Experiment 10

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

Theory:

1. What is streaming? Explain batch and stream data.

Ans) Streaming is the process of transmitting and processing data continuously and in real-time. Instead of waiting for all the data to be collected (like in batch processing), streaming handles data as soon as it's generated. This allows for immediate analysis and actions — useful in scenarios like fraud detection, live video, and real-time analytics.

A. **Batch Data Analysis**:

- Processed in groups after collecting over time.
- Not real-time, higher latency.
- Examples: Monthly billing, daily sales reports.
- Tools: Hadoop, Hive.

B. Streamed Data Analysis:

- Processed in real-time as it's generated.
- Low latency, continuous flow.
- Examples: Live chat, stock market feeds.
- Tools: Kafka, Spark Streaming.

The main difference is that in batch processing, data is processed later in chunks, while in stream processing, data is processed continuously in real-time as it arrives.

2. How does data streaming take place in Apache Spark?

Ans) In Apache Spark, data streaming is handled by Spark Streaming, which allows for the processing of real-time data in a distributed and fault-tolerant manner. Here's how the process works:

1. **Data Ingestion**: Spark Streaming can ingest real-time data from various sources like Kafka, Flume, Sockets, or file systems like HDFS or Amazon S3. The data is continuously received in small chunks.
2. **Micro-batches**: Spark Streaming divides the incoming data stream into micro-batches, which are small, manageable batches that are processed in intervals (typically in milliseconds or seconds). This approach allows Spark to handle streaming data while leveraging its batch processing engine.
3. **Processing**: Each micro-batch is processed using Spark's standard transformations (like map, filter, etc.) and actions (like reduce, count, etc.). These operations are applied on the data in the batch, similar to how traditional Spark processes batch data.
4. **Output**: After processing, the results of the micro-batches are typically written to external storage systems (such as HDFS, databases, or dashboards) or used to trigger further actions.
5. **Fault Tolerance:** Spark Streaming ensures fault tolerance by replicating the data across multiple nodes and periodically checkpointing the data, so that the system can recover from failures without losing progress.

Conclusion: In this experiment, we learned how Apache Spark Streaming processes real-time data using micro-batches. It enables efficient, scalable, and fault-tolerant real-time analytics, making it ideal for applications like monitoring and fraud detection. Spark Streaming provides a powerful solution for large-scale data processing.