USN :_____ **INTERNAL ASSESSMENT TEST 1**

Sub:	Big Data Analysis	Subject Code:	BAD 601	Branch :	AI & DS
Date:	24.03.2025	Duration: 90 min	Marks : 50	Sem : VI	OBE

Slno	Answer any FIVE question	Marks	CO	RBT
Q1	Explain CAP theorem in big data. How to deal with unstructured data?	10	CO1	L2
Q2	What is Hadoop Ecosystem? Discuss various components of Hadoop Ecosystem with proper diagram.	10	CO2	L1
Q3	What are the key advantages in Hadoop. Explain each of them.	10	CO2	L3
Q4	Explain the HDFS daemons. Explain the Anatomy of file read operation.	5+5	CO3	L1
Q5	What is Map Reduce? Explain working of various phases of Map Reduce with appropriate example and diagram.	5+5	CO3	L2
Q6.	What is MongoDB. Explain all the features of MongoDB.	3+7	CO3	L3

CI **CCI** **HOD**

Answer Key

Answer 1. CAP Theorem (Consistency, Availability, Partition Tolerance): The CAP theorem, proposed by Eric Brewer, states that in a distributed data system, it is impossible to achieve all three of the following properties simultaneously:

Consistency (C) – Every read operation receives the most recent write or an error. This ensures that all nodes have the same data at the same time.

Availability (A) – Every request receives a response, whether it is the latest data or not. The system remains operational even if some nodes fail.

Partition Tolerance (P) – The system continues to function even if there is a network failure that prevents communication between nodes.

Types of Systems Based on CAP

CP (Consistency + Partition Tolerance) – Guarantees consistency but may sacrifice availability during network failures. Example: MongoDB, HBase.

AP (Availability + Partition Tolerance) – Guarantees availability but may return stale data in case of network partitions. Example: Cassandra, DynamoDB.

CA (Consistency + Availability) – Works only in systems without partitions (not practical for distributed systems). Example: Traditional relational databases (single-node MySQL).

Unstructured data (e.g., text, images, videos, logs, social media content) requires specialized techniques to extract meaningful insights. Below are key methods:

Data Mining: Extracts patterns and useful information from large datasets. Uses clustering, classification, and association rule mining.

Text Mining: Analyzes large amounts of textual data to find insights. Includes sentiment analysis, keyword extraction, and topic modeling.

Natural Language Processing (NLP): Enables computers to understand and process human language. Uses tokenization, named entity recognition (NER), and machine learning models.

Manual Tagging with Metadata: Adding structured information (metadata) to unstructured content. Helps in organizing, searching, and filtering data.

Noisy Text Analysis: Deals with messy, inconsistent, or misspelled text data. Techniques include spell correction, normalization, and stemming.

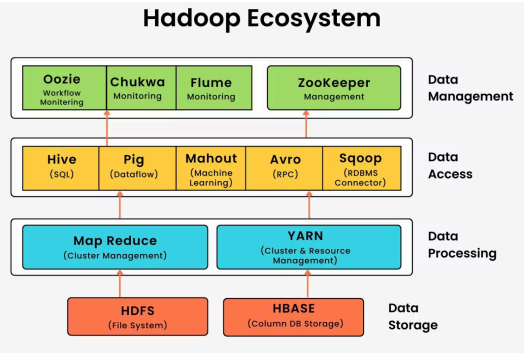
Part of Speech (POS) Tagging: Identifies and labels words as nouns, verbs, adjectives, etc. Helps in syntactic and semantic analysis for NLP applications.

Unstructured Information Architecture: Organizing unstructured data using a framework or strategy. Includes indexing, categorization, and search optimization.

Answer 2. Hadoop Ecosystem is a platform or a suite which provides various services to solve the big data problems. It includes Apache projects and various commercial tools and solutions.

There are four major elements of Hadoop i.e. HDFS, MapReduce, YARN, and Hadoop Common Utilities. Most of the tools or solutions are used to supplement or support these major elements. All these tools work collectively to provide services such as absorption, analysis, storage and maintenance of data etc. Following are the components that collectively form a Hadoop ecosystem:

HDFS: Hadoop Distributed File System
YARN: Yet Another Resource Negotiator
MapReduce: Programming based Data Processing
Spark: In-Memory data processing
PIG, HIVE: Query based processing of data services
HBase: NoSQL Database
Mahout, Spark MLLib: Machine Learning algorithm libraries
Solar, Lucene: Searching and Indexing
Zookeeper: Managing cluster
Oozie: Job Scheduling



Answer 3. Hadoop is a popular framework for big data processing and storage. It provides a scalable, fault-tolerant, and cost-effective solution for handling large datasets. Here are the key advantages of Hadoop:

1. Scalability: Hadoop is designed to scale horizontally by adding more nodes to the cluster. It can handle petabytes and even exabytes of data efficiently. The distributed nature of Hadoop allows parallel processing across multiple nodes, improving performance.
2. Fault Tolerance: Hadoop ensures data reliability and fault tolerance using data replication. The Hadoop Distributed File System (HDFS) replicates data across multiple nodes. If a node fails, the system automatically switches to another replica without data loss.
3. Cost-Effectiveness: Hadoop is an open-source framework, reducing licensing costs. It runs on commodity hardware, eliminating the need for expensive high-end servers. Organizations can store and process large volumes of data at a lower cost compared to traditional databases.
4. Flexibility in Data Handling: Hadoop can process structured, semi-structured, and unstructured data. It supports various data formats such as JSON, XML, images, and videos. Unlike traditional databases that require structured schemas, Hadoop allows schema-on-read processing.
5. High Throughput: Hadoop enables parallel data processing using the MapReduce programming model. Data is processed in a distributed manner, leading to high throughput and efficiency. Large datasets are processed faster compared to traditional methods.
6. Reliability: HDFS ensures reliable data storage with multiple replicas. If a hardware failure occurs, data recovery is automatic, ensuring continuous availability.
7. Easy to Use and Maintain: Hadoop provides high-level tools like Pig, Hive, and Spark to simplify big data processing. These tools allow users to write SQL-like queries without deep programming knowledge. Cluster management tools like Apache Ambari help in easy monitoring and maintenance.

8. Integration with Cloud and Other Technologies: Hadoop integrates with cloud platforms such as AWS, Azure, and Google Cloud. It supports various big data frameworks like Apache Spark, Apache Flink, and Apache Kafka. This makes it adaptable to different data processing needs.
9. Community Support: Hadoop has a large global community and strong industry adoption. Continuous improvements and updates are provided by the open-source community. Numerous resources, tutorials, and forums are available for learning and troubleshooting.
10. Security Features: Hadoop offers security mechanisms like Kerberos authentication, access control lists (ACLs), and encryption. It integrates with enterprise security solutions to ensure data protection.

Answer 4. HDFS Daemons

NameNode (Master): The NameNode is the central authority of HDFS. It stores metadata (file system namespace, directories, and file-to-block mapping). It does not store actual data; it only keeps track of where data blocks are stored in DataNodes.

• Functions:
  ◦ Manages file system operations (open, close, rename).
  ◦ Maintains information about DataNodes.
  ◦ Ensures data replication and fault tolerance.
  • If the NameNode fails, the HDFS system can become non-operational unless a backup mechanism (e.g., Secondary NameNode or High Availability NameNode) is in place.

DataNode (Slave): The DataNodes are responsible for storing actual data blocks. They communicate with the NameNode to receive instructions and send periodic heartbeats to indicate they are alive.

• Functions:
  ◦ Stores, retrieves, and replicates data blocks.
  ◦ Performs block-level read/write operations.
  ◦ Periodically reports block health to the NameNode.
  ◦ Replaces missing blocks when failures occur.

Secondary NameNode (Checkpoint Node): The Secondary NameNode is NOT a backup for the NameNode. It helps optimize the performance of the NameNode by taking periodic checkpoints.

• Functions:
  ◦ Merges edit logs with the FsImage (file system image) to prevent excessive memory usage in the NameNode.
  ◦ Reduces the time required for NameNode recovery in case of failure.
  ◦ Helps in reducing the size of metadata logs, preventing the NameNode from becoming overloaded.

Step anatomy of file reading in HDFS:

1. Client Request to NameNode

The client initiates a read request by specifying the file path.
The HDFS Client (using Hadoop API) communicates with the NameNode.

The NameNode checks metadata and returns:
The list of DataNodes storing the file blocks.
The block locations for the requested file.
2. Client Communicates with DataNodes
The client directly connects to the first DataNode that holds the first block of the file.
Instead of sending data through the NameNode (which would create a bottleneck), the client reads directly from DataNodes.
3. Data Transfer from DataNodes to Client
The DataNode streams the requested data blocks to the client over the network.
The client reads one block at a time, following the sequence of blocks as indicated by the NameNode.
4. Reading from Multiple DataNodes
If a file is large and distributed across multiple blocks, the client will contact different DataNodes to fetch each block.
The client follows the nearest replica approach, prioritizing local or nearby DataNodes for faster access.
5. Data Assembly and Processing
The HDFS client assembles the blocks in order to reconstruct the original file.
The retrieved data is then processed by the application, such as MapReduce, Spark, or any other Hadoop-based framework.
6. Block Replication Consideration
If a DataNode is down, HDFS automatically retrieves the block from another replica stored on a different DataNode.
HDFS ensures fault tolerance by maintaining multiple copies of each block (default replication factor is 3).

Answer 5. 1. Mapping Phase (Processing Input Data)

Step 1: Record Reader
- Reads input data stored in HDFS (Hadoop Distributed File System).
- Converts raw input into key-value pairs (e.g., line number as key and text as value).
- Splits large input files into smaller chunks for parallel processing.

Step 2: Mapper
- Processes each key-value pair independently.
- Extracts meaningful information and converts it into an intermediate format.
- Emits new key-value pairs for further processing.

Step 3: Combiner (Optional)
- Acts as a mini-reducer at the mapper level.
- Locally aggregates data before sending it to the next phase.
- Reduces network congestion by minimizing duplicate data transfer.

Step 4: Partitioner
- Determines which reducer will handle each key.

- Uses a hash function to distribute data across reducers evenly.
- Ensures that all values for a particular key go to the same reducer.

2. Reducing Phase (Data Aggregation)

Step 5: Shuffle

- Collects and transfers intermediate key-value pairs from mappers to reducers.
- Ensures that all values of the same key reach the same reducer.

Step 6: Sort

- Organizes the shuffled data by key.
- Groups all values associated with a particular key for efficient processing.

Step 7: Reducer

- Aggregates values for each key.
- Performs operations like summing, counting, or finding max/min based on the use case.
- Emits the final key-value pairs as output.

Step 8: Final Output

- The final processed data is written to HDFS.
- The output is usually in a structured format (text, CSV, JSON, etc.).
- Can be used for further data analysis, machine learning, or reporting.

Answer 6. MongoDB is a NoSQL database that stores data in a document-oriented format using JSON-like structures called BSON (Binary JSON). Unlike traditional relational databases (RDBMS), MongoDB is schema-less, meaning that data can have flexible structures. It is widely used for big data applications, real-time analytics, and scalable cloud applications.

Features of MongoDB

1. Document-Oriented Database
   - Stores data in BSON (Binary JSON) format, allowing embedded documents and arrays.
   - Provides flexibility to store data in a nested structure.
2. Schema-Less Structure
   - Unlike RDBMS, MongoDB does not require a fixed schema.
   - Each document in a collection can have a different structure, making it highly adaptable.
3. Scalability
   - Uses horizontal scaling (Sharding) to distribute data across multiple servers.
   - Can handle large-scale applications with high volumes of data.
4. High Performance
   - Supports indexing to speed up queries.
   - Uses in-memory processing for faster performance.

- Performs better than traditional SQL databases for read and write operations.

5. Automatic Sharding
   - Distributes data across multiple servers for load balancing.
   - Ensures fault tolerance and prevents system overload.
6. Replication for High Availability
   - Uses Replica Sets to maintain copies of data across multiple servers.
   - If the primary server fails, a secondary server automatically takes over.
7. Query Language Support
   - Supports a rich query language for filtering, updating, and aggregation.
   - Can perform CRUD operations (Create, Read, Update, Delete) efficiently.
8. Indexing for Fast Search
   - Supports various types of indexes (single-field, compound, text, geospatial, etc.).
   - Improves the performance of queries.
9. Aggregation Framework
   - Provides powerful data processing capabilities.
   - Supports functions like filtering, sorting, grouping, and transformation.
10. Load Balancing
    - Distributes read and write operations across multiple servers.
    - Ensures high availability and efficiency in large-scale applications.
11. GridFS for File Storage
    - Stores and retrieves large files, such as images, audio, and videos.
    - Splits files into smaller chunks for efficient storage.
12. ACID Transactions (Since MongoDB 4.0)
    - Supports multi-document ACID transactions, ensuring data consistency.
    - Provides rollback mechanisms in case of failure.