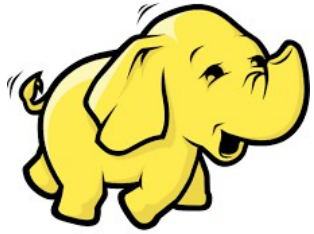
HDFS Overview

Tushar B. Kute,
http://tusharkute.com



HDFS

- Hadoop File System was developed using distributed file system design.
- It is run on commodity hardware. Unlike other distributed systems, HDFS is highly fault-tolerant and designed using low-cost hardware.
- HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines.
- These files are stored in redundant fashion to rescue the system from possible data losses in case of failure.
- HDFS also makes applications available to parallel processing.

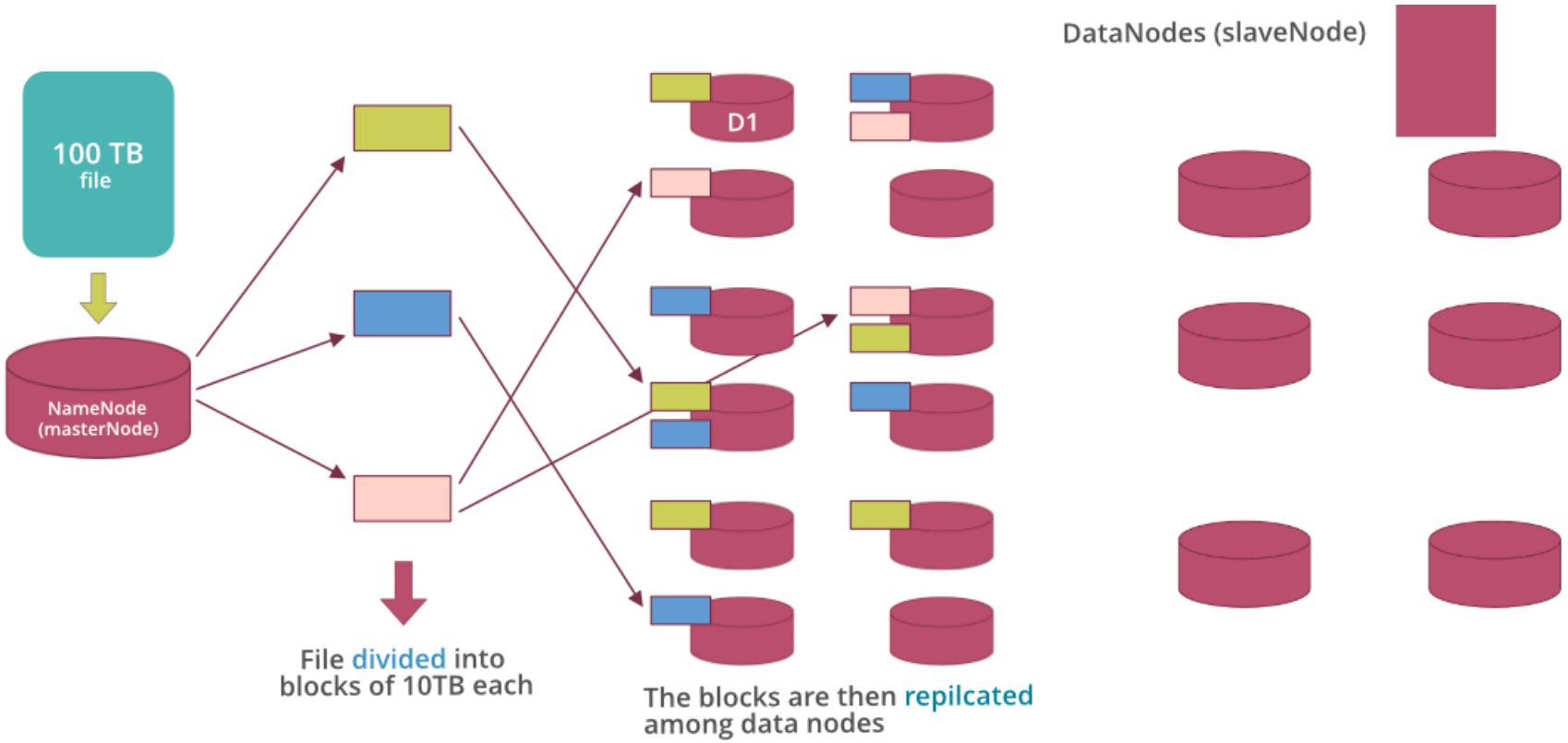
Features of HDFS

- It is suitable for the distributed storage and processing.
- Hadoop provides a command interface to interact with HDFS.
- The built-in servers of namenode and datanode help users to easily check the status of cluster.
- Streaming access to file system data.
- HDFS provides file permissions and authentication

HDFS Terminology

- Let’s elaborate the terms:
  - 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 Architecture



Architecture

- 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 1 means for each block 1 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.

Namenode

- The namenode is the commodity hardware that contains the GNU/Linux operating system and the namenode software.
- It is a software that can be run on commodity hardware.
- The system having the namenode acts as the master server and it does the following tasks:
  - Manages the file system namespace.
  - Regulates client's access to files.
  - It also executes file system operations such as renaming, closing, and opening files and directories.

Datanode

- The datanode is a commodity hardware having the GNU/Linux operating system and datanode software. For every node (Commodity hardware/ System) in a cluster, there will be a datanode.
- These nodes manage the data storage of their system.
  - Datanodes perform read-write operations on the file systems, as per client request.
  - They also perform operations such as block creation, deletion, and replication according to the instructions of the namenode.

Block

- Generally the user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual data nodes.
- These file segments are called as blocks. In other words, the minimum amount of data that HDFS can read or write is called a Block.
- The default block size is 128MB, but it can be increased as per the need to change in HDFS configuration.

Block

- 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.

Replication

- 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.

Goals of HDFS

- Fault detection and recovery: Since HDFS includes a large number of commodity hardware, failure of components is frequent. Therefore HDFS should have mechanisms for quick and automatic fault detection and recovery.
- Huge datasets: HDFS should have hundreds of nodes per cluster to manage the applications having huge datasets.
- Hardware at data: A requested task can be done efficiently, when the computation takes place near the data. Especially where huge datasets are involved, it reduces the network traffic and increases the throughput.

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

Thank you

This presentation is created using LibreOffice Impress 5.1.6.2, can be used freely as per GNU General Public License

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com