Master Executive di II Livello

BIG DATA ANALYSIS AND BUSINESS INTELLIGENCE

Vamsi Krishna Varma Gunturi
Data science intern at ISTAT
vamsivarmagunturi@gmail.com

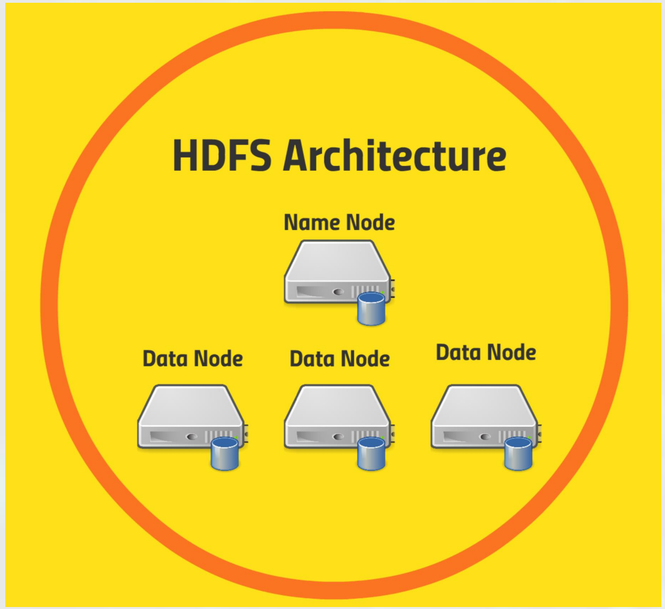
HDFS Overview

fondazione INUIT TORVERGATA

What is HDFS ?

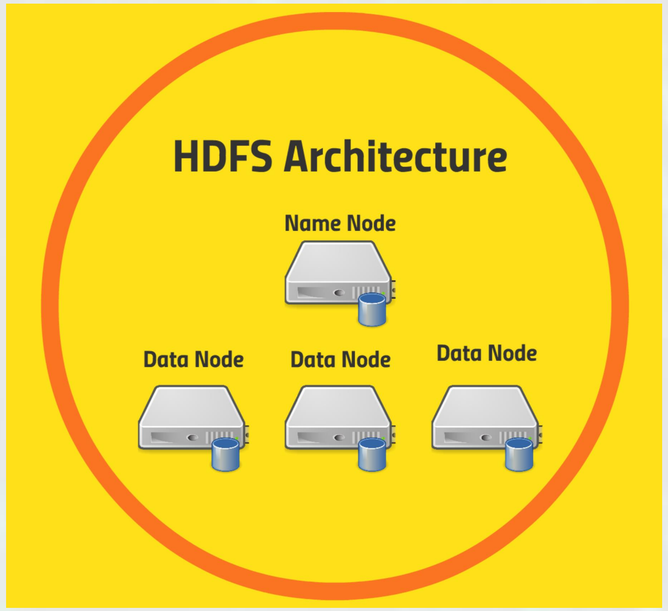
- Hadoop distributed file system which handles big files by breaking them in to small blocks stored across several commodity computers
- HDFS allows your big data to be stored across an entire cluster in a distributed manner in a reliable manner and allows your applications to analyse that data to access that data quickly and reliably.
- Each block is about 128 MB’s large so it can accommodate pretty large files
- In order to handle failure. it will actually store more than one copy of each block and so that way if one of these individual computers goes down HDFS can deal with that and actually start retrieving information from a different computer that had a backup copy of that block.

HDFS Architecture



- Based on master-slave architecture
- Name node is what keeps track of where all those blocks live. So basically it's maintaining a big old table of a given file name under some virtual directory structure in HDFS and it knows where to go to actually find every copy of every block that's associated with that file.
- Data node is where the data is actually stored and replicated

HDFS Daemons



- Daemons are the processes running in background.

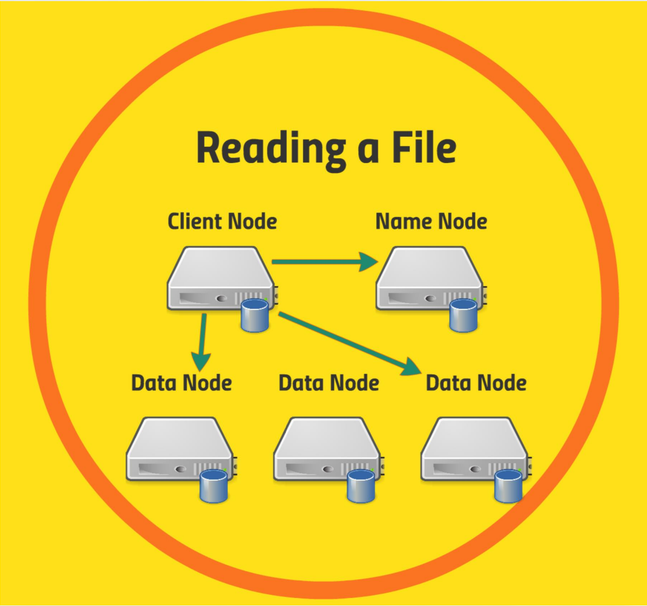
❖ Name node:

- Run on the master node.
- Store metadata (data about data) like file path, the number of blocks, block Ids. etc.
- Require high amount of RAM.
- Store meta-data in RAM for fast retrieval i.e to reduce seek time.
- Though a persistent copy of it is kept on disk.

❖ Data node:

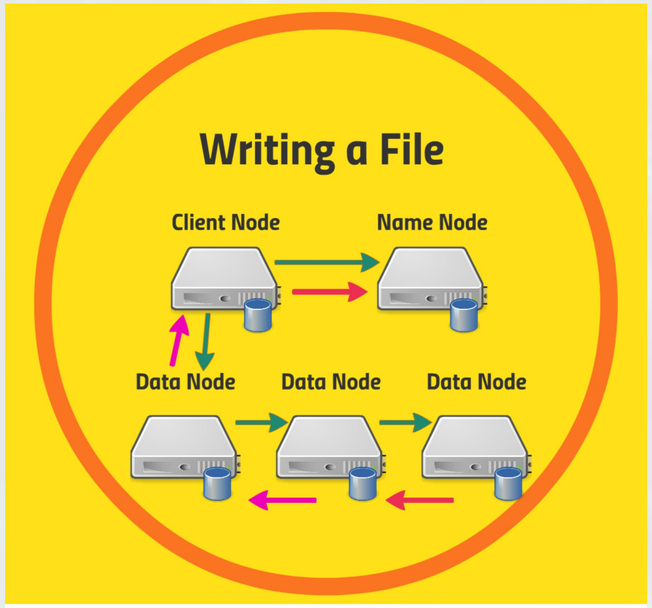
- Run on slave nodes.
- Require high memory as data is actually stored here.
- They can talk to each other for replication and data retrieval.

Reading a file



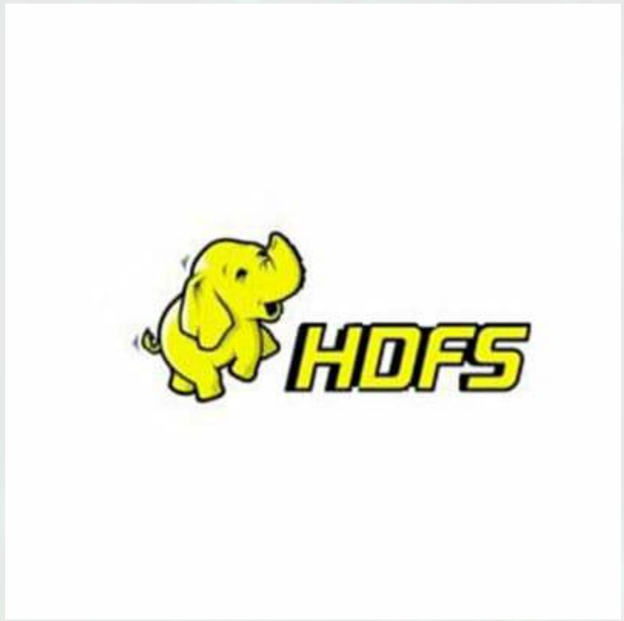
- **Step 1**: Client node would first talk to the name node
- **Step 2**: Name node figures out where that file is stored which blocks are on which data nodes and which ones are the most efficient data nodes for you to reach from your client
- **Step 3**: Finally name node directs your client to go retrieve data directly from those data nodes that it wants.

Writing a file



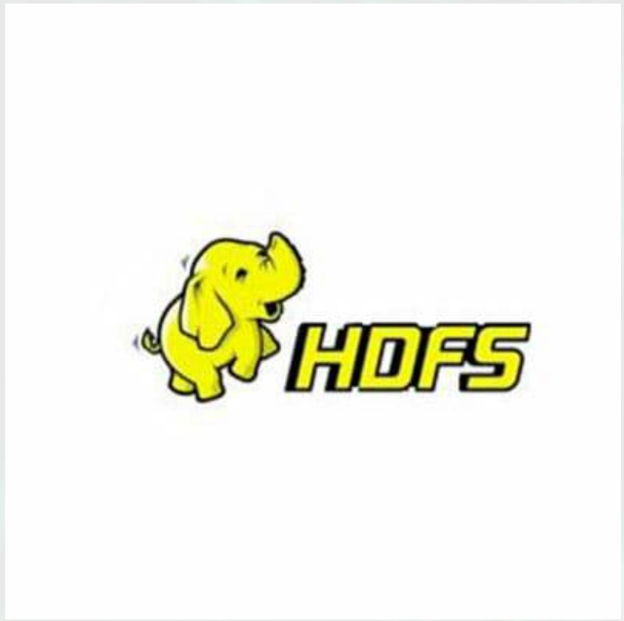
- **Step 1**: Client node first talks to the name node
- **Step 2**: Name node says OK we're going to create a new file on these data nodes
- **Step 3**: Client node talks to that first data node and the data node says OK I'm going to store this block. That block's going to be replicated on this data node. And then we're going to pass it onto this data node.
- **Step 4**: When all the data knows that we want to replicate that file have written the data successfully acknowledgements get sent back through and back up to the client node
- **Step 5**: Finally back to the name node where it can then record the fact that that file has been written successfully. And now future requests to read that file can be served.

HDFS high availability



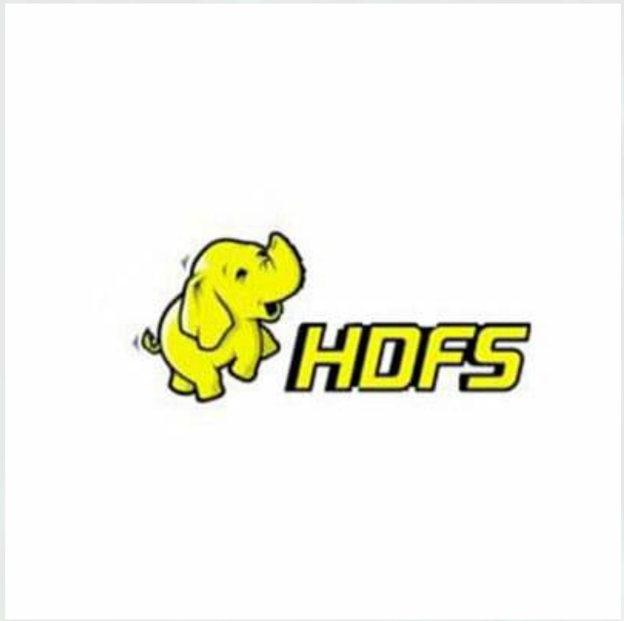
- Hot standby name node using shared edit log
- Zookeeper tracks active name node
- Uses extreme measures to ensure only one name node is used at a time

HDFS Features



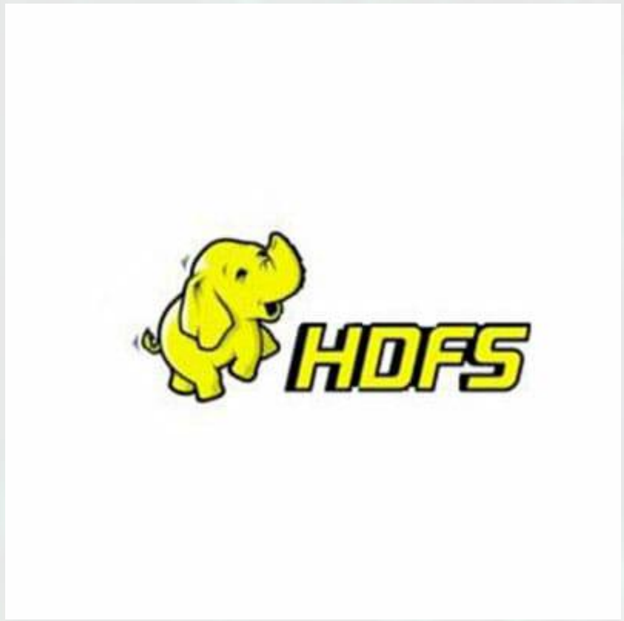
- Distributed data storage.
- Blocks reduce seek time.
- The data is highly available as the same block is present at multiple data nodes.
- Even if multiple data nodes are down we can still do our work, thus making it highly reliable.
- High fault tolerance.

HDFS Limitations



- **Low latency data access**: Applications that require low-latency access to data i.e in the range of milliseconds will not work well with HDFS, because HDFS is designed keeping in mind that we need high-throughput of data even at the cost of latency.
- **Small file problem**: Having lots of small files will result in lots of seeks and lots of movement from one data node to another data node to retrieve each small file, this whole process is a very inefficient data access pattern.

Useful HDFS Commands



- **> hdfs dfs - ** Check all the HDFS commands
- **> sbin/start-all.sh** : To start all the Hadoop services
- **> jps** : To check list of active services and their port numbers
- > **hdfs dfs -ls /** : Prints all the directories present in HDFS
- > **dfs -mkdir <folder name>** : Create a new directory in HDFS
- > **hdfs dfs -touchz <file_path>** : Creates an empty file
- > **hdfs dfs -copyFromLocal <local file path> <dest(present on hdfs)>** : To copy the files/folders from local file system to HDFS store
- > **hdfs dfs -cat <path>** : Prints the file contents

Useful HDFS Commands

- **> hdfs dfs -copyToLocal <<srcfile(on hdfs)> <local file dest>** : To copy files/folders from HDFS store to the local file system
- **> hdfs dfs -cp <src(on hdfs)> <dest(on hdfs)>** : Copy files with in HDFS
- > **hdfs dfs -rmr <filename/directoryName>** : Deletes a file from HDFS recursively
- > **hdfs dfs -du <dirName>** : Gives the size of each file in the directory
- > **hdfs dfs -setrep -R -w 6 <filename/directoryName>** : Change the replication factor of file/folder inside HDFS. By default it is 3.

Master Executive di II Livello

BIG DATA ANALYSIS AND BUSINESS INTELLIGENCE

Vamsi Krishna Varma Gunturi
Data science intern at ISTAT
vamsivarmagunturi@gmail.com

Grazie

fondazione INUIT TORVERGATA