Experiment No. 02

Aim: To study Hadoop using Hive, Pig, HBase, Sqoop, NoSQL, MongoDB. Also study the architecture of Pig Latin, MapReduce and APIs.

Theory:

Apache Hive:

Apache Hive is a distributed, fault-tolerant data warehouse system that enables analytics at a massive scale. A data warehouse provides a central store of information that can easily be analyzed to make informed, data driven decisions.
Hive allows users to read, write, and manage petabytes of data using SQL. Hive is built on top of Apache Hadoop, which is an open-source framework used to efficiently store and process large datasets.
Advantages of Hive-

1. Fast - Hive is designed to quickly handle petabytes of data using batch processing.
2. Familiar - Hive is designed to quickly handle petabytes of data using batch processing.
3. Scalable - Hive is easy to distribute and scale based on your needs.

Apache HBase:

Apache HBase is an open-source, NoSQL, distributed big data store. It enables random, strictly consistent, real-time access to petabytes of data. HBase is very effective for handling large, sparse datasets.

HBase integrates seamlessly with Apache Hadoop and the Hadoop ecosystem and runs on top of the Hadoop Distributed File System (HDFS) or Amazon S3 using Amazon Elastic MapReduce (EMR) file system, or EMRFS. HBase serves as a direct input and output to the Apache MapReduce framework for Hadoop, and works with Apache Phoenix to enable SQL-like queries over HBase tables.

Advantages:

1. Scalable - HBase is designed to handle scaling across thousands of servers and managing access to petabytes of data. With the elasticity of Amazon EC2, and the scalability of Amazon S3, HBase is able to handle online access to massive data sets.
2. Fast - HBase provides low latency random read and write access to petabytes of data by distributing requests from applications across a cluster of hosts. Each host has access to data in HDFS and S3, and serves read and write requests in milliseconds.
3. Fault-Tolerant - HBase splits data stored in tables across multiple hosts in the cluster and is built to withstand individual host failures. Because data is stored on HDFS or S3, healthy hosts will automatically be chosen to host the data once served by the failed host, and data is brought online automatically.

Apache Sqoop:

Apache Sqoop

Apache SQOOP (SQL-to-Hadoop) is a tool designed to support bulk export and import of data into HDFS from structured data stores such as relational databases, enterprise data warehouses, and NoSQL systems. Apache Sqoop is a tool for transferring data between Amazon S3, Hadoop, HDFS, and RDBMS databases.

Features of Sqoop

1. Parallel Import/Export - Sqoop uses the YARN framework to import and export data. This provides fault tolerance on top of parallelism.
2. Import Results of an SQL Query - Sqoop enables us to import the results returned from an SQL query into HDFS.
3. Connectors For All Major RDBMS Databases - Sqoop provides connectors for multiple RDBMSs, such as the MySQL and Microsoft SQL servers.
4. Kerberos Security Integration - Sqoop supports the Kerberos computer network authentication protocol, which enables node communication over an insecure network to authenticate users securely.
5. Provides Full and Incremental Load - Sqoop can load the entire table or parts of the table with a single command.

NoSQL:

A NoSQL database organizes large distributed data sets into tuples - key value pairs and objects. NOSQL databases can be of different types –Graph Stores, Wide-Column Stores, Document Databases or Key Value stores. NoSQL -Not Only SQL implies that NoSQL databases do not adhere to any of the relational database concepts. NoSQL databases were developed to handle big data which relational databases were not capable of. NoSQL databases do not follow any strict schema. They distribute the database load on multiple hosts to scale out better. Valu pairs and aggregate them to produce desired results. The input and output of the map and reduce jobs are stored in HDFS.

The three main types of NoSQL are.

1. Column Database (column-oriented)
2. Key-Value Database (key/value oriented)
3. Document Database (document-oriented)

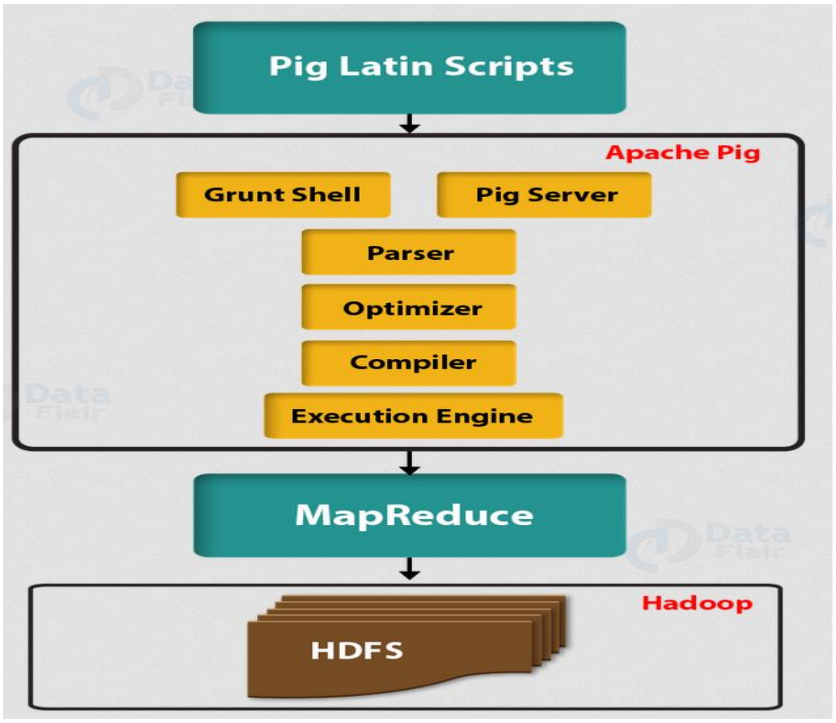
MongoDB:

MongoDB is a highly flexible and scalable NoSQL database management platform that is document-based, can accommodate different data models, and stores data in key-value sets. It was developed as a solution for working with large volumes of distributed data that cannot be processed effectively in relational models, which typically accommodate rows and tables.

Advantages:

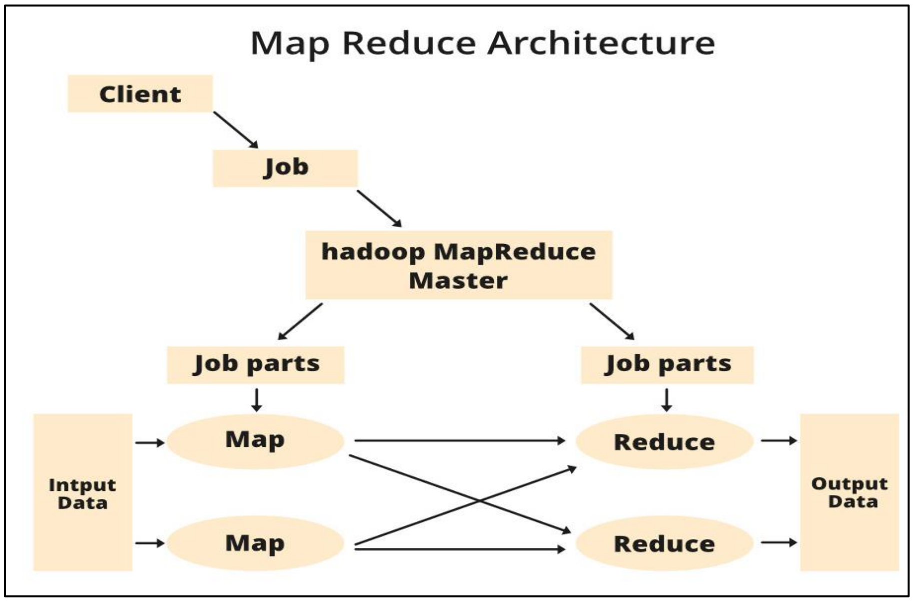
1. When using relational databases, you need several tables for a construct. With Mongo's document-based model, you can represent a construct in a single entity, especially for immutable data.
2. The query language used by MongoDB supports dynamic querying.
3. The schema in MongoDB is implicit, meaning you do not have to enforce it. This makes it easier to represent inheritance in the database in addition to improving polymorphism data storage.
4. Horizontal storage makes it easy to scale.

Architecture of Pig Latin:



1. Atom -Atom is defined as any single value in Pig Latin, irrespective of their data.
2. Tuple - Tuple is a record that is formed by an ordered set of fields.
3. Bag - An unordered set of tuples is what we call Bag.
4. Map - A set of key-value pairs is what we call a map (or data map).
5. Relation - A bag of tuples is what we call Relation. In Pig Latin, the relations are unordered.

MapReduce:



MapReduce is the second of the two most crucial modules, and it’s what allows you to work with data within Hadoop. It performs two tasks:

- Mapping - which involves transforming a set of data into a format that can be easily analyzed. It accomplishes this by filtering and sorting.
- Reducing - which follows mapping. Reducing performs mathematical operations (e.g., counting the number of customers over the age of 21) on the map job output.

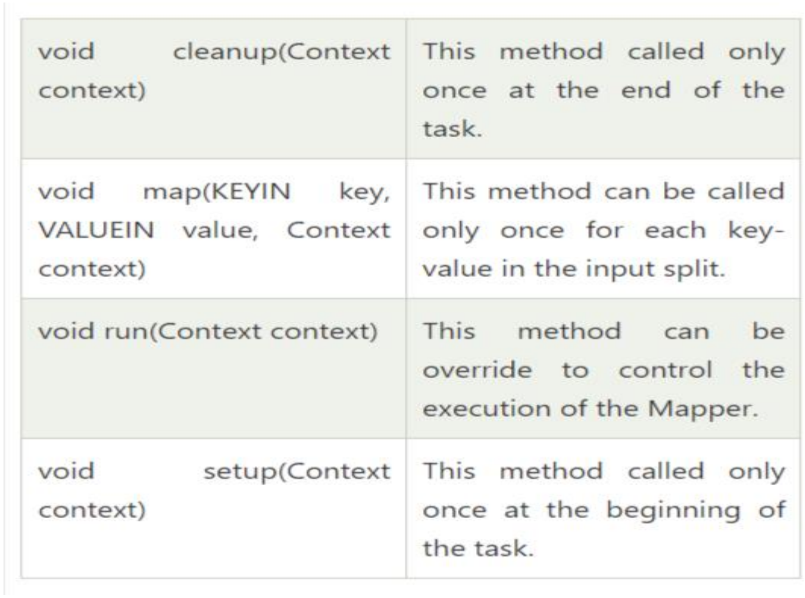
Components of MapReduce Architecture:

1. Client: The MapReduce client is the one who brings the Job to the MapReduce for processing. There can be multiple clients available that continuously send jobs for processing to the Hadoop MapReduce Manager.
2. Job: The MapReduce Job is the actual work that the client wants to do which is comprised of so many smaller tasks that the client wants to process or execute.
3. Hadoop MapReduce Master: It divides the particular job into subsequent job-parts.
4. Job-Parts: The task or sub-jobs that are obtained after dividing the main job. The result of all the job-parts combined to produce the final output.
5. Input Data: The data set that is fed to the MapReduce for processing.
6. Output Data: The final result is obtained after the processing.

APIs of MapReduce:

a) MapReduce Mapper Class – In MapReduce, the role of the Mapper class is to map the input key-value pairs to a set of intermediate key-value pairs. It transforms the input records into intermediate records.

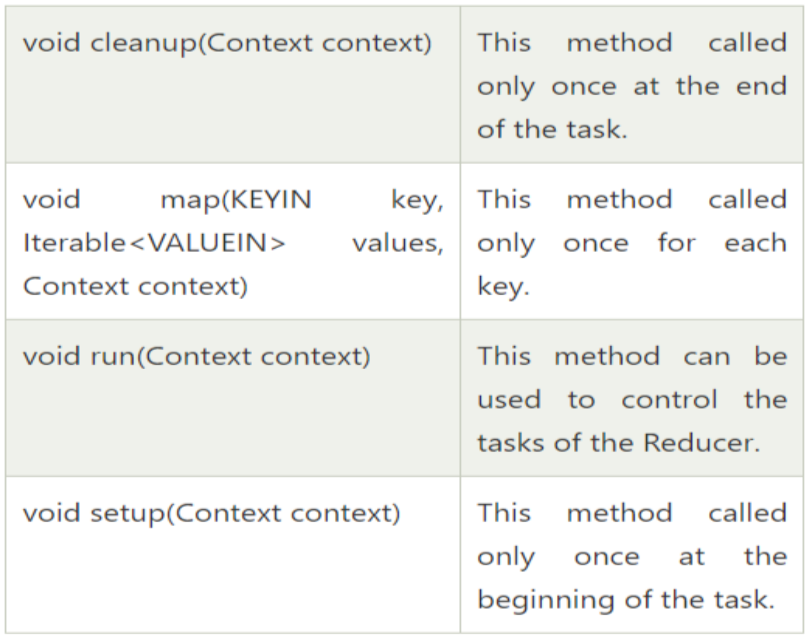
Methods of Mapper Class -



void cleanup(Context context)	This method called only once at the end of the task.
void map(KEYIN key, VALUEIN value, Context context)	This method can be called only once for each key-value in the input split.
void run(Context context)	This method can be override to control the execution of the Mapper.
void setup(Context context)	This method called only once at the beginning of the task.

b) MapReduce Reducer Class - In MapReduce, the role of the Reducer class is to reduce the set of intermediate values

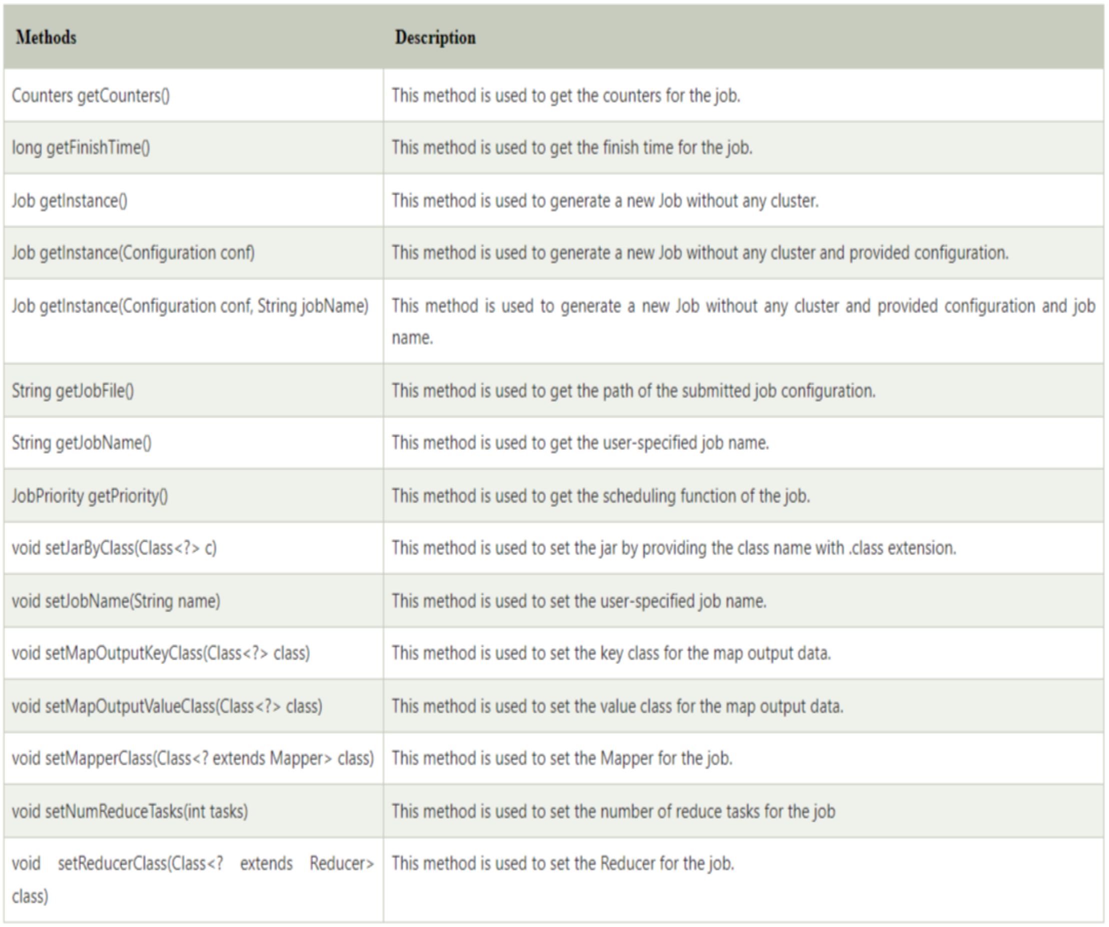
Methods of Reducer Class –



void cleanup(Context context)	This method called only once at the end of the task.
void map(KEYIN key, Iterable<VALUEIN> values, Context context)	This method called only once for each key.
void run(Context context)	This method can be used to control the tasks of the Reducer.
void setup(Context context)	This method called only once at the beginning of the task.

c) MapReduce Job Class - The Job class is used to configure the job and submit it. It also controls the execution and query the state. Once the job is submitted, the set method throws IllegalStateException.

Methods of Job Class –



Methods	Description
Counters getCounters()	This method is used to get the counters for the job.
long getFinishTime()	This method is used to get the finish time for the job.
Job getInstance()	This method is used to generate a new Job without any cluster.
Job getInstance(Configuration conf)	This method is used to generate a new Job without any cluster and provided configuration.
Job getInstance(Configuration conf, String jobName)	This method is used to generate a new Job without any cluster and provided configuration and job name.
String getJobFile()	This method is used to get the path of the submitted job configuration.
String getJobName()	This method is used to get the user-specified job name.
JobPriority getPriority()	This method is used to get the scheduling function of the job.
void setJarByClass(Class<?> c)	This method is used to set the jar by providing the class name with .class extension.
void setJobName(String name)	This method is used to set the user-specified job name.
void setMapOutputKeyClass(Class<?> class)	This method is used to set the key class for the map output data.
void setMapOutputValueClass(Class<?> class)	This method is used to set the value class for the map output data.
void setMapperClass(Class<? extends Mapper> class)	This method is used to set the Mapper for the job.
void setNumReduceTasks(int tasks)	This method is used to set the number of reduce tasks for the job
void setReducerClass(Class<? extends Reducer> class)	This method is used to set the Reducer for the job.