Big Data and Hadhoop

Assignment - 1

Page : Scribble
Date : / /

Que.1 what are the "5Vs" of Big Data, and how do they Define the challenges associated with managing Big Data in an enterprise setting? Provide examples of each.

Ans.1 The "5 Vs" of Big Data are :-

1. Volume : Refers to the massive amounts of data generated.
   - Challenge : storing and processing large datasets.
   - Example : Social media platforms handling billions of posts daily.

2. Velocity : Refers to the speed of data generation and processing.
   - challenge : Real-time data analysis and decision making
   - Example : Fraud detection in banking systems.

3. Variety : Refers to the diversity of data types (structured, semi-structured, unstructured).
   - Challenge : Integrating and analyzing heterogeneous data sources.
   - Example : combining patient records, lab reports and medical images in healthcare.

Page : Scribble
Date : / /

Que. 2 What are the essential steps involved in setting up a demo environment for Big Data projects ? Discuss the hardware, software, data and security considerations.

Ans. 2 Setting up a demo environment for Big Data projects involves careful planning and configuration of hardware, software, data, and security to mimic real world scenarios. Here are the essential steps :

1. Define Objectives : Clarify the demo's purpose, such as analytics or pipeline testing.
2. Hardware : Use high-performance local systems or cloud platforms (AWS, Azure) for scalability.
3. Software : Install Big Data tools (Hadoop, Spark), databases (HDFS, Hive, NoSQL), visualization tools (Tableau, Power BI), and development environments (PyCharm, Jupyter).
4. Data : Use public datasets or generate synthetic data in varied formats (CSV, JSON). Set up ETL pipelines with tools like Apache NiFi.
5. Security : Implement access controls (RBAC), encryption (SSL/TLS) and monitoring (ELK stack).
6. Test : Validate workflows, optimize configurations and ensure performance.

Que. 3

Ans. 3

21100BTCSE09775
ANSHIKA JAIN

7. Document : Prepare a demo script and setup guide for clarity and reproducibility.

Ans. 3 Describe the MapReduce programming model and its key components. How does the shuffle and sort phase work in the context of MapReduce jobs, and why is it crucial for the efficiency of the process?

Ans. 3 Map Reduce is a programming model for processing large datasets in a distributed environment. It consists of two primary phases:

1. Map Phase :
- Input data is split into key-value pairs and processed independently by mapper functions.
- Output : Intermediate key-value pairs

2. Reduce Phase :
- Reducers aggregate and process intermediate key-value pairs with the same key to produce the final result.

Key Components

1. Mapper : Processes input data and generates intermediate key: value pairs.
2. Reducer : Processes grouped intermdi -ate data to produce the

final output.

3. InputFormat : Defines how input data is split and read.
4. OutputFormat : Defines how the output data is written.
5. Job Tracker / Resource Manager : Manages and schedules MapReduce jobs across the cluster.
6. Task Tracker / Node Manager : Executes individual map and reduce tasks.

Shuffle and Sort Phase

What happens :

1. Intermediate data is shuffled (transferred) from mappers to reducers.
2. Data is sorted by keys so that all values for a given key are grouped together.

Importance :

- Ensures that reducers process data in a structured way by key.
- Reduces data movement across the network for efficiency.
- Allows parallel processing of grouped data in the reduce phase.

Efficiency : Optimized shuffling and sorting reduce network bottlenecks and improve job completion.

ANSHIKA JAIN 2110OBTCSE04473

Page: Scribble
Date: / /

Que. 4 Explain the roles of the NameNode and DataNodes in the Hadoop Distributed File System (HDFS). How does HDFS ensure data reliability and fault tolerance ?

Ans. 4 1. NameNode :
- Acts as the master node.
- Manages metadata, including file directories, block locations, and permissions.
- Coordinates client requests and assigns tasks to Data Nodes.

2. DataNodes :
- Serve as the worker nodes.
- Store data blocks and handle read/write requests from clients as directed by the NameNode.
- Periodically send heartbeats and block reports to the NameNode.

3 For Data Reliability and Fault Tolerance in HDFS

1. Data Replication :
- Each data block is replicated across multiple Data Nodes (default : 3 replicas).
- Ensures availability even if a node fails.

2. Heartbeat Mechanism :
- DataNodes send heartbeats to the NameNode to signal their availability.

Scribble
Page :
Date : / /

- If a DataNode fails, the NameNode re-replicates its blocks to healthy nodes.

3. Checksums : Data blocks are verified using checksums to detect and recover from corruption.

4. Rack-Awareness : Replicas are distributed across different racks to protect against rack-level failures.

ANSHIKA JAIN 21100BTCSE09473

Assignment - 2

Que 1 How do the key components of the hadhoop ecosystem, such as hive, pig, Hbase and sqoop is integrated with the hadhoop to enhance its capabilities of data storage, processing & analysis.

Ans. 1

1. Hive (SQL for Hadoop)
   - Integration : Runs on top of Hadoop's MapReduce or Tez engine
   - Purpose: Enables SQL-like quering (HiveQL) for structured data stored in HDFS.
   - Enhancement :
     → Simplies data quering for users familier with SQL
     → Ideal for data wearehousing and batch processing tasks.

Example : Running a SELECT query on large data sets stored in HDFS.

2. Pig (Scripting for Hadhoop)
   - Integration : Executes pig latin scripts using Hadoop's MapReduce or Tez engine.
   - Purpose : Simplifies data transformation and processing through a high level scripting language.
   - Enhancement :
     → Easier for ETL (Extract, Transform, load) operations.
     → Optimized for semi-structured or unstructured data.

Example : Processing logs or data pipelines with fewer lines of code than raw MapReduce.

3. HBase (NoSQL for Hadoop)
- Integration : Directly uses HDFS for storage and works with MapReduce of data processing.
- Purpose : Provides real time read/write access to large datasets.
- Enhancement :
  → Adds NoSQL capabilities for handling unstructured and semi-structured data.
  → Useful for applications requiring low-latency queries, like social media or IOT.

Example : Storing and retrieving real time sensor data for analytics.

4. Sqoop (Data Integration)
- Integration : Bridges Hadoop and relational databases (eg. MySQL, Oracle)
- Purpose : Transfers data efficiently between HDFS & relational database.
- Enhancement :
  → Facilitates ingestion of structured data into Hadoop for analysis.
  → Enables exporting processed results back to databases for reporting.

Example : Importing customer data from MYSQL into HDFS for analysis.

ANSHIKA JAIN 21100BTCSE09473

Page: Date: Scribble

Que.2 What are the steps involved in installing & configuring the hadhoop ecosystem on a local environment and data warehousing?

Ans.2 Prerequisites

- Install Java (JDK 8 or higher) and set JAVA_HOME
- Install SSH for password-less communication between nodes (even in pseudo distributed mode).

Hadoop Installation

1. Download Hadoop: Get the latest stable release from Apache Hadoop.
2. Extract Hadoop: Unzip the package into a chosen directory (eg. /usr/local/hadhoop).
3. Set Environment Variables: Add HADOOP_HOME and update the PATH in .bashrc or equivalent.
4. Edit configuration files:
   - core-site.xml: Set the default file system (hdfs://localhost:9000).
   - hdfs-site.xml: Define replication factor and directory paths.
   - mapred-site.xml: Configure the MapReduce framework.
   - yarn-site.xml: Set up the YARN resource Manager.
5. Format the Namenode: Run ~~hdfs~~ hdfs namenode -format.

Page: Date: / /

1.6. Start Hadoop Services : Use start-dfs.sh and start-yarn.sh

Hive Installation and Integration :

1. Download Hive : Get the latest release from Apache Hive
2. Extract Hive : Unzip the package to a chosen directory (eg. /usr/local/hive).
3. Set Environment Variables : Add HIVE_HOME and update the PATH
4. Configure Hive : hive-site.xml
   - Set the metastore database (default: Derby or external database like MySQL).
   - Point to Hadoop's fs.defaultFS.
5. Initialize Metastore : Use schematool -initschema to setup the metastore database
6. Start Hive : Run hive from the terminal.

Efficient Querying and Data Warehousing with Hive :

1. Create Tables : Use HiveQL to define tables (eg. CREATE TABLE ...)
2. Load Data : LOAD DATA INPATH '/path' INTO TABLE table_name).
3. Query Data

Verification

Run a sample MapReduce Job and query data using Hive to verify proper Integration.

Qus. 3

Ans. 3

1.
2.
3.
4.

ANSHIKA JAIN 21100BTCSE09473

Page: Date: Scribble

Que.3 What role does Zookeeper play in the hadoop ecosystem, and how does its installation and configuration contribute to managing distributed applications like HBase and Kafka?

Ans.3 Zookeeper in the Hadoop ecosystem provides distributed coordination and synchronization for applications like HBase and Kafka.

→ Role :

- Manages coordination, configuration, leader election and fault-tolerance in distributed systems.
- Ensures consistency and synchronization across nodes.
- In HBase : Manages region server coordination and master election.
- In Kafka : Handles broker coordination and partial leader election

Installation and configuration

1. Download and install Zookeeper.
2. Configure zoo.cfg for coordination settings
3. Start Zookeeper with zkServer.sh start
4. Integrate with HBase/Kafka by updating their respective config files (hbase-site.xml in hbase OR zookeeper.connect in Kafka).

Qur. 4 How does Sqoop enable data transfer between hadoop and relational databases and what are the primary challenges and considerations when installing Sqoop within a hadoop environment?

Ans. 4. Sqoop enables data transfer between Hadoop and relational databases ~~and what are the primary challenges and~~ by importing data from relational databases (eg. MySQL, oracle) into Hadoop (HDFS, Hive) and exporting processed data from hadoop back to relational databases. It does this using optimized data transfer techniques, leveraging parallel processing for fast & efficient bulk data movement.

Primary challenges & considerations

1. Database Connectivity: Ensuring correct JDBC drivers for connecting to the relational database.
2. Data format Compatibility: Handling different data types and ensuring compatibility between the database schema and HDFS formats.
3. Performance Tuning: Configuring parallelism, batching and split size to optimize large data transfers.

4. Security : Managing authentication (Kerberos, user credentials) and data encryption during transfers.
5. Resource Allocation : Ensuring sufficient resources (memory, CPU) to handle large data loads with overloading the cluster.

Que.5 What are the key advantages of using Mahout for machine learning within the hadoop ecosystem and how does its installation enhance data processing capabilities for large scale predictive analysis ?

Ans.5 Apache Mahout enhances machine learning within the hadoop ecosystem by enabling scalable ~~data~~, efficient data processing for large datasets.

Key Advantages :

1. Scalability : Leverages hadoop's distributed processing for large scale tasks.
2. Efficient Algorithms : optimized for clustering, classification, regression, and recommendations.
3. Seamless Integration : Works directly with HDFS and MapReduce.
4. Distributed Processing : Uses MapReduce and spark for parallel computation.
5. Flexibility : Supports both batch and real time processing.

Scribble
Page :
Date : / /

Installation impact

- Mahout's Algorithms, when deployed on a hadoop cluster, leverage parallel computation for faster processing of large scale data.
- Improves predictive analysis by handling complex computations efficiently across large datasets, enabling data scientists to scale their models.

ANSHIKA JAIN

Assignment 3

21100BTCSE09773

Page: Scribble
Date: / /

Ques: Explain how Hadoop can be used to store and manage large datasets efficiently.

Ans: Hadoop efficiently stores and manages large datasets using HDFS and MAPReduce.

HDFS (Hadoop Distributed File System)

- Data Distribution: Splits large files into smaller blocks and distributes them across multiple nodes.
- Replication: Each block is replicated for fault tolerance.
- Scalability: Easily scales adding more nodes to the cluster.
- Data Locality: Processes data close to where it's stored, reducing transfer time.

MapReduce

- Parallel Processing: Breaks tasks into smaller chunks, distributing them across nodes for faster processing.
- Efficient computation: Uses Map and Reduce functions to process and aggregate data.

Date : / /

Fault Tolrance
- Data is replicated to ensure availability even if a node fails.
- Automatic recovery ensures no data loss

Que.2 What are the primary differences between SQL and NoSQL data management systems, and when is it beneficial to use NoSQL over SQL?

Ans.2

Feature	SQL	NoSQL
Data Model	Relational (structured, tables)	Non-Relational (document, key-value, graph)
Schema	Fixed schema	Flexible or schema less
Scalability	Vertical (scale up)	Horizontal (scale out)
Query language	standard SQL	Varies (eg. JSON queries, APIs)
consistency	Strong ACID compliance	often eventual consistency (CAP theorem)
Use case	structured data, complex queries	Unstructured / semi-structured data, high scalability

NoSQL over SQL →

- When handling large, unstructured, or semi structured datasets that require horizontal scaling.
- For applications requiring fast reads / writes, such as caching or real time analytics.

- When the data structure evolves frequently, like in IOT or dynamic web apps.
- Ideal for handling large scale distributed data with high availability (eg. social networks).
- Use NoSQL when flexibility, scalability and performance for distributed systems outweigh the need for strict consistency and relational integrity.

Que.3 Describe the role of Hive in big data processing and explain how it simplifies querying in Hadoop.

Ans.3 Hive plays a key role in big data processing by providing a SQL-like interface (HiveQL) to query and analyze data stored in Hadoop's HDFS. It simplifies querying in Hadoop in the following ways:

Role in Big Data Processing

- Data warehousing: Acts as a data warehouse infrastructure for managing and querying structured data.
- Integration with Hadoop: Runs on top of Hadoop's MapReduce or Tez engine, leveraging distributed processing for scalability.
- Support for large datasets: Handles large scale batch processing and analysis efficiently.

How Hive Simplifies Querying

1. SQL-like syntax : HiveQL enables users familiar with SQL to query hadoop data without learning complex programming frameworks like MapReduce.
2. Abstraction : Abstracts low-level hadoop processes, converting SQL queries into MapReduce jobs automatically.
3. Data Formats : Supports various file formats (eg. CSV, parquet, ORC), making it versatile for different data types.
4. Ease of use : Simplifies tasks like data transformation, aggregation, and joins for non-programmers.

Que. 4 Compare and contrast SQL and HiveQL. How does HiveQL adapt SQL to work with Hadoop's distributed file system?

Ans. 4

Aspect	SQL	HiveQL
Purpose	Queries Relational databases	Queries data in Hadoop
Underlying system	Works with RDBMS	Built for Hadoop's distributed system
Execution	Executes directly on a database	Translates queries into MapReduce, Tez or Spark Jobs.
Schema	Fixed Schema	Schema on Read (applied during query time)

Performance	Low latency for transactions	Batch processing, optimized for large datasets.
File formats	Works with structured data only	Supports structured & semi-structured

How HiveQL adapts SQL for Hadoop

- Distributed Processing : Converts SQL-like queries into MapReduce, Tez or Spark jobs to process data across multiple nodes in hadoop.
- Schema on Read : Allows defining table schemas over raw data stored in HDFS, enabling flexibility with unstructured and semi-structured data.
- Batch Processing : Optimized for high throughput batch jobs rather than low latency queries.
- Support for Big Data formats : Works with Hadoop-native file formats for efficient storage & querying.

Que.5 How would you query big data stored in Hadoop using Hive? Provide a basic example query in HiveQL.

Ans.5 Steps :

1. Load data into Hive
   - Create a Hive table mapped to data in HDFS.
   - Load the data into table.

2. Write & Execute HiveQL Queries
- Use SQL-like syntax (HiveQL) to query and analyze the data.

Example
Scenario : Query a dataset of sales records stored in HDFS.

1. CREATE A TABLE

```
CREATE TABLE sales_data (
    order_id INT,
    customer_name STRING,
    amount FLOAT,
    order_date STRING )
ROW FORNAT DELIMITED
FIELDS TERMINATED BY ',';
STORED AS TEXTILE ;
```

2. Load Data

```
LOAD DATA INPATH '/hdfs_path/sales.csv'
    INTO TABLE sales_data ;
```

3. Query Data

```
SELECT customer_name , SUM (amount)
    AS total_spent
FROM sales_data
WHERE order_date >= '2024-01-01'
GROUP BY customer_name
ORDER BY total_spent DESC ;
```

ANSHIKA JAIN 21100 BTCSE09773

Output :

The query fetches customers and their total spending since January 1, 2021, sorted by spending in descending order.

Hive enables efficient querying of big data stored in HDFS with familiar SQL like syntax.

Que.6 Explain the steps involved in moving data from an RDBMS to Hive and discuss the reasons for performing this migration.

Ans.6 Steps to Move Data from RDBMS to Hive

1. Prepare Data : Clean and optimize RDBMS data for migration
2. Use sqoop : Export data from RDBMS to HDFS using sqoop :

```
sqoop import --connect jdbc:mysql://db --
username user --table table --target-dir/
hdfs_path
```

3. Create Hive Table : Define a Hive table matching the schema.
4. Load Data into Hive : Import HDFS data into the Hive table :

```
LOAD DATA INPATH '/hdfs_path' INTO
TABLE hive_table;
```

Reasons for Migration :

1. Scalability : Handles massive datasets efficiently.

2. Cost - Effective : Runs on low cost commodity hardware
3. Big Data Analytics : Optimized for batch processing and analysis
4. Schema on read : offers flexibility for working data formats.
5. Integration : works well with big data tools like Spark and Pig.

Que.. 7 What are the key differences between moving data from and RDBMS to HBASE versus moving it to Hive?

Aspect	HBase	Hive
Data Model	NoSQL, column oriented, key-value store	SQL-like, table based relational model
Schema	Schema-less, flexible column families	Schema-on-read, supports structured data.
Use Case	Real time read/write access	Batch processing & analytics
Data format	supports unstructured/ semi-structured data	Works best with structured data.
Processing Engine	Direct interaction (via APIs or tools)	Executes via MapReduce, Tez or Spark
Scalability	Highly scalable for real time workloads	Scalable for large scale batch queries

Page: Scribble
Date: / /

Integration with Hadoop	Directly integrated with HDFS for storage	Relies on HDFS for storage & querying.

When to use HBase

- Real-time access and updates
- Handling unstructured or semi-structured data.
- High write / read throughput with low latency.

When to use Hive

- large scale data analysis or batch processing
- When SQL-like querying is needed
- For structured data warehousing and reporting.

Group Based Question (LETTER : A)
Q1 : GROUP1

Page : Scribble
Date : / /

Que. 1 What is a hadoop cluster and how do you maintain it ?

Ans.. A Hadoop cluster is a group of connected computers (nodes) working together to store and process large datasets. It operates in a master-slave architecture, with :

- Master Nodes : Manage & coordinate cluster activities (eg. NameNode, ResourceManager)
- SlaveNodes : Store Data (via Data Nodes) & perform computations (via NodeManagers).

* Maintaining a hadoop cluster

1. Monitor : Use tools like Ambari or Nagios to track health & performance.
2. Ensure Data Reliability : check block replication and use fsck for errors
3. Optimize : Balance data and allocate resources efficiently in YARN.
4. Routine Maintenance : clean logs and Update components regularly.
5. Secure : Use Kerberos for authentication and set access controls.
6. Backups : Backup metadata & enable NameNode high availability.

* Regular monitoring, optimization, and security ensure smooth cluster operations.