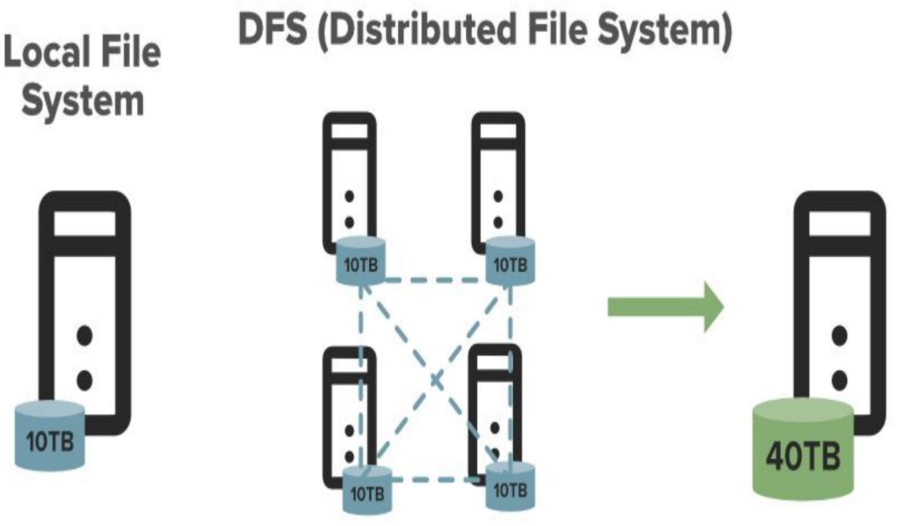
HDFS

What is a Local File System

- The file system is a kind of Data structure which we use in an operating system to manage file on disk space. This means it allows the user to keep maintain and retrieve data from the local disk.
- Windows File Systems
  - NTFS(New Technology File System)
  - FAT32(File Allocation Table 32). FAT32 is used in some older versions of windows but can be utilized on all versions of *windows xp*.
- Linux File Systems
  - ext3, ext4 etc.

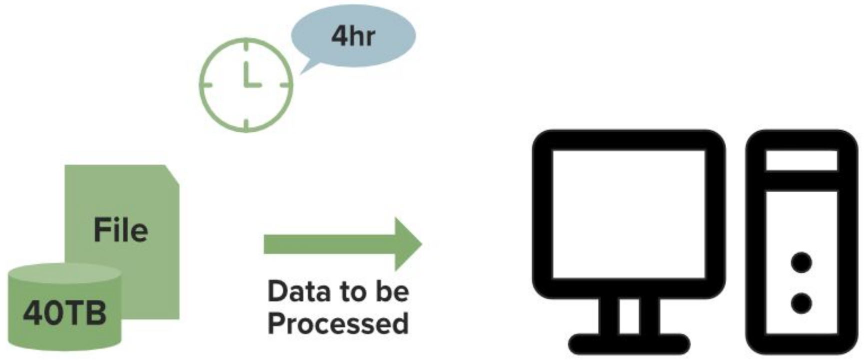
What is a DFS

- DFS stands for the distributed file system, it is a concept of storing the file in multiple nodes in a distributed manner.
- DFS actually provides the Abstraction for a single large system whose storage is equal to the sum of storage of other nodes in a cluster.

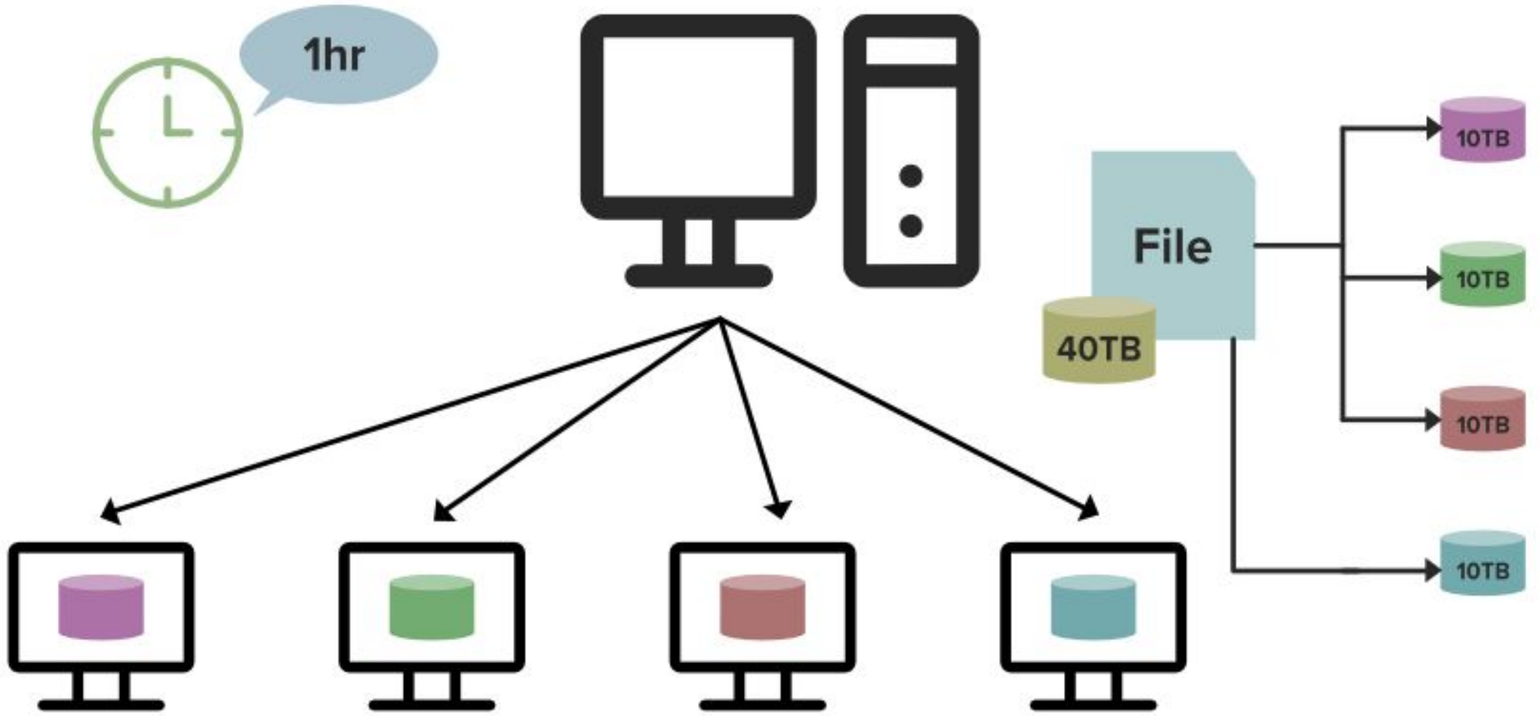


Why need DFS

- The disk capacity of a system can only increase up to an extent. If somehow you manage the data on a single system then you'll face the processing problem, processing large datasets on a single machine is not efficient.



Why need DFS



Goals and Assumptions of HDFS

- Originally built as infrastructure for ***Apache Nutch*** web search engine
- Distributed File system to manage ***extremely large files*** and do ***computation*** on them on ***commodity hardwares***
- Resource Requirements
  - Extremely Large Files (we are talking about petabytes : 1000 TBs)
  - Perform efficient Computation ((e.g. per word frequency count on these files)
  - Commodity Hardware (around ½ GB)
- Functional Goals
  - Fault tolerant
  - High Throughput even at the cost of latency

Assumptions

- Hardware Failures
- Large Data Sets
- Simple Coherency Model
  - Write once Read many
  - No file modifications allowed. Only delete, create (append in future)
- Portabiity across Heterogenous Hardware and Software
  - Intel, AMD, Windows, Linux etc.
- Moving Computation is more efficient than moving data

- Lets see a working example

HDFS Architecture

- HDFS Cluster
  - MasterNode (aka NameNode)
  - SlaveNode (aka DataNodes)
- HDFS Daemons
  - Background Processes running on HDFS cluster machines
  - MasterDaemons
  - SlaveDaemons
- Data Balancing
  - Balancing of Data during data loss due to node crash or disk crash
- HeartBeat
  - DataNode to NameNode. Absence of heartbeat for a prolonged time implies death of the datanode
- Replication : Done by DataNode

HDFS Cluster

HDFS Cluster

- **Scalability:** Hadoop clusters can easily scale up or down by adding or removing nodes. For example, if data grows from 5PB to 7PB, more servers can be added to handle it.
- **Flexibility:** Hadoop can store and process all types of data structured, semi-structured or unstructured making it highly adaptable.
- **Speed/Parallelization:** Due to its distributed nature and use of MapReduce, Hadoop processes data quickly across multiple nodes.
- **Less likelihood of Data Loss:** Hadoop keeps multiple copies of data across nodes. If one node fails, the data is still safe on another.
- **Economical:** Hadoop runs on low-cost commodity hardware, making it a budget-friendly solution for big data storage and processing.

HDFS Cluster

- Nodes ⇒ Collection of individual hardwares

- MasterNode
  - *Include the NameNode (handles file system metadata) and the ResourceManager (manages cluster resources).*

- SlaveNode
  - *Include DataNodes (store actual data) and NodeManagers (manage execution of tasks).*

HDFS Cluster Nodes

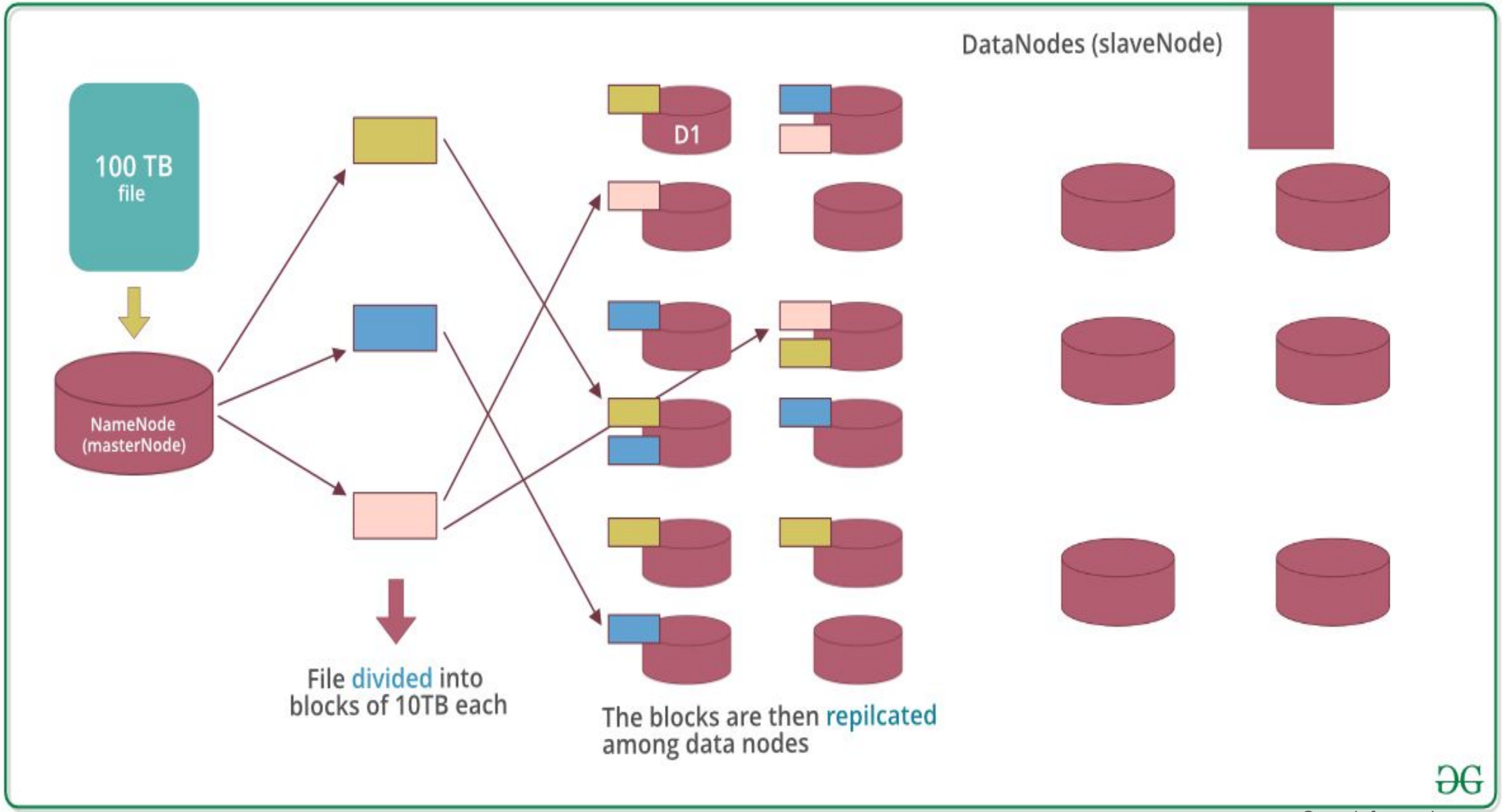
- MasterNode
  - Requires reliable hardware less prone to failures
    - In Hadoop 2.0 you have at least 2 MNs Active and Standby NameNode
  - Executes FS operations like opening, closing, renaming of files and directories
  - Manages Slaves Nodes & Assigns work to them
- SlaveNode
  - Actual worker nodes 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.

Types of HDFS Cluster

- Single Node Hadoop Cluster
    - In a single node cluster, all Hadoop components (NameNode, DataNode, ResourceManager, NodeManager, etc.) run on the same machine. This setup uses a single JVM (Java Virtual Machine) and is mainly used for learning or testing purposes.
- Multi Node HDFS Cluster
    - A multi node cluster has multiple machines (nodes). Master daemons like NameNode and ResourceManager run on powerful systems, while slave daemons like DataNode and NodeManager run on other, often less powerful, machines. This setup is used in real-world, large-scale data processing.

Data Storage

- Lets take an example of how a 100 TB file is stored in HDFS Cluster
- Request comes to NameNode/MasterNode
- NameNode Divides it into blocks of 10 TB
- NameNode then sends these 10 blocks to 10 Different Data Nodes. These 10 data nodes are picked up based on some criteria
- Data Nodes store these blocks along with information which is sent back once it is done
- NameNode does the same based on replication factor
- Note : NameNode has record of everything, it knows location fo every single file along with its blocks and the datanode which holds it. It si stored in teh metadata file


DataNodes (slaveNode)
100 TB file
NameNode (masterNode)
D1
File divided into blocks of 10TB each
The blocks are then repilcated among data nodes
ƏG

HDFS Blocks

- In HDFS files are broken into units of blocks of fixed size in NameNode

- These Blocks are then sent to DataNodes for storing and replication

- Size of the Block Size is 64 MB by default and is configurable
    - Industry Standard is 128MB

Rationale Behind Large Block Size

- Disk Access TIme :

  Sector-Search (87%) + Transfer Time (0.5%) + Overhead

- *To minimize sector search time block sizes needs to be increased.*
- Data Transfer time of random 128 MB assuming 1K sector size

  16.7ms * 128 MB / 1K ⇒ 16.7 * 128*1000*K/1K ⇒ 16.7*128*1000 ⇒ ***2137 s***

- Data Transfer time of sequential 128 MB where sectors are contiguous

  (9+5.55) + 0.1*128MB/1K ⇒ 14.55 + 0.1*128*1000 = 14.55+12800 = ***12.8 s***

  166x latency reduction!!!

HDFS Daemons

HDFS Daemons

- MasterDaemons
  - ***NameNode***
  - Resource Manager
- Slave Daemons
  - ***Date Node***
  - Node Manager

HDFS Daemons (Name Node)

NameNode works on the Master System. The primary purpose of Namenode is to manage all the MetaData. Metadata is the list of files stored in HDFS(Hadoop Distributed File System). As we know the data is stored in the form of blocks in a Hadoop cluster. So the DataNode on which or the location at which that block of the file is stored is mentioned in MetaData. All information regarding the logs of the transactions happening in a Hadoop cluster (when or who read/wrote the data) will be stored in MetaData. MetaData is stored in the memory.

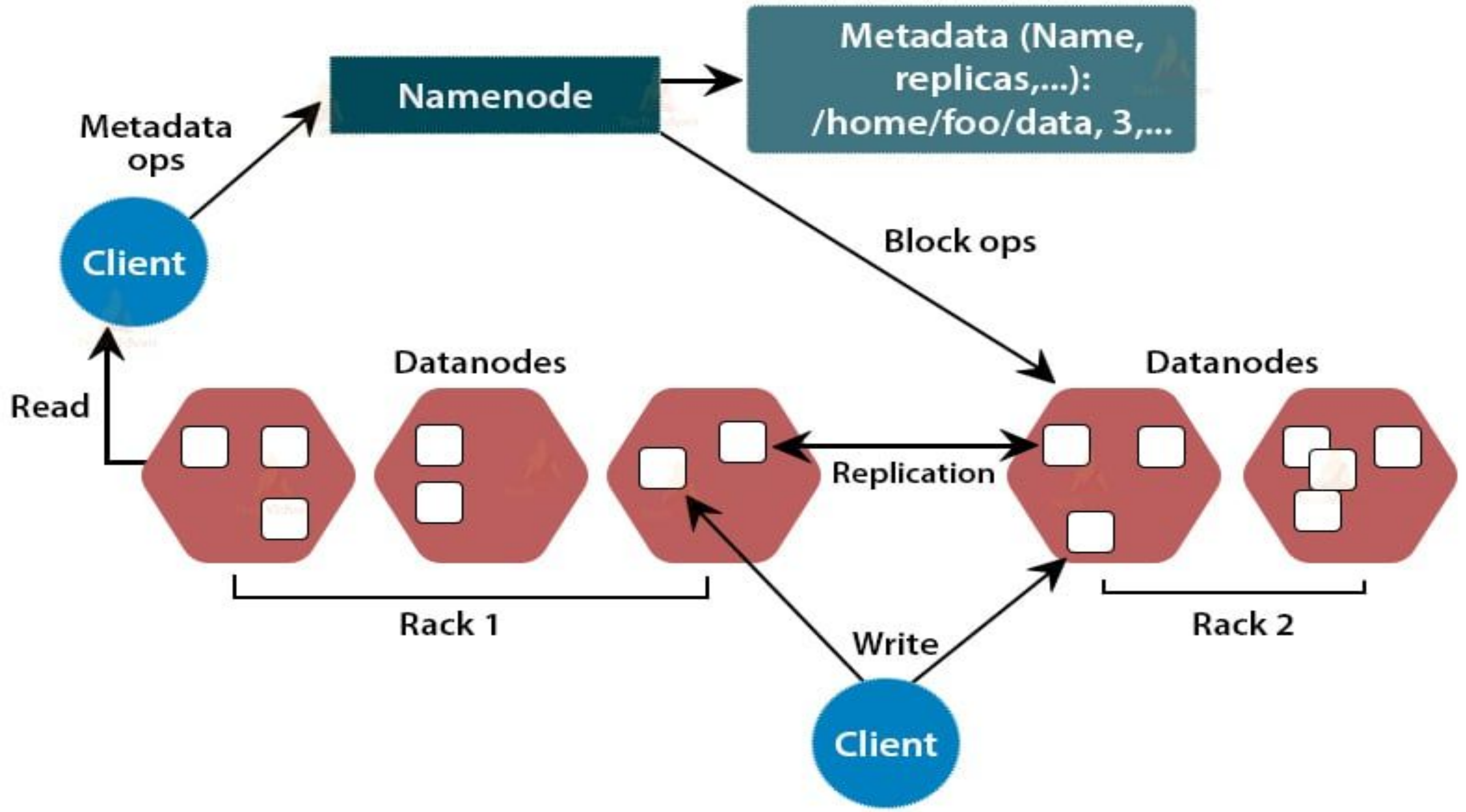
- It never stores the data that is present in the file.
- It stores the information of DataNode such as their Block id's and Number of Blocks
- Stores meta-data in RAM for fast retrieval i.e to reduce seek time. Though a persistent copy of it is kept on disk.
- Requires a high amount of RAM.

HDFS Daemon (Data Node)

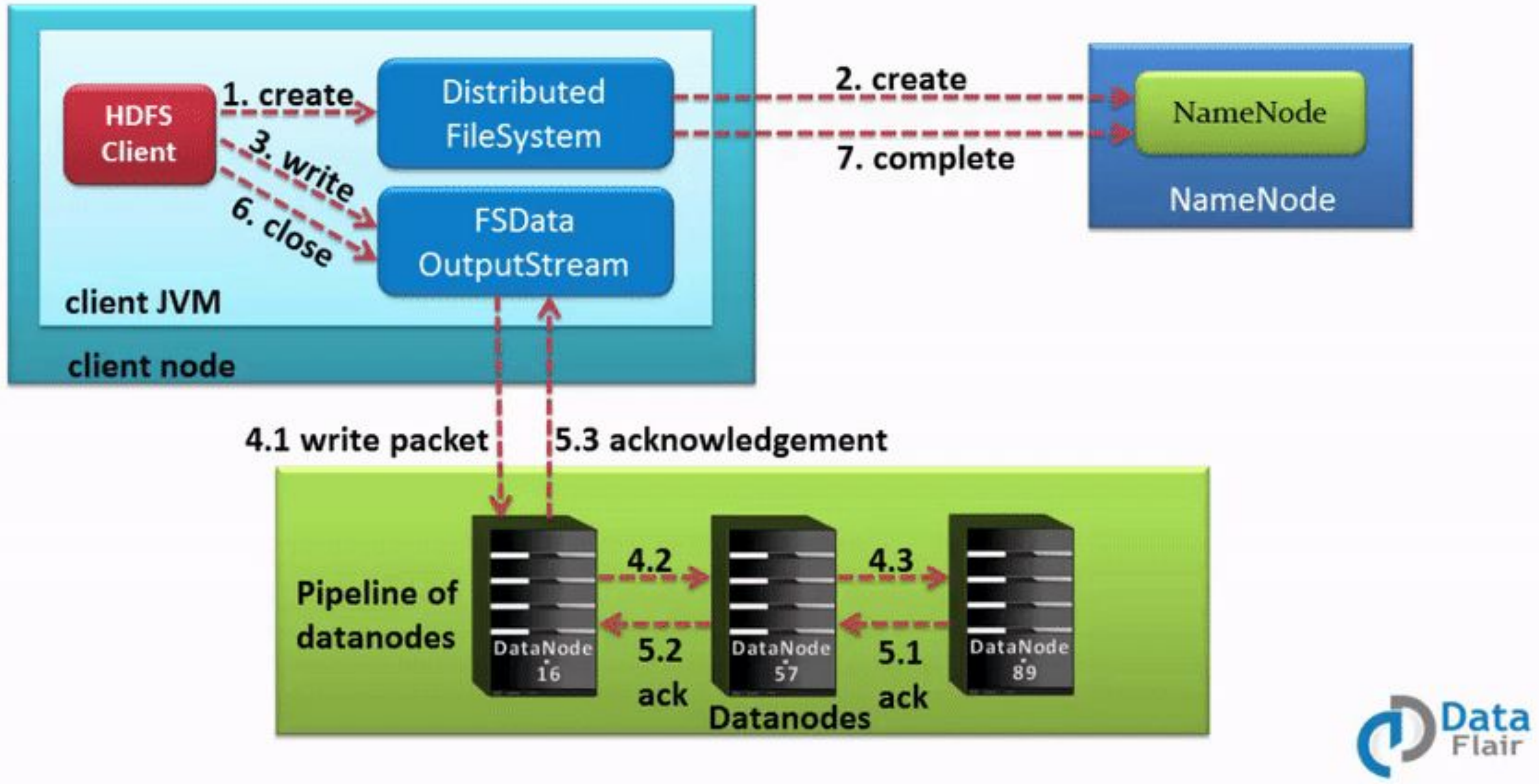
DataNode process works on the Slave Nodes. The NameNode always instructs DataNode for storing the Data. DataNode is a program that runs on the slave system that serves the read/write request from the client. As the data is stored in this DataNode, they should possess high memory to store more Data.

- DataNodes:
  - Run on ***slave*** nodes.
  - Require high memory as data is actually stored here.

HDFS Architecture



HDFS File Write



HDFS Client
1. create
Distributed FileSystem
3. write
6. close
FSData OutputStream
client JVM
client node
2. create
7. complete
NameNode
NameNode
4.1 write packet
5.3 acknowledgement
Pipeline of datanodes
DataNode 16
4.2
5.2 ack
DataNode 57
4.3
5.1 ack
DataNode 89
Datanodes
Data Flair