Reference Material: Problem Solving Using Big Data Models and Approaches

Introduction to Big Data Problem Solving

Big Data refers to massive volumes of structured, semi-structured, and unstructured data that require specialized techniques for storage, processing, and analysis. Solving problems using Big Data requires understanding various models and approaches that help in handling data efficiently.

Big Data Characteristics (5Vs)

1. **Volume** - Large amounts of data generated every second.
2. **Velocity** - Speed at which data is generated and processed.
3. **Variety** - Different types of data (structured, semi-structured, unstructured).
4. **Veracity** - Reliability and accuracy of data.
5. **Value** - Extracting meaningful insights from data.

Big Data Processing Models

1. Batch Processing Model

- Processes data in large blocks at scheduled intervals.
- Example: Apache Hadoop MapReduce.
- **Use Case:** Large-scale ETL operations, historical data analysis.

2. Real-Time (Stream) Processing Model

- Processes data as it arrives in real-time.
- Example: Apache Spark Streaming, Apache Flink.
- **Use Case:** Fraud detection, live dashboards, sensor data processing.

3. Lambda Architecture

- Hybrid model combining batch and real-time processing.
- **Layers:** Batch Layer, Speed Layer, Serving Layer.
- **Use Case:** Recommendation engines, real-time analytics with historical context.

4. Kappa Architecture

- Simplified architecture using only stream processing.
- Example: Apache Kafka, Apache Samza.
- **Use Case:** Internet of Things (IoT) applications, real-time monitoring.

Big Data Approaches for Problem Solving

1. Data Storage Solutions

- **HDFS (Hadoop Distributed File System):** Stores large datasets across clusters.
- **NoSQL Databases (MongoDB, Cassandra):** Handles semi-structured and unstructured data.
- **Cloud Storage (AWS S3, Azure Blob):** Scalable data storage solutions.

2. Data Processing Techniques

- **MapReduce:** Parallel processing framework.
- **Spark RDDs & DataFrames:** Efficient in-memory computing.
- **Hive & SQL-on-Hadoop:** Querying large datasets with SQL-like syntax.

3. Data Ingestion Methods

- **Batch Ingestion:** Sqoop, Apache Nifi.
- **Stream Ingestion:** Apache Kafka, Apache Flume.

4. Machine Learning in Big Data

- **Supervised Learning:** Predictive modeling using labeled data.
- **Unsupervised Learning:** Clustering large datasets.
- **Big Data ML Tools:** Apache Mahout, MLlib (Spark).

5. Security & Governance in Big Data

- **Data Privacy Regulations (GDPR, CCPA).**
- **Access Control (Kerberos, Ranger, Atlas).**
- **Data Encryption & Masking Techniques.**

Hands-on Demonstrations

1. **Batch Processing with Hive** - Creating tables, loading data, running queries.
2. **Real-time Data Processing using PySpark** - RDD operations, DataFrame transformations.
3. **Stream Processing with Kafka** - Setting up topics, producing & consuming messages.

Conclusion

Big Data problem solving involves selecting the right model and approach based on the problem's requirements. Understanding batch vs. real-time processing, leveraging cloud solutions, and applying machine learning techniques enables organizations to make data-driven decisions efficiently.

Topic Covered	**Key Details**	**Application (Use Cases)**
Introduction to Hive	- Overview of Apache Hive - Importance of Hive in Big Data - Hive Architecture and Components	- Used in Data Warehousing for querying large datasets - Enables SQL-like querying on structured data in Hadoop
Setting Up Hive & Database	- Starting Hive shell - Creating a database (employee_db) - Switching to the database	- Helps in organizing large datasets into different databases - Useful in enterprise-level data management
Creating Hive Table	- Defined schema for employees table - Used ROW FORMAT DELIMITED and STORED AS TEXTFILE	- Structuring raw data for efficient querying - Used in retail and finance for customer data storage
Loading Data into Hive	- Created sample data employees.txt - Moved data to HDFS - Used LOAD DATA INPATH command	- Data ingestion in ETL pipelines - Used in real-time analytics for data lakes
Performing Hive Queries	- Basic queries (SELECT *, COUNT(*)) - Filtering data (WHERE, ORDER BY) - Aggregation (GROUP BY, AVG(Salary))	- Used for generating business intelligence reports - Common in fraud detection and trend analysis
Hive Optimization & Wrap-Up	- Discussed partitioning & bucketing - Dropping tables (DROP TABLE) - Best practices for query performance	- Improves query performance for large-scale data processing - Common in financial transactions and IoT analytics
Introduction to PySpark & RDDs	- Overview of Apache Spark & PySpark - Resilient Distributed Datasets (RDDs) - Creating and transforming RDDs	- Used for distributed parallel computing - Common in real-time data processing and ETL workflows
RDD Operations in PySpark	- Transformation Operations: ✔ map(), flatMap(), filter() ✔ distinct(), union(), intersection() ✔ groupByKey(), reduceByKey(), sortByKey() - Action Operations: ✔ collect(), count(), first(), take() ✔ reduce(), foreach()	- Used for log processing, IoT data analysis, and real-time streaming - Common in recommendation engines and large-scale computations

DataFrame Operations in PySpark	- Creating DataFrames from in-memory data - Transformation Operations: ✔ select(), filter(), where() ✔ groupBy(), agg(), orderBy() ✔ withColumn(), drop(), fillna(), dropDuplicates() - Action Operations: ✔ show(), count(), describe(), summary() - Handling missing values & data cleansing	- Efficient data manipulation in analytics pipelines - Used in fraud detection, risk analysis, and machine learning
PySpark SQL Queries	- Registering DataFrames as temporary SQL tables - Writing SQL queries using Spark SQL - Performing joins (INNER, LEFT, RIGHT, FULL) - Aggregations (SUM, AVG, MIN, MAX)	- Useful in building real-time data pipelines - Used for querying large-scale structured datasets in finance, e-commerce, and healthcare

✅ Outcome: Participants gained hands-on experience with Hive, PySpark RDDs, DataFrame transformations, and SQL queries, enabling them to handle structured and semi-structured big data efficiently.