Name: Div: Roll no:

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 and unbounded flow of data generated in real-time from various sources such as sensors, logs, online transactions, or social media platforms. Unlike traditional processing methods that wait for the data to be collected, streaming processes each piece of data almost instantly as it arrives, often within milliseconds or seconds. This allows for real-time analytics, monitoring, and event-driven applications.

- Batch Data is collected and stored over a time period, then processed as a single unit. This method is effective when immediate results are not required. It is widely used for ETL processes, periodic reporting, and historical data analysis, where latency is acceptable.

- Stream Data, on the other hand, is continuously produced and must be processed in near real-time. It is ideal for applications that require instant feedback, like fraud detection systems, stock trading platforms, live metrics dashboards, or user activity tracking. The main advantage of stream processing is its ability to provide timely insights and actions.

2. How Data Streaming Takes Place Using Apache Spark.

Apache Spark Streaming is a powerful component built on top of the core Spark engine, enabling real-time stream processing by using a micro-batch architecture. It processes data streams in small batches, which balances performance and latency. Here's how it works:

- Input Sources: Spark Streaming can ingest data from a variety of real-time sources such as Apache Kafka, Apache Flume, Amazon Kinesis, TCP sockets, or even files monitored in real-time.

- Micro-batch Architecture: Spark collects the data for a short interval (e.g., every 1 or 2 seconds) and creates a batch. These batches are then processed using the same

APIs used for batch jobs, which makes it easier for developers to switch between batch and stream use cases.

- DStreams (Discretized Streams): Spark represents incoming streaming data as DStreams, which are essentially a continuous series of RDDs. This abstraction allows developers to apply familiar operations such as map(), reduceByKey(), filter(), etc., to process data.

- Transformations and Actions: Developers can perform complex operations on the stream such as aggregations, joins, and windowed computations. The results can be sent to databases, file systems, or real-time dashboards.

- Output: The final output of stream processing can be stored or visualized in real-time, enabling instant feedback and data-driven decisions. Spark also supports stateful operations to maintain data across batches.

Conclusion:

Apache Spark offers a robust and unified platform for handling both batch and stream data processing. While batch processing is efficient for handling large volumes of historical data, Spark Streaming brings the capability to process data in near real-time, helping organizations gain insights instantly. Its micro-batch architecture strikes a good balance between performance and complexity, making it suitable for both small-scale and enterprise-grade real-time applications. Spark's ability to connect to a variety of input/output systems, combined with fault tolerance and scalability, makes it a leading choice for modern data processing needs. Moreover, Spark Structured Streaming now provides even more powerful and user-friendly APIs for developing continuous applications with built-in support for event time, stateful processing, and fault recovery.