SNS COLLEGE OF TECHNOLOGY

COIMBATORE

AN AUTONOMOUS INSTITUTION

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade

Approved by AICTE New Delhi & affiliated to the Anna University, Chennai

DEPARTMENT OF MCA

Course Name : 16CAT702 - BIG DATA ANALYTICS

Class : II Year / III Semester

Unit II - Introduction

Topic II – Hadoop Distributed File System(HDFS)

HDFS

- With growing data velocity the data size easily outgrows the storage limit of a machine.
- A solution would be to store the data across a network of machines. Such filesystems are called *distributed filesystems*.
- It provides one of the most reliable filesystems. HDFS (Hadoop Distributed File System) is a unique design that provides storage for *extremely large files* with streaming data access pattern and it runs on *commodity hardware*.

Hadoop Distributed File System(HDFS)

Extremely large files: Here we are talking about the data in range of petabytes(1000 TB).

Streaming Data Access Pattern: HDFS is designed on principle of *write-once and read-many-times*. Once data is written large portions of dataset can be processed any number times.

Commodity hardware: Hardware that is inexpensive and easily available in the market. This is one of feature which specially distinguishes HDFS from other file system.

HDFS

- **Nodes:** Master-slave nodes typically forms the HDFS cluster.

NameNode(MasterNode):

- Manages all the slave nodes and assign work to them.
- It executes filesystem namespace operations like opening, closing, renaming files and directories.
- It should be deployed on reliable hardware which has the high config. not on commodity hardware.

DataNode(SlaveNode):

- Actual worker nodes, who do the actual work like reading, writing, processing etc.
- They also perform creation, deletion, and replication upon instruction from the master.
- They can be deployed on commodity hardware.

HDFS

HDFS daemons: Daemons are the processes running in background.

Namenodes:

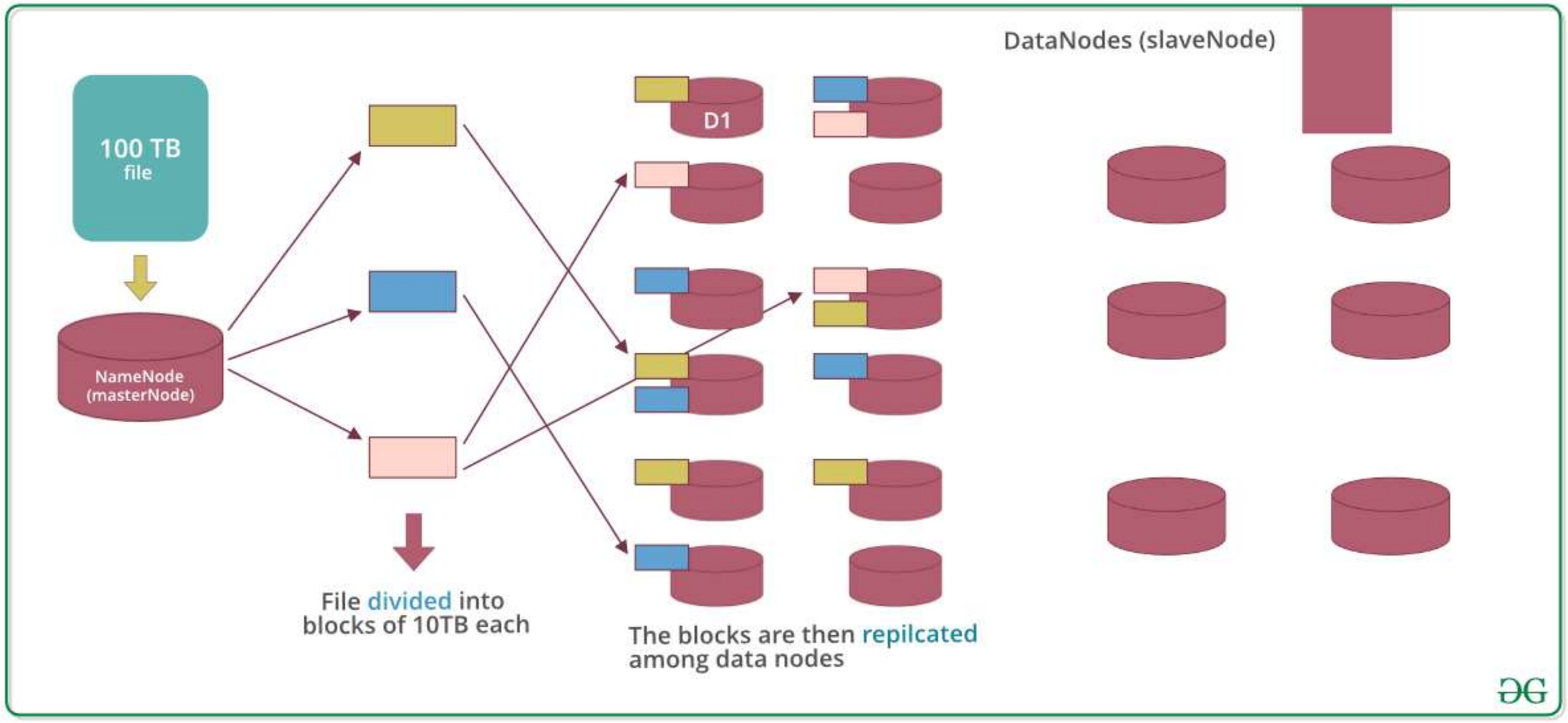
- Run on the master node.
- Store metadata (data about data) like file path, the number of blocks, block Ids. etc.
- Require high amount of RAM.
- Store meta-data in RAM for fast retrieval i.e to reduce seek time. Though a persistent copy of it is kept on disk.

DataNodes:

- Run on slave nodes.
- Require high memory as data is actually stored here.
- Data storage in HDFS: Now let's see how the data is stored in a distributed manner.

Data storage in HDFS

Data storage in HDFS: Now let's see how the data is stored in a distributed manner.



Data storage in HDFS

- Lets assume that 100TB file is inserted, then masternode(namenode) will first *divide* the file into blocks of 10TB (default size is *128 MB* in Hadoop 2.x and above).
- Then these blocks are stored across different datanodes(slavenode). Datanodes(slavenode)*replicate* the blocks among themselves and the information of what blocks they contain is sent to the master. Default replication factor is *3* means for each block 3 replicas are created (including itself). In hdfs.site.xml we can increase or decrease the replication factor i.e we can edit its configuration here.

- **Note:** MasterNode has the record of everything, it knows the location and info of each and every single data nodes and the blocks they contain, i.e. nothing is done without the permission of masternode.

Why divide the file into blocks?

Let’s assume that we don’t divide, now it’s very difficult to store a 100 TB file on a single machine. Even if we store, then each read and write operation on that *whole file* is going to take very high seek time.

But if we have multiple blocks of size 128MB then its become easy to perform various read and write operations on it compared to doing it on a whole file at once. So we divide the file to have faster data access i.e. reduce seek time.

Why replicate the blocks in data nodes while storing?

Let's assume we don't replicate and only one yellow block is present on datanode D1. Now if the data node D1 crashes we will lose the block and which will make the overall data inconsistent and faulty. So we replicate the blocks to achieve *fault-tolerance.*

Terms related to HDFS:

HeartBeat : It is the signal that datanode continuously sends to namenode. If namenode doesn't receive heartbeat from a datanode then it will consider it dead.

Balancing : If a datanode is crashed the blocks present on it will be gone too and the blocks will be *under-replicated* compared to the remaining blocks. Here master node(namenode) will give a signal to datanodes containing replicas of those lost blocks to replicate so that overall distribution of blocks is balanced.

Replication:: It is done by datanode.

Note: No two replicas of the same block are present on the same datanode.

HDFS

Features:

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

HDFS

Limitations: Though HDFS provide many features there are some areas where it doesn't work well.

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 datanode to another datanode to retrieve each small file, this whole process is a very inefficient data access pattern.

Reference

1. https://www.javatpoint.com/what-is-hadoop

THANK YOU