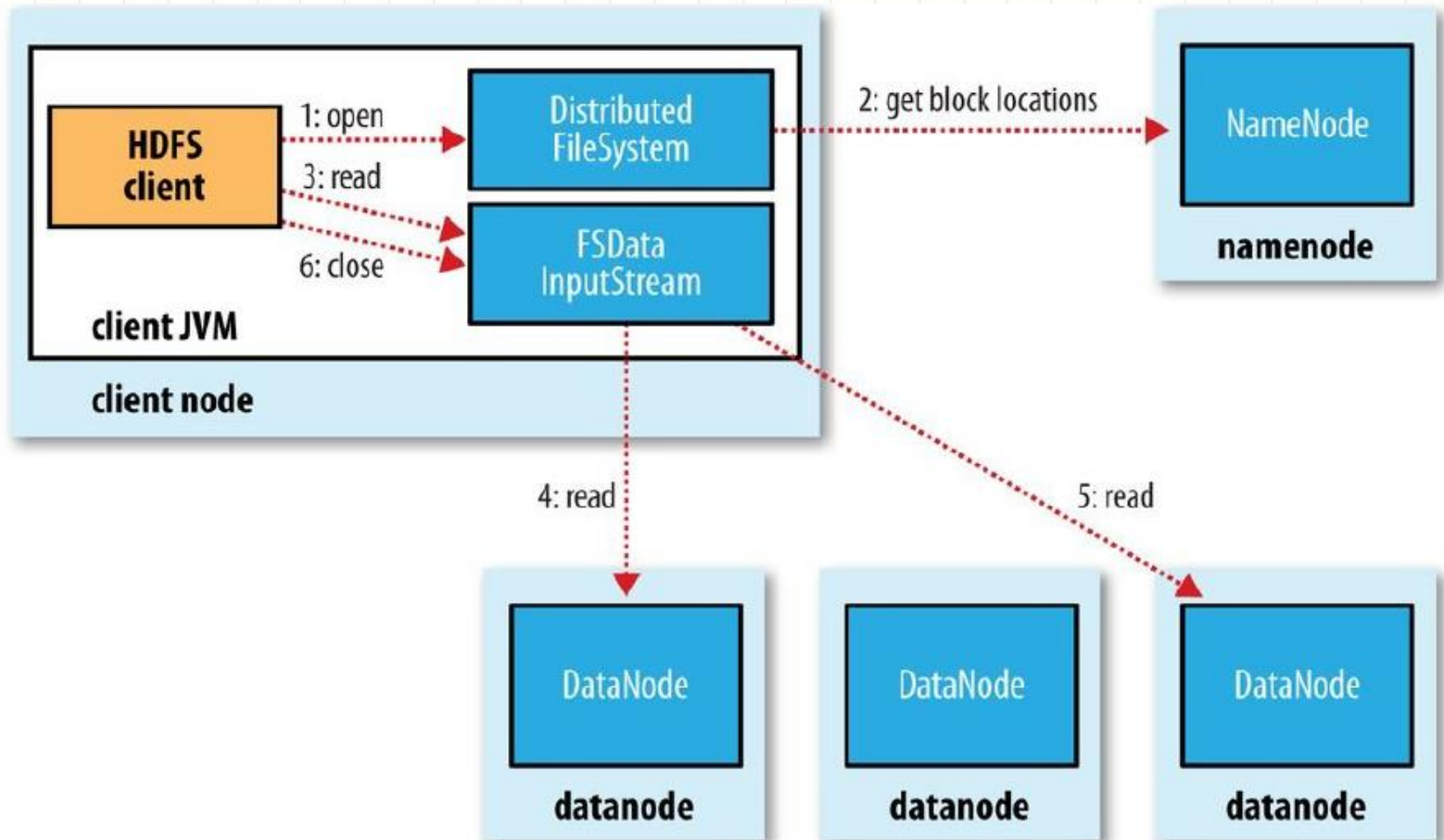


Data Flow

- File Read
 - RPC
 - Data Locality
 - Data Node address
 - Next Batch Block
 - Remember Failed Data Node
 - Checksum
 - Client connects to Data Node Directly Guided by Name Node

Data Flow

- File Read



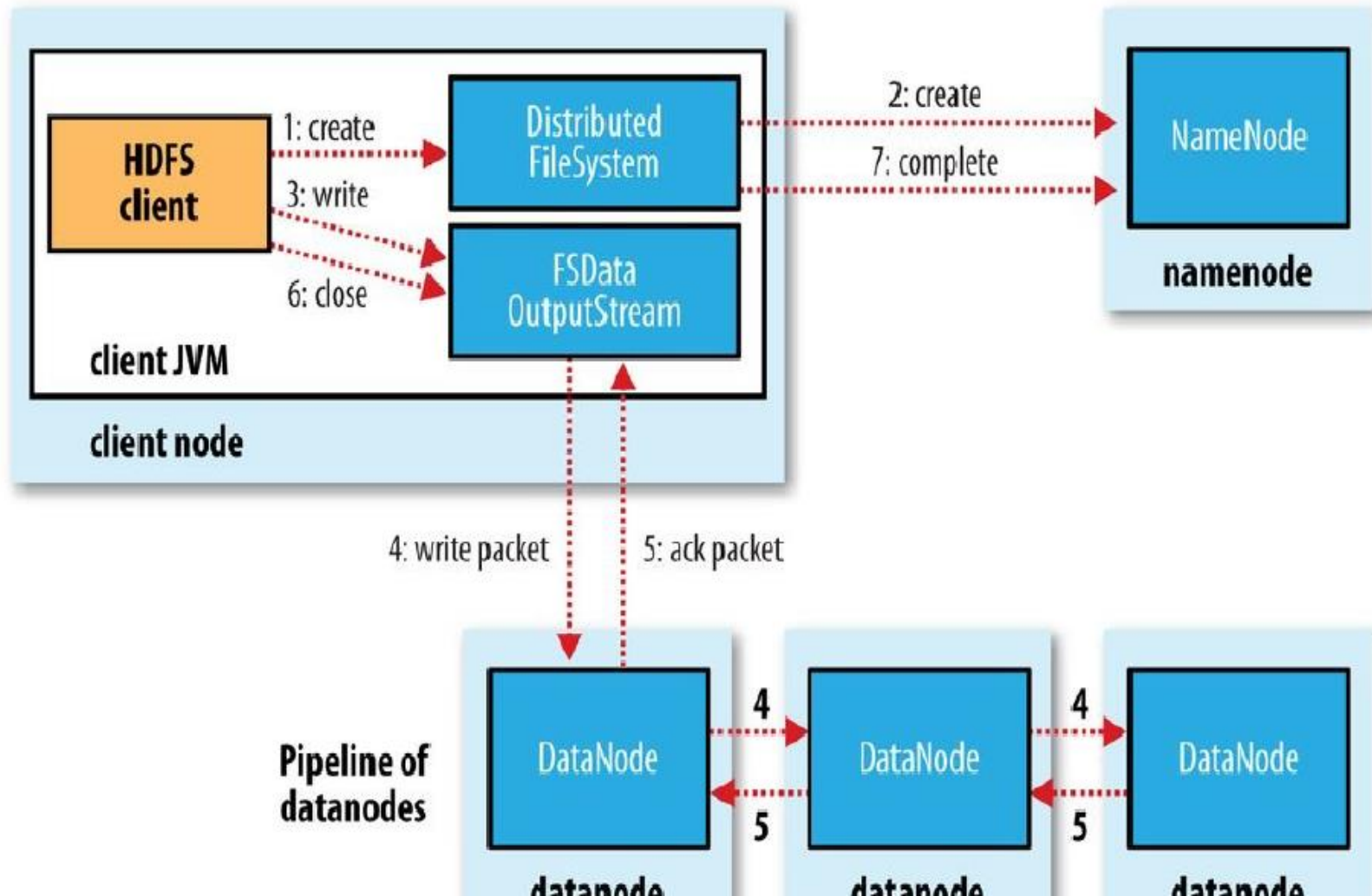
Data Flow

- File Write

- Name Node Checks the File
- Data Queue
- Data Streamer (Ask Name Node to Allocate Blocks)
- Next Batch Block
- Ack Queue
- Action if fail to write to Data Node
- Partial Blocks will be deleted from failed data node
- New Good Pipeline
- Name Node for under-replicated
- `dfs.namenode.replication.min`
- Asynchronously replicated

Data Flow

- File Write



Data Flow

- Coherency Model
 - Hflush (data node memory)
 - Hsync
 - Lose data without hsync

Data Flow

- Parallel Copying with distcp
 - `hadoop distcp file1 file2`
 - try it out on a small test directory tree first
 - transferring data between two HDFS clusters
 - keeping an HDFS Cluster Balanced (set mapper)

File Permission in HDFS

- POSIX Model
 - three type of permissions
 - owner, group and mode
 - dfs.permissions.enabled
 - chmod, chown and chgrp

Fine Grained Permission in HDFS

- HDFS ACL
 - background
 - `dfs.namenode.acls.enabled` (`hdfs-site.xml`)
 - `dfs.permissions.enabled`
 - `chmod`, `chown` and `chgrp`

Fine Grained Permission in HDFS

- Granting Access to Another Named Group
 - `hadoop fs -mkdir /tmp/permission-acl-1`
 - `hadoop fs -mkdir /tmp/permission-acl-2`
 - `hadoop fs -chmod 750 /tmp/permission-acl-1`
 - `hadoop fs -setfacl -m group:hadoop:r-x /tmp/permission-acl-1`
 - `hadoop fs -getfacl /tmp/permission-acl-1`
 - `hadoop fs -ls /tmp/permission-acl-1`

Fine Grained Permission in HDFS

- Using a Default ACL for Automatic Application to New Children
 - `hadoop fs -setfacl -m default:group:hadoop:r-x /tmp/permission-acl-2`
 - `hadoop fs -mkdir /tmp/permission-acl-2/sub`
 - `hadoop fs -getfacl -R /tmp/permission-acl-2`

Fine Grained Permission in HDFS

- Blocking Access to a Sub-tree for a specific User
 - `hadoop fs -setfacl -m user:yarn:--- /tmp/permission-acl-2`
 - `hadoop fs -getfacl /tmp/permission-acl-2`