Spark Vs. Flink

Lecturer: Mr. Chan Sophal

Team 2: Touch Sopheak
Pov Phannet
Luk Chamnan
Sam Lyheng

vs.

Table of Content

Introduction

Apache Flink is an open-source, high-performance framework designed for large-scale data processing, excelling at real-time stream processing. It features low-latency and stateful computations, enabling users to process live data and generate insights on-the-fly. Flink is fault-tolerant, scalable, and provides powerful data processing capabilities that cater to various use cases.

Introduction (Cont.)

Apache Spark is a versatile, open-source data processing framework that offers an all-in-one solution for batch processing, machine learning, and graph processing. It is known for its ease of use and comprehensive library of built-in tools and algorithms. Like Flink, Spark is fault-tolerant, scalable, and delivers high-performance data processing. Spark's versatility makes it suitable for a wide range of applications and industries.

Comparison

Processing Models:

- Apache Flink: Primarily focused on real-time stream processing, Flink efficiently processes large volumes of data with low-latency. Flink's processing engine is built on top of its own streaming runtime and can also handle batch processing.
- Apache Spark: Originally designed for batch processing, Spark later introduced a micro-batching model for handling streaming data. While it can process streaming data, its performance in terms of latency is generally higher than Flink's.

Comparison

APIs and Libraries:

- Apache Flink: Provides a robust set of APIs in Java, Scala, and Python for developing data processing applications. Flink's libraries include FlinkML for machine learning, FlinkCEP for complex event processing, and Gelly for graph processing.
- Apache Spark: Offers APIs in Java, Scala, Python, and R, making it accessible to a wider range of developers. Spark also has comprehensive libraries, such as MLlib for machine learning, GraphX for graph processing, and Spark Streaming for processing real-time data.

Comparison

Fault Tolerance:

- Apache Flink: Utilizes a distributed snapshotting mechanism, allowing for quick recovery from failures. The state of the processing pipeline is periodically checkpointed, ensuring data consistency in case of failures.
- Apache Spark: Employs a lineage information-based approach for fault tolerance. Spark keeps track of the data transformation sequence, enabling it to recompute lost data in case of failures.

Comparison

Windowing:

- Apache Flink: Offers advanced windowing capabilities, including event-time and processing-time windows, as well as session windows for handling complex event patterns. Flink's windowing features are particularly suitable for real-time stream processing.
- Apache Spark: Provides basic windowing functionality, such as tumbling and sliding windows, which work well for batch and micro-batching scenarios but may not be as suited for real-time stream processing.

Performance & Scalability

Real-time Processing:

- Flink: Generally outperforms Spark with lower latency due to its stream-first architecture and efficient state management. Benchmarks show Flink can process millions of events per second with minimal delay.
- Spark: Spark Streaming offers near real-time capabilities but typically incurs slightly higher latency than Flink due to micro-batching. Latency depends on micro-batch size and processing time.

Performance & Scalability

Batch Processing:

- Spark: Excels in large-scale batch processing tasks. Benchmarks demonstrate Spark's ability to handle petabyte-scale datasets efficiently, making it ideal for ETL pipelines and iterative algorithms.
- Flink: Can also handle batch processing with micro-batching, but is not as optimized as Spark for purely batch workloads. Its performance may be comparable or slightly lower than Spark depending on the specific task.

Performance & Scalability

Scalability:

- Spark: Generally considered easier to scale due to its wider adoption and mature cluster management tools. Existing infrastructure and expertise in Spark might make scaling smoother.
- Flink: Requires careful configuration for stateful streaming applications to ensure efficient scaling and load balancing. However, once properly configured, Flink can scale well for both streaming and batch workloads.

Use Case

- Use Cases of Apache Spark

Apache Spark is ideal for use cases that require batch processing, such as data warehousing, ETL, and machine learning.

- Use Cases of Apache Flink

Apache Flink is ideal for use cases that require real-time data processing and stateful stream processing, such as real-time analytics, machine learning, and event-driven architectures.

Use Case

- Real-time processing: Flink excels due to its design for streaming data with near-instantaneous processing.
- Batch processing: Spark's in-memory processing and optimized engine make it ideal for large-scale batch tasks.
- Machine Learning: Spark's MLlib provides a comprehensive suite of machine learning tools, making it suitable for ML-focused projects.

Use Case

- Graph processing: Spark's GraphX excels in graph computations; Flink's Gelly is available but less mature.
- Stateful processing: Flink is superior for stateful processing, ideal for maintaining state during stream processing.
- Community & ecosystem: Spark offers a larger community and ecosystem, providing extensive resources and third-party integrations.

Conclusion

Apache Flink and Apache Spark are both powerful data processing frameworks, each with its unique strengths and capabilities. The choice between the two depends on your specific use case and requirements. Flink is particularly well-suited for stateful and real-time stream processing, while Spark excels in machine learning and graph processing.

Thank you