UNIT III- HADOOP

QUESTION BANK

PART A (2 Marks each)

1. Briefly discuss about the history of Hadoop. Imp
2. List out the companies used by Hadoop.
3. Name the two Specialist Hadoop companies.
4. List out the vendors established by Hadoop.
5. List out the modules of Apache Hadoop framework. Imp
6. Define NameNode. Imp
7. Define DataNode. Imp
8. Explain the Job Scheduling in MapReduce. Imp
9. Define shuffle or sort. Imp

PART B (5 Marks each)

10. Explain the features of Hadoop. Imp
11. Explain HDFS features. Imp
12. Explain the components of Hadoop. **V.Imp**
13. Explain scaling out. Imp
14. What is Hadoop streaming? Explain. Imp
15. Explain Java Interfaces to HDFS. Imp
16. Discuss the failures of MapReduce. Imp
17. Explain MapReduce types and formats. **V.Imp**
18. Explain the features of MapReduce. Imp

PART C (15 Marks each)

19. Explain in detail the architecture of HDFS. **V.V.Imp**
20. Explain how analysing the data with Hadoop. **OR** Explain the working of MapReduce in detail. **OR** Explain the anatomy of MapReduce in detail. **V.V.Imp** *(Same Answer)*
21. Explain the design of HDFS. **OR** Explain the features of HDFS. **V.V.Imp** *(Same Answer)*

NOTES

HISTORY OF HADOOP

- Created by **Doug Cutting**.
- He was the creator of **Apache Lucene** (Widely used text search library).
- **Apache Nutch** (Open-Source Web Search Engine) was started in 2002. It was a part of Lucene Project.
- The origin of Hadoop was **Apache Nutch** in 2006.
- Hadoop is a made-up name.
- This name given by Doug Cutting's kid to a yellowed elephant.

- Projects in Hadoop ecosystem have names are unrelated to their function, often with an elephant or other animal names.
- 2004: - Nutch Developers set about writing an open-source implementation called **Nutch Distributed File System (NDFS)**.
- 2004:- Google published a paper introduced MapReduce.
- 2008 January:- Hadoop made its own top-level project at Apache.
- At that time Hadoop was used by many other **companies** like:
  - ✓ Yahoo!
  - ✓ Facebook
  - ✓ New York Times
- April 2008:- **Hadoop got an world record of fastest system to sort an entire terabytes of data.**
- **Running on 910 node cluster, Hadoop sorted 1Tb in 209 seconds (3.5 minutes).**

November 2008	April 2009
Google	Yahoo!
MapReduce Implementation	Hadoop
Sorted 1Tb in 68 sec.	Sorted same in 62 sec.

- **Widely used in Mainstream enterprises.**
- **Hadoop is used for general purpose storage & analysis platform for big data**.
- Hadoop support established **vendors** like:
  - ✓ EMC: Electronic Membership Corporation
  - ✓ IBM
  - ✓ Microsoft
  - ✓ Oracle
- Specialist **Hadoop companies** are:
  - ✓ Cloudera
  - ✓ Horton Works

FEATURES OF HADOOP

1) **Storage & process Big Data**: - Process Big Data of 3V characteristics.
2) **Open-Source Framework**:-
   - Open-Source access & cloud services enable large data stores.
   - Hadoop uses a cluster of multiple inexpensive servers of cloud.
3) **Java & Linux based**:-
   - Hadoop uses Java Interfaces.
   - Hadoop base is Linux.
   - Hadoop has its own set of shell commands supports.
4) **Fault-efficient scalable**:-
   - System provides servers at high scalability
   - System provides scalable by adding new nodes to handles large data.
5) **Flexible modular design**:-
   - Simple & modular programming.
   - Hadoop very helpful in:
     - ✓ Storing

- ✓ Manipulating
- ✓ Processing
- ✓ Analysing of Big Data

- Modular functions make system flexible.
- One can add or replace components at ease.

6) **Robust design of HDFS**:-
   - Execution of big data application continue even individual server or cluster fails.
   - Because Hadoop helps for backup & data recovery mechanism.
   - HDFS has high reliability.
7) **Hardware fault tolerant**:-
   - A fault does not affect data & application processing.
   - If a node goes down, other nodes takes care.
   - This is due to multiple copies of all data blocks which replicate automatically.
   - Default 3 copies of data blocks.

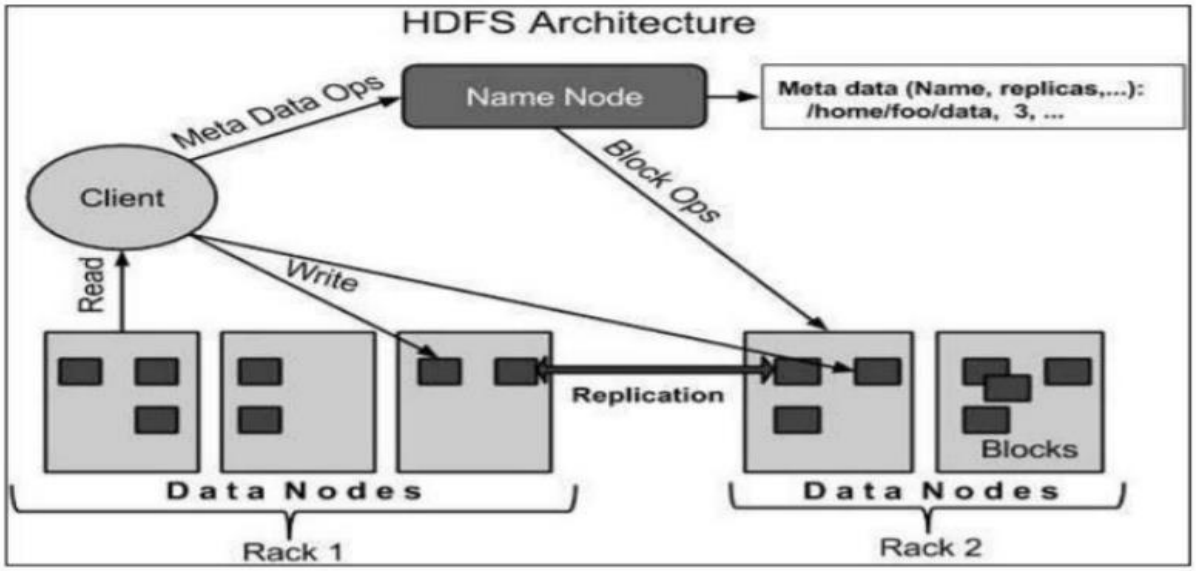
MODULES OF APACHE HADOOP FRAMEWORK

1) **Hadoop common**:
   - Contains libraries & utilities needed by other Hadoop components.
2) **HDFS**:
   - It is distributed file system.
   - Store data on machines.
   - Providing very high aggregate bandwidth across clusters.
3) **Hadoop YARN**:
   - Introduced in 2012.
   - Platform responsible for managing computing resources in clusters & using them for scheduling users' applications.
4) **Hadoop MapReduce**:
   - Implementation of MapReduce programming model for large scale data processing.

HADOOP DISTRIBUTED FILE SYSTEM (HDFS)

- Big Data analytics applications are software applications that make use of large-scale data.
- The applications **analyse** Big Data using **massive parallel processing frameworks**.
- **HDFS** is a **core component of Hadoop**.
- HDFS is **designed to run on a cluster of computers** and **servers at cloud-based utility services**.
- HDFS **stores Big Data** which may **range from GBs to PBs**.
- HDFS **stores the data in a distributed manner** in order to **compute fast**.
- The distributed data store in **HDFS stores data in any format regardless of schema**.
- **HDFS provides high throughput** access to data-centric applications that require large-scale data processing workloads.

- Figure 3-1 shows the HDFS architecture.



NAMENODE

- The name node is the commodity **hardware** that contains the **GNU/Linux operating system** and the name node software.
- It is a software that can be run on commodity hardware.
- The system having the name node 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 data node is a commodity hardware having the **GNU/Linux operating system** and data node software.
- For every node (Commodity hardware/System) in a cluster, there will be a data node.
- 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 **blocks**.

- In other words, the minimum amount of data that HDFS can read or write is called a Block.
- **The default block size** is **64MB**, but it can be increased as per the need to change in HDFS configuration.
- Data at the stores enable running the distributed applications including analytics, data mining, OLAP using the clusters.
- Hadoop **HDFS features** are as follows:
  - Create, append, delete, rename and attribute modification functions
  - Content of individual files cannot be modified or replaced but appended with new data at the end of the file.
  - It is suitable for **distributed storage and processing**.
  - Hadoop provides a command interface to interact with HDFS
  - The built-in servers of name node and data node help users to easily check the status of the cluster.
  - Streaming access to file system data
  - HDFS provides file permissions and authentication.

COMPONENTS OF HADOOP

- **Apache Pig** :
  - ✓ A software for analysing large data sets that consists of a high-level language similar to SQL for expressing data analysis programs, coupled with infrastructure for evaluating these programs.
  - ✓ It contains a compiler that produces sequences of Map- Reduce programs.
- **HBase:**
  - ✓ A **non-relational columnar distributed database** designed to run on top of Hadoop Distributed File system (HDFS).
  - ✓ It is written in Java and modelled after Google's Big Table.
  - ✓ HBase is an example of a NoSQL data store.
- **Hive:**
  - ✓ It is a Data warehousing application that provides the SQL interface and relational model.
  - ✓ Hive infrastructure is built on the top of Hadoop that help in providing summarization, query and analysis.
- **Cascading** :
  - ✓ A software abstraction layer for Hadoop, intended to hide the underlying complexity of Map Reduce jobs.
  - ✓ Cascading allows users to create and execute data processing workflows on Hadoop clusters using any JVM based language.
- **Avro:**
  - ✓ A data serialization system and data exchange service.
  - ✓ It is basically used in Apache Hadoop.
  - ✓ These services can be used together as well as independently.

- **Big Top**:
  - ✓ It is used for packaging and testing the Hadoop ecosystem.
- **Oozie**:
  - ✓ Oozie is a java based web-application that runs in a java servlet.
  - ✓ Oozie uses the database to store a definition of Workflow that is a collection of actions.
  - ✓ It manages the Hadoop jobs.

ANALYSING THE DATA WITH HADOOP

- Hadoop can be used for data analysis technologies to analyse the huge stock data being generated very frequently.
- Important **Hadoop data analysis technology is MapReduce.**

MapReduce

- **MapReduce is a framework** in which we can write applications to **process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable** manner.
- **MapReduce is a processing technique** and a program model for distributed computing **based on java**.
- The MapReduce algorithm contains two **important tasks**, namely **Map** and **Reduce**.
- **Map takes a set of data and converts it into another set of data**, where individual **elements are broken down into tuples (key/value pairs**).
- Secondly, **reduce the task**, which takes the **output from a map as an input** and **combines** those **data tuples into a smaller set of tuples**.
- As the sequence of the name MapReduce implies, the **reduce task is always performed after the map job**.
- The major **advantage** of MapReduce is that it is **easy to scale data processing** over multiple computing nodes.
- Under the MapReduce model, the **data processing primitives** are called **mappers** and **reducers**.
- Once we **write an application in the MapReduce form**, **scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change.**
- This **simple scalability** is what has **attracted many programmers** to use the MapReduce model.

Algorithm

- MapReduce program executes in **three stages**
  1. Map stage
  2. Shuffle stage
  3. Reduce stage

Map Stage

- The **map** or **mapper's job is to process the input data**.
- Generally the **input data is in the form of file or directory** and is **stored in the Hadoop filesystem (HDFS)**.
- The **input file is passed to the mapper function line by line**.
- The **mapper processes the data** and **creates several small chunks of data**.

Reduce Stage

- This stage is the **combination of the Shuffle stage and the Reduce stage**.
- The **Reducer's job is to process the data that comes from the mapper.**
- **After processing**, it **produces a new set of output**, which will be **stored in the HDFS**.
- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
- Most of the **computing takes place on nodes** with **data on local disks** that **reduces the network traffic**.
- **After completion of the given tasks**, the cluster collects and **reduces the data** to form an **appropriate result**, and **sends it back to the Hadoop server.**
- The MapReduce framework operates on pairs, the framework views the input to the job as a set of pairs and produces a set of pairs as the output of the job.

SCALING OUT

- To scale out, we need to store the data in a distributed file system (typically HDFS).
- This allows Hadoop to move the MapReduce computation to each machine hosting a part of the data, using Hadoop's resource management system, called **YARN**.
- A MapReduce job is a unit of work that the client wants to be performed.
- It consists of the input data, the MapReduce program, and configuration information.
- Hadoop runs the job by dividing it into tasks, of which there are two types: map tasks and reduce tasks.
- The tasks are scheduled using YARN and run on nodes in the cluster.
- If a task fails, it will be automatically rescheduled to run on a different node.

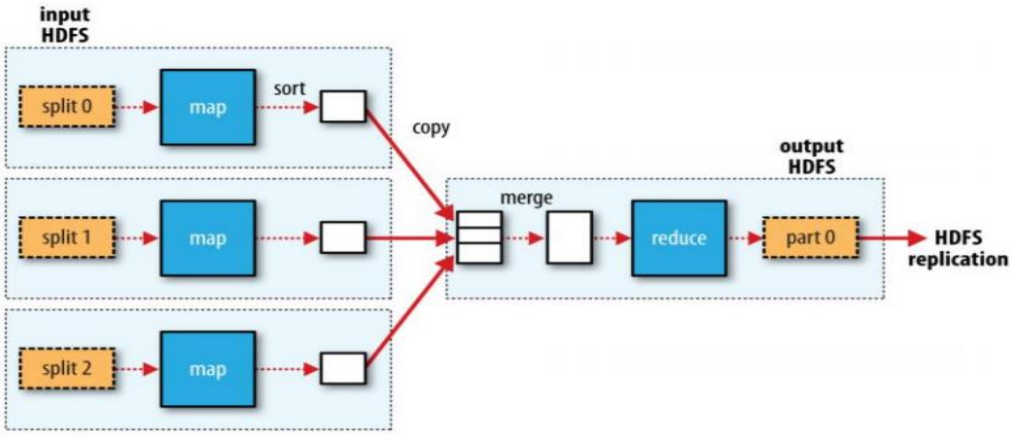


Figure MapReduce data flow with a single reduce task

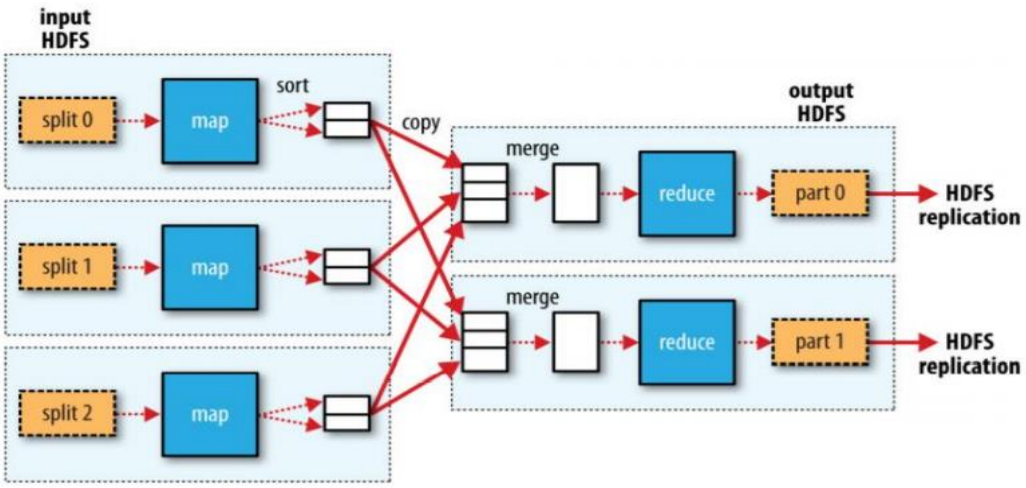
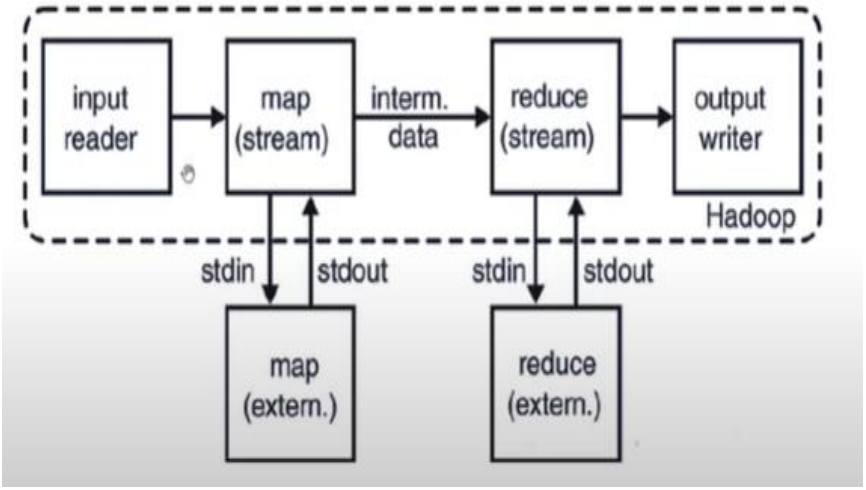


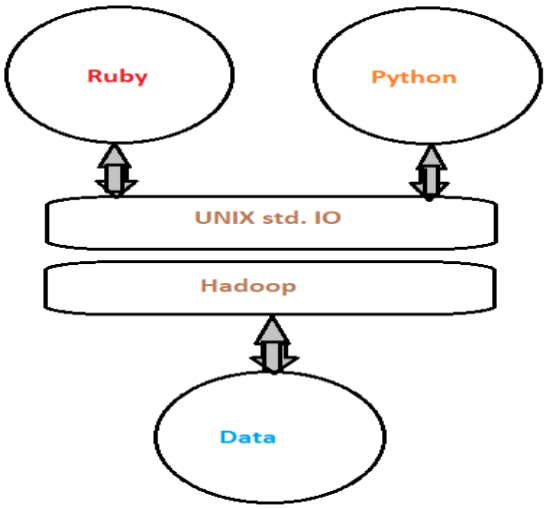
Figure MapReduce data flow with multiple reduce tasks

- Hadoop divides the input to a MapReduce job into fixed-size pieces called input splits, or just splits.
- Hadoop creates one map task for each split, which runs the user-defined map function for each record in the split.
- The output of the reduce is normally stored in HDFS for reliability.
- For each HDFS block of the reduce output, the first replica is stored on the local node, with other replicas being stored on off-rack nodes for reliability.

HADOOP STREAMING

- The **core ideology of Hadoop** is the **data processing should be independent of the language**. i.e., it is **flexible** because the **programs can be designed in any languages** to do the processing.
- Hadoop streaming is an **ability of Hadoop** to **interface with Map & Reduce programs written in Java/ Non-Java** like Ruby, PHP, C++, Python etc.
- Hadoop streaming uses **Unix standard streams** as the **interface between Hadoop** and our **program**.





ADVANTAGES OF HADOOP STREAMING

- Availability: **No separate Software is need** to install.
- Learning: Doesn't require anything new to learn concept/technology.
- Faster development time
- Faster conversion: Take less time to convert the data from one format to another format.
- No need for non-Java code developer to learn Java language.

DISADVANTAGES OF HADOOP STREAMING

- Not good in performance
- Suited for handling data represented in text.
- Uses excessive amount of RAM.

DESIGN OF HDFS

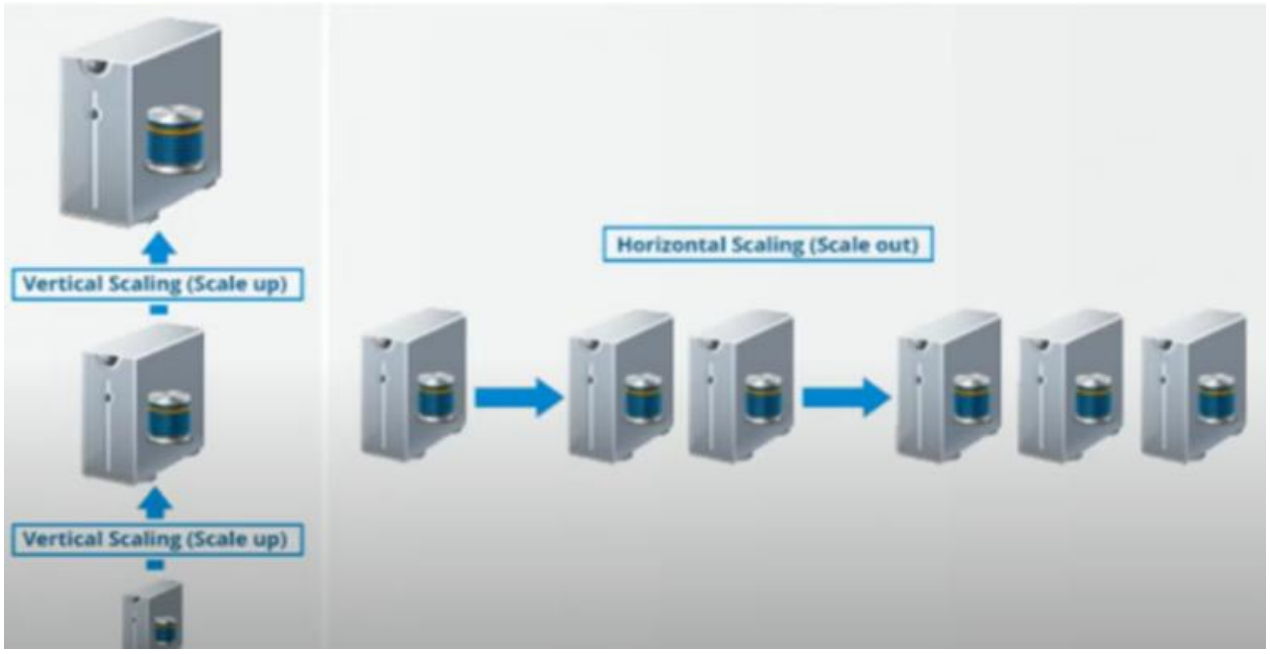
- The HDFS is designed for big data processing.
- It is a core part of Hadoop, which is used for data storage.
- It is designed to run on commodity hardware. i.e., to run HDFS, we don't have specialized hardware or it can be easily installed and run on commodity hardware such as low-cost hardware, easily available in the market.

FEATURES OF HDFS

- **Distributed**: In HDFS, the data is divided into multiple data blocks & stored into different nodes. This is one of the most important features of HDFS that makes Hadoop very powerful.
- **Parallel computation**: Data is divided and stored in different nodes; it allows parallel computation.

- **Highly scalable**: HDFS is highly scalable as it can scale hundreds of nodes in a single cluster.
  - ✓ Scaling is divided into two types:

1) **Vertical scaling (scale up)**:
   - For more processing power, can increase the number of cores in CPU or can increase the RAM size.
   - But the disadvantage is that there is a limit in which increase the number of cores in CPU/ increase the RAM size.
   - So that particular machine will not properly working & entire system will be collapsed, not able to access data, can't processing on that system.
2) **Horizontal scaling (scale out**):
   - Initially have one machine.
   - If want more processing power & storage, add one more machine.
   - If still want more processing power & storage, add one more machine.
   - In case, if any one of the machine is not working, other machines will be used for processing the data.
   - The above is the advantage of horizontal scaling than vertical scaling.
   - Hence HDFS follows **horizontal scaling**.



- **Replication**:
  - Due to some unfavourable conditions, the node containing the data may be failed to work.
  - So, to overcome such problems, HDFS always maintains the copy of data on a different machine.
- **Fault tolerance**:
  - The HDFS is highly fault-tolerant.
  - i.e., if any machine fails, the other machine containing the copy of that data automatically become active.
  - In HDFS, the fault-tolerant signifies the robustness of the system in the event of failure.
- **Streaming data access**:
  - It follows **write-once/ read-many design**.

- Which means that once the data stored in the HDFS can't be altered.
- But we can append the data at the end.

- **Portable**:
  - HDFS is designed in such a way that it can easily portable from one platform to another.

JAVA INTERFACES TO HDFS

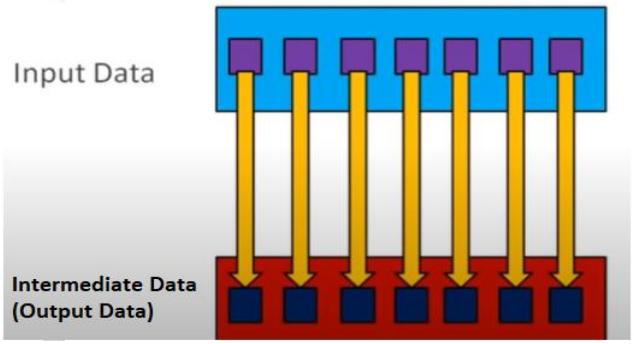
- JAVA interfaces are used for accessing Hadoop's file system.
- JAVA is a high-level/ OOP language.
- In order to interact with Hadoop's file system programmatically, Hadoop provides multiple JAVA classes.
- Package name: **org.apache.hadoop.fs**
- org: Organization, apache: inventor of Hadoop, fs: file system
- This package contains classes useful in manipulation of a file in Hadoop's Filesystem.
- HDFS cluster primarily consist of **NameNode (Master Node**) that manages the file system's metadata & **DataNode (Slave Node**) that stores the actual data.

READ OPERATION IN HDFS

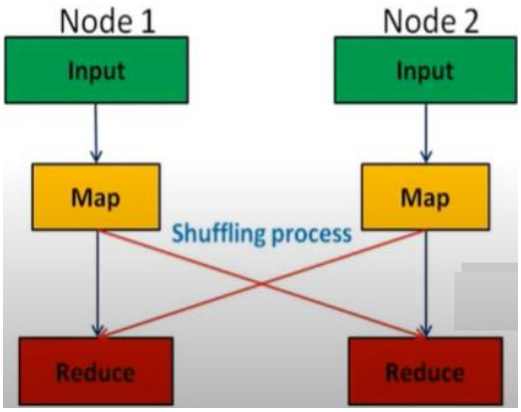
- Data read request is provided by a **client**.
- A client initiate read request by calling '**open()**' method of filesystem object.
- Data is read in the form of streams where in client invokes "**read()**" method repeatedly.
- This process of read() operation continues till it reaches the end of block.
- Once a client has done with reading, it calls '**close()'** method.

HOW MAP REDUCE WORKS

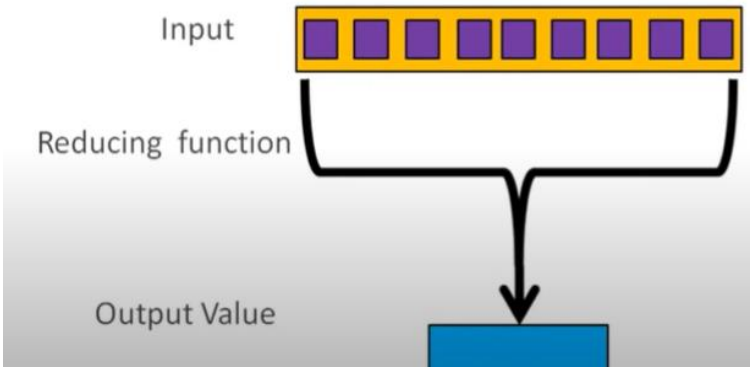
- **Massive parallel processing** technique for processing data which is **distributed on a commodity cluster**.
- **Parallel processing** will **save time**.
- Map reducing technique has two phases.
  1. **Map:**
     - Divide the whole data into chunks.
     - Maps input key/value pairs to a set of intermediate key/value pairs.
     - After mapping, There is another phase is called **shuffling** & **sorting**.



- **Shuffling** is the movement of intermediate records form mapper to reducer.
- **The process of exchanging the intermediate outputs from the map task to the reducer task is known as shuffling.**



2. **Reduce**:
   - After map process, reducer will obtain result.
   - Reduces a set of intermediate values which share a key to a smaller set of values.

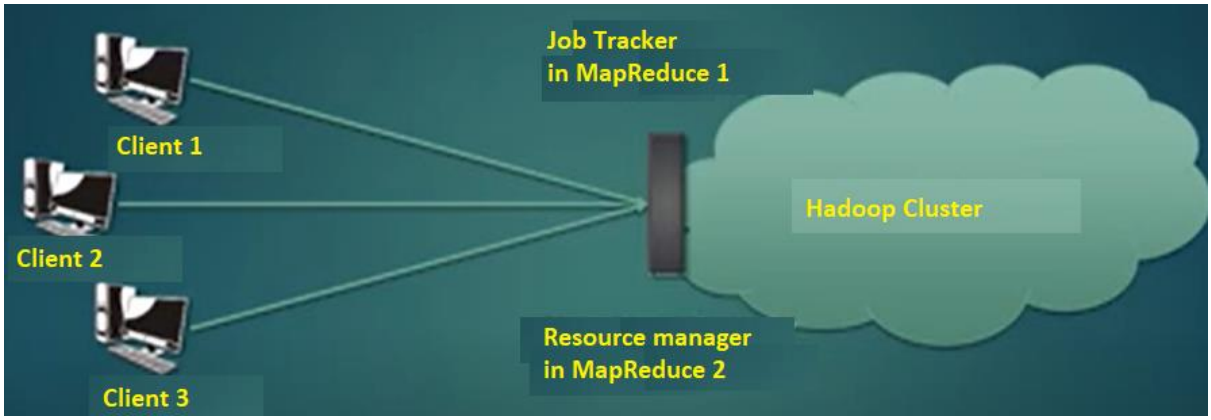


FAILURES OF MAP REDUCE

- There are 3 types of failures:
  1. **Task failure**:
     - Tasks are the user codes.
     - It may be code caused issues like infinite loop/ Hung code.
     - In those cases, **mapred.task.timeout** property will set.
     - Another reason for the failure of user tasks can be run time errors.
     - Another error is JVM (JAVA Virtual Machine) errors.
  2. **TaskTracker Failure**:
     - In the case of TaskTracker failure, JobTracker will reschedule task on another TaskTracker.
     - Rerun the following cases:
       - In-completed tasks.
       - Completed task but whose job is not completed.
       - Blacklisted TaskTrackers.
  3. **Job Tracker Failure**:
     - Most serious failure in Map Reduce.
     - Better to run on a more efficient hardware to avoid failure.

JOB SCHEDULING IN MAP REDUCE

- Multiple users issuing the jobs on Hadoop distributed Network.
- The scheduling scheme would be employed at **JobTracker** in case of **MapReduce 1** and **Source manager** in case of **MapReduce 2**.



- The following schemes can be set up in MapReduce framework:
  - **FIFO**: First in First Out:
    - Default method in MapReduce 1.
  - **Fair Scheduler**:
    - Concept is very similar to Capacitive scheduler with minor differences.
    - Gives **better user experience.**
  - **Capacity Scheduler**:
    - Default method in MapReduce 2.
    - It provides more security than other schedulers.

SHUFFLE & SORT

- Every MapReduce job goes to shuffle and sort phase.
- Map program processes input key/value then the map output is sorted and transferred to reduce it. This is known as **Shuffle**.
- The processes are written to **memory buffer**. Its default size is 100 MB.
- Map writes in the memory buffer fills up and when the buffer reaches a **threshold**, where threshold limit is **80%.**
- If the maps output is very large, recommended to **compress** the maps output to reduce the amount of data.

MAP REDUCE TYPES & FORMATS

- MapReduce has a simple model of data processing input & output for the map & reduce functions.
- The map & Reduce functions in Hadoop having the following general form:
  Map : K1, V1 → list(K2,V2)
  Reduce: (K2, list(V2))→ list(K3,V3)
- When two process intercommunicate (ex: Map communicate to Reduce), the data is transferred in terms of objects.

- **Serialization** is the process of turning structured objects into a byte stream for transmission over a network.
- **Deserialization** is the reverse process of serialization, in which turning a byte stream into series of structured objects.
- The commonly used data types are:
  - Boolean
  - Byte
  - Short
  - Int
  - Float
  - Long
  - Double
  - string

MAP REDUCE FEATURES

- High Scalability
- Flexibility
- Cost effective
- Fast
- Security & Authentication
- Compatible
- Reliable
- Highly available
- Simple programming model
- Parallel & distributed computing
- Counters *(Imp)*
  - Task counters
  - Job counters etc
- Sorting *(Imp)*