CMR
INSTITUTE OF
TECHNOLOGY

USN | | | | | | | | | | |

Internal Assessment Test 1 – Nov 2025

Sub:	Big Data Analytics							Sub Code:	MMCA311B
Date:	05.11.25	Duration:	90 min's	Max Marks:	50	Sem:	III	Branch:	MCA

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

	MARKS	OBE	
		CO	RBT
PART I			
1 Define Big Data. Briefly explain its importance in today's digital world.	[10]	CO1	L1
OR			
2 Differentiate between structured, semi-structured, and unstructured data with examples.	[10]	CO1	L3
PART II			
3 Define Hadoop. What are its main components?	[10]	CO1	L1
OR			
4 What are the advantages of Apache Spark over Hadoop MapReduce?	[10]	CO1	L1
PART III			
5 Define HDFS and explain its purpose in the Hadoop ecosystem.	[10]	CO2	L1
OR			
6 What is the function of the NameNode and DataNode in HDFS.	[10]	CO2	L1
PART IV			
7 Differentiate between the Map and Reduce phases in the MapReduce model.	[10]	CO2	L3
OR			
8 What are Combiners, Partitioners, and Counters in Hadoop? Explain with examples.	[10]	CO2	L1
PART V			
9 Explain the roles of the Driver**,** Executors**,** and Cluster Manager in the Spark ecosystem.	[10]	CO3	L2
OR			
10 List the main components of the Spark framework and mention one function of each.	[10]	CO3	L2

Solution

1 Define Big Data. Briefly explain its importance in today's digital world.

Big Data refers to extremely large, diverse, and complex datasets that are too massive for traditional processing tools. Its importance lies in its ability to be analyzed for insights that improve decision-making, personalize customer experiences, enhance operational efficiency, and drive innovation in today's digital world.

Definition

Vast volume: Big data involves petabytes or exabytes of information, a volume too large to be stored or processed by conventional databases.

High velocity: It is generated at high speed from sources like social media, sensors, and transactions, requiring real-time or near-real-time processing.

Great variety: This includes structured (like spreadsheets), semi-structured, and unstructured data (like text, images, and videos) from a wide range of sources.

Need for advanced processing: Due to its scale and complexity, big data requires advanced technologies and analytics tools, such as machine learning, to extract meaningful patterns and insights.

Importance in the digital world

Enhanced decision-making: Big data analytics helps organizations identify trends, patterns, and correlations to make more informed and confident strategic decisions.

Improved customer experiences: By analyzing customer behavior, companies can personalize products, services, and interactions, leading to higher satisfaction and loyalty.

Greater operational efficiency: Analyzing data can reveal inefficiencies in processes, allowing businesses to optimize supply chains, manage resources more effectively, and reduce costs.

Faster innovation: Insights from big data can help identify new market opportunities, leading to faster development of innovative products and services.

Advanced risk management: Financial institutions, for example, use big data to detect fraud and manage risk more effectively by analyzing vast amounts of transaction data in real-time.

The 5 V's of Big Data

Volume: Refers to the huge amount of data generated every second-ranging from terabytes to petabytes. Example: YouTube uploads 500+ hours of video every minute.

Velocity: The speed at which data is created, shared, and processed. Data streams in from sensors, social media, and transactions in real-time.

Variety: Data comes in multiple formats-text, audio, images, videos, logs, sensor data, etc. Handling all these types together is complex

Veracity: Refers to the trustworthiness and accuracy of the data. Inconsistent, duplicated, or noisy data can lead to wrong insights.

Value: Not all data is useful. The key is extracting relevant data and turning it into business value through analytics.

2 Differentiate between structured, semi-structured, and unstructured data with examples.

Big Data includes huge volume, high velocity, and extensible variety of data. There are 3 types: Structured data, Semi-structured data, and Unstructured data.

1. **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.

2. **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.

3. **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.

Differences between Structured, Semi-structured and Unstructured data:

Properties	Structured data	Semi-structured data	Unstructured data
Technology	It is based on Relational database table	It is based on XML/RDF(Resource Description Framework).	It is based on character and binary data
Transaction management	Matured transaction and various concurrency techniques	Transaction is adapted from DBMS not matured	No transaction management and no concurrency
Version management	Versioning over tuples,row,tables	Versioning over tuples or graph is possible	Versioned as a whole
Flexibility	It is schema dependent and less flexible	It is more flexible than structured data but less flexible than unstructured data	It is more flexible and there is absence of schema
Scalability	It is very difficult to scale DB schema	It's scaling is simpler than structured data	It is more scalable.
Robustness	Very robust	New technology, not very spread	--
Query performance	Structured query allow complex joining	Queries over anonymous nodes are possible	Only textual queries are possible

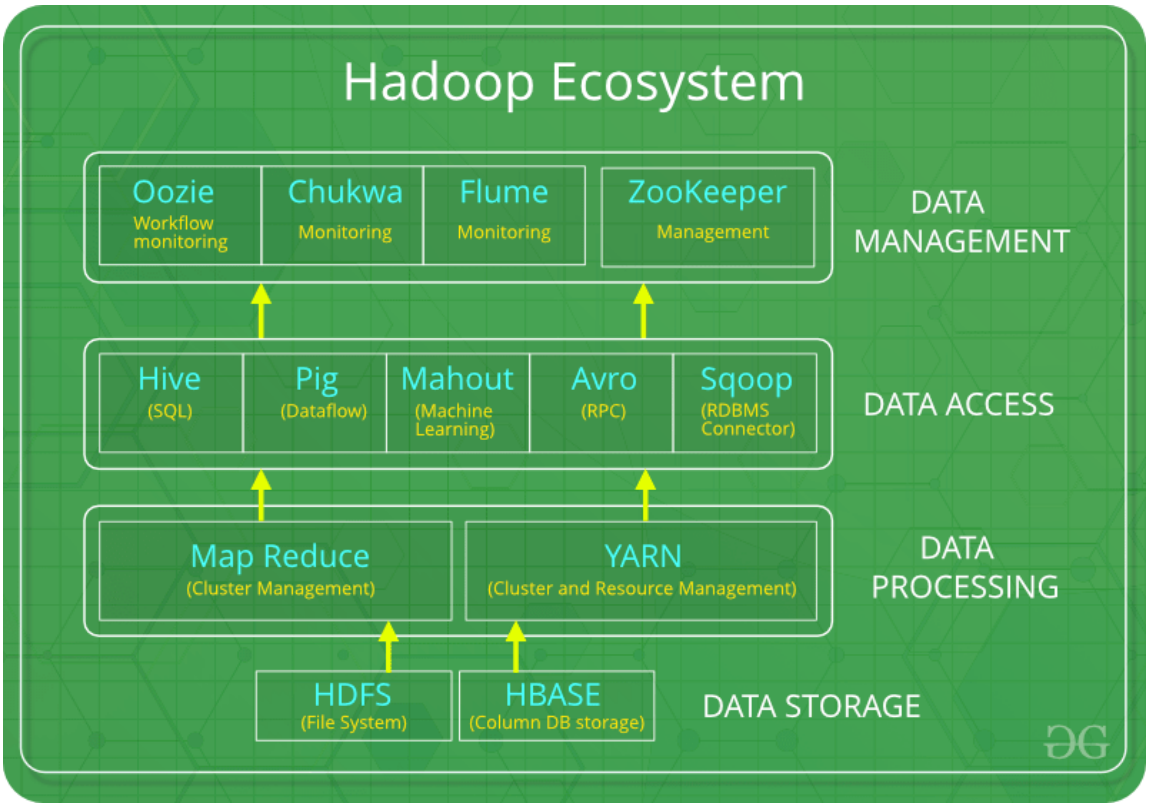
3. Define Hadoop. What are its main components?

Hadoop is an open-source framework for storing and processing large-scale data across distributed clusters using commodity hardware. The Hadoop Ecosystem is a suite of tools and technologies built around Hadoop's core components (HDFS, YARN, MapReduce and Hadoop Common) to enhance its capabilities in data storage, processing, analysis and management.

Components of Hadoop Ecosystem

Hadoop Ecosystem comprises several components that work together for efficient big data storage and processing:

- **HDFS (Hadoop Distributed File System):** Stores large datasets across distributed nodes.
- **YARN (Yet Another Resource Negotiator):** Manages cluster resources and job scheduling.
- **MapReduce:** A programming model for batch data processing.
- **Spark:** Provides fast, in-memory data processing.
- **Hive & Pig:** High-level tools for querying and analyzing large datasets.
- **HBase:** A NoSQL database for real-time read/write access.
- **Mahout & Spark MLlib:** Libraries for scalable machine learning.
- **Solr & Lucene:** Tools for full-text search and indexing.
- **Zookeeper:** Manages coordination and configuration across the cluster.
- **Oozie:** A workflow scheduler for managing Hadoop jobs.



4. What are the advantages of Apache Spark over Hadoop MapReduce?

MapReduce is a framework the use of which we can write functions to process massive quantities of data, in parallel, on giant clusters of commodity hardware in a dependable manner. It is also a processing method and an application model for dispensed computing primarily based on java. The MapReduce algorithm incorporates two necessary tasks, particularly Map and Reduce. The map takes a set of records and converts it into every other set of data, where individual factors are broken down into tuples that are present in key-value pairs. Also, it helps in minimizing task, which takes the output from a map as an enter and combines those statistics tuples into a smaller set of tuples. As the sequence of the title MapReduce implies, the decrease assignment is continually carried out after the map job.

Apache Spark is a data processing framework that can rapidly operate processing duties on very massive information sets, and can additionally distribute information processing duties throughout a couple of computers, either on its very own or in tandem with different allotted computing tools. These two features are key to the worlds of massive information and machine learning, which require the marshaling of large computing energy to crunch via massive information stores. Spark additionally takes some of the programming burdens of these duties off the shoulders of developers with an easy use API that abstracts away a whole lot of the grunt work of distributed computing and large information processing.

S.No.	MapReduce	Spark
1.	It is a framework that is open-source which is used for writing data into the Hadoop Distributed File System.	It is an open-source framework used for faster data processing.
2.	It is having a very slow speed as compared to Apache Spark.	It is much faster than MapReduce.
3.	It is unable to handle real-time processing.	It can deal with real-time processing.
4.	It is difficult to program as you required code for every process.	It is easy to program.
5.	It supports more security projects.	Its security is not as good as MapReduce and continuously working on its security issues.
6.	For performing the task, It is unable to cache in memory.	It can cache the memory data for processing its task.
7.	Its scalability is good as you can add up to n different nodes.	It is having low scalability as compared to MapReduce.
8.	It actually needs other queries to perform the task.	It has Spark SQL as its very own query language.

5. Define HDFS and explain its purpose in the Hadoop ecosystem

The Hadoop Distributed File System (HDFS) is a key component of the Apache Hadoop ecosystem, designed to store and manage large volumes of data across multiple machines in a distributed manner. It provides high-throughput access to data, making it suitable for applications that deal with large datasets, such as big data analytics, machine learning, and data warehousing. This article will delve into the architecture of HDFS, explaining its key components and mechanisms, and highlight the advantages it offers over traditional file systems.

HDFS Architecture

HDFS is designed to be highly scalable, reliable, and efficient, enabling the storage and processing of massive datasets. Its architecture consists of several key components:

1. NameNode
2. DataNode
3. Secondary NameNode
4. HDFS Client
5. Block Structure

NameNode

The NameNode is the master server that manages the filesystem namespace and controls access to files by clients. It performs operations such as opening, closing, and renaming files and directories. Additionally, the NameNode maps file blocks to DataNodes, maintaining the metadata and the overall structure of the file system. This metadata is stored in memory for fast access and persisted on disk for reliability.

Key Responsibilities:

- Maintaining the filesystem tree and metadata.
- Managing the mapping of file blocks to DataNodes.
- Ensuring data integrity and coordinating replication of data blocks.

Characteristics of HDFS
Below are some characteristics of HDFS:

- HDFS has high fault-tolerance
- HDFS may consist of thousands of server machines. Each machine stores a part of the file system data. HDFS detects faults that can occur on any of the machines and recovers it quickly and automatically.
- HDFS has high throughput
- HDFS is designed to store and scan millions of rows of data and to count or add some subsets of the data. The time required in this process is dependent on the complexities involved.
- It has been designed to support large datasets in batch-style jobs. However, the emphasis is on high throughput of data access rather than low latency.
- HDFS is economical
- HDFS is designed in such a way that it can be built on commodity hardware and heterogeneous platforms, which is low-priced and easily available.

6. What is the function of the NameNode and DataNode in HDFS

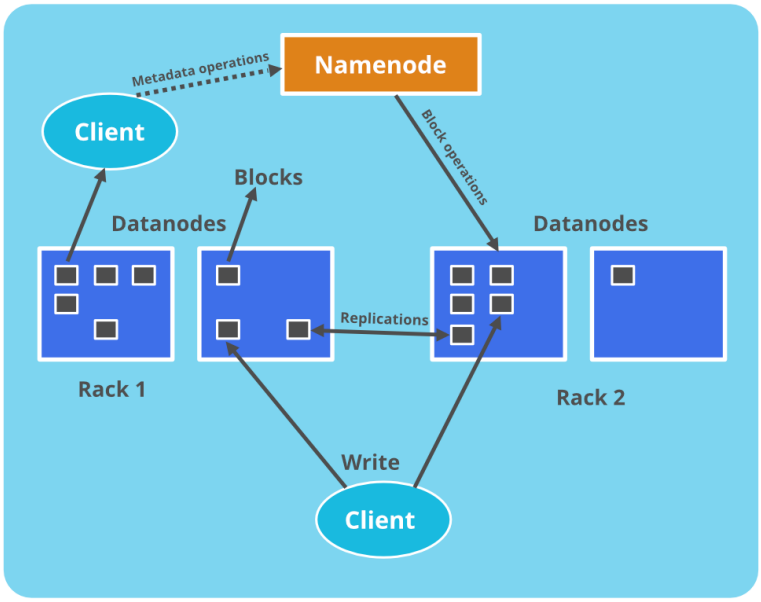
DataNode

DataNodes are the worker nodes in HDFS, responsible for storing and retrieving actual data blocks as instructed by the NameNode. Each DataNode manages the storage attached to it and periodically reports the list of blocks it stores to the NameNode.

HDFS Client

The HDFS client is the interface through which users and applications interact with the HDFS. It allows for file creation, deletion, reading, and writing operations. The client communicates with the NameNode to determine which DataNodes hold the blocks of a file and interacts directly with the DataNodes for actual data read/write operations.
Key Responsibilities:

- Facilitating interaction between the user/application and HDFS.
- Communicating with the NameNode for metadata and with DataNodes for data access.



7. Differentiate between the Map and Reduce phases in the MapReduce model.

Map Reduce is a framework in which we can write applications to run huge amount of data in parallel and in large cluster of commodity hardware in a reliable manner.

Phases of MapReduce

MapReduce model has three major and one optional phase.

1. Mapping
2. Shuffling and Sorting

3. Reducing
4. Combining

1) Mapping

It is the first phase of MapReduce programming. Mapping Phase accepts key-value pairs as input as (k, v), where the key represents the Key address of each record and the value represents the entire record content.The output of the Mapping phase will also be in the key-value format (k', v').

2) Shuffling and Sorting

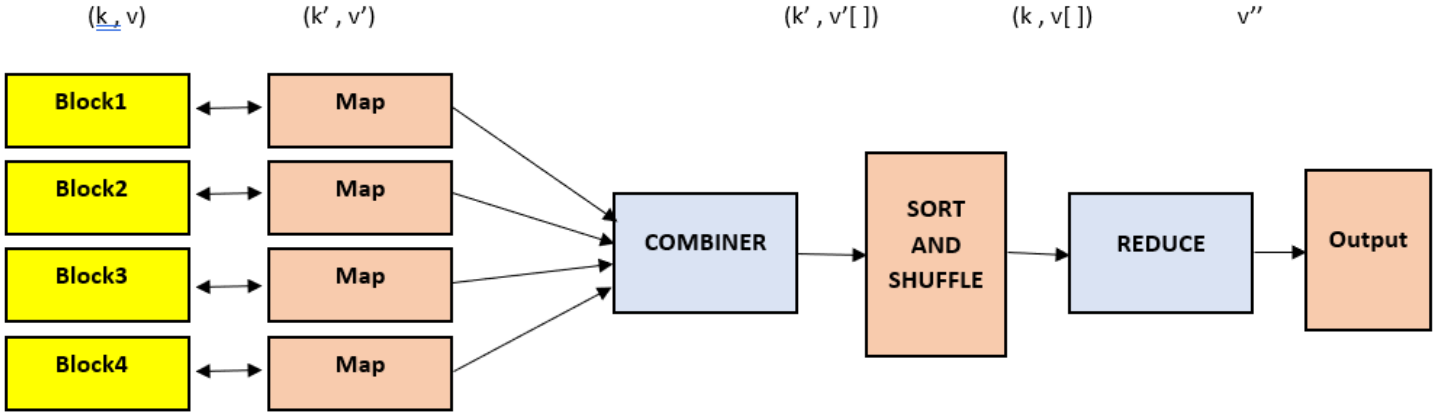
The output of various mapping parts (k', v'), then goes into Shuffling and Sorting phase. All the same values are deleted, and different values are grouped together based on same keys. The output of the Shuffling and Sorting phase will be key-value pairs again as key and array of values (k, v[ ]).

3) Reducer

The output of the Shuffling and Sorting 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 Collected against their corresponding keys. Reducer stabilize outputs of various mappers and computes the final output.

4) Combining

It is an optional phase in the MapReduce phases . The combiner phase is used to optimize the performance of MapReduce phases. This phase makes the Shuffling and Sorting phase work even quicker by enabling additional performance features in MapReduce phases.



8. What are Combiners, Partitioners, and Counters in Hadoop? Explain with examples.

In Hadoop MapReduce, Combiners, Partitioners, and Counters are components that enhance the efficiency and monitoring of data processing.

Combiners

A Combiner, also known as a "mini-reducer," is an optional optimization that performs local aggregation of intermediate key-value pairs produced by the Mappers before they are sent to the Reducers. This reduces the amount of data shuffled across the network, improving performance. The Combiner's logic must be commutative and associative, meaning the order of operations does not affect the final result.

Example: In a word count job, if a Mapper processes a text file and outputs `(word, 1)` for each occurrence, a Combiner can aggregate these locally. Instead of sending `(apple, 1), (apple, 1), (banana, 1)` to the Reducer, the Combiner would send `(apple, 2), (banana, 1)`.

Partitioners

A Partitioner determines which Reducer will receive a particular key-value pair from the Mapper's output. It ensures that all values for a given key are sent to the same Reducer, which is crucial for correct aggregation. The default Partitioner uses a hash function on the key (e.g., `key.hashCode() % numberOfReducers`).

Example: In a dataset of sales transactions, if you want to sum sales by region, a custom Partitioner could ensure that all transactions for "North America" go to one Reducer, "Europe" to another, and so on, based on the region key.

Counters

Counters are used to track various statistics and events during a MapReduce job. They can be framework-defined (e.g., bytes read, records written) or user-defined. User-defined counters allow developers to track application-specific metrics. Counters are aggregated globally and displayed in the job's progress reports.

Example: In a log analysis job, a user-defined counter could track the number of "ERROR" messages encountered. The Mapper would increment this counter each time it processes a log entry containing "ERROR," providing a summary of errors at the end of the job.

9. Explain the roles of the Driver, Executors, and Cluster Manager in the Spark ecosystem.

In the Apache Spark ecosystem, the Driver, Executors, and Cluster Manager are key components that work together to execute Spark applications.

1. Spark Driver:

- **Role:**

The Driver is the central coordinator of a Spark application. It runs the main() function of the application and creates the SparkContext, which is the entry point to Spark functionality.

- **Responsibilities:**
  - Translates the user's Spark code (transformations and actions) into a Directed Acyclic Graph (DAG) of execution stages.
  - Communicates with the Cluster Manager to request and acquire resources (Executors) for the application.
  - Divides the application into tasks and schedules them to be run on the Executors.
  - Monitors the progress of tasks and collects the results from the Executors.
  - Manages the lifecycle of the Spark application.

2. Spark Executors:

- **Role:**

Executors are worker processes that run on the worker nodes within the Spark cluster. They are responsible for executing the individual tasks assigned by the Driver.

- **Responsibilities:**
  - Execute the computational tasks assigned by the Driver.
  - Store data in memory or on disk for caching and iterative computations.
  - Report the results of executed tasks back to the Driver.
  - Communicate with each other for shuffle operations (data exchange between stages).

3. Cluster Manager:

- **Role:**

The Cluster Manager is an external service responsible for resource management and allocation across the cluster. Spark applications can be deployed on various cluster managers like Standalone, YARN, Mesos, or Kubernetes.

- **Responsibilities:**
  - Acquires resources (CPU, memory) on the worker nodes for the Spark application.
  - Launches the Executors on the worker nodes as requested by the Driver.
  - Manages the overall cluster resources, including resource arbitration between multiple applications.
  - Provides a mechanism for the Driver to communicate with the Executors.

10 List the main components of the Spark framework and mention one function of each.

Spark is a cluster computing system. It is faster as compared to other cluster computing systems (such as Hadoop). It provides high-level APIs in Python, Scala, and Java. Parallel jobs are easy to write in Spark. In this article, we will discuss the different components of Apache Spark.

Spark processes a huge amount of datasets and it is the foremost active Apache project of the current time. Spark is written in Scala and provides API in Python, Scala, Java, and R. The most vital feature of Apache Spark is its in-memory cluster computing that extends the speed of the data process. Spark is an additional general and quicker processing platform. It helps us to run programs relatively quicker than Hadoop (i.e.) a hundred times quicker in memory and ten times quicker even on the disk. The main features of spark are:

1. Multiple Language Support: Apache Spark supports multiple languages; it provides API's written in Scala, Java, Python or R. It permits users to write down applications in several languages.
2. Quick Speed: The most vital feature of Apache Spark is its processing speed. It permits the application to run on a Hadoop cluster, up to one hundred times quicker in memory, and ten times quicker on disk.
3. Runs Everywhere: Spark will run on multiple platforms while not moving the processing speed. It will run on Hadoop, Kubernetes, Mesos, Standalone, and even within the Cloud.
4. General Purpose: It is powered by plethora libraries for machine learning (i.e.) MLlib, DataFrames, and SQL at the side of Spark Streaming and GraphX. It is allowed to use a mix of those libraries which are coherently associated with the application. The feature of mix streaming, SQL, and complicated analytics, within the same application, makes Spark a general framework.
5. Advanced Analytics: Apache Spark also supports "Map" and "Reduce" that has been mentioned earlier. However, at the side of MapReduce, it supports Streaming data, SQL queries, Graph algorithms, and Machine learning.

Thus, Apache Spark may be used to perform advanced analytics.

The above figure illustrates all the spark components. Let's understand each of the components in detail:

1. Spark Core: All the functionalities being provided by Apache Spark are built on the highest of the Spark Core. It delivers speed by providing in-memory computation capability. Spark Core is the foundation of parallel and distributed processing of giant dataset. It is the main backbone of the essential I/O functionalities and significant in programming and observing the role of the spark cluster. It holds all the components related to scheduling, distributing and monitoring jobs on a cluster, Task dispatching, Fault recovery. The functionalities of this component are:
    1. It contains the basic functionality of spark. (Task scheduling, memory management, fault recovery, interacting with storage systems).
    2. Home to API that defines RDDs.
2. Spark SQL Structured data: The Spark SQL component is built above the spark core and used to provide the structured processing on the data. It provides standard access to a range of data sources. It includes Hive, JSON, and JDBC. It supports querying data either via SQL or via the hive language. This also works to access structured and semi-structured information. It also provides powerful, interactive, analytical application across both streaming and historical data. Spark SQL could be a new module in the spark that integrates the relative process with the spark with programming API. The main functionality of this module is:
    1. It is a Spark package for working with structured data.
    2. It Supports many sources of data including hive tablets, parquet, json.
    3. It allows the developers to intermix SQK with programmatic data manipulation supported by RDDs in python, scala and java.
3. Spark Streaming: Spark streaming permits ascendible, high-throughput, fault-tolerant stream process of live knowledge streams. Spark can access data from a source like a flume, TCP socket. It will operate different algorithms in which it receives the data in a file system, database and live dashboard. Spark uses Micro-batching for real-time streaming. Micro-batching is a technique that permits a method or a task to treat a stream as a sequence of little batches of information. Hence spark streaming groups the live data into small batches. It delivers it to the batch system for processing. The functionality of this module is:
    1. Enables processing of live streams of data like log files generated by production web services.
    2. The API's defined in this module are quite similar to spark core RDD API's.
4. Mllib Machine Learning: MLlib in spark is a scalable Machine learning library that contains various machine learning algorithms. The motive behind MLlib creation is to make the implementation of machine learning simple. It contains machine learning libraries and the implementation of various algorithms. For example, clustering, regression, classification and collaborative filtering.
5. GraphX graph processing: It is an API for graphs and graph parallel execution. There is network analytics in which we store the data. Clustering, classification, traversal, searching, and pathfinding is also possible in the graph. It generally optimizes how we can represent vertex and edges in a graph. GraphX also optimizes how we can represent vertex and edges when they are primitive data types. To support graph computation, it supports fundamental operations like subgraph, joins vertices, and aggregate messages as well as an optimized variant of the Pregel API.