Assignment No:- 2

- Aim:- Comparative analysis between Batch and Streamed data processing tools like Map-reduce, Apache spark, Apache Flink, Apache Samza, Apache kafka and Apache Storm.

- Theory:-

- Batch Processing:-
Involves processing data in chunks or batches, typically over period of time. It is suited for scenarios where real-time data processing is not required.

- Stream Processing:-
Involves processing data in real time as it arrives.
It is ideal for time-sensitive data and applications requiring continuous updates and low latency processing.

- MapReduce (Batch Processing)

- Primary use case:-
Batch processing of large datasets

typically in Hadoop ecosystems.

- Processing mode:-
  Batch (processes large chunks of stored data on disk)

- Strengths:-
  - Scalability for large datasets
  - Distributed and fault-tolerant
  - Good for offline analytics

- Limitations:-
  - High latency (due to disk-based storage)
  - Complex to manage and requires significant resources for low-latency applications.
  - Not suitable for real-time data processing

- Apache Spark (Batch & Stream processing):-

- Primary use case:-
  Both batch and stream proces

- Processing mode:-
  Batch and Stream (via stru
  Streaming)

- Processing Mode :-
Stream (low- latency processing with kafka integration).

- Strengths :-
  - Strong integrations with Apache kafka
  - Designed for distributed stream Processing with built - in support for Stateful Processing
  - Supports exactly - once processing
  - Easily scales to handle large data volumes.

- Limitations :-
  - Limited flexibility in terms of advanced event processing Compared to Apache flink.
  - less mature in comparison to flink and Spark
  - Requires kafka for messaging, So not suitable for systems without kafka.

• Apache kafka (Stream Processing):-

- Primary Use case:-
Distributed messaging and event Streaming

- Processing Mode:- Stream (message queue for continuous stream data processing)

- Strength:-
  - Distributed, fault-tolerant, and high throughput message broker.
  - Enables event-driven architectures and stream Processing.
  - Provides durability and fault tolerance for stream data
  - Widely used for real-time data pipelines

- Limitations:-
  - Not a processing engine itself, typically paired with tools like Samza, Flink or storm for data processing.
  - Not optimized for complex Computation or real-time analytics on its own.

- Apache Storm (Stream Processing):

- Primary Use case:-
Real-time stream processing for low latency use cases.

- Processing Mode:-
Stream (real-time, low latency processing)

- Strengths:-
  - Highly suitable for real-time, low latency processing
  - Supports both batch and stream-based operations (for some use cases).
  - Robust fault-tolerance and scalability
  - Highly configurable with complex event processing capabilities.

- Limitations:-
  - Not suitable for batch processing or high-throughput scenarios that don't require ultra-low latency

- Managing large clusters and scalability can be challenging

- Not as flexible as Spark or Flink for complex data transformations.

• Real-Time vs Near Real-Time Processing:-

- Batch Processing (MapReduce):-

- Works on large datasets and processes in descrete chunks, so it is not suitable for real-time processing.

- Typically has high latency, meaning it may take minutes, hours, or even days to process batches.

- Stream Processing (Apache Flink, Apache Storm, Apache Samza):-

- Designed for continuous data streams, providing real-time processing.

- Flink and Storm focus on microsecon to milliseconds latency for processing incoming data, making them ideal for real-time

- Strengths :-
  - Can handle both batch and streaming data.
  - In - memory processing, which provides much faster processing compared to MapReduce.
  - Highly flexible and supports machine learning (MLib) and graph processing (GraphX).
  - Scalability of large datasets
  - Built-in libraries for SQl queries, machine learning and graph processing.

- Limitations:-
- Stream processing in micro-based, So it is not low-latency as true stream processing systems

- Requires good understanding of Spark architecture to tune performanc

* Apache Flink (Stream Processing):-

- Primary use case:-
  Real- time Stream processing

- Processing Model-

Stream (true event-driven processing with low latency)

- Strengths :-
- High throughput and low-latency stream processing.
- Supports exactly-one processing semantics
- Fault-tolerant with stateful processing
- Can handle both stream & batch processing, but optimized for real-time stream processing
- Advanced event-time processing and windowing capabilities

- Limitations :-
- More complex to deploy and manage compared to batch tools.
- Can be harder to integrate with batch-based legacy systems.

- Apache Samza (Stream Processing) :-

- Primary use cases:
Stream processing in conjuction with Apache kafka.

decision - making systems.

- Samza, integrated with Apache kafka, provides low-latency Stream processing, ideal for applications that need to react to events as they occur.

- Data Consistency and Processing Semantics:

- MapReduce:-

- Primarily operates in "batch" mode, so consistency and handling of data changes over time are not concern.

- However, any inconsistency or failure during MapReduce job can cause entire job to fail.

- Apache Spark:-

- Supports "atleast once" Processing semantics for stream processi (using micro-batches) but can also configured for "exactly one" sematecs with more advanced setups

- Apache Plink:

- Highly effeciencent in handling stateful stream processing and guarantees exactly-one semantecs with ensures no data duplication.

- CONCLUSION :-

Hence, we have successfully learned about comparative analysis between batched and streamed data processing tools like Mapredu apache spark, apache plink, apache samza, apache kafka, apache storm.