Experiment-10

Aim: To perform Batch and Streamed Data Analysis using Apache Spark.

Theory:

1. **What is Streaming? Explain Batch and Stream Data:**
    Streaming refers to the continuous processing of real-time data as it arrives. It is commonly used in applications that require immediate action such as fraud detection, stock market analysis, and live dashboards. Streaming data is unbounded, time-sensitive, and flows in continuously.

Batch data processing, in contrast, involves collecting data over a period and processing it together. It is widely used in data warehousing, periodic reporting, and data transformation tasks. The data is bounded and processed in chunks with scheduled jobs.

Examples:

● Batch: Generating monthly sales reports.

● Stream: Real-time user click analysis on a website.

2. **How data streaming takes place using Apache Spark:**
    Apache Spark handles stream processing through its Structured Streaming engine. Structured Streaming treats incoming data streams as an unbounded table and performs incremental computation using the same DataFrame API used for batch jobs.

The streaming data can be ingested from various sources such as Kafka, sockets, directories, or cloud storage. Spark then processes the data using transformations like filter, select, groupBy, and aggregations. Developers can apply window operations, manage late-arriving data using watermarking, and use checkpointing for fault tolerance.

Internally, Spark divides the live stream into micro-batches. These micro-batches are processed using the Spark engine and then output to sinks like HDFS, databases, or dashboards. With its high scalability and distributed nature, Apache Spark ensures that real-time data processing can be performed with low latency and high throughput.

Key Features:

● Unified APIs for batch and streaming

● Support for stateful computations

- Integration with structured data sources

- Fault-tolerant and scalable architecture

Use Case Examples:

- Real-time transaction monitoring

- Streamed log analysis

- Live social media analytics

Conclusion:
In this experiment, I gained a strong understanding of the differences between batch and streaming data processing. I learned that batch processing is ideal for historical and periodic tasks, while streaming suits real-time, continuous data needs. Through Apache Spark, I explored Structured Streaming, which provides a powerful, unified framework to handle both types of workloads. I learned how to ingest live data from sources like Kafka or files, apply transformations, and output results dynamically. This helped me appreciate Spark's capabilities in managing complex data pipelines and real-time analytics. Overall, I understood how Spark's architecture enables scalable and fault-tolerant processing, making it a preferred tool for modern data-driven applications.