Assignment - 1

1) What is Big Data? Explain characteristics of Big Data.

Big Data is a collection of data that is huge in volume, yet growing exponentially with time. It is a data with so large size and complexity that none of traditional data management tools can store it or process it efficiently. Big data is also a data but with huge size.

Four characteristics define Big Data:

Volume
The main characteristic that makes data "big" is the sheer volume. It makes no sense to focus on minimum storage units because the total amount of information is growing exponentially every year. In 2010, Thomson Reuters estimated in its annual report that it believed the world was "awash with over 800 exabytes of data and growing."
For that same year, EMC, a hardware company that makes data storage devices, thought it was closer to 900 exabytes and would grow by 50 percent every year. No one really knows how much new data is being generated, but the amount of information being collected is huge.

Variety
Variety is one the most interesting developments in technology as more and more information is digitized. Traditional data types (structured data) include things on a bank statement like date, amount, and time. These are things that fit neatly in a relational database.
Structured data is augmented by unstructured data, which is where things like Twitter feeds, audio files, MRI images, web pages, web logs are put anything that can be captured and stored but doesn't have a meta model (a set of rules to frame a concept or idea it defines a class of information and how to express it) that neatly defines it.
Unstructured data is a fundamental concept in big data. The best way to understand unstructured data is by comparing it to structured data. Think of structured data as data that is well defined in a set of rules. For example, money will always be numbers and have at least two decimal points; names are expressed as text; and dates follow a specific pattern.
With unstructured data, on the other hand, there are no rules. A picture, a voice recording, a tweet they all can be different but express ideas and thoughts based on human understanding. One of the goals of big data is to use technology to take this unstructured data and make sense of it.

Veracity
Veracity refers to the trustworthiness of the data. Can the manager rely on the fact that the data is representative? Every good manager knows that there are inherent discrepancies in all the data collected.

Velocity
Velocity is the frequency of incoming data that needs to be processed. Think about how many SMS messages, Facebook status updates, or credit card swipes are being sent on a particular telecom carrier every minute of every day, and you'll have a good appreciation of velocity. A streaming application like Amazon Web Services Kinesis is an example of an application that handles the velocity of data.

2) What are the benefits of Big Data? Discuss challenges under Big Data. How Big Data Analytics can be useful in the development of smart cities.

- **Proper use of big data offers several advantages, including:**

Opportunities to Make Better Decisions
When a lot of information is available in a form that organizations can readily manage and analyze, there's a greater probability of discovering patterns and insights that can inform operational and strategic decisions. Data-driven insights provide a foundation for more informed and reliable decision-making.

Increasing Productivity and Efficiency
With big data analytics tools enabling organizations to process more information at faster speeds, personal productivity levels for individual workers can increase, and the enterprise can gain access to information and insights about its own operations that enable management to recognize areas where the organization itself could be more productive.

Reducing Costs
Streamlining operations, improving efficiency, and increasing productivity by using big data can introduce significant cost savings to an enterprise and positively impact overall profitability. The use of big data in predictive or prescriptive analytics processes powered by machine learning and artificial intelligence generates further cost reductions. This can be through the identification of even more efficient ways of working or via mechanisms such as preventative maintenance and enhanced quality control management.

Improving Customer Service and Customer Experience
Technical support and Helpline services powered by big data, machine learning (ML), and artificial intelligence (AI) can greatly improve the standard of response and follow-up that organizations can give to their consumers. Responsible use and analysis of customer and transaction data enable organizations to personalize their outreach to individual consumers, leading to greater engagement with brands and more satisfying user or buyer experiences.

Fraud and Anomaly Detection
In industries like financial services or healthcare, it’s just as important to know what’s going wrong as it is to know what’s going right. AI and machine learning systems with big data can easily detect erroneous transactions, fraudulent activity indicators, and anomalies in data sets that may point to various ongoing or potential issues.

Greater Agility and Speed to Market
Developments in the real-time processing of big data through stream analytics enable organizations to become more agile. Both in their internal operations and product development, innovation, and speed to market.

- **Challenges under Big Data**

Insufficient understanding and acceptance of big data
Confusing variety of big data technologies
Paying loads of money
Complexity of managing data quality
Dangerous big data security holes
Tricky process of converting big data into valuable insights

- **Big data can have an impact on various sectors of a city, including transport, public safety, city budgets, and more.**

Public Safety
Smart cities must provide security for their citizens. Cities can use predictive big data analytics to identify which areas are prone to be hubs of crime and predict the exact crime location. Information like historical and geographical data helps cities create a much safer environment.

Transportation
Traffic congestion is a major problem in many cities since it can cost cities millions in revenue. Cities can manage transportation by analyzing data from transport authorities. The analyzed data can uncover patterns that help reduce traffic congestion and help authorities implement data-driven road optimization.

Cost Reduction
Cities invest a lot of money in transforming a city into a smart city. These investments can be either for remodeling or renovation. Analysis of big data can suggest which areas require transformation and what kind of transformation. As a result, cities can make dedicated investments for required areas.

Sustainable Growth

Regular analysis of the growth of a smart city enables city officials to get continuous updates about needed changes. Continuous updates are the key growth drivers of sustainability because they provide a clear idea regarding the required developments. Data plays a key role in determining the outcomes of development in a smart city.

3) What are the advantages of Hadoop

Major Advantages of Hadoop:

Scalable
Hadoop is a highly scalable storage platform because it can store and distribute very large data sets across hundreds of inexpensive servers that operate in parallel. Unlike traditional relational database systems (RDBMS) that can't scale to process large amounts of data.

Cost-effective
Hadoop also offers a cost-effective storage solution for businesses exploding data sets. The problem with traditional relational database management systems is that it is extremely cost prohibitive to scale to such a degree in order to process such massive volumes of data. In an effort to reduce costs, many companies in the past would have had to down-sample data and classify it based on certain assumptions as to which data was the most valuable. The raw data would be deleted, as it would be too cost-prohibitive to keep.

Flexible
Hadoop enables businesses to easily access new data sources and tap into different types of data (both structured and unstructured) to generate value from that data. This means businesses can use Hadoop to derive valuable business insights from data sources such as social media, email conversations.

Fast
Hadoop's unique storage method is based on a distributed file system that basically 'maps' data wherever it is located on a cluster. The tools for data processing are often on the same servers where the data is located, resulting in much faster data processing. If you're dealing with large volumes of unstructured data, Hadoop is able to efficiently process terabytes of data in just minutes, and petabytes in hours.

Resilient to failure
A key advantage of using Hadoop is its fault tolerance. When data is sent to an individual node, that data is also replicated to other nodes in the cluster, which means that in the event of failure, there is another copy available for use.

4) Explain Hadoop Architecture.

As we all know Hadoop is a framework written in Java that utilizes a large cluster of commodity hardware to maintain and store big size data. Hadoop works on MapReduce Programming Algorithm that was introduced by Google. Today lots of Big Brand Companys are using Hadoop in their Organization to deal with big data for eg. Facebook, Yahoo, Netflix, eBay, etc.

The Hadoop Architecture Mainly consists of 4 components.

- MapReduce
- HDFS(Hadoop distributed File System)
- YARN(Yet Another Resource Framework)
- Common Utilities or Hadoop Common

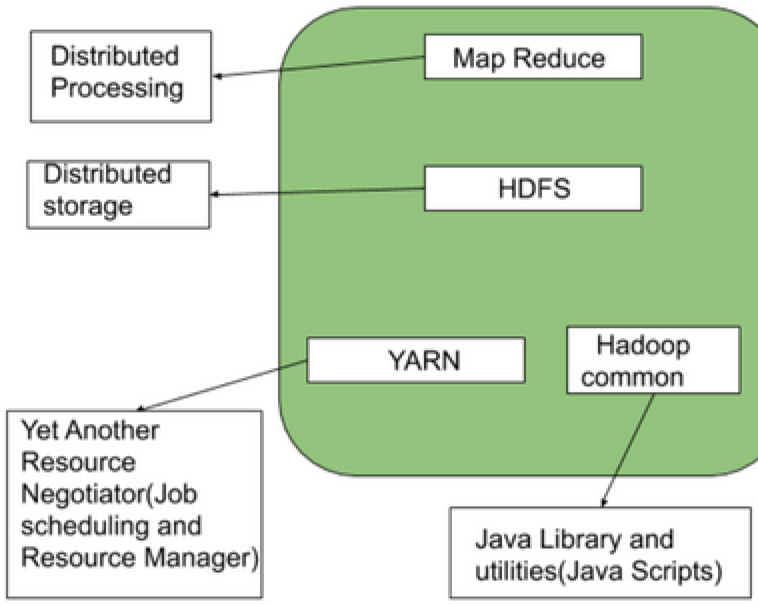

Distributed Processing
Map Reduce
Distributed storage
HDFS
YARN
Hadoop common
Yet Another Resource Negotiator(Job scheduling and Resource Manager)
Java Library and utilities(Java Scripts)

Fig 1 - Hadoop Architecture

MapReduce

MapReduce nothing but just like an Algorithm or a data structure that is based on the YARN framework. The major feature of MapReduce is to perform the distributed processing in parallel in a Hadoop cluster which Makes Hadoop working so fast. When you are dealing with Big Data, serial processing is no more of any use. MapReduce has mainly 2 tasks which are divided phase-wise:

In first phase, Map is utilized and in next phase Reduce is utilized.

Here, we can see that the Input is provided to the Map() function then it's output is used as an input to the Reduce function and after that, we receive our final output. Let's understand What this Map() and Reduce() does.

As we can see that an Input is provided to the Map(), now as we are using Big Data. The Input is a set of Data. The Map() function here breaks this DataBlocks into Tuples that are nothing but a key-value pair. These key-value pairs are now sent as input to the Reduce(). The Reduce() function then combines this broken Tuples or key-value pair based on its Key value and form set of Tuples, and perform some operation like sorting, summation type job, etc. which is then sent to the final Output Node. Finally, the Output is Obtained.

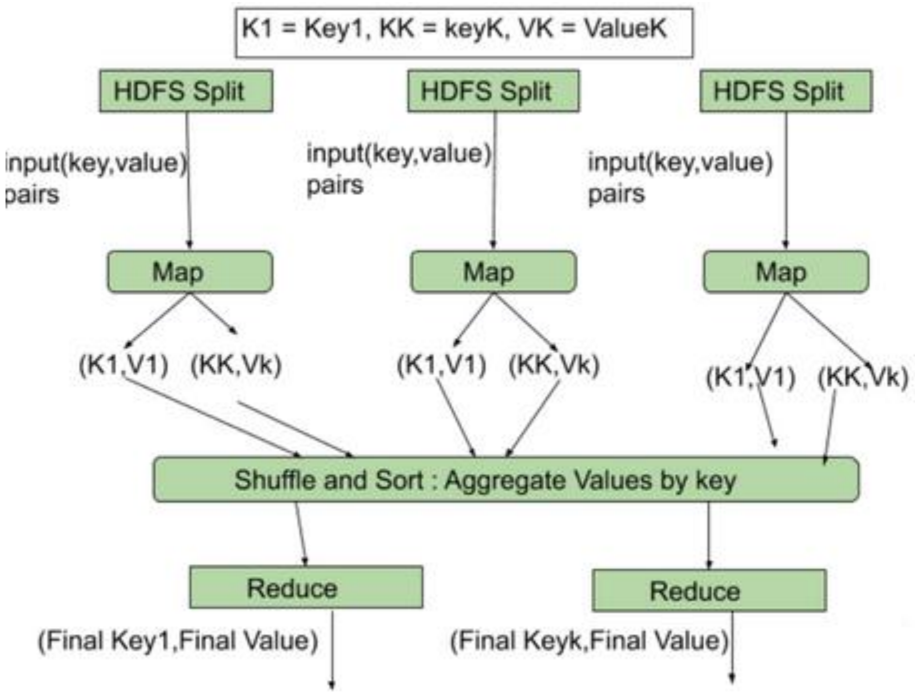


Fig 2 – MapReduce Task

HDFS

HDFS(Hadoop Distributed File System) is utilized for storage permission is a Hadoop cluster. It mainly designed for working on commodity Hardware devices(inexpensive devices), working on a distributed file system design. HDFS is designed in such a way that it believes more in storing the data in a large chunk of blocks rather than storing small data blocks.
HDFS in Hadoop provides Fault-tolerance and High availability to the storage layer and the other devices present in that Hadoop cluster. Data storage Nodes in HDFS.

NameNode(Master)
DataNode(Slave)

NameNode: NameNode works as a Master in a Hadoop cluster that guides the Datanode(Slaves). Namenode is mainly used for storing the Metadata i.e. the data about the data. Meta Data can be the transaction logs that keep track of the user's activity in a Hadoop cluster.

Meta Data can also be the name of the file, size, and the information about the location(Block number, Block ids) of Datanode that Namenode stores to find the closest DataNode for Faster Communication. Namenode instructs the DataNodes with the operation like delete, create, Replicate, etc.

DataNode: DataNodes works as a Slave DataNodes are mainly utilized for storing the data in a Hadoop cluster, the number of DataNodes can be from 1 to 500 or even more than that. The more number of DataNode, the Hadoop cluster will be able to store more data. So it is advised that the DataNode should have High storing capacity to store a large number of file blocks.

YARN(Yet Another Resource Negotiator)

YARN is a Framework on which MapReduce works. YARN performs 2 operations that are Job scheduling and Resource Management. The Purpose of Job schedular is to divide a big task into small jobs so that each job can be assigned to various slaves in a Hadoop cluster and Processing can be Maximized. Job Scheduler also keeps track of which job is important, which job has more priority, dependencies between the jobs and all the other information like job timing, etc. And the use of Resource Manager is to manage all the resources that are made available for running a Hadoop cluster.

Features of YARN
- Multi-Tenancy
- Scalability
- Cluster-Utilization
- Compatibility

Hadoop common or Common Utilities

Hadoop common or Common utilities are nothing but our java library and java files or we can say the java scripts that we need for all the other components present in a Hadoop cluster. these utilities are used by HDFS, YARN, and MapReduce for running the cluster. Hadoop Common verify that Hardware failure in a Hadoop cluster is common so it needs to be solved automatically in software by Hadoop Framework.

5) Explain working of following phases of Map Reduce with one common example. (i) Map Phase (ii) Combiner Phase (iii) Shuffle and Sort Phase (iv) Reducer Phase.

MapReduce model has three major and one optional phase:

1. Mapper
It is the first phase of MapReduce programming and contains the coding logic of the mapper function.The conditional logic is applied to the ‘n’ number of data blocks spread across various data nodes.Mapper function accepts key-value pairs as input as (k, v), where the key represents the offset address of each record and the value represents the entire record content.The output of the Mapper phase will also be in the key-value format as (k’, v’).

2. Shuffle and Sort
The output of various mappers (k’, v’), then goes into Shuffle and Sort phase.All the duplicate values are removed, and different values are grouped together based on similar keys.The output of the Shuffle and Sort phase will be key-value pairs again as key and array of values (k, v[]).

3. Reducer
The output of the Shuffle and Sort phase (k, v[]) will be the input of the Reducer phase.In this phase reducer function’s logic is executed and all the values are aggregated against their corresponding keys. Reducer consolidates outputs of various mappers and computes the final job output. The final output is then written into a single file in an output directory of HDFS.

4. Combiner
It is an optional phase in the MapReduce model. The combiner phase is used to optimize the performance of MapReduce jobs. In this phase, various outputs of the mappers are locally reduced at the node level. For example, if different mapper outputs (k, v) coming from a single node contains duplicates, then they get combined i.e. locally reduced as a single (k, v[]) output. This phase makes the Shuffle and Sort phase work even quicker thereby enabling additional performance in MapReduce jobs.

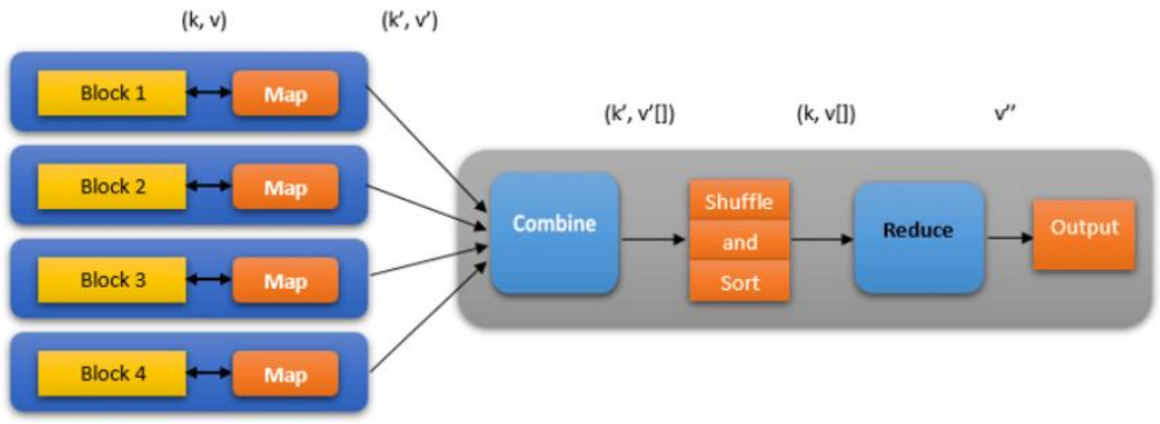


Fig 3 – Map Reduce Phases

6) HDFS commands with syntax and at least one example of each. (i) copyFromLocal (ii)setrep (iii) checksum

copyFromLocal (or) put: To copy files/folders from local file system to hdfs store. This is the most important command. Local filesystem means the files present on the OS.
Syntax:
bin/hdfs dfs -copyFromLocal <local file path> <dest(present on hdfs)>

Example: Let's suppose we have a file AI.txt on Desktop which we want to copy to folder bda present on hdfs.
bin/hdfs dfs -copyFromLocal ../Desktop/AI.txt /bda
(OR)
bin/hdfs dfs -put ../Desktop/AI.txt /bda

setrep: This command is used to change the replication factor of a file/directory in HDFS. By default it is 3 for anything which is stored in HDFS (as set in hdfs core-site.xml).

Example 1: To change the replication factor to 6 for geeks.txt stored in HDFS.
bin/hdfs dfs -setrep -R -w 6bda.txt
Example 2: To change the replication factor to 4 for a directory bda Input stored in HDFS.
bin/hdfs dfs -setrep -R 4 /bda
The -w means wait till the replication is completed. And -R means recursively, we use it for directories as they may also contain many files and folders inside them.

checksum -Returns the Checksum Information of a File
The checksum command is used to Returns the Checksum Information of a File. Returns the checksum information of a file.
$ hadoop fs -checksum [-v] URI
or
$ hdfs dfs -checksum [-v] URI

7) Draw and explain HDFS architecture

Hadoop is an open source framework from Apache and is used to store process and analyze data which are very huge in volume. Hadoop is written in Java and is not OLAP (online analytical processing). It is used for batch/offline processing.It is being used by Facebook, Yahoo, Google, Twitter, LinkedIn and many more. Moreover it can be scaled up just by adding nodes in the cluster.

Modules of Hadoop
HDFS: Hadoop Distributed File System. Google published its paper GFS and on the basis of that HDFS was developed. It states that the files will be broken into blocks and stored in nodes over the distributed architecture.
Yarn: Yet another Resource Negotiator is used for job scheduling and manage the cluster.
Map Reduce: This is a framework which helps Java programs to do the parallel computation on data using key value pair. The Map task takes input data and converts it into a data set which can be computed in Key value pair. The output of Map task is consumed by reduce task and then the out of reducer gives the desired result.
Hadoop Common: These Java libraries are used to start Hadoop and are used by other Hadoop modules.

Hadoop Architecture

The Hadoop architecture is a package of the file system, MapReduce engine and the HDFS (Hadoop Distributed File System). The MapReduce engine can be MapReduce/MR1 or YARN/MR2.

A Hadoop cluster consists of a single master and multiple slave nodes. The master node includes Job Tracker, Task Tracker, NameNode, and DataNode whereas the slave node includes DataNode and TaskTracker.

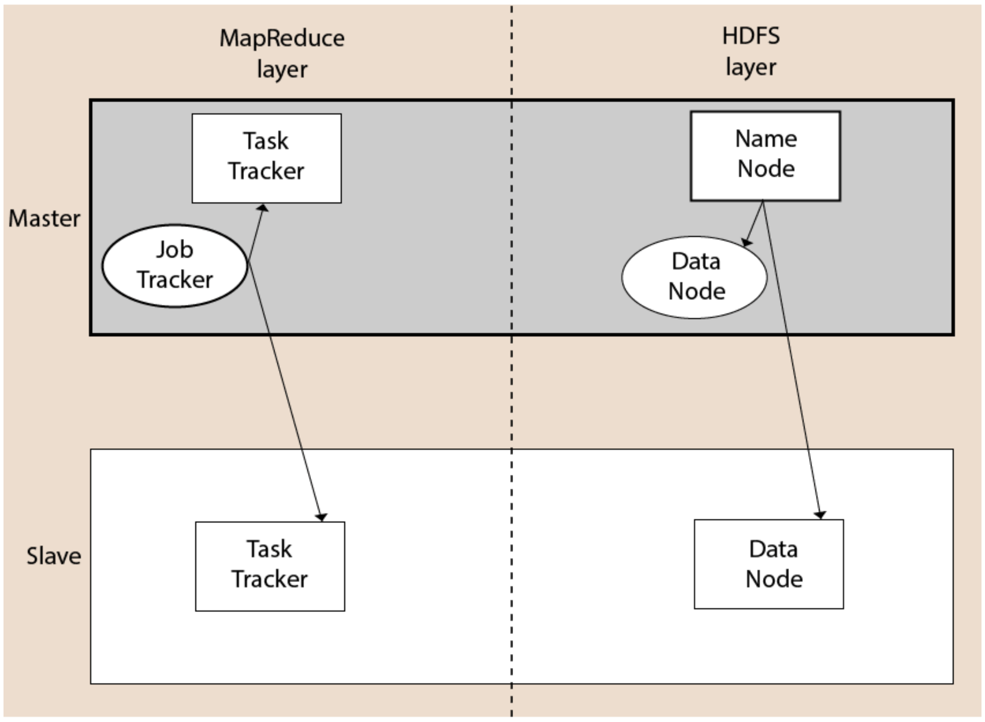


Fig 4 – Hadoop Architecture

Hadoop Distributed File System

The Hadoop Distributed File System (HDFS) is a distributed file system for Hadoop. It contains a master/slave architecture. This architecture consist of a single NameNode performs the role of master, and multiple DataNodes performs the role of a slave.

Both NameNode and DataNode are capable enough to run on commodity machines. The Java language is used to develop HDFS. So any machine that supports Java language can easily run the NameNode and DataNode software.

NameNode
It is a single master server exist in the HDFS cluster.
As it is a single node, it may become the reason of single point failure.
It manages the file system namespace by executing an operation like the opening, renaming and closing the files.
It simplifies the architecture of the system.

DataNode
The HDFS cluster contains multiple DataNodes.
Each DataNode contains multiple data blocks.
These data blocks are used to store data.
It is the responsibility of DataNode to read and write requests from the file system's clients.
It performs block creation, deletion, and replication upon instruction from the NameNode.

Job Tracker
The role of Job Tracker is to accept the MapReduce jobs from client and process the data by using NameNode. In response, NameNode provides metadata to Job Tracker.

Task Tracker
It works as a slave node for Job Tracker.It receives task and code from Job Tracker and applies that code on the file. This process can also be called as a Mapper.

MapReduce Layer
The MapReduce comes into existence when the client application submits the MapReduce job to Job Tracker. In response, the Job Tracker sends the request to the appropriate Task Trackers. Sometimes, the TaskTracker fails or time out. In such a case, that part of the job is rescheduled.

8) What is Zookeeper? List the benefits of it

ZooKeeper is a distributed co-ordination service to manage large set of hosts. Co-ordinating and managing a service in a distributed environment is a complicated process. ZooKeeper solves this issue with its simple architecture and API. ZooKeeper allows developers to focus on core application logic without worrying about the distributed nature of the application.

The ZooKeeper framework was originally built at “Yahoo!” for accessing their applications in an easy and robust manner. Later, Apache ZooKeeper became a standard for organized service used by Hadoop, HBase, and other distributed frameworks. For example, Apache HBase uses ZooKeeper to track the status of distributed data.

Before moving further, it is important that we know a thing or two about distributed applications. So, let us start the discussion with a quick overview of distributed applications.
Here are the benefits of using ZooKeeper

- **Simple distributed coordination process**
- **Synchronization** − Mutual exclusion and co-operation between server processes. This process helps in Apache HBase for configuration management.
- **Ordered Messages**
- **Serialization** − Encode the data according to specific rules. Ensure your application runs consistently. This approach can be used in MapReduce to coordinate queue to execute running threads.
- **Reliability**
- **Atomicity** − Data transfer either succeed or fail completely, but no transaction is partial.

9) Explain HBase Architecture. Explain working of Hive with proper steps and diagram.

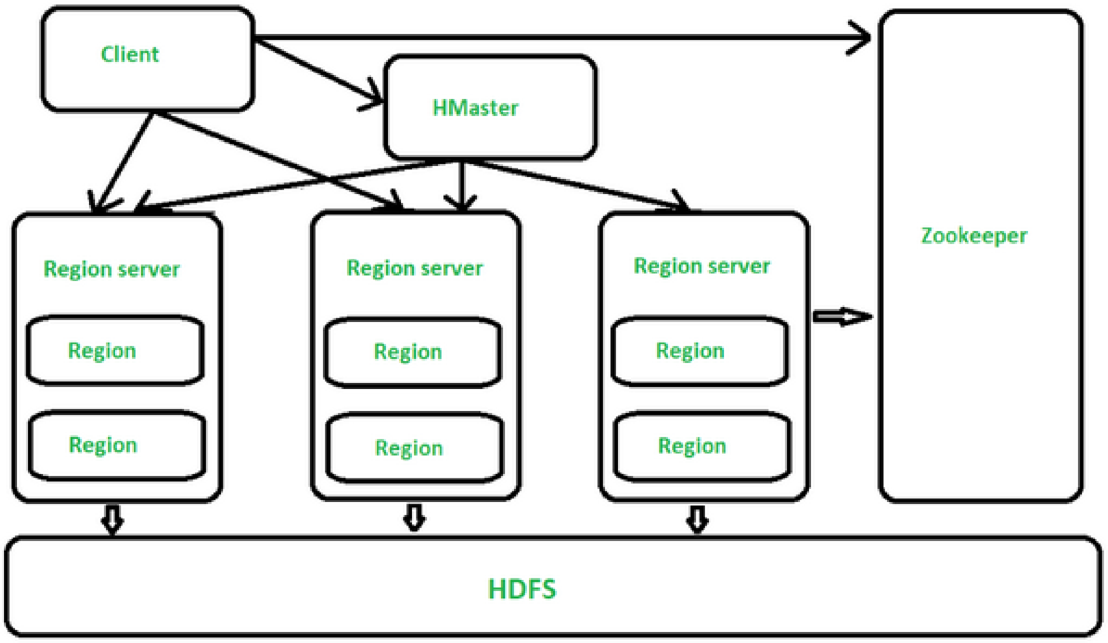


Fig 5 – Hbase Architecture

All the 3 components are described below:

HMaster
The implementation of Master Server in HBase is HMaster. It is a process in which regions are assigned to region server as well as DDL (create, delete table) operations. It monitor all Region Server instances present in the cluster. In a distributed environment, Master runs several background threads. HMaster has many features like controlling load balancing, failover etc.

Region Server
HBase Tables are divided horizontally by row key range into Regions. **Regions** are the basic building elements of HBase cluster that consists of the distribution of tables and are comprised of Column families. Region Server runs on HDFS DataNode which is present in Hadoop cluster. Regions of Region Server are responsible for several things, like handling, managing, executing as well as reads and writes HBase operations on that set of regions. The default size of a region is 256 MB.

Zookeeper
It is like a coordinator in HBase. It provides services like maintaining configuration information, naming, providing distributed synchronization, server failure notification etc. Clients communicate with region servers via zookeeper.

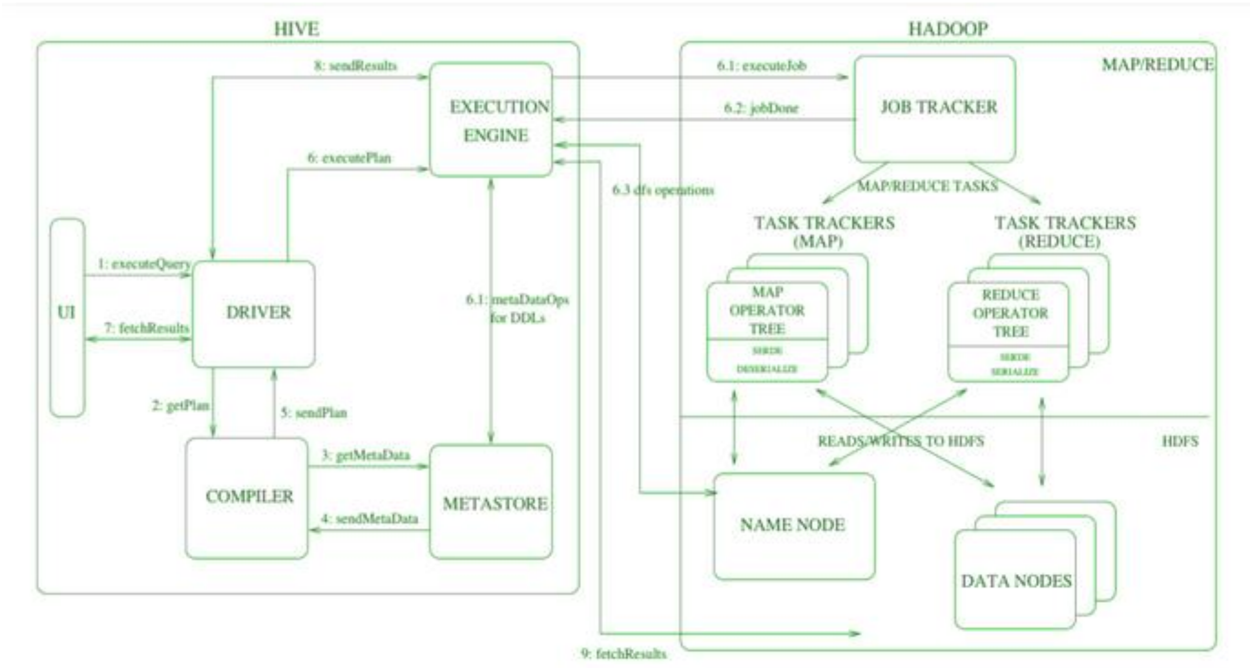


Fig 6 – Hive Execution

in the above diagram along with architecture, *job execution flow in Hive with Hadoop is demonstrated step by step*.

- **Step-1: Execute Query –**
  Interface of the Hive such as Command Line or Web user interface delivers query to the driver to execute. In this, UI calls the execute interface to the driver such as ODBC or JDBC.

- **Step-2: Get Plan –**
  Driver designs a session handle for the query and transfer the query to the compiler to make execution plan. In other words, driver interacts with the compiler.

- **Step-3: Get Metadata –**
  In this, the compiler transfers the metadata request to any database and the compiler gets the necessary metadata from the metastore.

- **Step-4: Send Metadata –**
  Metastore transfers metadata as an acknowledgment to the compiler.

- **Step-5: Send Plan –**
  Compiler communicating with driver with the execution plan made by the compiler to execute the query.

- **Step-6: Execute Plan –**
  Execute plan is sent to the execution engine by the driver.
  - Execute Job
  - Job Done
  - Dfs operation (Metadata Operation)
- **Step-7: Fetch Results –**
  Fetching results from the driver to the user interface (UI).

- **Step-8: Send Results –**
  Result is transferred to the execution engine from the driver. Sending results to Execution engine. When the result is retrieved from data nodes to the execution engine, it returns the result to the driver and to user interface (UI).

10) What do you mean by HiveQL Data Definition Language? Explain any three HiveQL DDL commands with its syntax and example.

Hive Data Definition Language (DDL) is a subset of Hive SQL statements that describe the data structure in Hive by creating, deleting, or altering schema objects such as databases, tables, views, partitions, and buckets. Most Hive DDL statements start with the keywords CREATE , DROP , or ALTER .

Create Database
In Hive, CREATE DATABASE statement is used to create a Database, this takes an optional clause IF NOT EXISTS, using this option, it creates only when database not already exists.

If the Database already exists you will get an error Database db_name already exists.

To check if the database already exists before creating, use IF NOT EXISTS clause.
You can change the location of the database using LOCATION clause

CREATE DATABASE IF NOT EXISTS emp;

Show Databases
Hive by default contains a default database. You can get all databases in Hive using SHOW DATABASES; statement.

```
hive> SHOW DATABASES;
OK
default
emp
Time taken: 0.059 seconds, Fetched: 2 row(s)
```

Use Database
By using the USE command you can set the current database for all subsequent HiveQL statements.

```
hive>USE emp;
```

11) What is Apache Spark

Apache Spark
Industries are using Hadoop extensively to analyze their data sets. The reason is that Hadoop framework is based on a simple programming model (MapReduce) and it enables a computing solution that is scalable, flexible, fault-tolerant and cost effective. Here, the main concern is to maintain speed in processing large datasets in terms of waiting time between queries and waiting time to run the program.
Spark was introduced by Apache Software Foundation for speeding up the Hadoop computational computing software process.
As against a common belief, **Spark is not a modified version of Hadoop** and is not, really, dependent on Hadoop because it has its own cluster management. Hadoop is just one of the ways to implement Spark.
Spark uses Hadoop in two ways – one is **storage** and second is **processing**. Since Spark has its own cluster management computation, it uses Hadoop for storage purpose only.

Apache Spark is a lightning-fast cluster computing technology, designed for fast computation. It is based on Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. The main feature of Spark is its **in-memory cluster computing** that increases the processing speed of an application.
Spark is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries and streaming. Apart from supporting all these workload in a respective system, it reduces the management burden of maintaining separate tools.

Evolution of Apache Spark
Spark is one of Hadoop's sub project developed in 2009 in UC Berkeley's AMPLab by Matei Zaharia. It was Open Sourced in 2010 under a BSD license. It was donated to Apache software foundation in 2013, and now Apache Spark has become a top level Apache project from Feb-2014.

Features of Apache Spark
Apache Spark has following features.

- **Speed** − Spark helps to run an application in Hadoop cluster, up to 100 times faster in memory, and 10 times faster when running on disk. This is possible by reducing number of read/write operations to disk. It stores the intermediate processing data in memory.
- **Supports multiple languages** − Spark provides built-in APIs in Java, Scala, or Python. Therefore, you can write applications in different languages. Spark comes up with 80 high-level operators for interactive querying.
- **Advanced Analytics** − Spark not only supports 'Map' and 'reduce'. It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.

12) Differentiate: Apache pig Vs Map Reduce

MapReduce is a model that works over Hadoop to access big data efficiently stored in HDFS (Hadoop Distributed File System). It is the core component of Hadoop, which divides the big data into small chunks and process them parallelly.

Pig is an open-source tool that is built on the Hadoop ecosystem for providing better processing of Big data. It is a high-level scripting language that is commonly known as Pig Latin scripts. It works on the HDFS (Hadoop Distributed File System), which supports the use of various types of data.

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.

13) Write differences between NoSQL and SQL.

SQL	**NoSQL**
RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)	Non-relational or distributed database system.
These databases have fixed or static or predefined schema	They have dynamic schema
These databases are not suited for hierarchical data storage.	These databases are best suited for hierarchical data storage.
These databases are best suited for complex queries	These databases are not so good for complex queries
Vertically Scalable	Horizontally scalable
Follows ACID property	Follows CAP(consistency, availability, partition tolerance)

14) How is spark faster than Mapreduce?

One of the main differences between Spark and MapReduce is how they process the data. Spark does everything In-Memory while MR persists the data on the disk after map or reduce jobs. So, by any standard Spark can outperform MR quite easily.

So, as long as you have enough memory to fit the data, Spark will do better than Mapreduce. And, this can be clearly observed in the 2014 Daytons GraySort contest (sorting a 100 TB data). Spark completed the task 3 times faster using only a 10th of Hardware compared to Mapreduce. But, this speed comes at a cost. Spark needs a lot of memory. It keeps the data in memory to cache it and so if we have to run Spark alongside some other resource heavy application, we could see a significant drop in performance. Mapreduce does much better in these situations as it removes data immediately when its not required.

So, essentially we can say that Spark is faster than MR when it comes to processing the same amount of data multiple times rather than unloading and loading new data.
So, In conclusion Spark is simpler and usually much faster than Mapreduce for the usual Machine learning and Data Analytics applications. But, that won't put an end to Mapreduce or replace it entirely anytime soon.

To evaluate these improvements, we decided to participate in the Sort Benchmark. With help from Amazon Web Services, we participated in the Daytona Gray category, an industry benchmark on how fast a system can sort 100 TB of data (1 trillion records). Although our entry is still under review, we are eager to share with you our submission. The previous world record was 72 minutes, set by Yahoo using a Hadoop MapReduce cluster of 2100 nodes. Using Spark on 206 EC2 nodes, we completed the benchmark in 23 minutes. This means that Spark sorted the same data 3X faster using 10X fewer machines. All the sorting took place on disk (HDFS), without using Spark’s in-memory cache.

Additionally, while no official petabyte (PB) sort competition exists, we pushed Spark further to also sort 1 PB of data (10 trillion records) on 190 machines in under 4 hours. This PB time beats previously reported results based on Hadoop MapReduce in 2009 (16 hours on 3800 machines). To the best of our knowledge, this is the first petabyte-scale sort ever done in a public cloud.

15) Differentiate between HBase Vs. RDBMS.

Relational Database Management System (RDBMS)
RDBMS is for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access. A Relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as introduced by E. F. Codd. An RDBMS is a type of DBMS with a row-based table structure that connects related data elements and includes functions that maintain the security, accuracy, integrity and consistency of the data. The most basic RDBMS functions are create, read, update and delete operations.Hbase follows the ACID Properties.

HBase
HBase is a column-oriented database management system that runs on top of Hadoop Distributed File System (HDFS). It is well suited for sparse data sets, which are common in many big data use cases. It is an opensource, distributed database developed by Apache software foundations. Initially, it was named Google Big Table, afterwards it was re-named as HBase and is primarily written in Java. It can store massive amount of data from terabytes to petabytes. It is built for low-latency operations and is used extensively for read and write operations. It stores large amount of data in the form of tables.

RDBMS	**HBase**
It requires SQL (structured query language)	NO SQL
It has a fixed schema	No fixed schema
It is row oriented	It is column oriented
It is not scalable	It is scalable
It is static in nature	Dynamic in nature
Slower retrieval of data	Faster retrieval of data
It follows the ACID (Atomicity, Consistency, Isolation and Durability) property.	It follows CAP (Consistency, Availability,Partition-tolerance) theorem.
It can handle structured data	It can handle structured, unstructured as well as semi-structured data
It cannot handle sparse data	It can handle sparse data

16) What is RDD? Explain properties of RDD.

Resilient Distributed Datasets
Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which

may be computed on different nodes of the cluster. RDDs can contain any type of Python, Java, or Scala objects, including user-defined classes.

Formally, an RDD is a read-only, partitioned collection of records. RDDs can be created through deterministic operations on either data on stable storage or other RDDs. RDD is a fault-tolerant collection of elements that can be operated on in parallel.

There are two ways to create RDDs − parallelizing an existing collection in your driver program, or referencing a dataset in an external storage system, such as a shared file system, HDFS, HBase, or any data source offering a Hadoop Input Format.

Spark makes use of the concept of RDD to achieve faster and efficient MapReduce operations. Let us first discuss how MapReduce operations take place and why they are not so efficient.

In-memory computation
The data inside RDD are stored in memory for as long as you want to store. Keeping the data in-memory improves the performance by an order of magnitudes. refer this comprehensive guide to Learn Spark in-memory computation in detail.

ii. Lazy Evaluation
The data inside RDDs are not evaluated on the go. The changes or the computation is performed only after an action is triggered. Thus, it limits how much work it has to do. Follow this guide to learn Spark lazy evaluation in great detail.

iii. Fault Tolerance
Upon the failure of worker node, using lineage of operations we can re-compute the lost partition of RDD from the original one. Thus, we can easily recover the lost data. Learn Fault tolerance is Spark in detail.

iv. Immutability
RDDS are immutable in nature meaning once we create an RDD we can not manipulate it. And if we perform any transformation, it creates new RDD. We achieve consistency through immutability.

v. Persistence
We can store the frequently used RDD in in-memory and we can also retrieve them directly from memory without going to disk, this speedup the execution. We can perform Multiple operations on the same data, this happens by storing the data explicitly in memory by calling persist() or cache() function. Follow this guide for the detailed study of RDD persistence in Spark.

vi. Partitioning
RDD partition the records logically and distributes the data across various nodes in the cluster. The logical divisions are only for processing and internally it has no division. Thus, it provides parallelism.

vii. Parallel
Rdd, process the data parallelly over the cluster.

viii. Location-Stickiness
RDDs are capable of defining placement preference to compute partitions. Placement preference refers to information about the location of RDD. The DAGScheduler places the partitions in such a way that task is close to data as much as possible. Thus speed up computation.

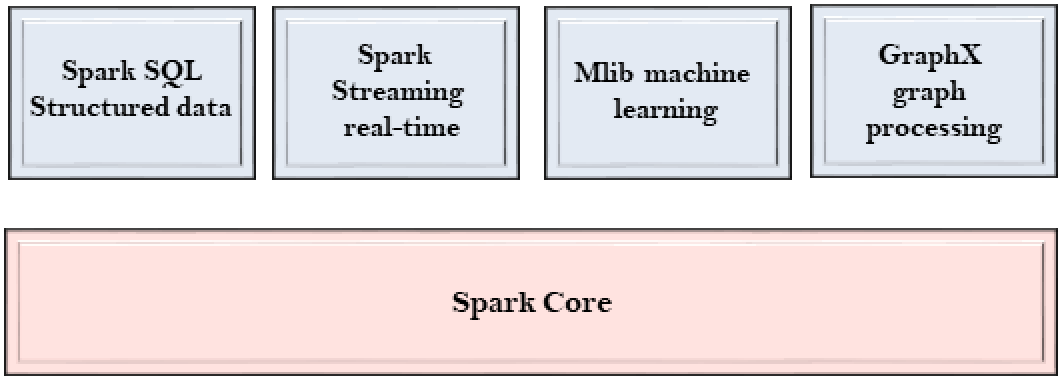
ix. Coarse-grained Operation
We apply coarse-grained transformations to RDD. Coarse-grained meaning the operation applies to the whole dataset not on an individual element in the data set of RDD.

17) Differentiate between HDFS Vs. HBase.

HDFS	HBase
HDFS is a java based **file distribution system**	Hbase is **hadoop database** that runs on top of HDFS
HDFS is **highly fault-tolerant** and **cost-effective**	HBase is **partially tolerant** and **highly consistent**
HDFS Provides only **sequential read/write operation**	**Random access** is possible due to hash table
HDFS is based on **write once read many times**	HBase supports **random read and write**operation into filesystem
HDFS has a **rigid architecture**	HBase support **dynamic changes**
HDFS is prefereable for **offline batch processing**	HBase is preferable for **real time processing**
HDFS provides **high latency** for access operations.	HBase provides **low latency** access to small amount of data

18) Explain Component of Spark.

The Spark project consists of different types of tightly integrated components. At its core, Spark is a computational engine that can schedule, distribute and monitor multiple applications.
Let's understand each Spark component in detail.



Spark Core

- The Spark Core is the heart of Spark and performs the core functionality.
- It holds the components for task scheduling, fault recovery, interacting with storage systems and memory management.

Spark SQL

- The Spark SQL is built on the top of Spark Core. It provides support for structured data.
- It allows to query the data via SQL (Structured Query Language) as well as the Apache Hive variant of SQL?called the HQL (Hive Query Language).
- It supports JDBC and ODBC connections that establish a relation between Java objects and existing databases, data warehouses and business intelligence tools.
- It also supports various sources of data like Hive tables, Parquet, and JSON.

Spark Streaming

- Spark Streaming is a Spark component that supports scalable and fault-tolerant processing of streaming data.
- It uses Spark Core's fast scheduling capability to perform streaming analytics.
- It accepts data in mini-batches and performs RDD transformations on that data.
- Its design ensures that the applications written for streaming data can be reused to analyze batches of historical data with little modification.
- The log files generated by web servers can be considered as a real-time example of a data stream.

MLlib

- The MLlib is a Machine Learning library that contains various machine learning algorithms.
- These include correlations and hypothesis testing, classification and regression, clustering, and principal component analysis.
- It is nine times faster than the disk-based implementation used by Apache Mahout.

GraphX

- The GraphX is a library that is used to manipulate graphs and perform graph-parallel computations.
- It facilitates to create a directed graph with arbitrary properties attached to each vertex and edge.
- To manipulate graph, it supports various fundamental operators like subgraph, join Vertices, and aggregate Messages.

19) Write a short note on Pig.

Pig is a high-level platform or tool which is used to process the large datasets. It provides a high-level of abstraction for processing over the MapReduce. It provides a high-level scripting language, known as *Pig Latin* which is used to develop the data analysis codes. First, to process the data which is stored in the HDFS, the programmers will write the scripts using the Pig Latin Language. Internally *Pig Engine*(a component of Apache Pig) converted all these scripts into a specific map and reduce task. But these are not visible to the programmers in order to provide a high-level of abstraction. Pig Latin and Pig Engine are the two main components of the Apache Pig tool. The result of Pig always stored in the HDFS.

Need of Pig: One limitation of MapReduce is that the development cycle is very long. Writing the reducer and mapper, compiling packaging the code, submitting the job and retrieving the output is a time-consuming task. Apache Pig reduces the time of development using the multi-query approach. Also, Pig is beneficial for the programmers who are not from java background. 200 lines of Java code can be written in only 10 lines using the Pig Latin language. Programmers who have SQL knowledge needed less effort to learn Pig Latin.

Evolution of Pig: Earlier in 2006, Apache Pig was developed by Yahoo's researchers. At that time, the main idea to develop Pig was to execute the MapReduce jobs on extremely large datasets. In the year 2007, it moved to Apache Software Foundation(ASF) which makes it an open source project. The first version(*0.1*) of Pig came in the year 2008. The latest version of Apache Pig is *0.18* which came in the year 2017.

Features of Apache Pig:

- For performing several operations Apache Pig provides rich sets of operators like the filters, join, sort, etc.
- Easy to learn, read and write. Especially for SQL-programmer, Apache Pig is a boon.
- Apache Pig is extensible so that you can make your own user-defined functions and process.
- Join operation is easy in Apache Pig.
- Fewer lines of code.
- Apache Pig allows splits in the pipeline.
- The data structure is multivalued, nested, and richer.
- Pig can handle the analysis of both structured and unstructured data.

20) Compare Raw oriented and Column Oriented database structures.

Row oriented data stores	**Column oriented data stores**
Data is stored and retrieved one row at a time and hence could read unnecessary data if some of the data in a row are required.	In this type of data stores, data are stored and retrieve in columns and hence it can only able to read only the relevant data if required.
Records in Row Oriented Data stores are easy to read and write.	In this type of data stores, read and write operations are slower as compared to row-oriented.
Row-oriented data stores are best suited for online transaction system.	Column-oriented stores are best suited for online analytical processing.
These are not efficient in performing operations applicable to the entire datasets and hence aggregation in row-oriented is an expensive job or operations.	These are efficient in performing operations applicable to the entire dataset and hence enables aggregation over many rows and columns.
Typical compression mechanisms which provide less efficient result than what we achieve from column-oriented data stores.	These type of data stores basically permits high compression rates due to little distinct or unique values in columns.

21) What is data serialization? With proper examples discuss and differentiate structured, unstructured and semi-structured data. Make a note on how type of data affects data

Serialization
Data serialization is the process of converting data objects present in complex data structures into a byte stream for storage, transfer and distribution purposes on physical devices.

Structured data
Structured data is data whose elements are addressable for effective analysis. It has been organized into a formatted repository that is typically a database. It concerns all data which can be stored in database SQL in a table with rows and columns. They have relational keys and can easily be mapped into pre-designed fields. Today, those data are most processed in the development and simplest way to manage information. Example: Relational data.

Semi-Structured data
Semi-structured data is information that does not reside in a relational database but that has some organizational properties that make it easier to analyze. With some processes, you can store them in the relation database (it could be very hard for some kind of semi-structured data), but Semi-structured exist to ease space. Example: XML data.

Unstructured data
Unstructured data is a data which is not organized in a predefined manner or does not have a predefined data model, thus it is not a good fit for a mainstream relational database. So for Unstructured data, there are alternative platforms for storing and managing, it is increasingly prevalent in IT systems and is used by organizations in a variety of business intelligence and analytics applications. Example: Word, PDF, Text, Media logs.

Computer systems may vary in their hardware architecture, OS, addressing mechanisms. Internal binary representations of data also vary accordingly in every environment. Storing and exchanging data between such varying environments requires a platform-and-language-neutral data format that all systems understand.

Once the serialized data is transmitted from the source machine to the destination machine, the reverse process of creating objects from the byte sequence called deserialization is carried out. Reconstructed objects are clones of the original object.

Choice of data serialization format for an application depends on factors such as data complexity, need for human readability, speed and storage space constraints. XML, JSON, BSON, YAML, MessagePack, and protobuf are some commonly used data serialization formats.

22) What is transformation and actions in Apache Spark? Discuss various commands available for this activity in Apache Spark?

Spark Transformation is a function that produces new RDD from the existing RDDs. It takes RDD as input and produces one or more RDD as output. Each time it creates new RDD when we apply any transformation. Thus, the so input RDDs, cannot be changed since RDD are immutable in nature.

There are various functions in RDD transformation. Let us see RDD transformation with examples

1. map(func)
The map function iterates over every line in RDD and split into new RDD. Using map() transformation we take in any function, and that function is applied to every element of RDD. In the map, we have the flexibility that the input and the return type of RDD may differ from each other. For example, we can have input RDD type as String, after applying the map() function the return RDD can be Boolean.
For example, in RDD {1, 2, 3, 4, 5} if we apply "rdd.map(x=>x+2)" we will get the result as (3, 4, 5, 6, 7).

2. FlatMap()
With the help of flatMap() function, to each input element, we have many elements in an output RDD. The most simple use of flatMap() is to split each input string into words.

3.filter(func)
Spark RDD filter() function returns a new RDD, containing only the elements that meet a predicate. It is a narrow operation because it does not shuffle data from one partition to many partitions.

4. mapPartitions(func)
5.mapPartitionWithIndex()
6.union(dataset)
7.intersection(other-dataset)
8.distinct()

RDD Action
Transformations create RDDs from each other, but when we want to work with the actual dataset, at that point action is performed. When the action is triggered after the result, new RDD is not formed like transformation. Thus, Actions are Spark RDD operations that give non-RDD values. The values of action are stored to drivers or to the external storage system. It brings laziness of RDD into motion.

1.count()
Action count() returns the number of elements in RDD.
2.collect()
The action collect() is the common and simplest operation that returns our entire RDDs content to driver program.
3.take(n)
The action take(n) returns n number of elements from RDD.
4.top()
If ordering is present in our RDD, then we can extract top elements from our RDD using top(). Action top() use default ordering of data.
5.countByValue()
The countByValue() returns, many times each element occur in RDD.
6.foreach()
When we have a situation where we want to apply operation on each element of RDD, but it should not return value to the driver. In this case, foreach() function is useful. For example, inserting a record into the database.

23. Explain traditional stream processing Architecture.

Traditional Processing is a kind of offline processing that involves simple computations on data when it is being processed. It mainly stores raw data that is not so much aggregated and structured. It has various operations like pre-processing and extraction of data which is performed on the raw data. The data goes through a chain of algorithms in order to be processed.

To process the data, most traditional stream processing systems are designed with a continuous operator model, which works as follows:

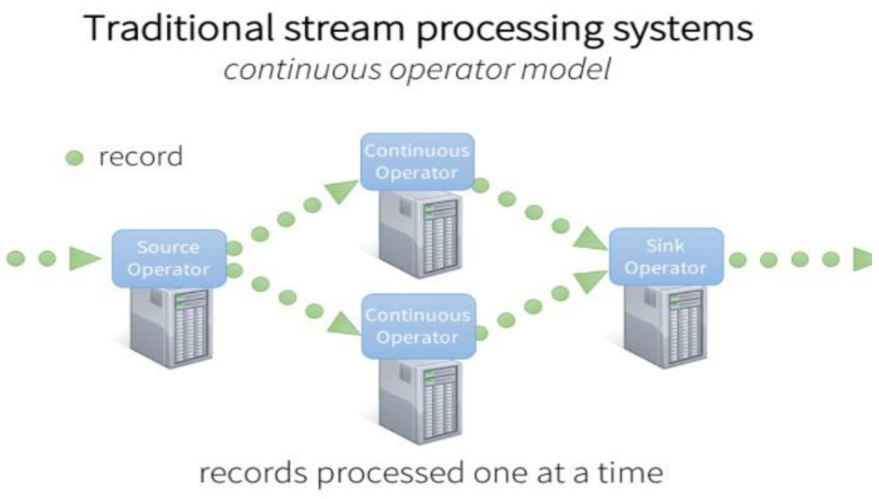

Traditional stream processing systems
continuous operator model
record
Continuous Operator
Source Operator
Sink Operator
Continuous Operator
records processed one at a time

Figure 1: Architecture of traditional stream processing systems

There is a set of worker nodes, each of which run one or more continuous operators.

Each continuous operator processes the streaming data one record at a time and forwards the records to other operators in the pipeline.

There are “source” operators for receiving data from ingestion systems, and “sink” operators that output to downstream systems.

Continuous operators are a simple and natural model. However, with today’s trend towards larger scale and more complex real-time analytics, this traditional architecture has also met some challenges. We designed Spark Streaming to satisfy the following requirements:

Fast failure and straggler recovery – With greater scale, there is a higher likelihood of a cluster node failing or unpredictably slowing down (i.e. stragglers). The system must be able to automatically recover from failures and stragglers to provide results in real time. Unfortunately, the static allocation of continuous operators to worker nodes makes it challenging for traditional systems to recover quickly from faults and stragglers.

Load balancing – Uneven allocation of the processing load between the workers can cause bottlenecks in a continuous operator system. This is more likely to occur in large clusters and dynamically varying workloads. The system needs to be able to dynamically adapt the resource allocation based on the workload.

Unification of streaming, batch and interactive workloads – In many use cases, it is also attractive to query the streaming data interactively (after all, the streaming system has it all in memory), or to combine it with static datasets (e.g. pre-computed models). This is hard in continuous operator systems as they are not designed to the dynamically introduce new operators for ad-hoc queries. This requires a single engine that can combine batch, streaming and interactive queries.

Advanced analytics like machine learning and SQL queries – More complex workloads require continuously learning and updating data models, or even querying the “latest” view of streaming data with SQL queries. Again, having a common abstraction across these analytic tasks makes the developer’s job much easier.

To address these requirements, Spark Streaming uses a new architecture called discretized streams that directly leverages the rich libraries and fault tolerance of the Spark engine.