**SRM Institute of Science and Technology**

**College of Engineering and Technology**

**School of Computing**

SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamilnadu **Set B**

**Academic Year: 2023-24 (Even)**

**Test: CLA-T2** **Date: 03-04-2024**

**Course Code & Title:** **21CSE222T BIG DATA TOOLS &TECHNIQUES** **Duration:** 1:30 Hours

**Year & Sem: II Year / IV Sem** **Max. Marks:** 50

**Course Articulation Matrix: *(to be placed)***

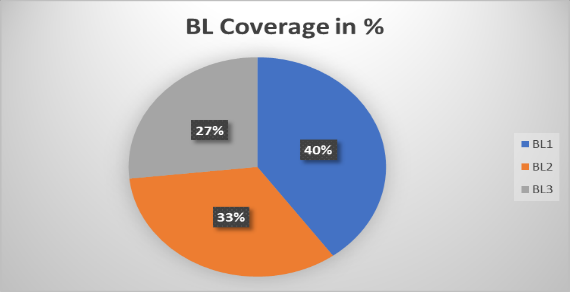
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S.No.** | **Course Outcome** | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** |
| 1 | **CO1** | - | - | - | - | 1 | - |  | - | - | - | - | - |
| 2 | **CO2** | - | - | - | - | 1 | - | - | - | - | - | - | - |
| 3 | **CO3** | - | - | - | - | 1 | - | - | - | - | - | - | - |
| 4 | **CO4** | - | - | - | - | 1 | - | - | - | - | - | - | - |
| 5 | **CO5** | - | - | - | - | 1 | - | - | - | - | - | - | - |

**Part- A (10 x 1 = 10 Marks)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Q. No** | **Question** | **Marks** | **BL** | **CO** | **PO** |
| 1 | The usual way of detecting corrupted data is by computing a \_\_\_\_\_\_\_\_\_ for the data.  a) Error detection  b) Cyclic Redundancy Check  **c) Check Sum**  d) Data Correction | **1** | **L1** | **2** | **5** |
| 2 | In \_\_\_\_\_\_\_\_ mode, Pig runs in a single JVM and accesses the local filesystem.  **a) Local**  b) Map reduce  c) Complex  d) Hadoop | **1** | **L2** | **2** | **5** |
| 3 | \_\_\_\_\_\_\_\_\_\_\_is an interactive shell for running Pig commands.  a) Script  **b) Grunt**  c) Embedded  d) Grant | **1** | **L1** | **3** | **5** |
| 4 | The Hive \_\_\_\_\_\_\_\_\_ makes it easy to run Hive commands from a wide range of programming languages.  a) Thrift Client  b) JDBC Driver  **c) ODBC Driver**  d) Metastore | **1** | **L2** | **3** | **5** |
| 5 | \_\_\_\_\_\_\_\_\_\_\_\_ determine the distribution of data within the subdirectories.  a) Table  **b) Partitions**  c) Buckets  d) Tokens | **1** | **L1** | **3** | **5** |
| 6 | \_\_\_\_\_\_\_\_\_\_\_ engine executes the query and generates results as same as MapReduce results.  a) Process Engine  b) Execution Engine  c) HBASE  **d) MapReduce** | **1** | **L1** | **4** | **5** |
| 7 | \_\_\_\_\_\_\_\_\_\_\_\_\_ is a distributed co- ordination service to manage large set of hosts.  **a) Zoo Keeper**  b) HIVE  c) PIG  d) SCOOP | **1** | **L2** | **4** | **5** |
| 8 | Oozie workflow are written in  a) HTML  **b) XML**  c) PHP  d) JS | **1** | **L1** | **4** | **5** |
| 9 | Oozie Workflow is collection of actions arranged in\_\_\_\_\_\_\_\_\_\_\_  a) DCG  b) UDAG  c) UDCG  **d) DAG** | **1** | **L2** | **4** | **5** |
| 10 | Extend the role of Sqoop \_\_\_\_\_\_\_\_  a) SQL to Hadoop  b) SQL to HBase  **c) MySQL to Hadoop**  d) SQL Hadoop | **1** | **L2** | **4** | **5** |
|  | **Part B**  **Answer any two (2x5=10 Marks)** |  |  |  |  |
| 11 | Describe the Hadoop I/O concept of serialization.   🡪 Serialization in Hadoop refers to the process of converting complex data structures or objects into a binary format for efficient storage, transfer, and processing within a distributed system. Hadoop uses the Writable interface to enable custom object serialization, allowing data to be transmitted between nodes in a compact and optimized manner. Serialization plays a key role in reducing network bandwidth usage, enabling efficient data processing in frameworks like MapReduce and Spark by converting data into a format suitable for distributed computing. Overall, serialization in Hadoop is essential for handling large-scale data operations effectively across a cluster of nodes. | **5** | **L2** | **2** | **5** |
| 12 | Write the differences between PIG and SQL.   Purpose and Domain:  SQL: Designed for querying and manipulating structured data in relational databases.  PIG: Used for processing and analyzing large datasets, particularly in Hadoop, using a procedural scripting language.  Data Model:  SQL: Works with structured data stored in tables with predefined schemas.  PIG: Handles semi-structured or unstructured data with flexible schemas, suitable for Big Data processing.  Syntax:  SQL: Declarative syntax specifies what data to retrieve or manipulate.  PIG: Procedural data flow language (Pig Latin) specifies a sequence of data transformations using operators.  Optimization:  SQL: Relational databases optimize queries internally using indexing and query plans.  PIG: Relies on MapReduce for distributed processing and performance optimization.  Schema:  SQL: Requires a defined schema before querying; schema changes may require alterations to tables.  PIG: Adopts schema-on-read, determining data schema at runtime based on data loading and processing.  Extensibility:  SQL: Limited extensibility beyond standard SQL operations.  PIG: Offers greater extensibility with User Defined Functions (UDFs) for custom processing logic.  Usage:  SQL: Widely used in relational databases and data warehouses for business analytics.  PIG: Popular in Hadoop environments for preprocessing and transforming large datasets before analysis. | **5** | **L2** | **3** | **5** |
| 13 | Explain the components of spark. 🡪 Spark Core:  Foundation of Spark framework handling task scheduling, memory management, and interacting with storage systems. Uses Resilient Distributed Datasets (RDDs) for distributed data processing.  Spark SQL:  Enables integration of structured data processing with Spark's functional programming APIs. Provides DataFrame abstraction for SQL-like querying of distributed datasets.  Spark Streaming:  Extension of Spark Core for real-time stream processing of data from sources like Kafka or Flume. Allows applying batch-like transformations on live data streams.  MLlib (Machine Learning Library):  Scalable machine learning library offering algorithms for classification, regression, clustering, and feature engineering. Leverages Spark's distributed computing for large-scale model training and deployment.  GraphX:  Graph processing library enabling manipulation and computation on graph-structured data. Supports algorithms like PageRank and Connected Components for analyzing large-scale graphs.  SparkR:  R package allowing interactive Spark jobs from the R shell. Provides DataFrame-based API for data manipulation and analysis within R environment.  Spark Ecosystem:  Spark has a rich ecosystem of additional libraries and tools for various use cases, including real-time analytics (Spark Streaming), graph analytics (GraphX), and machine learning (MLlib). | **5** | **L3** | **4** | **5** |
|  | **Part C**  **Answer any two (15x2=30 Marks)** |  |  |  |  |
| 14 | A company has a vast amount of log data generated from various applications and systems. They want to perform analytics on this data to gain insights into user behavior and system performance. Explain how Hive can be leveraged in this scenario.  Data Storage and Schema Definition:  Hive allows the log data, stored as files in HDFS or other compatible file systems, to be structured and queried using a schema. A Hive table can be created to define the structure of the log data, specifying columns, data types, and file formats (e.g., CSV, JSON, Parquet) used to store the logs.  Data Ingestion and ETL:  The log data can be ingested into Hive tables using Hive's LOAD DATA command or by directly writing data into HDFS directories that are associated with Hive tables. Prior to ingestion, Extract-Transform-Load (ETL) processes can be performed to clean, transform, and enrich the log data as needed.  Querying with SQL-Like Interface:  Once the data is ingested into Hive tables, users can leverage Hive's SQL-like query language called HiveQL to write and execute complex analytical queries on the log data. HiveQL allows users to perform aggregations, filtering, joins, and other SQL operations to extract meaningful insights from the log data.  Optimized Query Execution:  Hive optimizes query execution by translating HiveQL queries into MapReduce jobs (or other execution engines like Tez or Spark SQL) that run on the Hadoop cluster. This enables parallel and distributed processing of the log data, making it scalable and efficient for large-scale analytics.  Integration with Ecosystem Tools:  Hive integrates seamlessly with other tools and frameworks in the Hadoop ecosystem. For example, data analysts can use Apache Zeppelin or BI tools like Tableau or Power BI to visualize Hive query results and create interactive dashboards based on log data analytics.  Schema Evolution and Flexibility:  Hive supports schema evolution, allowing changes to the table schema without affecting the underlying data. This flexibility is beneficial when dealing with evolving log formats or when adding new types of log data to the analytics pipeline.  Performance Tuning:  Hive provides various optimization techniques such as partitioning, bucketing, and indexing to improve query performance on large datasets. By optimizing data organization and storage, Hive can efficiently handle complex analytics tasks on log data. | **15** | **L3** | **3** | **5** |
| 15 | A distributed system requires coordination of distributed tasks with ordered execution guarantees. Each task should be executed only after the completion of its predecessor. Explain how Apache ZooKeeper can be utilized for implementing distributed coordination and sequencing. How would you use ZooKeeper's sequential znodes to enforce task ordering?   ZooKeeper provides primitives like sequential znodes that can be used to enforce task ordering and ensure that tasks are executed in a predefined sequence. Here's how you can use ZooKeeper for this purpose:  Utilizing ZooKeeper for Distributed Coordination:  ZooKeeper is a distributed coordination service that provides features like distributed locks, leader election, and synchronization mechanisms.  Each node in the distributed system can connect to ZooKeeper to coordinate tasks and maintain a consistent view of the system's state.  Sequential Znodes in ZooKeeper:  ZooKeeper allows the creation of znodes with sequential numbering. When a znode is created as sequential (CreateMode.SEQUENTIAL), ZooKeeper appends a unique sequence number to the znode's name.  The sequence number is based on a monotonically increasing counter maintained by ZooKeeper, ensuring that znodes are created in a predictable order.  Implementing Task Ordering with Sequential Znodes:  To enforce task ordering using ZooKeeper:  When a task needs to be executed, the client creates a sequential znode under a designated parent znode in ZooKeeper.  The task's execution logic is associated with the znode creation process. The task's predecessor znode (the previous task in the sequence) can be determined based on the sequence numbers.  By monitoring the sequence numbers of the znodes, clients can determine the execution order of tasks. Each client waits until the znode corresponding to its predecessor task is completed (i.e., deleted or marked as finished) before proceeding with its own task execution.  Steps for Task Execution:  Client A wants to execute Task A.  Client A creates a sequential znode (e.g., /tasks/task\_, with sequential number appended by ZooKeeper).  Client A monitors the znodes to determine when its predecessor (the last completed task in the sequence) is finished.  Once the predecessor's znode is deleted or marked as completed, Client A proceeds with executing Task A.  Task execution logic is triggered upon successful znode creation and predecessor completion.  Handling Failures and Recovery:  ZooKeeper provides reliability and fault-tolerance, ensuring that even in the event of failures (e.g., client disconnects or node crashes), the sequencing logic can be maintained and tasks can be executed in the correct order.  Clients can implement retry mechanisms and session management to handle temporary failures and ensure consistency in task sequencing. | **15** | **L3** | **3** | **5** |
| 16 | A financial institution wants to develop a comprehensive risk management system that can handle large-scale data processing and complex analytics on financial transactions in real-time. Explain how Apache Spark can be leveraged to build such a risk management system.   Real-time Data Processing:  Spark Streaming, a component of Apache Spark, can be used to ingest and process real-time data streams of financial transactions. Streaming data can be received from various sources such as Kafka, Flume, or TCP sockets.  Spark Streaming enables continuous processing of streaming data, allowing the risk management system to analyze transactions as they occur and respond in near real-time to potential risks or anomalies.  Data Transformation and Enrichment:  Spark's core processing engine (Spark Core) can be utilized to perform data transformation and enrichment on financial transaction data.  Spark provides a rich set of APIs (in Scala, Java, Python, or SQL) that allow developers to write complex data manipulation logic, apply business rules, and derive new features from raw transaction data.  Complex Analytics and Machine Learning:  Apache Spark's MLlib (Machine Learning Library) offers scalable machine learning algorithms that can be applied to perform predictive analytics and risk modeling.  MLlib supports various machine learning tasks such as classification, regression, clustering, and anomaly detection, allowing the risk management system to identify and mitigate potential risks in financial transactions.  Interactive Data Analysis:  Spark SQL provides a SQL-like interface for querying and analyzing financial transaction data stored in distributed datasets (e.g., Parquet files, Hive tables).  Data analysts and risk managers can use Spark SQL to interactively explore transaction data, perform ad-hoc queries, and generate custom reports for risk assessment and compliance purposes.  Scalability and Performance:  Apache Spark is designed for distributed computing and can scale horizontally to handle large volumes of data across a cluster of machines.  Spark's in-memory processing capabilities and efficient task scheduling (using Directed Acyclic Graphs) ensure high performance and low latency for complex analytics on financial datasets.  Integration with Ecosystem Tools:  Spark integrates seamlessly with other components in the Hadoop ecosystem (e.g., HDFS, Hive, Kafka), enabling seamless data integration and interoperability with existing data pipelines and infrastructure.  Fault Tolerance and Reliability:  Spark's built-in fault tolerance mechanisms (like lineage graphs and RDD lineage) ensure data integrity and recoverability in case of node failures or data processing errors, critical for maintaining the reliability of a risk management system. | **15** | **L3** | **4** | **5** |

**\*Program Indicators are available separately for Computer Science and Engineering in AICTE examination reforms policy.**

**Course Outcome (CO) and Bloom’s level (BL) Coverage in Questions**



**Approved by the Audit Professor/Course Coordinator**