Big data

Assignment Questions

PW SKILLS

1. Explain the core components of the Hadoop ecosystem and their respective roles in processing and storing big data. Provide a brief overview of HDFS, MapReduce, and YARN.

2. Discuss the Hadoop Distributed File System (HDFS) in detail. Explain how it stores and manages data in a distributed environment. Describe the key concepts of HDFS, such as NameNode, DataNode, and blocks, and how they contribute to data reliability and fault tolerance.

3. Write a step-by-step explanation of how the MapReduce framework works. Use a real-world example to illustrate the Map and Reduce phases. Discuss the advantages and limitations of MapReduce for processing large datasets.

4. Explore the role of YARN in Hadoop. Explain how it manages cluster resources and schedules applications. Compare YARN with the earlier Hadoop 1.x architecture and highlight the benefits of YARN.

5. Provide an overview of some popular components within the Hadoop ecosystem, such as HBase, Hive, Pig, and Spark. Describe the use cases and differences between these components. Choose one component and explain how it can be integrated into a Hadoop ecosystem for specific data processing tasks.

6. Explain the key differences between Apache Spark and Hadoop MapReduce. How does Spark overcome some of the limitations of MapReduce for big data processing tasks?

7. Write a Spark application in Scala or Python that reads a text file, counts the occurrences of each word, and returns the top 10 most frequent words. Explain the key components and steps involved in this application.

8. Using Spark RDDs (Resilient Distributed Datasets), perform the following tasks on a dataset of your choice:
a. Filter the data to select only rows that meet specific criteria.
b. Map a transformation to modify a specific column in the dataset.
c. Reduce the dataset to calculate a meaningful aggregation (e.g., sum, average).

9. Create a Spark DataFrame in Python or Scala by loading a dataset (e.g., CSV or JSON) and perform the following operations:
a. Select specific columns from the DataFrame.
b. Filter rows based on certain conditions.
c. Group the data by a particular column and calculate aggregations (e.g., sum, average).
d. Join two DataFrames based on a common key.

10. Set up a Spark Streaming application to process real-time data from a source (e.g., Apache Kafka or a simulated data source). The application should:
a. Ingest data in micro-batches.
b. Apply a transformation to the streaming data (e.g., filtering, aggregation).
c. Output the processed data to a sink (e.g., write to a file, a database, or display it).

11. Explain the fundamental concepts of Apache Kafka. What is it, and what problems does it aim to solve in the context of big data and real-time data processing?

12. Describe the architecture of Kafka, including its key components such as Producers, Topics, Brokers, Consumers, and ZooKeeper. How do these components work together in a Kafka cluster to achieve data streaming?

13. Create a step-by-step guide on how to produce data to a Kafka topic using a programming language of your choice and then consume that data from the topic. Explain the role of Kafka producers and consumers in this process.

14. Discuss the importance of data retention and data partitioning in Kafka. How can these features be configured, and what are the implications for data storage and processing?

15. Give examples of real-world use cases where Apache Kafka is employed. Discuss why Kafka is the preferred choice in those scenarios, and what benefits it brings to the table.

Submission Guidelines:

- Answer all the questions in a single Jupyter Notebook file (.ipynb).
- Include necessary code, comments, and explanations to support your answers and implementation.
- Ensure the notebook runs without errors and is well-organized.
- Create a GitHub repository to host your assignment files.
- Rename the Jupyter Notebook file using the format "date_month_topic.ipynb" (e.g., "21st_September_GAN.ipynb").
- Place the Jupyter Notebook file in the repository.
- Commit and push any additional files or resources required to run your code (if applicable) to the repository.
- Ensure the repository is publicly accessible.
- Submit the link to your GitHub repository as the assignment submission.

Grading Criteria:

- Understanding and completeness of answers: 40%
- Clarity and depth of explanations: 25%
- Correct implementation and evaluation of optimizer techniques: 15%
- Analysis and comparison of different optimizers: 10%
- Proper code implementation and organization: 10%

Note: Create your assignment in Jupyter notebook and upload it to GitHub & share that uploaded assignment file link through your dashboard. Make sure the repository is public.