CMR INSTITUTE OF TECHNOLOGY

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

**Internal Assessment Test 2 – Dec 2025**

| Sub: | Big Data Analytics | | | | | | | Sub Code: | MMCA311B |
|---|---|---|---|---|---|---|---|---|---|
| Date: | 16.12.25 | 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 Explain the architecture of Apache Spark with a neat diagram. Describe the roles of RDDs, DAG, and Executors in detail. | [10] | CO1 | L1 |
| **OR** | | | |
| 2 What is Spark MLlib? Explain the machine learning workflow in MLlib with a suitable example. | [10] | CO1 | L3 |
| **PART II** | | | |
| 3 Write a detailed note on the different types of NoSQL databases. Give examples for each. | [10] | CO1 | L1 |
| **OR** | | | |
| 4 Explain the architecture of HBase. Describe CRUD operations in HBase with examples. | [10] | CO1 | L1 |
| **PART III** | | | |
| 5 What is Cassandra? Explain its data model, storage architecture and replication mechanisms. | [10] | CO2 | L1 |
| **OR** | | | |
| 6 Explain data modeling techniques for scalability and performance in NoSQL systems. | [10] | CO2 | L1 |
| **PART IV** | | | |
| 7 Describe the architecture and working of Apache Flume for real-time data ingestion. Give a use-case example. | [10] | CO2 | L3 |
| **OR** | | | |
| 8 Discuss real-time data streaming using Kafka and Spark Streaming. Explain typical use cases in industry. | [10] | CO2 | L1 |
| **PART V** | | | |
| 9 Explain how Hive and Pig support data analytics in Big Data environments. Compare HiveQL with Pig Latin. | [10] | CO3 | L2 |
| **OR** | | | |
| 10 Write a detailed note on Big Data applications in healthcare, finance, e-commerce, IoT, and social media with examples. | [10] | CO3 | L2 |

**1 Explain the architecture of Apache Spark with a neat diagram. Describe the roles of RDDs, DAG, and Executors in details.**

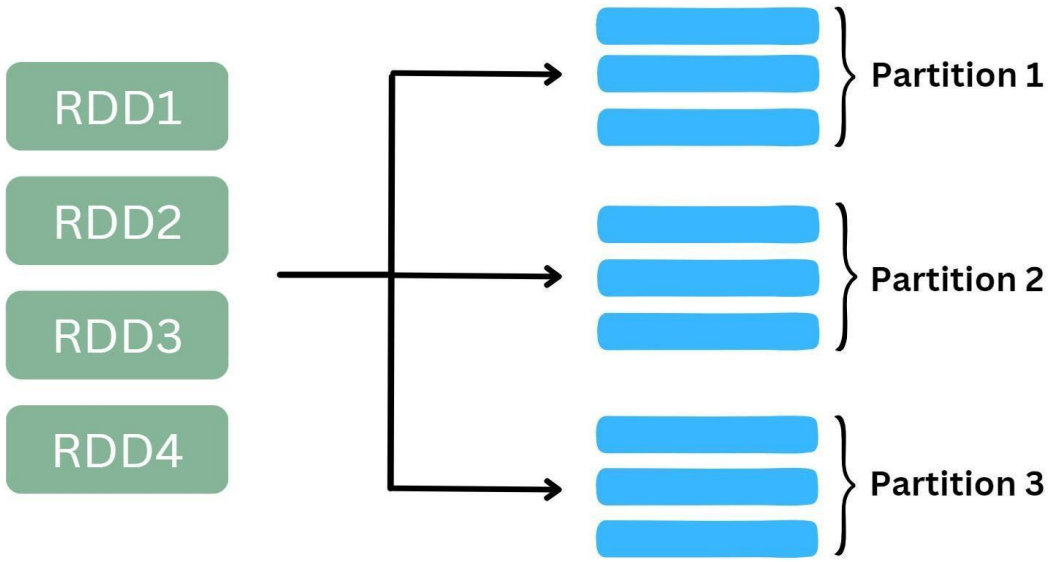
The architecture of Spark is based on two main abstraction layers, which are given below.

Resilient Distributed Dataset (RDD)

Directed Acyclic Graph (DAG)

**Resilient Distributed Dataset (RDD)**

RDD is an essential building block of Spark. It is a key data computation tool and an interface for immutable data. It is a data structure that is useful to recompute data in case of failure. RDD includes all sorts of Python, Scala, or Java objects for users to use.




Each dataset in RDD is divided into logical partitions. These partitions are stored and processed on various machines of a cluster.

The concept of RDD in Spark is to achieve a fast and more efficient MapReduce operation.

Using RDDs, one can perform two types of operations:

1. **Transformations:** These are the set of operations that are applied to create a new RDD from the existing RDDs. It takes one RDD as an input and produces one or more RDD as an output.

2. **Actions:** Actions are the operations that trigger a Spark job to perform computations on the dataset. After the computations are done, the result is sent to the Spark driver program.

**Directed Acyclic Graph (DAG)**

In Spark, DAG is the logical flow of plans that need to be applied to data to achieve the desired outcome. The DAG can be broken down as follows:

- **Directed:** It means a direct connection from one node to another.

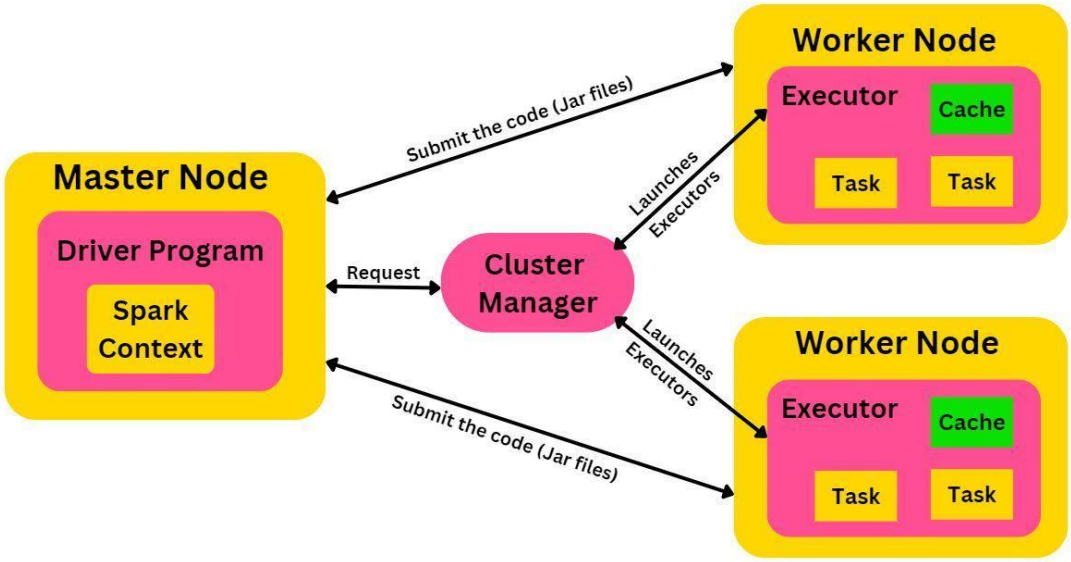
- **Acyclic:** It means that there are no cycles or loops present. Whenever a transformation occurs, it cannot return to its earlier state.

- **Graph:** A graph is a combination of *vertices* and *edges*. Here *vertices* represent the **RDDs**, and the *edges*, represent the operation to be applied to the RDDs.

DAG helps Spark to optimize executions, achieve parallelism, and provide fault tolerance.

**What is Apache Spark Architecture?**

Apache Spark has a structured architecture that is integrated with various libraries and extensions. Spark follows a master-slave architecture where all the components and layers of Spark are loosely coupled. The Spark cluster consists of a single master and multiple slaves.




**2.What is Spark MLlib? Explain the machine learning workflow in MLlib with a suitable example.**

MLlib is a machine learning framework built on top of Apache Spark. It is designed to make machine learning tasks faster and more efficient. MLlib supports various ML algorithms and utilities.

MLlib is a machine learning framework built on top of Apache Spark. It is designed to make machine learning tasks faster and more efficient. MLlib supports various ML algorithms and utilities.

MLlib is a machine learning framework built on top of Apache Spark. It is designed to make machine learning tasks faster and more efficient. MLlib supports various ML algorithms and utilities.

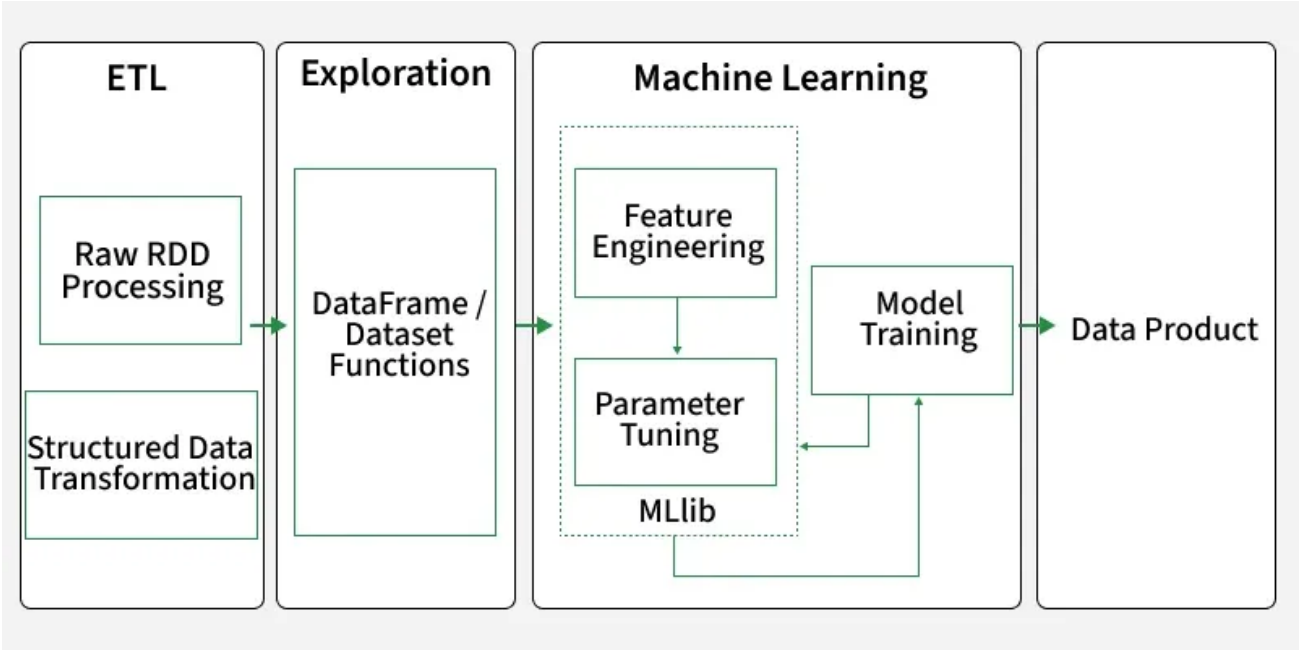
**Core Features of MLlib**

- MLlib is a scalable, easy to use and comprehensive library
- Algorithms like Classification, Regression, Clustering, Collaborative Filtering, etc can be implemented using it.
- Similar to scikit-learn hence streamlines the process of building and tuning machine learning workflows.
- Includes feature selection, extraction, scaling, etc.
- Provides tools to evaluate models with accuracy, precision, etc.

- It provides 2 main APIs: RDD-based API and Data Frame-based APIWorking Principle

**Workflow of MLlib Models**

1. **Data Ingestion**: Load data using Spark DataFrames.
2. **Data Preprocessing**: Cleaning, handling missing values. Feature selection and transformation
3. **Model Selection**: Choose a machine learning algorithm based on data.
4. **Training**: Fit the model on training data.
5. **Prediction**: Use the trained model to make predictions on test or new data.
6. **Evaluation**: Evaluate model performance using various metrics.
7. **Pipeline Deployment**: Build a pipeline combining all steps.




The above image illustrates the Workflow of a Model using MLlib.

**Major Algorithms in MLlib**

**1. Classification Models**

- Used to categorize data into predefined labels.
- Used in Email spam detection
- Handle binary and multiclass problems

**2. Regression Models**

- Used to predict continuous values.
- Used in House price prediction
- Minimize error between predicted and actual values

**3. Clustering Models**

- Used to group data points without labels.
- Used in Customer segmentation
- Unsupervised learning based on similarity

**4. Recommendation Algorithms**

- Used for personalized content delivery.
- Used in Movie recommendations
- Collaborative filtering based on user-item interactions

**5. Dimensionality Reduction**

- Used to reduce feature space while preserving data variance.
- Used in Visualization or preprocessing

- Helps in improving performance and reducing noise

**6. Feature Transformation**

- Essential for preparing raw data into a usable format.
- Encoding categorical variables and scaling features
- Required before model training

**3 Write a detailed note on the different types of NoSQL databases. Give examples for each.**

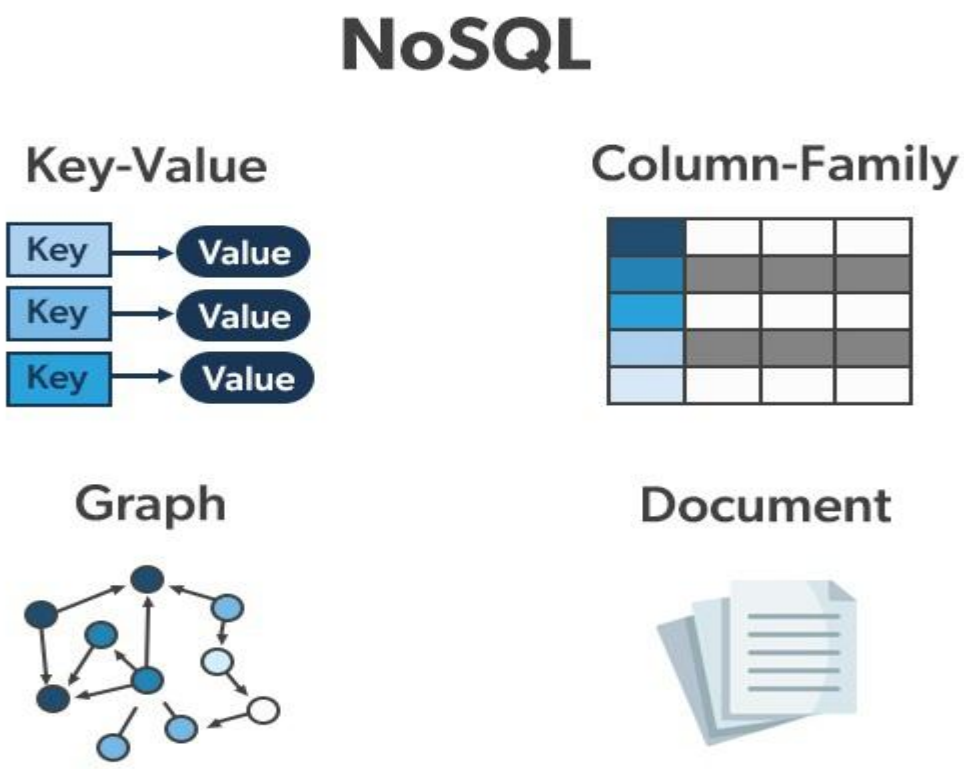
Databases store organized data for easy access and management. Traditional relational databases use structured tables, but modern applications and big data have driven the rise of NoSQL systems. NoSQL databases handle large volumes of unstructured and semi-structured data, offering the scalability and flexibility today's diverse workloads demand.

**Types of NoSQL Database**

NoSQL databases can be classified into **four main types**, based on their data storage and retrieval methods:

1. Document-based databases
2. Key-value stores
3. Column-oriented databases
4. Graph-based databases

Each type has unique advantages and use cases, making NoSQL a preferred choice for big data applications, real-time analytics, cloud computing and distributed systems




**1. Document-Based Database**

The document-based database is a nonrelational database. Instead of storing the data in rows and columns (tables), it uses the documents to store the data in the database. A document database stores data in JSON, BSON or XML documents.

Documents can be stored and retrieved in a form that is much closer to the data objects used in applications which means less translation is required to use these data in the applications. In the Document database, the particular elements can be accessed by using the index value that is assigned for faster querying.
Collections are the group of documents that store documents that have similar contents. Not all the documents are in any collection as they require a similar schema because document databases have a flexible schema.

**2. Key-Value Stores**

A key-value store is a nonrelational database. The simplest form of a NoSQL database is a **key-value store**. Every data element in the database is stored in key-value pairs. The data can be retrieved by using a unique key allotted to each element in the database. The values can be simple data types like strings, numbers or complex objects. A key-value store is like a relational database with only two columns which is the key and the value.

**3. Column Oriented Databases**

A column-oriented database is a non-relational database that stores the data in columns instead of rows. That means when we want to run analytics on a small number of columns, we can read those columns directly without consuming memory with the unwanted data. Columnar databases are designed to read data more efficiently and retrieve the data with greater speed. A columnar database is used to store a large amount of data.

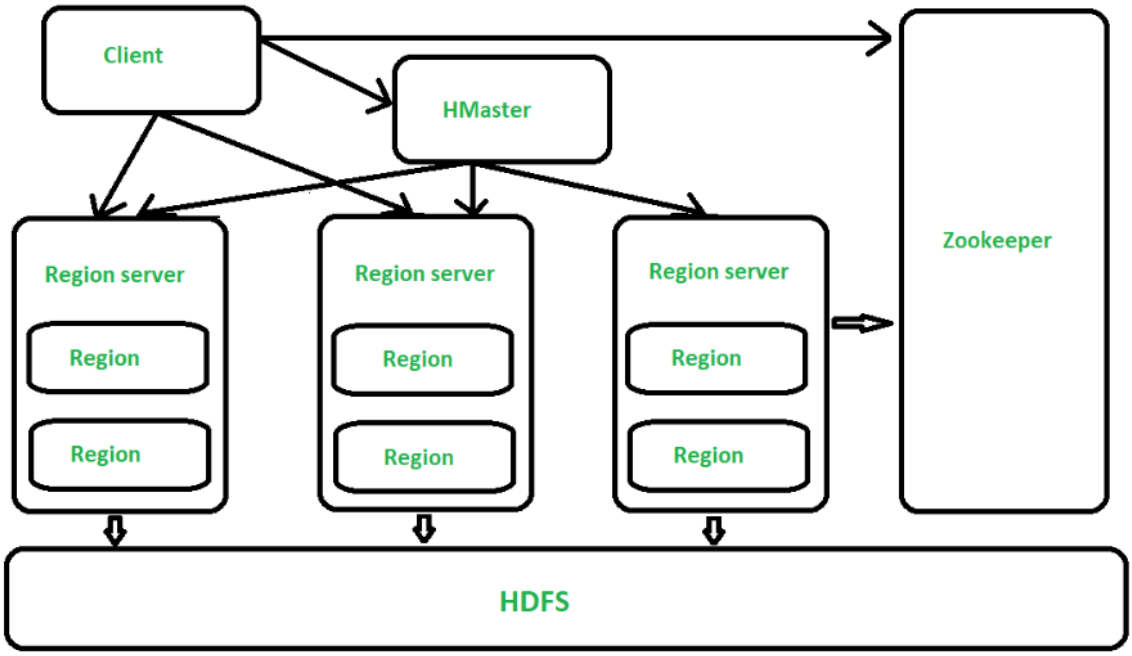
**4. Graph-Based Databases**

Graph-based databases focus on the relationship between the elements. It stores the data in the form of nodes in the database. The connections between the nodes are called links or relationships, making them ideal for complex relationship-based queries.

- Data is represented as nodes (objects) and edges (connections).
- Fast graph traversal algorithms help retrieve relationships quickly.
- Used in scenarios where relationships are as important as the data itself.

**4. Explain the architecture of HBase. Describe CRUD operations in HBase with examples**

HBase is a distributed, scalable, NoSQL database built on top of Hadoop. It is designed to store huge amounts of structured or semi-structured data and provide fast, random read/write access. To achieve this, HBase relies on three main components in its architecture: HMaster, Region Server, and ZooKeeper.




**HMaster**

The **HMaster** acts as the **main coordinator** of the HBase cluster.
Think of it as the *manager* that oversees how data is distributed and how the cluster functions.

**Key Roles of HMaster**

- Assigns regions (data chunks) to Region Servers
- Manages table operations like **create**, **delete**, and **modify**
- Monitors the health of Region Servers
- Balances load across servers
- Handles failover when a server crashes

In large clusters, multiple backup HMasters run to ensure high availability.

**Region Server**

HBase tables are very large, so they are divided horizontally into smaller parts called Regions. A Region Server is responsible for managing these regions.

**What Region Servers Do**

- Store and manage regions, each containing data for a specific row-key range
- Handle read and write requests from clients
- Store data in column families, which are the basic storage units in HBase
- Run on top of HDFS DataNodes, making use of Hadoop's storage

Each region is around 256 MB by default, and new regions are automatically created as the table grows.

**ZooKeeper**

ZooKeeper works like a **traffic controller** for the HBase cluster.

**ZooKeeper Responsibilities**

- Helps clients find which Region Server holds which data
- Monitors server failures and helps in quick recovery
- Maintains cluster configuration
- Provides distributed synchronization

Without ZooKeeper, coordination between HMaster, Region Servers, and clients would not be possible.

## Standard CRUD Operations

This table lists the standard create, read, update, and delete (CRUD) operations for a business object service.

| Operation | Method Name | Description |
|---|---|---|
| create Operation | `create`*`ServiceDataObjectName* | Creates a service data object and its descendants. |
| delete Operation | `delete`*`ServiceDataObjectName* | Deletes a business object. |
| get Operation | `get`*`ServiceDataObjectName* | Retrieves a single business object by the primary key. |
| find Operation | `find`*`ServiceDataObjectName* | Finds and returns a list of business objects that meet the specified search criteria. |
| find by additional predefined search criteria Operation | `find`*`ServiceDataObjectNameSearchCriteriaName* | Finds and returns a list of business objects that meet the specified search criteria and the additional predefined search criteria. |
| update Operation | `update`*`ServiceDataObjectName* | Updates a business object. |
| merge Operation | `merge`*`ServiceDataObjectName* | Updates a business object, if it exists. Otherwise creates a new business object. |
| process Operation | `process`*`ServiceDataObjectName* | Performs create, update, delete, or merge operation on a list of business objects. The specified operation is applied to all the objects in a given list. |
| process change summary Operation | `processCS`*`ServiceDataObjectName* | Performs create, update, or delete operations on a list of business objects. Allows you to specify different operations for different objects. |

**5. What is Cassandra? Explain its data model, storage architecture and replication mechanisms.**

**Apache Cassandra** is an open-source, distributed NoSQL database designed for handling large volumes of data across commodity servers, providing high availability and linear scalability without a single point of failure. It is a wide-column store that uses a peer-to-peer, masterless architecture.

### Data Model

Cassandra's data model is a partitioned row store that is "query-driven," meaning the schema is designed around the specific data access patterns of the application, rather than the entity relationships typical of relational databases.

Key components include:

- **Keyspace:** The outermost container, similar to a database or schema in RDBMS. It defines the replication strategy and options for its data.
- **Table:** Stores data in rows and columns (formerly called column families). Unlike RDBMS, different rows in the same table can have different columns, offering a flexible schema.
- **Row:** Uniquely identified by a **primary key**.
- **Primary Key:** Consists of a **partition key** and optional **clustering keys**.
  - **Partition Key:** Determines how data is distributed across the nodes in the cluster (via consistent hashing).
  - **Clustering Keys:** Determine the sort order of rows within a single partition.
- **Column:** The basic unit of data, with a name, value, and timestamp.

### Storage Architecture

Cassandra's storage engine uses a Log-Structured Merge (LSM) tree mechanism, which optimizes for very high write throughput. The storage components and write process are as follows:

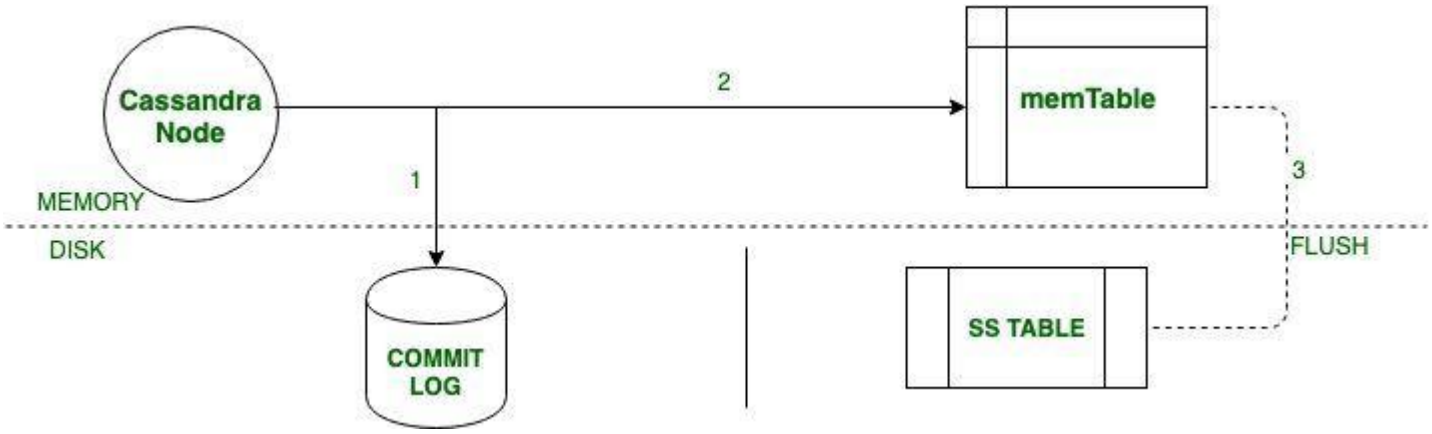
1. **Commit Log:** Every write operation is first appended to a commit log on disk, ensuring data durability in case of a crash.
2. **Memtable:** The data is simultaneously written to an in-memory data structure called a memtable.
3. **SSTables (Sorted String Tables):** When a memtable reaches a certain threshold, it is flushed to disk as an immutable (unchangeable) SSTable file. Each update creates a new entry, and deletes are marked with a "tombstone" marker.
4. **Compaction:** Periodically, multiple SSTables are merged into one through a process called compaction, which discards obsolete data and tombstones to reclaim disk space and improve read performance.
5. **Bloom Filters:** For read operations, bloom filters (probabilistic data structures residing in memory) are used to quickly check if a specific row key exists in an SSTable, avoiding unnecessary disk I/O.

### Replication Mechanisms

Replication in Cassandra ensures high availability and fault tolerance across a decentralized, peer-to-peer cluster.

- **Replication Factor (RF):** A keyspace-level setting that determines the number of copies of each row stored across the cluster. A common factor in production is three.
- **Replication Strategies:**
  - **SimpleStrategy:** Used for single-datacenter clusters. Replicas are placed on nodes clockwise around the ring.
  - **NetworkTopologyStrategy:** Recommended for multi-datacenter deployments. It allows for specifying the replication factor per datacenter and attempts to place replicas in different racks to survive physical rack failures.

- **Tunable Consistency:** Cassandra allows developers to choose the consistency level (CL) for each read and write operation, balancing between consistency, availability, and latency.
  - **Strong Consistency:** Achieved when R (read CL) + W (write CL) > RF (e.g., if RF=3, R=2, W=2, then 2+2 > 3).
  - **Eventual Consistency:** The system ensures data will eventually be consistent across all replicas, a model aided by mechanisms like **hinted handoff** (storing writes for temporarily down nodes) and **read repair** (fixing inconsistencies during read requests).
- **Gossip Protocol:** Nodes use a peer-to-peer gossip protocol to discover each other and exchange state information about the cluster topology and node health, eliminating the need for a master node




6. Explain data modeling techniques for scalability and performance in NoSQL systems.

NoSQL (Not Only SQL) databases are designed to handle large volumes of unstructured and semi-structured data. Unlike traditional relational databases that rely on fixed schemas and tables, NoSQL offers flexible data models and supports horizontal scaling. This makes them well-suited for modern applications that require high performance, scalability, and the ability to manage diverse data types efficiently.

**Key Features of NoSQL Databases**

- **Dynamic schema:** Allow flexible shaping of data to meet new requirements without the need to migrate or change schemas.
- **Horizontal scalability:** They scale horizontally for adding more nodes into the existing ones and acquire enough storage for even bigger datasets and much higher traffic by distributing the load on multiple servers.
- **Document-based:** Data are presented in flexible, semi-structured formats like JSON/BSON (e.g., MongoDB).
- **Key-value-based:** They possess a simple but fast access pattern (e.g., Redis) by storing data as pairs of keys and values.
- **Column-based:** Data are organized into columns instead of rows (e.g., CASSANDRA).
- **Distributed and high availability:** They are designed to be highly available and to automatically handle node failures and data replication across multiple nodes in a database cluster.
- **Flexibility:** Allow developers to store and retrieve data in a flexible and dynamic manner, with support for multiple data types and changing data structures.
- **Performance:** Perfect for big data and real-time analytics and high volume applications.

**Challenges of NoSQL Databases**

- **Lack of standardization:** NoSQL systems can be vastly different from one another, making it even harder to choose the right one for a specific use case.
- **Lack of ACID compliance:** NoSQL databases may not provide consistency, which is a disadvantage for applications that need strict data integrity.

- **Narrow focus:** Great for storage but lack functionalities as transaction management, in which relational databases are great.
- **Absence of Complex Query Support:** They are not designed to handle complex queries, which means that they are not a good fit for applications that require complex data analysis or reporting.
- **Lack of maturity:** Being relatively new, NoSQL may not have the reliability, security and feature set of traditional relational databases.
- **Management complexity:** For large datasets, maintaining a NoSQL database could be quite more complicated than managing a relational database.
- **Limited GUI Tools**: While some NoSQL databases, like MongoDB offer GUI tools like MongoDB Compass, not all NoSQL databases provide flexible or user-friendly GUI tools.

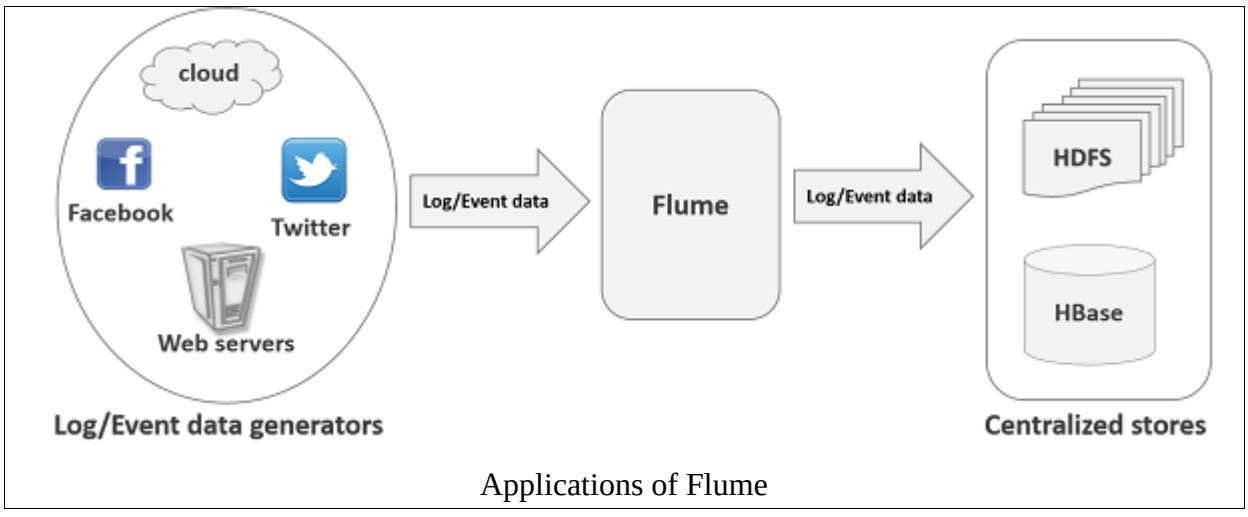
**Popular NoSQL Databases & Their Use Cases**

| NoSQL Database | Type | Use Cases |
|---|---|---|
| **MongoDB** | Document-based | Content management, product catalogs |
| **Redis** | Key-Value Store | Caching, real-time analytics, session storage |
| **Cassandra** | Column-Family Store | Big data, high availability systems |
| **Neo4j** | Graph Database | Fraud detection, social networks |

**7. Describe the architecture and working of Apache Flume for real-time data ingestion. Give a use-case example.**

Apache Flume is a tool/service/data ingestion mechanism for collecting aggregating and transporting large amounts of streaming data such as log files, events (etc...) from various sources to a centralized data store.

Flume is a highly reliable, distributed, and configurable tool. It is principally designed to copy streaming data (log data) from various web servers to HDFS.




Applications of Flume

Assume an e-commerce web application wants to analyze the customer behavior from a particular region. To do so, they would need to move the available log data in to Hadoop for analysis. Here, Apache Flume comes to our rescue.

Flume is used to move the log data generated by application servers into HDFS at a higher speed.

## Advantages of Flume

Here are the advantages of using Flume −

- Using Apache Flume we can store the data in to any of the centralized stores (HBase, HDFS).
- When the rate of incoming data exceeds the rate at which data can be written to the destination, Flume acts as a mediator between data producers and the centralized stores and provides a steady flow of data between them.
- Flume provides the feature of **contextual routing**.
- The transactions in Flume are channel-based where two transactions (one sender and one receiver) are maintained for each message. It guarantees reliable message delivery.
- Flume is reliable, fault tolerant, scalable, manageable, and customizable.

## Features of Flume

Some of the notable features of Flume are as follows −

- Flume ingests log data from multiple web servers into a centralized store (HDFS, HBase) efficiently.
- Using Flume, we can get the data from multiple servers immediately into Hadoop.
- Along with the log files, Flume is also used to import huge volumes of event data produced by social networking sites like Facebook and Twitter, and e-commerce websites like Amazon and Flipkart.
- Flume supports a large set of sources and destinations types.
- Flume supports multi-hop flows, fan-in fan-out flows, contextual routing, etc.
- Flume can be scaled horizontally.

**Apache Flume - Data Transfer In Hadoop**

**Big Data,** as we know, is a collection of large datasets that cannot be processed using traditional computing techniques. Big Data, when analyzed, gives valuable results. **Hadoop** is an open-source framework that allows to store and process Big Data in a distributed environment across clusters of computers using simple programming models.

## Streaming / Log Data

Generally, most of the data that is to be analyzed will be produced by various data sources like applications servers, social networking sites, cloud servers, and enterprise servers. This data will be in the form of **log files** and **events**.

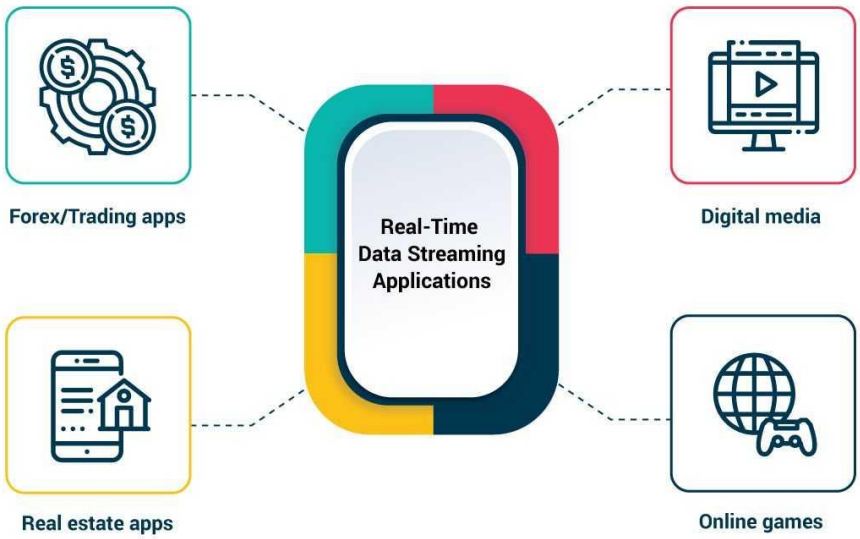
**Log file** − In general, a log file is a **file** that lists events/actions that occur in an operating system. For example, web servers list every request made to the server in the log files.

On harvesting such log data, we can get information about −

- the application performance and locate various software and hardware failures.
- the user behavior and derive better business insights.

The traditional method of transferring data into the HDFS system is to use the **put** command. Let us see how to use the **put** command.

**9. Discuss real-time data streaming using Kafka and Spark Streaming. Explain typical use**




Data is being generated at an unprecedented rate from IoT sensors, mobile devices, servers, applications, and security logs. To make sense of this data deluge, businesses need more than just traditional batch systems — they need a way to ingest, process, and act on data the moment it's created.

That's where **real-time data streaming** comes in.

**Real-time data streaming** is the continuous transfer and processing of data as it's generated. It enables organizations to detect anomalies, personalize user experiences, make timely decisions, and respond to events instantly — all without waiting for batch cycles or manual ETL jobs.

Unlike traditional pipelines that collect and store data before processing it, **streaming architectures allow you to consume, enrich, transform, and analyze data continuously,** as it flows through your systems. This means you can act on data in motion, rather than data at rest.

In this comprehensive guide, you'll learn:

- What real-time data streaming is and how it works
- The core components of a streaming data architecture
- Popular tools and platforms used to implement it

- Benefits of adopting real-time pipelines
- Real-world use cases across industries

**10. Explain how Hive and Pig support data analytics in Big Data environments. Compare HiveQL with Pig Latin.**

**1. Pig :**

Pig is used for the analysis of a large amount of data. It is abstract over MapReduce. Pig is used to perform all kinds of data manipulation operations in Hadoop. It provides the Pig-Latin language to write the code that contains many inbuilt functions like join, filter, etc. The two parts of the Apache Pig are Pig-Latin and Pig-Engine. Pig Engine is used to convert all these scripts into a specific map and reduce tasks. Pig abstraction is at a higher level. It contains less line of code as compared to MapReduce.

**2. Hive :**

Hive is built on the top of Hadoop and is used to process structured data in Hadoop. Hive was developed by Facebook. It provides various types of querying language which is frequently known as Hive Query Language. Apache Hive is a data warehouse and which provides an SQL-like interface between the user and the Hadoop distributed file system (HDFS) which integrates Hadoop.

**Difference between Pig and Hive :**

| S.No. | Pig | Hive |
|---|---|---|
| 1. | Pig operates on the client side of a cluster. | Hive operates on the server side of a cluster. |
| 2. | Pig uses pig-latin language. | Hive uses HiveQL language. |
| 3. | Pig is a Procedural Data Flow Language. | Hive is a Declarative SQLish Language. |
| 4. | It was developed by Yahoo. | It was developed by Facebook. |
| 5. | It is used by Researchers and Programmers. | It is mainly used by Data Analysts. |
| 6. | It is used to handle structured and semi-structured data. | It is mainly used to handle structured data. |

10. Write a detailed note on Big Data applications in healthcare, finance, e-commerce, IoT,

In today's world, there are a lot of data. Big companies utilize those data for their business growth. By analyzing this data, the useful decision can be made in various cases as discussed below:

**1. Tracking Customer Spending Habit, Shopping Behavior:** In big retails store (like Amazon, Walmart, Big Bazar etc.) management team has to keep data of customer's spending habit (in which product customer spent, in which brand they wish to spent, how frequently they spent), shopping behavior, customer's most liked product (so that they can keep those products in the store). Which product is being searched/sold most, based on that data, production/collection rate of that product get fixed.

Banking sector uses their customer's spending behavior-related data so that they can provide the offer to a particular customer to buy his particular liked product by using bank's credit or debit card with discount or cashback. By this way, they can send the right offer to the right person at the right time.

**2. Recommendation:** By tracking customer spending habit, shopping behavior, Big retails store provide a recommendation to the customer. E-commerce site like Amazon, Walmart, Flipkart does product recommendation. They track what product a customer is searching, based on that data they recommend that type of product to that customer.
As an example, suppose any customer searched bed cover on Amazon. So, Amazon got data that customer may be interested to buy bed cover. Next time when that customer will go to any google page, advertisement of various bed covers will be seen. Thus, advertisement of the right product to the right customer can be sent.

YouTube also shows recommend video based on user's previous liked, watched video type. Based on the content of a video, the user is watching, relevant advertisement is shown during video running. As an example suppose someone watching a tutorial video of Big data, then advertisement of some other big data course will be shown during that video.

**3. Smart Traffic System:** Data about the condition of the traffic of different road, collected through camera kept beside the road, at entry and exit point of the city, GPS device placed in the vehicle (Ola, Uber cab, etc.). All such data are analyzed and jam-free or less jam way, less time taking ways are recommended. Such a way smart traffic system can be built in the city by Big data analysis. One more profit is fuel consumption can be reduced.
**4. Secure Air Traffic System:** At various places of flight (like propeller etc) sensors present. These sensors capture data like the speed of flight, moisture, temperature, other environmental condition. Based on such data analysis, an environmental parameter within flight are set up and varied.
By analyzing flight's machine-generated data, it can be estimated how long the machine can operate flawlessly when it to be replaced/repaired.

**5. Auto Driving Car:** Big data analysis helps drive a car without human interpretation. In the various spot of car camera, a sensor placed, that gather data like the size of the surrounding car, obstacle, distance from those, etc. These data are being analyzed, then various calculation like how many angles to rotate, what should be speed, when to stop, etc carried out. These calculations help to take action automatically.
**6. Virtual Personal Assistant Tool:** Big data analysis helps virtual personal assistant tool (like Siri in Apple Device, Cortana in Windows, Google Assistant in Android) to provide the answer of the various question asked by users. This tool tracks the location of the user, their local time, season, other data related to question asked, etc. Analyzing all such data, it provides an answer.
**7. IoT:**

- Manufacturing company install IOT sensor into machines to collect operational data. Analyzing such data, it can be predicted how long machine will work without any problem when it requires repairing so that company can take action before the situation when machine facing a lot of issues or gets totally down. Thus, the cost to replace the whole machine can be saved.
- In the Healthcare field, Big data is providing a significant contribution. Using big data tool, data regarding patient experience is collected and is used by doctors to give better treatment. IoT device can sense a symptom of probable coming disease in the human body and prevent it from giving advance treatment. IoT Sensor placed near-patient, new-born baby constantly keeps track of various health condition like heart bit rate, blood presser, etc. Whenever any parameter crosses the safe limit, an alarm sent to a doctor, so that they can take step remotely very soon.

**8. Education Sector:** Online educational course conducting organization utilize big data to search candidate, interested in that course. If someone searches for YouTube tutorial video on a subject, then online or offline course provider organization on that subject send ad online to that person about their course.

**9. Energy Sector:** Smart electric meter read consumed power every 15 minutes and sends this read data to the server, where data analyzed and it can be estimated what is the time in a day when the power load is less throughout the city. By this system manufacturing unit or housekeeper are suggested the time when they should drive their heavy machine in the night time when power load less to enjoy less electricity bill.

**10. Media and Entertainment Sector:** Media and entertainment service providing company like Netflix, Amazon Prime, Spotify do analysis on data collected from their users. Data like what type of video, music users are watching, listening most, how long users are spending on site, etc are collected and analyzed to set the next business strategy.