

## → HDFS

- a. Hadoop Distributed File System
- b. Based on Master-Slave Architecture
- c. Basic rule : *more the partitioning/division, faster the processing*
- d. HDFS = Master (nameNodes) + Slaves(NameNodes)
- e. Metadata of name Node
  - i. Contains FS image and edit logs
  - ii. In case metadata is destroyed or Name Node goes down, all files in hadoop cluster become inaccessible
  - iii. Is a very critical part of Hadoop cluster to access files
- f. Metadata of block
  - i. Block IDs + location/IP of Data Node storing a block
  - ii. Block pool (BP)
- g. Block size
  - i. Size of split being created
  - ii. default block size in HDFS (hadoop3) is 128MB (but hadoop2 or previous had default block size 64MB)
  - iii. e.g. : a file of 10GB size with default block size of 128MB splits into 80 blocks  
 $10\text{GB} / 128\text{MB} = 10\text{GB} \times 1024\text{MB} / 128\text{MB} = 80 \text{ splits/blocks}$   
 $1\text{TB} / 128\text{MB} = 1\text{TB} \times 1024\text{GB} \times 1024\text{MB} / 128\text{MB} = 8192 \text{ splits/blocks}$
- h. Client Machine
  - i. Also known as gateway
  - ii. User can connect to HDFS using Client/Gateway
  - iii. User can fire queries using client
  - iv. Can R/W data to Data Nodes using Name Node
  - v. While writing a file, client Splits the file as per default block size and send files to Name Node for storage
  - vi. Client only writes the first replica, but other replicas are created by Name Node
  - vii. While reading a file, client only sends the filename to Name node and it fetches the data
- i. Has two types of Nodes in Hadoop cluster:
  - i. Master node/Name Node
    - 1. Only one master node / Name Node which manage the Hadoop cluster
    - 2. Is a very powerful machine as it has to do cluster management tasks
    - 3. Should always be in a running state, otherwise files become inaccessible
    - 4. Based on UNIX + Hadoop + Java
    - 5. Assigns Block Pool, block IDs and locations/IPs of Data Node to splits of file received from client to store on Data Node
    - 6. Contains metadata(FS image and edit logs)

7. manages all the Slave/Data nodes in Hadoop cluster, manage data and communication between Slave/data nodes
8. Sends heartbeat signal to Zookeeper
9. runs Resource Manager service to manage resources and jobs on Data Nodes

ii. Slave Node/Data Node

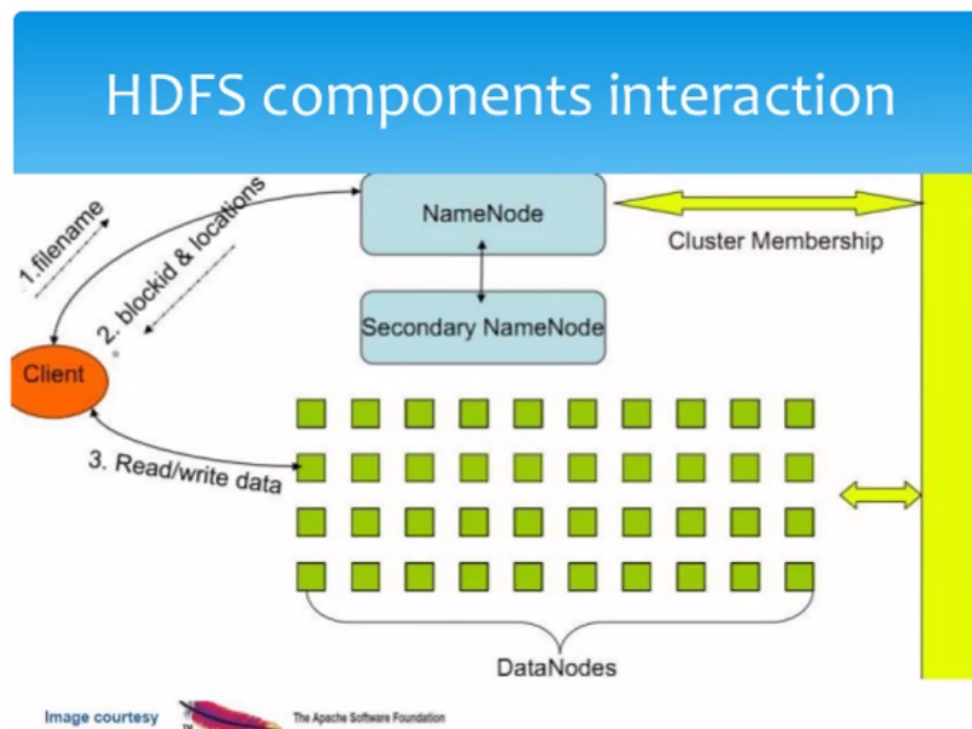
1. There may be number of Slave nodes called Data Nodes
2. minimum one Data Node is required for cluster
3. Can be of heterogeneous configuration
4. Sends heartbeat signal to Name Node every 3 secs
5. Sends block report to Name Node every 30 secs
6. Runs Node Manager service which accepts tasks and resources(files) from Resource manager on name Node to store & process data
7. Data is stored on Slave Nodes/Data nodes as splits distributed over different Data Nodes

j. Zookeeper

- i. Separate set of services
- ii. Monitors health of every nameNode

→ Objectives of cluster:

- a. Store data
- b. Parallel processing



## → Fault tolerance in Hadoop


- a. Fault tolerance is the ability to handle any faults in the system
- b. Fault tolerance in Hadoop is implemented using two methods
  - i. Replication Factor (R.F.)
    1. In case one node goes down, another replica can be created from nodes which are active, so it is needed for distributed system
    2. number of replicas being made
    3. It is an automatic procedure performed by Name node while writing a file or when a data node goes down
    4. Default replication factor is 3, minimum Replication Factor is 1, maximum Replication Factor is 512
    5. RF can be changed to factor 'n' using command

```
hadoop fs -setrep <n> file1
```

```
hadoop fs -setrep 4 file1 # sets RF to 4
```

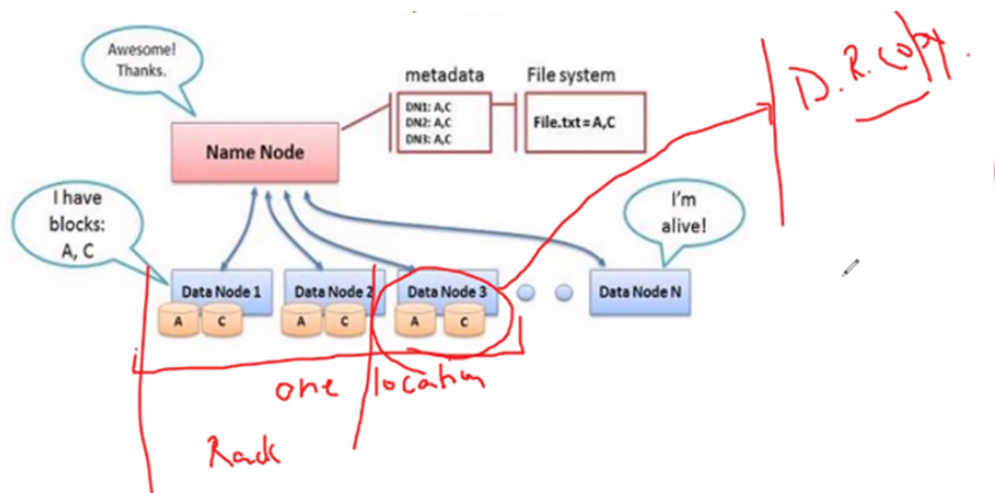
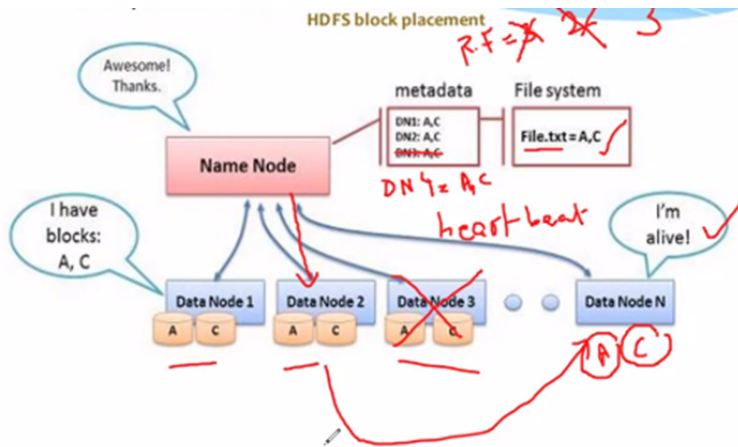
6. RF should not be more than number of Data Nodes in cluster [ $RF \leq \text{No. of DataNodes}$ ]
7. Client only writes the first replica, but other replicas are created by Name Node
8. Under-replication
  - a. when there are less replicas than the set replication factor
  - b. E.g. a file of 950GB is stored on 1TB cluster will have only 1 replica but RF is 3
  - c. Immediate soln is to delete the extra files to create more space, so that Hadoop can automatically create more replicas as per set replication factor OR user can reduce the replication factor to the existing number of replicas
9. No two replicas should be on same node

DataNodes

Image courtesy  The Apache Software Foundation

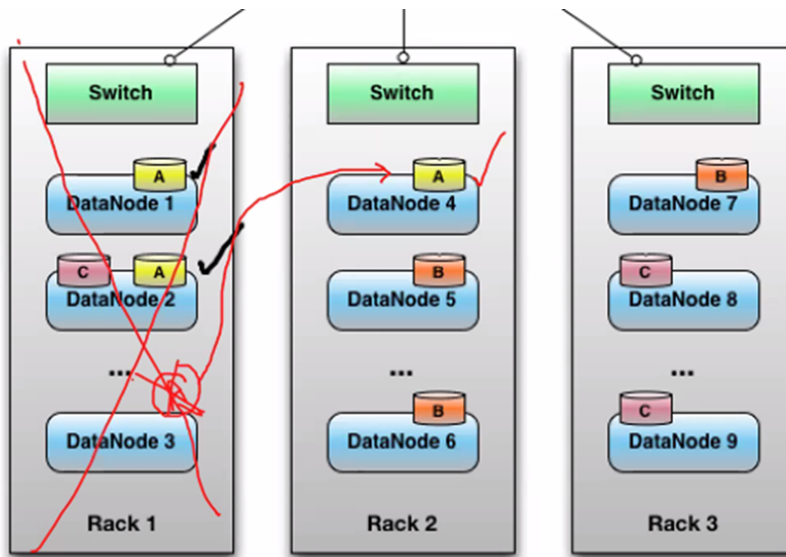
$$1 \text{ PB} [R.F=3] = \text{DNs} [10 \text{ TB}]$$
$$1 \text{ PB} = 1024 \text{ TB} \times 3 = 3072 \text{ TB} = 307.2$$

308



## ii. Rack Awareness

1. 3rd copy or disaster copy
2. Placing the nodes not in one location , but at atleast 2 locations
3. Storing data in different racks at different locations to avoid failure of the cluster all at once
4. Hadoop cluster consists of all the racks at different locations
5. Each rack contains multiple data nodes



### iii. Backup Node

#### 1. Secondary Name Node

- existed till hadoop v2
- serves as backup for Active name Node
- Backup of metadata(FS image and edit logs) is taken every 1 hour from Active Name Node to Secondary Name Node
- Can recover to last backup which was taken within last 1 hour
- Causes lot of downtime, as recovering becomes difficult as it can only be recovered to the last backup and the jobs are lost which are launched after backup was taken

#### 2. Standby Name Node

- Exists from Hadoop v3
- Is an “perfect” replica of Active Name Node
- Backup of metadata(FS image and edit logs) is taken simultaneously
- Can recover to immediate state of Active Name Node, so no running job is lost
- Near to zero downtime as Standby Name Node starts working immediately after Active Name Node goes down



→ Webshell connects to gateway

## # hadoop fs commands

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs
Usage: hadoop fs [generic options]
    [-appendToFile <localsrc> ... <dst>]
    [-cat [-ignoreCrc] <src> ...]
    [-checksum <src> ...]
    [-chgrp [-R] GROUP PATH...]
    [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
    [-chown [-R] [OWNER][:[GROUP]] PATH...]
    [-copyFromLocal [-f] [-p] [-l] [-d] [-t <thread count>] <localsrc> ... <dst>]
    [-copyToLocal [-f] [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
    [-count [-q] [-h] [-v] [-t <storage type>]] [-u] [-x] [-e] <path> ...]
    [-cp [-f] [-p] [-p[topax]] [-d] <src> ... <dst>]
    [-createSnapshot <snapshotDir> [<snapshotName>]]
    [-deleteSnapshot <snapshotDir> <snapshotName>]
    [-df [-h] [<path> ...]]
    [-du [-s] [-h] [-v] [-x] <path> ...]
    [-expunge]
    [-find <path> ... <expression> ...]
    [-get [-f] [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
    [-getfacl [-R] <path>]
    [-getfattr [-R] {-n name | -d} [-e en] <path>]
    [-getmerge [-nl] [-skip-empty-file] <src> <localdst>]
    [-help [cmd ...]]
    [-ls [-C] [-d] [-h] [-q] [-R] [-t] [-S] [-r] [-u] [-e] [<path> ...]]
    [-mkdir [-p] <path> ...]
    [-moveFromLocal <localsrc> ... <dst>]
    [-moveToLocal <src> <localdst>]
    [-mv <src> ... <dst>]
    [-put [-f] [-p] [-l] [-d] <localsrc> ... <dst>]
    [-renameSnapshot <snapshotDir> <oldName> <newName>]
    [-rm [-f] [-r] [-R] [-skipTrash] [-safely] <src> ...]
    [-rmdir [--ignore-fail-on-non-empty] <dir> ...]
    [-setfacl [-R] [{-b|-k} {-m|-x <acl_spec>} <path>]][--set <acl_spec> <path>]]
    [-setfattr {-n name [-v value] | -x name} <path>]
    [-setrep [-R] [-w] <rep> <path> ...]
    [-stat [format] <path> ...]
    [-tail [-f] <file>]
    [-test [-defsz] <path>]
    [-text [-ignoreCrc] <src> ...]

    [-touch [-a] [-m] [-t TIMESTAMP ] [-c] <path> ...]
    [-touchz <path> ...]
    [-truncate [-w] <length> <path> ...]
    [-usage [cmd ...]]

Generic options supported are:
-conf <configuration file>      specify an application configuration file
-D <property=value>             define a value for a given property
-fs <file:///hdfs://namenode:port> specify default filesystem URL to use, overrides 'fs.defaultFS' property from configurations.
-jt <local|resource|emanager:port> specify a ResourceManager
-files <file1,...>              specify a comma-separated list of files to be copied to the map reduce cluster
-libjars <jar1,...>             specify a comma-separated list of jar files to be included in the classpath
-archives <archive1,...>       specify a comma-separated list of archives to be unarchived on the compute machines

The general command line syntax is:
command [genericOptions] [commandOptions]
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$
```

## # listing root dir of hadoop fs

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -ls /
Found 7 items
drwxr-xr-x - hbase hbase          0 2023-06-03 01:03 /hbase
drwxrwxrwx - hdfs supergroup      0 2023-02-06 07:53 /markovData
drwxrwxr-x - solr solr            0 2023-04-13 19:15 /solr
drwxr-xr-x - hdfs supergroup      0 2021-05-22 18:21 /system
drwxrwxrwt - hdfs supergroup      0 2023-05-20 07:39 /tmp
drwxr-xr-x - hdfs supergroup      0 2023-06-05 08:42 /user
drwxr-xr-x - hdfs supergroup      0 2021-10-29 14:36 /userTest2
[bigdatalab456422@ip-10-1-1-204 ~]$
```

## # create a file in machine file system

```
[bigdatalab456422@ip-10-1-1-204 ~]$ vi file1.txt
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ cat file1.txt
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ cat file1.txt
this is a unix file
we will be uploading on hdfs
[bigdatalab456422@ip-10-1-1-204 ~]$
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ ll
```

total 4

-rw-rw-r-- 1 bigdatalab456422 bigdatalab456422 50 May 16 12:19

file1.txt

```
# put one file to hadoop fs
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -put file1.txt  
/user/bigdatalab456422
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -ls  
/user/bigdatalab456422
```

```
Found 1 items
```

```
-rw-r--r--    3 bigdatalab456422 bigdatalab456422          50  
2023-05-16 12:22 /user/bigdatalab456422/file1.txt
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -cat  
/user/bigdatalab456422/file1.txt
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -cat /user/bigdatalab456422/file1.txt  
this is a unix file  
we will be uploading on hdfs  
[bigdatalab456422@ip-10-1-1-204 ~]$
```

```
# making a dir
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -mkdir  
/user/bigdatalab456422/training
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -ls  
/user/bigdatalab456422
```

```
Found 2 items
```

```
-rw-r--r--    3 bigdatalab456422 bigdatalab456422          50  
2023-05-16 12:22 /user/bigdatalab456422/file1.txt  
drwxr-xr-x    - bigdatalab456422 bigdatalab456422          0  
2023-05-16 12:25 /user/bigdatalab456422/training
```

```
# put another file
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -put file2.txt  
/user/bigdatalab456422/training
```

```
# move a file
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -mv  
/user/bigdatalab456422/file1.txt /user/bigdatalab456422/training
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -mv  
/user/bigdatalab456422/training/file1.txt  
/user/bigdatalab456422/training/newfile.txt
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -ls  
/user/bigdatalab456422/training
```

```
Found 2 items
```

```
-rw-r--r--    3 bigdatalab456422 bigdatalab456422          20  
2023-05-16 12:32 /user/bigdatalab456422/training/file2.txt
```



```
-rw-r--r--    3 bigdatalab456422 bigdatalab456422          50
2023-05-16 12:22 /user/bigdatalab456422/training/newfile.txt
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -ls
/user/bigdatalab456422
Found 1 items
```

```
drwxr-xr-x    - bigdatalab456422 bigdatalab456422          0
2023-05-16 12:34 /user/bigdatalab456422/training
```

# copying a file

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -cp
/user/bigdatalab456422/training/newfile.txt /user/bigdatalab456422/
```

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -ls
/user/bigdatalab456422/
Found 2 items
```

```
-rw-r--r--    3 bigdatalab456422 bigdatalab456422          50
2023-05-16 12:43 /user/bigdatalab456422/newfile.txt
drwxr-xr-x    - bigdatalab456422 bigdatalab456422          0
2023-05-16 12:34 /user/bigdatalab456422/training
```

# moving file to trash using rm

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -rm
/user/bigdatalab456422/training/newfile.txt
23/05/16 12:46:15 INFO fs.TrashPolicyDefault: Moved:
'hdfs://nameservice1/user/bigdatalab456422/training/newfile.txt' to
trash at: hdfs://nameservice1/user/bigdatalab45
6422/.Trash/Current/user/bigdatalab456422/training/newfile.txt
```

# cannot delete non-empty dir using rmdir command

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -rmdir
/user/bigdatalab456422/training
rmdir: `/user/bigdatalab456422/training': Directory is not empty
```

# but we can delete empty dir using rmdir command

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -mkdir
/user/bigdatalab456422/training2
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -rmdir
/user/bigdatalab456422/training2
```

# rmr command will move the dir to trash

```
[bigdatalab456422@ip-10-1-1-204 ~]$ hadoop fs -rmr
/user/bigdatalab456422/training
rmr: DEPRECATED: Please use '-rm -r' instead.
```

```
23/05/16 13:04:53 INFO fs.TrashPolicyDefault: Moved:  
'hdfs://nameservice1/user/bigdatalab456422/training' to trash at:  
hdfs://nameservice1/user/bigdatalab456422/.Trash/  
Current/user/bigdatalab456422/training
```