**BDI - Chapter 1**

1. **Define Big Data. List and discuss the 5V’s of Big Data. (5 Marks)**

* **Big data** refers to extremely large and diverse collections of structured, unstructured, and semi-structured data that continues to grow exponentially over time.
* These datasets are so huge and complex in volume, velocity, and variety, that traditional data management systems cannot store, process, and analyze them.
* The amount and availability of data is growing rapidly, spurred on by digital technology advancements, such as connectivity, mobility, the Internet of Things (IoT), and artificial intelligence (AI).
* As data continues to expand and proliferate, new big data tools are emerging to help companies collect, process, and analyze data at the speed needed to gain the most value from it.
* Big data describes large and diverse datasets that are huge in volume and also rapidly grow in size over time.
* Big data is used in machine learning, predictive modeling, and other advanced analytics to solve business problems and make informed decisions.
* **5V’s of Big Data**
  + **Volume: Amount of Data**
    - As its name suggests, the most common characteristic associated with big data is its high volume.
    - This describes the enormous amount of data that is available for collection and produced from a variety of sources and devices on a continuous basis.
  + **Velocity: Speed of Data Generation**
    - Big data velocity refers to the speed at which data is generated.
    - Today, data is often produced in real time or near real time, and therefore, it must also be processed, accessed, and analyzed at the same rate to have any meaningful impact.
  + **Variety: Diversity of Data**
    - Data is heterogeneous, meaning it can come from many different sources and can be structured, unstructured, or semi-structured.
    - More traditional structured data (such as data in spreadsheets or relational databases) is now supplemented by unstructured text, images, audio, video files, or semi-structured formats like sensor data that can’t be organized in a fixed data schema.
  + **Veracity: Accuracy of the Data**
    - Big data can be messy, noisy, and error-prone, which makes it difficult to control the quality and accuracy of the data.
    - Large datasets can be unwieldy and confusing, while smaller datasets could present an incomplete picture.
    - The higher the veracity of the data, the more trustworthy it is.
  + **Value: Worth of the Data**
    - It’s essential to determine the business value of the data you collect.
    - Big data must contain the right data and then be effectively analyzed in order to yield insights that can help drive decision-making.

1. **Define Big Data. List and discuss types of Big Data. (5 Marks)**

* **Structured Data**
  + Structured data can be crudely defined as the data that resides in a fixed field within a record.
  + A certain schema binds it, so all the data has the same set of properties.
  + Structured data is also called **relational data**.
  + It is split into multiple tables to enhance the integrity of the data by creating a single record to depict an entity. Relationships are enforced by the **application of table constraints**.
  + The business value of structured data lies within how well an organization can utilize its existing systems and processes for analysis purposes.
  + A Structured Query Language (**SQL**) is needed to bring the data together.
  + Structured data is easy to enter, query, and analyze.
  + All of the data follows the same format. However, forcing a consistent structure also means that any alteration of data is too tough as each record has to be updated to adhere to the new structure.
  + Examples of structured data include numbers, dates, strings, etc. The business data of an e-commerce website can be considered to be structured data.
  + Cons of Structured Data:
    - Structured data has limited flexibility and is suitable for certain specific use cases only.
    - Structured data is stored in a data warehouse with rigid constraints and a definite schema. Any change in requirements would mean updating all of that structured data to meet the new needs. This is a massive drawback in terms of resource and time management.
* **Semi-Structured Data**
  + Semi-structured data is **not bound by any rigid schema** for data storage and handling.
  + The data is not in the relational format and is not neatly organized into rows and columns like that in a spreadsheet. However, there are some features like key-value pairs that help in discerning the different entities from each other.
  + Since semi-structured data doesn’t need a structured query language, it is commonly called **NoSQL data**.
  + A data serialization language is used to exchange semi-structured data across systems that may even have varied underlying infrastructure.
  + Semi-structured content is often used to store metadata about a business process but it can also include files containing machine instructions for computer programs.
  + This type of information typically comes from external sources such as social media platforms or other web-based data feeds.
* **Unstructured Data**
  + Unstructured data is the kind of data that **doesn’t adhere to any definite schema** or set of rules. Its arrangement is unplanned and haphazard.
  + Photos, videos, text documents, and log files can be generally considered unstructured data.
  + Even though the metadata accompanying an image or a video may be semi-structured, the actual data being dealt with is unstructured.
  + Additionally, Unstructured data is also known as “dark data” because it cannot be analyzed without the proper software tools.

1. **Compare Traditional Data vs Big Data. (5 Marks)**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Traditional Data** | **Big Data** |
| **1** | Traditional data is the structured data that is being majorly maintained by all types of businesses starting from very small to big organizations. | Big data refers to extremely large and diverse collections of structured, unstructured, and semi-structured data that continues to grow exponentially over time. |
| **2** | Its volume ranges from Gigabytes to Terabytes. | Its volume ranges from Petabytes to Zettabytes or Exabytes. |
| **3** | Traditional database system deals with structured data. | Big data system deals with structured, semi-structured, database, and unstructured data. |
| **4** | Traditional data source is centralized and it is managed in centralized form. | Big data source is distributed and it is managed in distributed form. |
| **5** | Data integration is very easy. | Data integration is very difficult. |
| **6** | Normal system configuration is capable to process traditional data. | High system configuration is required to process big data. |
| **7** | The size of the data is very small. | The size is more than the traditional data size. |
| **8** | Normal functions can manipulate data. | Special kind of functions can manipulate data. |
| **9** | Its data model is strict schema based and it is static. | Its data model is a flat schema based and it is dynamic. |
| **10** | Traditional data is stable and inter relationship. | Big data is not stable and unknown relationship. |
| **11** | Traditional data is in manageable volume. | Big data is in huge volume which becomes unmanageable. |
| **12** | It is easy to manage and manipulate the data. | It is difficult to manage and manipulate the data. |
| **13** | Its data sources include ERP transaction data, CRM transaction data, financial data, organizational data, web transaction data etc. | Its data sources includes social media, device data, sensor data, video, images, audio etc. |

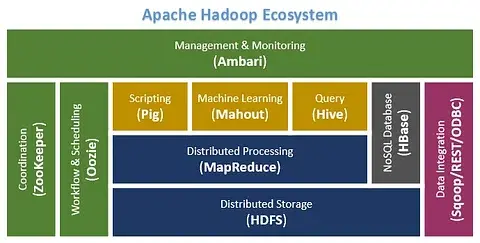
1. **Define Big Data Analytics. List and explain different types of Big Data Analytics. (5 Marks)**

* **Big data analytics** is the use of advanced analytic techniques against very large, diverse big data sets that include structured, semi-structured and unstructured data, from different sources, and in different sizes from terabytes to zettabytes.
* With big data analytics, you can ultimately fuel better and faster decision-making, modelling and predicting of future outcomes and enhanced business intelligence.
* Big-Data Analytics is like having a digital detective that helps us make sense of the massive amount of data we create in our online lives. Whether it’s the things we purchase online (on Flipkart, Amazon, etc.), the videos we watch on different platforms, or the posts we share, our digital actions generate a ton of information.
* Four main types of big data analytics support and inform different business decisions.

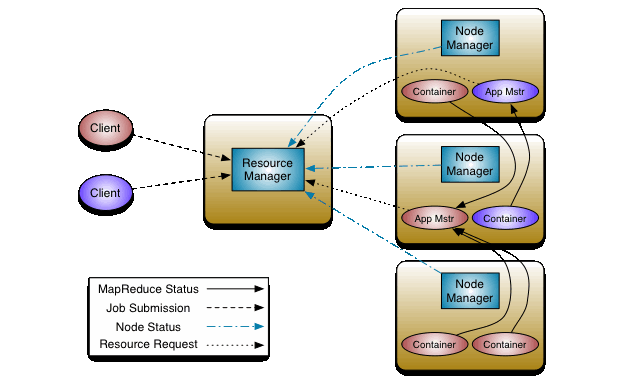
1. **Descriptive analytics**
   * Descriptive analytics refers to data that can be easily read and interpreted.
   * This data helps create reports and visualize information that can detail company profits and sales.
   * Example: During the pandemic, a leading pharmaceutical company conducted data analysis on its offices and research labs. Descriptive analytics helped them identify consolidated unutilized spaces and departments, saving the company millions of pounds.
2. **Diagnostics analytics**
   * Diagnostics analytics helps companies understand why a problem occurred.
   * Big data technologies and tools allow users to mine and recover data that helps dissect an issue and prevent it from happening in the future.
   * Example: An online retailer’s sales have decreased even though customers continue to add items to their shopping carts. Diagnostics analytics helped to understand that the payment page was not working correctly for a few weeks.
3. **Predictive analytics**
   * Predictive analytics looks at past and present data to make predictions.
   * With artificial intelligence (AI), machine learning, and data mining, users can analyses the data to predict market trends.
   * Example: In the manufacturing sector, companies can use algorithms based on historical data to predict if or when a piece of equipment will malfunction or break down.
4. **Prescriptive analytics**
   * Prescriptive analytics solves a problem, relying on AI and machine learning to gather and use data for risk management.
   * Example: Within the energy sector, utility companies, gas producers, and pipeline owners identify factors that affect the price of oil and gas to hedge risks.

**BDI - Chapter 2**

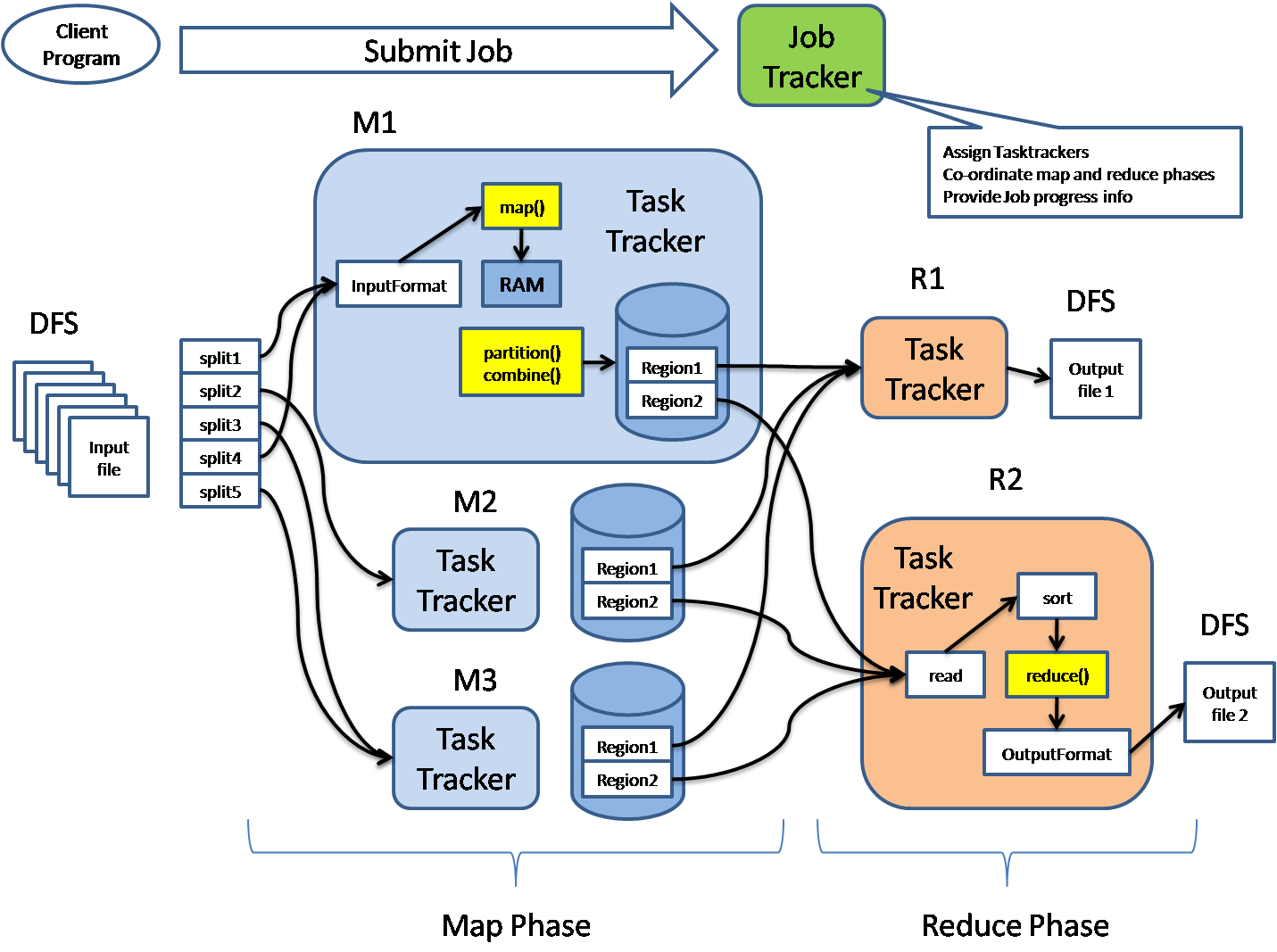
1. **Explain the core components of Hadoop. (5 Marks)**

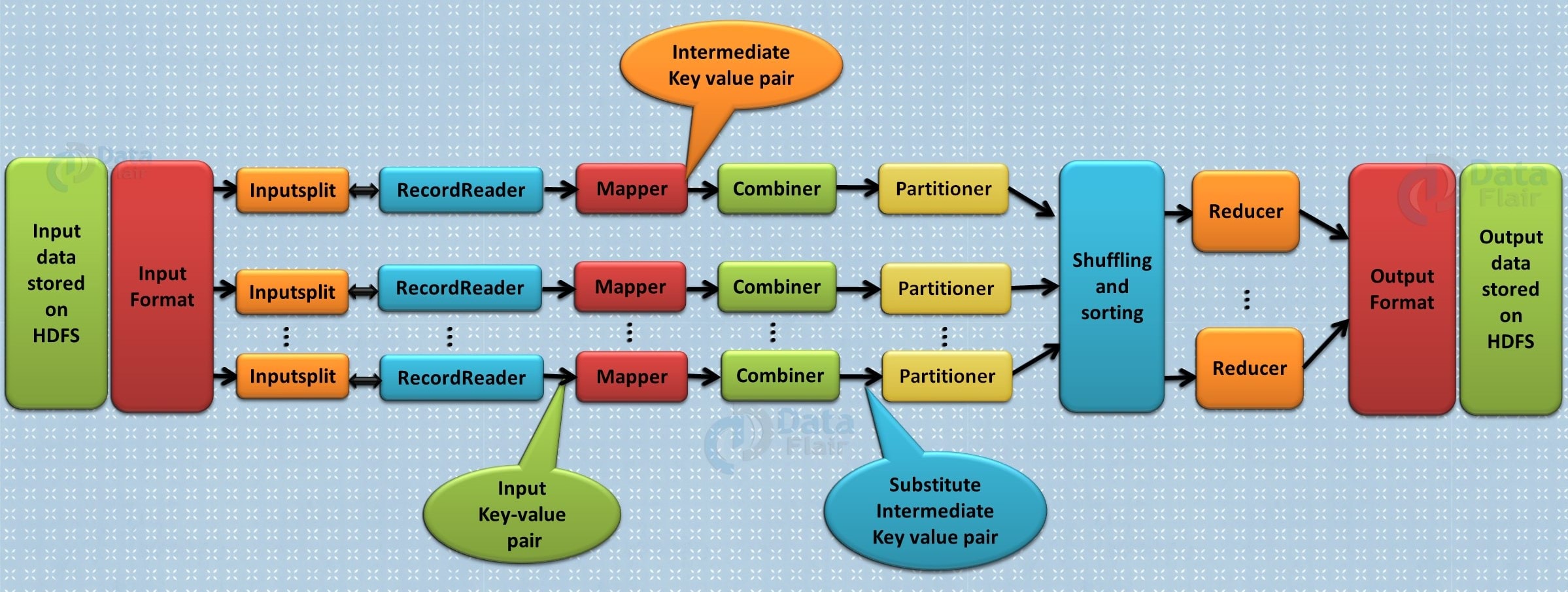
* Apache Hadoop is an open-source framework intended to make interaction with big data easier.
* Hadoop is a framework that enables processing of large data sets which reside in the form of clusters.
* Being a framework, Hadoop is made up of several modules that are supported by a large ecosystem of technologies.
* Stores and processes humongous data at a faster rate.
* The data may be structured, semi-structured, or unstructured
* Protects application and data processing against hardware failures.
* Whenever a node gets down, the processing gets redirected automatically to other nodes and ensures running of applications.
* Organizations can store raw data and processor filter it for specific analytic uses as and when required.
* As Hadoop is scalable, organizations can handle more data by adding more nodes into the systems.
* Supports real-time analytics, drives better operational decision-making and batch workloads for historical analysis
* **Core Components of Hadoop:**
  + **HDFS:** Hadoop Distributed File System
    - HDFS is designed to provide greater scalability and fault resistance to the Hadoop framework.
    - The primary function of HDFS is to act as a storage backend for other distributed applications, including MapReduce and HBase.
    - HDFS is a component of the Hadoop framework used for storing and managing large datasets across several connected computers.
    - HDFS is mainly designed for working on commodity Hardware devices (inexpensive devices), working on a distributed file system design.
    - HDFS has two important components: NameNode and DataNode

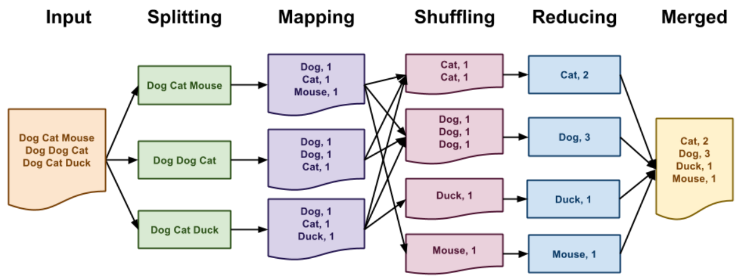
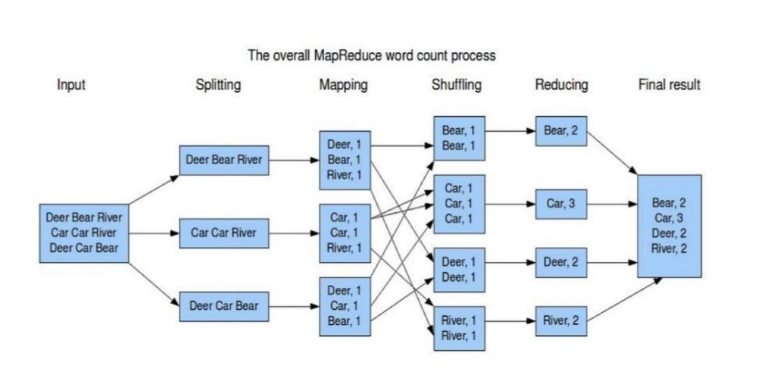
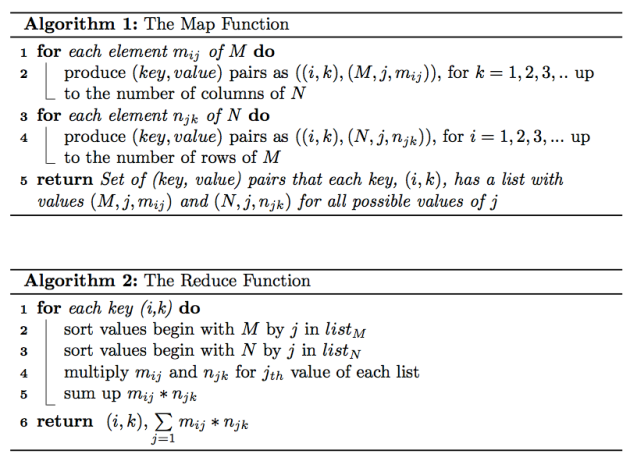
An HDFS cluster consists of a single NameNode and multiple DataNodes

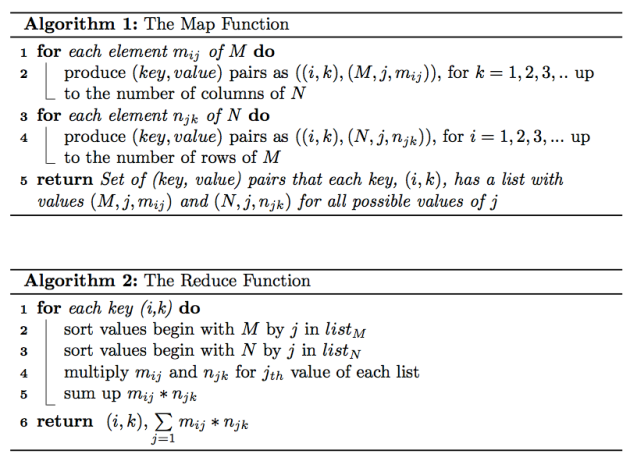
* + - NameNode:
      * A master server that manages the file system namespace and regulates access to files by clients.
      * NameNode executes file system namespace operations like opening, closing, and renaming files and directories.
      * It contains metadata, just like a log file or you can say as a table of content.
      * Hence, it requires less storage and high computational resources.
      * NameNode maintains the file system namespace
    - DataNode:
      * Manage storage attached to the nodes that they run on.
      * All data is stored on the DataNodes and hence it requires more storage resources.
    - Rack:
      * The rack is nothing but just the physical collection of nodes in our Hadoop cluster (maybe 30 to 40).
      * A large Hadoop cluster is consists of multiple Racks.
      * With the help of this Racks information, NameNode chooses the closest DataNode to achieve the maximum performance while performing the read/write information which reduces the Network Traffic.
    - Data Replication:
      * Ensures availability of data
      * Replication factor is the number of times a copy of something (here, data block) is made.
      * The blocks of a file are replicated for fault tolerance.
      * The block size and replication factor are configurable per file.
    - File System Namespace:
      * The file system namespace hierarchy is similar to most other existing file systems; one can create and remove files, move a file from one directory to another, or rename a file.
      * HDFS supports user quotas and access permissions.
      * Any change to the file system namespace or its properties is recorded by the NameNode.
      * An application can specify the number of replicas of a file that should be maintained by HDFS.
      * This information is stored by the NameNode.
    - File Block In HDFS:
      * Data in HDFS is always stored in terms of blocks.
      * So, the single block of data is divided into multiple blocks of size 128MB which is default and you can also change it manually.
  + **YARN: Yet Another Resource Negotiator.**
    - YARN is considered as the brain of Hadoop Ecosystem. It performs all your processing activities by allocating resources and scheduling tasks.
    - It handles the cluster of nodes and acts as Hadoop’s resource management unit.
    - YARN allocates RAM, memory, and other resources to different applications.
    - 2 major components of YARN: ResourceManager and NodeManager
      * ResourceManager: Main node in the processing department. It receives the processing requests, and then passes the parts of requests to corresponding Node Managers accordingly, where the actual processing takes place.
      * NodeManager: They are installed on every DataNode. It is responsible for execution of task on every single DataNode.
  + **MapReduce**
    - The major feature of MapReduce is to perform the distributed processing in parallel in a Hadoop cluster which Makes Hadoop working so fast.
    - MapReduce are two functions: Map and Reduce. They are sequenced one after the other.
      * Map(): takes input from the disk as <key,value> pairs, processes them, and produces another set of intermediate <key,value> pairs as output. **Splitting the input and mapping is part of Map()**
      * Reduce(): also takes inputs as <key,value> pairs (output of mapping function) and produces <key,value> pairs as output. **Shuffling and reducing is part of Reduce()**
  + **Pig**
    - Pig is a high-level data flow platform for executing MapReduce programs of Hadoop.
    - It was developed by Yahoo. The language for Pig is pig Latin.
    - PIG has two parts:
      * Pig Latin, the language
      * The pig runtime, for the execution environment.
    - The Pig scripts get internally converted to MapReduce jobs and get executed on data stored in HDFS.
    - Apart from that, Pig can also execute its job in Apache Tez or Apache Spark.
    - Pig can handle any type of data, i.e., structured, semi-structured or unstructured and stores the corresponding results into Hadoop Data File System.
    - Every task which can be achieved using PIG can also be achieved using java used in MapReduce.
    - The compiler internally converts pig latin to MapReduce.
    - It produces a sequential set of MapReduce jobs, and that’s an abstraction (which works like black box).
  + **HIVE**
    - HIVE is a data warehousing component which performs reading, writing and managing large data sets in a distributed environment using SQL-like interface.
    - The query language of Hive is called Hive Query Language (HQL), which is very similar like SQL.
    - It has 2 basic components:
      * Hive Command Line
      * JDBC/ODBC driver.
    - The Hive Command line interface is used to execute HQL commands.
    - Java Database Connectivity (JDBC) and Object Database Connectivity (ODBC) is used to establish connection from data storage.
  + **Mahout**
    - Apache Mahout is an open-source project that is primarily used for creating scalable machine learning algorithms.
    - It implements popular machine learning techniques such as:
      * Recommendation
      * Classification
      * Clustering
    - Mahout lets applications to analyze large sets of data effectively and in quick time.
    - Includes several MapReduce enabled clustering implementations such as k-means, fuzzy k-means, Canopy, Dirichlet, and Mean-Shift.
    - Supports Distributed Naive Bayes and Complementary Naive Bayes classification implementations.
  + **Sparks**
    - Spark is an alternative framework to Hadoop built on Scala but supports varied applications written in Java, Python, etc.
    - Compared to MapReduce it provides in-memory processing which accounts for faster processing.
    - In addition to batch processing offered by Hadoop, it can also handle real-time processing
  + **HBase**
    - HBase is an open source, non-relational distributed database. In other words, it is a NoSQL database.
    - It supports all types of data and that is why, it’s capable of handling anything and everything inside a Hadoop ecosystem.
    - It is modelled after Google’s BigTable, which is a distributed storage system designed to cope up with large data sets.
    - The HBase was designed to run on top of HDFS and provides BigTable like capabilities.
    - It gives us a fault tolerant way of storing sparse data, which is common in most Big Data use cases.
    - The HBase is written in Java, whereas HBase applications can be written in REST, Avro and Thrift APIs.
  + **Apache Ambari**
    - It is an open-source tool responsible for keeping track of running applications and their statuses.
    - Ambari manages, monitors, and provisions Hadoop clusters.
    - Also, it also provides a central management service to start, stop, and configure Hadoop services.
  + **Apache Flumes**
    - Apache Flume is an open-source tool for collecting, aggregating, and moving huge amounts of streaming data from the external web servers to the central store, say HDFS, HBase, etc.
    - It is a highly available and reliable service which has tunable recovery mechanisms.
    - The main purpose of designing Apache Flume is to move streaming data generated by various applications to Hadoop Distributed File System.
    - Apache Flume can efficiently ingest log data from various servers into a centralized repository.
    - With Flume, we can collect data from different web servers in real-time as well as in batch mode.
  + **Sqoop** 
    - Sqoop is a tool used to transfer bulk data between Hadoop and external datastores, such as relational databases (MS SQL Server, MySQL).
    - To process data using Hadoop, the data first needs to be loaded into Hadoop clusters from several sources
    - Sqoop in Hadoop helped to overcome all the challenges of the traditional approach and it could load bulk data from RDBMS to Hadoop with ease.
      * Maintaining data consistency
      * Ensuring efficient utilization of resources
      * Loading bulk data to Hadoop was not possible
      * Loading data using scripts was slow
  + **Kafka**
    - There are a lot of applications generating data and a commensurate number of applications consuming that data.
    - But connecting them individually is a tough task. That’s where Kafka comes in.
    - It sits between the applications generating data (Producers) and the applications consuming data (Consumers).
    - Kafka is distributed and has in-built partitioning, replication, and fault-tolerance.
    - It can handle streaming data and also allows businesses to analyze data in real-time.
  + **ZooKeeper**
    - Apache Zookeeper is the coordinator of any Hadoop job which includes a combination of various services in a Hadoop Ecosystem.
    - Apache Zookeeper coordinates with various services in a distributed environment.
    - It saves a lot of time by performing synchronization, configuration maintenance, grouping and naming**.**
  + **Oozie**
    - For Apache jobs, Oozie has been just like a scheduler.
    - It schedules Hadoop jobs and binds them together as one logical work.
    - There are two kinds of Oozie jobs:
      * Oozie workflow: These are sequential set of actions to be executed. You can assume it as a relay race. Where each athlete waits for the last one to complete his part.
      * Oozie Coordinator: These are the Oozie jobs which are triggered when the data is made available to it.

1. **Explain MapReduce architecture in brief. (5 Marks)**

* Mapreduce is processing unit of hadoop that processes and computes vast volume of data
* Two main components in Mapreduce:
  + Map tasks deal with splitting and mapping of data
  + Reduce tasks shuffle and reduce the data.
* Architecture:
  + MapReduce architecture consists of various components
  + Job: This is real work that needs to be done or processed
  + Task: This is a piece of real work that needs to be done or processed. The MapReduce task covers many small tasks that need to be done.
  + Job Tracker: This tracker plays a role in organizing tasks and tracking all tasks assigned to a task tracker.
  + Task Tracker: This tracker plays the role of tracking activity and reporting activity status to the task tracker.
  + Input data: This is used for processing in the mapping phase.
  + Exit data: This is the result of mapping and mitigation.
  + Client: This is a program or Application Programming Interface (API) that sends tasks to MapReduce. It can accept services from multiple clients.
  + Hadoop MapReduce Master: This plays the role of dividing tasks into sections.
  + Job Parts: These are small tasks that result in the division of the primary function.
  + **In MapReduce architecture**, clients submit tasks to MapReduce Master. This manager will then divide the work into smaller equal parts. The components of the function will be used for two main tasks in Map Reduce: mapping and subtraction.
  + The developer will write a concept that satisfies the organizations or company’s needs. Input data will be categorized and mapped.
  + The central data will then be filtered and merged. The slider that will produce the last one stored on HDFS will process the output.
  + The activity tracker works like a master. It ensures that we do all the work. The activity tracker lists tasks posted by clients, and it will provide job trackers for jobs. Each task tracker has a map function and minimizes tasks. Activity trackers report the status of each task assigned to the task tracker.
  + The following diagram shows a simplified flow diagram of the MapReduce program.
  + **Phases of MapReduce:**
    - **Mapping Phase**
      * This is the first phase of the program. There are two steps in this phase: classification and mapping.
      * The database is divided into equal units called units (input divisions) in the division step.
      * Hadoop contains a RecordReader that uses TextInputFormat to convert input variables into keyword pairs.
      * Key-value pairs are then used as input on the map step. This is the only data format a map editor can read or understand.
      * The map step contains the logic of the code used in these data blocks. In this step, the map analyzes key pairs and generates an output of the same form (key-value pairs).
    - **Shuffling phase**
      * This is the second phase that occurs after the completion of the Mapping phase.
      * It consists of two main steps: filtering and merging. In the filter step, keywords are filtered using keys and combining ensures that key-value pairs are included.
      * The shoplifting phase facilitates the removal of duplicate values ​​and the collection of values. Different values ​​with the same keys are combined. The output of this category will be keys and values, as in the Map section.
    - **Reducer phase**
      * In the reduction phase, the output of the push phase is user input. The subtractor continuously processes these inputs to reduce the median values ​​into smaller ones. Provides a summary of the entire database. Output in this category is stored in HDFS.
    - **Combiner phase**
      * This is the optional phase used to improve the MapReduce process. It is used to reduce pap output at the node level.
      * At this stage, duplicate output from the map output can be merged into a single output.
      * The integration phase accelerates the integration phase by improving the performance of tasks.



1. **Apply MapReduce on the following document to count the frequency of words. Show all the phases properly. (5 Marks)**
2. **Write the algorithm for Matrix Multiplication using MapReduce.**



1. **Write the HDFS commands for the following with suitable example: (5 Marks)**
   * 1. **To display recursively the contents in the directory.**

* Command: hdfs dfs -ls -R /path/to/directory
* Example: hdfs dfs -ls -R /user/hadoop/data
  + 1. **To copy files/folders from local file system to HDFS store.**

copyFromLocal:

* Command: hdfs dfs -copyFromLocal /path/to/local/file\_or\_directory /path/in/hdfs
* Example: hdfs dfs -copyFromLocal /home/user/file.txt /user/hadoop/data

put:

* Command: hdfs dfs -put /path/to/local/file\_or\_directory /path/in/hdfs
* Example: hdfs dfs -put /home/user/file.txt /user/hadoop/data
  + 1. **To copy files/folders from HDFS store to local file system.**

copyToLocal:

* Command: hdfs dfs -copyToLocal /path/in/hdfs /path/in/local
* Example: hdfs dfs -copyToLocal /user/hadoop/data/file.txt /home/user

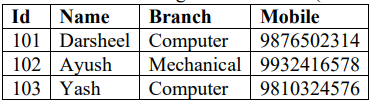
get:

* Command: hdfs dfs -get /path/in/hdfs /path/in/local
* Example: hdfs dfs -get /user/hadoop/data/file.txt /home/user
  + 1. **To give the size of each file in directory.**
* Command: hdfs dfs -du /path/to/directory
* Example: hdfs dfs -du /user
  + 1. **To change the replication factor of a file/directory in HDFS**
* Command: hdfs dfs -setrep -R <replication\_factor> /path/to/file\_or\_directory
* Example: hdfs dfs -setrep -R 3 /user/hadoop/data/file.txt

**BDI - Chapter 3**

* + - 1. **Consider the following database table: (10 Marks)**

**Create the following table in Hive with transactional property = true and partitions.**

****CREATE TABLE student (

Id INT,

Name VARCHAR(20),

Branch VARCHAR(20),

Mobile INT

)

PARTITIONED BY (load\_date date)

CLUSTERED BY(Id) INTO 3 BUCKETS

STORED AS ORC TBLPROPERTIES ('transactional'='true');

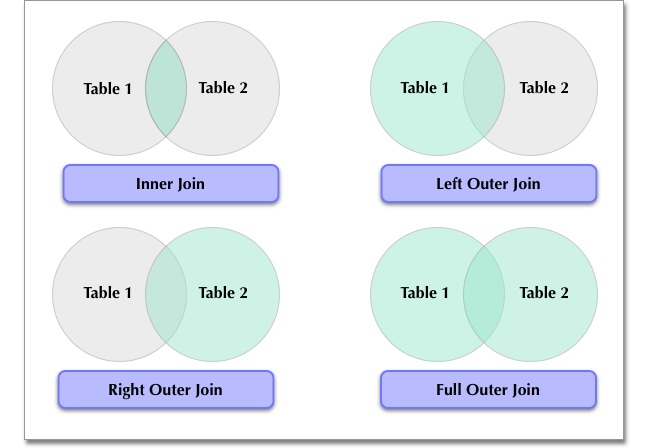
**Insert the following values in the table.**

INSERT INTO student PARTITION (load\_date = '2024-03-03') VALUES (101, 'Darsheel', 'Computer', 9876502314);

INSERT INTO student PARTITION (load\_date = '2024-03-03') VALUES (102, 'Ayush', 'Mechanical', 9932416578);

INSERT INTO student PARTITION (load\_date = '2024-03-03') VALUES (103, 'Yash', 'Computer', 9810324576);

**Display the count of students with respect to branch.**

SELECT Branch, COUNT(\*)

FROM student

GROUP BY Branch;

**Update the name to “Yashvi” where ID =103.**

UPDATE student

SET Name = 'Yashvi'

WHERE Id = 103;

**Alter the table to add new column “CGPA” with float datatype.**

ALTER TABLE student

ADD COLUMNS(CGPA FLOAT);

* + - 1. **Explain different Joins in Hive with suitable example. (5 Marks)**
* Basically, for combining specific fields from two tables by using values common to each one we use Hive JOIN clause.
* In other words, to combine records from two or more tables in the database we use JOIN clause.
* Types of Joins in Hive
  + - Inner join in Hive
      * To combine and retrieve the records from multiple tables we use Hive Join clause. Moreover, by using the primary keys and foreign keys of the tables JOIN condition is to be raised.
      * Command: **select \* from customers c join orders o on (c.id = o.customer\_id);**
    - Left Outer Join in Hive
      * On defining HiveQL Left Outer Join, even if there are no matches in the right table it returns all the rows from the left table.
      * To be more specific, even if the ON clause matches 0 (zero) records in the right table, then also this Hive JOIN still returns a row in the result. Although, it returns with NULL in each column from the right
      * table.
      * In addition, it returns all the values from the left table. Also, the matched values from the right table, or NULL in case of no matching JOIN predicate.
      * Command: **select \* from customers c left outer join orders o on (c.id = o.customer\_id);**
    - Right Outer Join in Hive
      * Even if there are no matches in the left table, HiveQL Right Outer Join returns all the rows from the right table.
      * To be more specific, even if the ON clause matches 0 (zero) records in the left table, then also this Hive JOIN still returns a row in the result. Although, it returns with NULL in each column from the left table.
      * In addition, it returns all the values from the right table. Also, the matched values from the left table or NULL in case of no matching join predicate.
      * Command: **select \* from customers c right outer join orders o on (c.id = o.customer\_id);**
    - Full Outer Join in Hive
      * The major purpose of this HiveQL Full outer Join is it combines the records of both the left and the right outer tables which fulfills the Hive JOIN condition.
      * Moreover, this joined table contains either all the records from both the tables or fills in NULL values for missing matches on either side.
      * Command: **select \* from customers c full outer join orders o on (c.id = o.customer\_id);**
      1. **Compare and contrast HBase with RDBMS. (5 Marks)**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Parameters** | **RDBMS** | **HBase** |
| **1.** | SQL | It requires SQL (Structured Query Language). | SQL is not required in HBase. |
| **2.** | Schema | It has a fixed schema. | It does not have a fixed schema and allows for the addition of columns on the fly. |
| **3.** | Database Type | It is a row-oriented database | It is a column-oriented database. |
| **4.** | Scalability | RDBMS allows for scaling up. That implies, that rather than adding new servers, we should upgrade the current server to a more capable server whenever there is a requirement for more memory, processing power, and disc space. | Scale-out is possible using HBase. It means that, while we require extra memory and disc space, we must add new servers to the cluster rather than upgrade the existing ones. |
| **5.** | Nature | It is static in nature | Dynamic in nature |
| **6.** | Data retrieval | In RDBMS, slower retrieval of data. | In HBase, faster retrieval of data. |
| **7.** | Rule | It follows the ACID (Atomicity, Consistency, Isolation, and Durability) property. | It follows CAP (Consistency, Availability, Partition-tolerance) theorem. |
| **8.** | Type of data | It can handle structured data. | It can handle structured, unstructured as well as semi-structured data. |
| **9.** | Sparse data | It cannot handle sparse data. | It can handle sparse data. |
| **10.** | Volume of data | The amount of data in RDBMS is determined by the server’s configuration. | In HBase, the amount of data depends on the number of machines deployed rather than on a single machine. |
| **11.** | Transaction Integrity | In RDBMS, mostly there is a guarantee associated with transaction integrity. | In HBase, there is no such guarantee associated with the transaction integrity. |
| **12.** | Referential Integrity | Referential integrity is supported by RDBMS. | When it comes to referential integrity, no built-in support is available. |
| **13.** | Normalize | In RDBMS, you can normalize the data. | The data in HBase is not normalized, which means there is no logical relationship or connection between distinct tables of data. |
| **14.** | Table size | It is designed to accommodate small tables.  Scaling is difficult. | It is designed to accommodate large tables. HBase may scale horizontally. |

* + - 1. **Write the HBase Commands for the following: (5 Marks)**
         1. Create a table with name Product.

**create 'product', 'shoe', 'tshirt'**

* 1. Add column family ‘Shoe’ and ‘Tshirt’.
  2. In column family Shoe, add the columns as below:

Brand = Nike, Price = $50, Size = 9, Description = For Men

**put 'Product', '1', 'Shoe:Brand', 'Nike'**

**put 'Product', '1', 'Shoe:Price', '$50'**

**put 'Product', '1', 'Shoe:Size', '9'**

**put 'Product', '1', 'Shoe:Description', 'For Men'**

* 1. In column family Tshirt, add the columns as below:

Brand = Polo, Price = $35, Size = XL, Description = Round Neck

**put 'Product', '2', 'Tshirt:Brand', 'Polo'**

**put 'Product', '2', 'Tshirt:Price', '$35'**

**put 'Product', '2', 'Tshirt:Size', 'XL'**

**put 'Product', '2', 'Tshirt:Description', 'Round Neck'**

* 1. Display the contents of table ‘Product’ for 10 row keys.

**scan 'Product', {LIMIT => 10}**

* 1. Check whether the table ‘Product’ is enabled or disabled.

**is\_enabled 'Product'**

1. **Write the HBase Commands for the following: (5 Marks)**
2. Checking the status of the HBase Server with 3 different parameters. (default parameter provided is “summary”.)

**status**

status 'simple'

status 'summary'

status 'detailed'

NOTE: When we execute this command status, it will give information about number of server’s present, dead servers and average load of server

1. Creating a table t1 with column families cf1, cf2, cf3.

**create 't1', 'cf1', 'cf2', 'cf3'**

1. Inserting value in the column c1 of cf1 as v1, column c2 of cf1 as v2.

**put 't1', 'row1', 'cf1:c1', 'v1'**

**put 't1', 'row1', 'cf1:c2', 'v2'**

1. Displaying the filters available.

**show\_filters**

1. Deleting the table t1 from HBase Server.

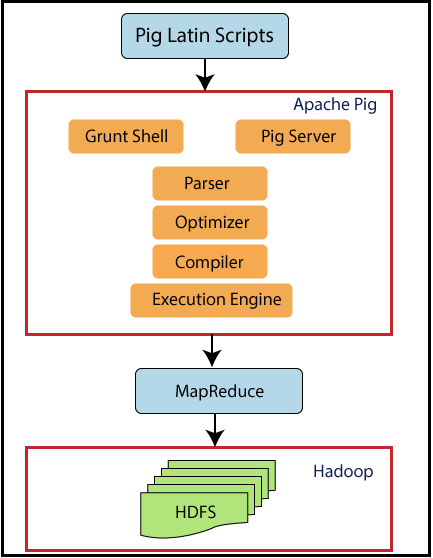
**disable 't1'**

**drop 't1'**

1. **Compare Apache Pig vs MapReduce. (5 Marks)**

|  |  |  |
| --- | --- | --- |
| **S.No** | **MapReduce** | **Pig** |
| **1.** | It is a Data Processing Language. | It is a Data Flow Language. |
| **2.** | It converts the job into map-reduce functions. | It converts the query into map-reduce functions. |
| **3.** | It is a Low-level Language. | It is a High-level Language |
| **4.** | It is difficult for the user to perform join operations. | Makes it easy for the user to perform Join operations. |
| **5.** | The user has to write 10 times more lines of code to perform a similar task than Pig. | The user has to write fewer lines of code because it supports the multi-query approach. |
| **6.** | It has several jobs therefore execution time is more. | It is less compilation time as the Pig operator converts it into MapReduce jobs. |
| **7.** | It is supported by recent versions of the Hadoop. | It is supported with all versions of Hadoop. |

1. **Explain the significance of Pig Grunt and Pig Latin. (5 Marks)**

* Pig is a high-level platform or tool for processing massive data sets. It provides a high level of abstraction for MapReduce computation. It includes a high-level scripting language called Pig Latin, used to create data analysis codes. The programmers will build scripts in the Pig Latin Language to process the data stored in the Hadoop distributed file system (HDFS). The Apache Pig tool's two primary components are Pig Latin and Pig Engine. Pig's output is always saved in HDFS.
* **Grunt** is Pig’s interactive shell.
* It enables users to enter Pig Latin interactively and provides a shell for users to interact with HDFS.
* To enter Grunt, invoke Pig with no script or command to run. Typing:

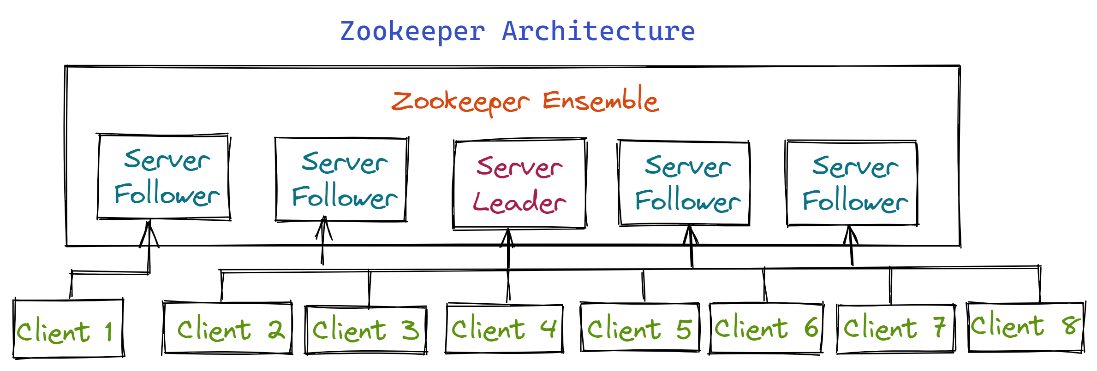
pig -x local

* will result in the prompt:

grunt>

* This gives you a Grunt shell to interact with your local filesystem. If you omit the -x local and have a cluster configuration set in PIG\_CLASSPATH, this will put you in a Grunt shell that will interact with HDFS on your cluster.
* Pig script can be executed with grunt shell which is a native shell provided by Apache pig to execute pig queries.
* We can invoke shell commands using sh and fs.
  + Syntax of sh command: grunt> sh ls
  + Syntax of fs command: grunt> fs -ls
* Grunt also provides commands for controlling Pig and MapReduce:
  + kill *jobid*: Kill the MapReduce job associated with jobid.
  + exec [[-param param\_name = param\_value]] [[-param\_file filename]] script: Execute the Pig Latin script script.
  + run [[-param param\_name = param\_value]] [[-param\_file filename]] script: Execute the Pig Latin script script in the current Grunt shell.
* Pig provides a high-level language known as **Pig Latin** for writing data analysis applications.
* Parser:
  + At first, all the Pig Scripts are handled by the Parser.
  + Parser basically checks the syntax of the script, does type checking, and other miscellaneous checks.
  + Afterwards, Parser’s output will be a DAG (directed acyclic graph) that represents the Pig Latin statements as well as logical operators.
* Optimiser:
  + The logical plan (DAG) is passed to the logical optimizer. It carries out the logical optimizations further such as projection and push down.
* Compiler:
  + Then compiler compiles the optimized logical plan into a series of MapReduce jobs.
* Execution Engine:
  + Finally, the MapReduce jobs are submitted to Hadoop in a sorted order.
  + Then these MapReduce jobs are executed on Hadoop producing the desired results
* To analyze data using Apache Pig, programmers must create scripts in the Pig Latin language.
* Internally, all of these scripts are turned to Map and Reduce jobs.
* The Pig Engine component of Apache Pig accepts Pig Latin scripts as input and turns them into MapReduce jobs.
* Apache Pig and Pig Latin come with the following features:
  + Rich set of operators: It has a variety of operators for performing operations like join, sort, filter, etc.
  + Handles all kinds of data: Apache Pig examines various data types, organized and unstructured. The findings are saved in HDFS.
  + User-Defined Functions (UDFs): Pig allows you to write user-defined functions in other programming languages, such as Java, and then invoke or embed them in Pig Scripts.
  + Extensibility: Users can create their functions to read, process, and write data using current operators.
  + Ease of programming: Pig Latin is comparable to SQL, and writing a Pig script is simple if you know SQL.
  + Optimization opportunities: The jobs in Apache Pig optimize their execution automatically, so programmers need to focus on language semantics.

1. **Explain the working of Zookeeper. Also state the benefits of Zookeeper. (10 Marks)**

* Apache ZooKeeper provides a wide range of good features to the user such as:
  + Updating the Node’s Status: Apache ZooKeeper is capable of updating every node that allows it to store updated information about each node across the cluster.
  + Managing the Cluster: This technology can manage the cluster in such a way that the status of each node is maintained in real time, leaving lesser chances for errors and ambiguity.
  + Naming Service: ZooKeeper attaches a unique identification to every node which is quite similar to the DNA that helps identify it.
  + Automatic Failure Recovery: Apache ZooKeeper locks the data while modifying which helps the cluster recover it automatically if a failure occurs in the database.
* Working of Apache Zookeeper
  + The first thing that happens as soon as the ensemble (a group of ZooKeeper servers) starts is, it waits for the clients to connect to the servers.
  + After that, the clients in the ZooKeeper ensemble will connect to one of the nodes. That node can be any of a leader node or a follower node.
  + Once the client is connected to a particular node, the node assigns a session ID to the client and sends an acknowledgement to that particular client.
  + If the client does not get any acknowledgement from the node, then it resends the message to another node in the ZooKeeper ensemble and tries to connect with it.
  + On receiving the acknowledgement, the client makes sure that the connection is not lost by sending the heartbeats to the node at regular intervals.
  + Finally, the client can perform functions like read, write, or store the data as per the need.
* Benefits of Apache ZooKeeper
  + Simplicity: Coordination is done with the help of a shared hierarchical namespace.
  + Reliability: The system keeps performing even if more than one node fails.
  + Order: It keeps track by stamping each update with a number denoting its order.
  + Speed: It runs with a ratio of 10:1 in the cases where ‘reads’ are more common.
  + Scalability: The performance can be enhanced by deploying more machines.
* ZOOKEEPER ARCHITECTURE
  + Client: Client node in our distributed applications cluster is used to access information from the server. It
  + sends a message to the server to let the server know that the client is alive, and if there is no response
  + from the connected server the client automatically resends the message to another server.
  + Server: The server gives an acknowledgement to the client to inform that the server is alive, and it provides all services to clients.
  + Leader: If any of the server nodes is failed, this server node performs automatic recovery.
  + Follower: It is a server node which follows the instructions given by the leader.
  + Ensemble: A cluster or ensemble is a group of Zookeeper servers. When running Apache, you can use ZooKeeper infrastructure in cluster mode to keep the system functioning at its best.