Comparison between Apache Flink and Apache Spark

Fernanda de Camargo Magano
Dylan Guedes
About Flink

- Open source streaming processing framework
- Stratosphere project started in 2010 in Berlin
- Flink started from a fork of this project
- Apache project in March 2014
- Flink Forward - annual Conference
Flink’s Architecture

Source: Introduction to Apache Flink book
Flink - Sources and sinks

- Flink programs are mapped to streaming dataflows (DAGs) that:
  - Start with one or more **sources**
  - End in one or more **sinks**

<table>
<thead>
<tr>
<th>Source/sink</th>
<th>Sink</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Kafka (source/sink)</td>
<td>Hadoop FileSystem (sink)</td>
</tr>
<tr>
<td>Apache Cassandra (sink)</td>
<td>RabbitMQ (source/sink)</td>
</tr>
<tr>
<td>Amazon Kinesis Streams (source/sink)</td>
<td>Apache NiFi (source/sink)</td>
</tr>
<tr>
<td>Elasticsearch (sink)</td>
<td>Twitter Streaming API (source)</td>
</tr>
</tbody>
</table>
Flink - Data formats

- Read/write in text files
- CSV files
- JSON
- Relational database (SQL)
- HDFS
Time

- Event, ingestion and processing time

**Source:** Flink website
To be able to travel back in time and reprocess the data correctly, the stream processor needs to support event time.
Flink - Windows

Source: Flink website
Flink - Session Windows

Windows with a better fit to how sessions naturally occur.

Source:
Flink book

Flink is currently the **only open source stream processing engine** that supports sessions.
Consistency

- Exactly once guarantee
- Both Spark Streaming and Flink have this guarantee
- In Spark comes with performance and expressiveness cost
- Flink is able to provide this guarantee, together with low-latency processing, and high throughput all at once.
Some benchmarks

Source: Apache Flink book
Why Flink?

- **Easy of working** with it compared with other technologies
- Deals with both **stream and batch** processing
- It has a growing and energetic **community**
- **Exactly-once** guarantees
- Correct time/window semantics
- High throughput and low latency (usually a trade-off in other tools)
Examples of Apache Flink in Production

**King.com** (more than 200 games in different countries)
- Flink allows to handle these massive data streams
- It keeps maximal flexibility for their applications.

**Zalando** (Online fashion platform in Europe)
- They employ a microservices style of architecture

**ResearchGate** (Academic social network)
- Adopt Flink since 2014 for batch and stream processing
Use Case at Ericsson

- Real-time analysis of logs and system performance
- Monitor a live cloud infrastructure
- Checks whether is behaving normally or an anomalous behavior
- Flink is important to this application to:
  - Correctly classifying anomalies
  - Produce the same result when running the same data twice (event time)
Use Case at Ericsson

Streaming architecture using Apache Flink at Ericsson.

Source: Introduction to Apache Flink Book
About

- **More than 1000** contributors (Apache Flink has less than 400)
- Started in 2009, at Berkeley
- Supports Python, R, Scala e Java
- Won the 2014 Daytona Sort, with a **4.27 TB/min** performance
- Used by Netflix, Amazon, Baidu, eBay, MyFitnessPal, NetEase, Yahoo, TripAdvisor...
Spark SQL

- Lazy processing
- Memory and disk for processing
- Great fault-tolerance mechanics
Spark Structured Streaming

- Uses micro-batches to achieve soft real time processing
- Great fault-tolerance mechanics
- Great throughput
When should I use it?

- Is non-hard real time a problem for you?
- The available sources and sinks match the ones that you have?
## Comparison table - Flink and Spark

<table>
<thead>
<tr>
<th></th>
<th>Flink</th>
<th>Spark</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Event size – stream</strong></td>
<td>single</td>
<td>micro-batch</td>
</tr>
<tr>
<td><strong>Delivery guarantees</strong></td>
<td>exactly once</td>
<td>exactly once</td>
</tr>
<tr>
<td><strong>State Management</strong></td>
<td>checkpoints (distributed snapshots)</td>
<td>checkpoints</td>
</tr>
<tr>
<td><strong>Fault tolerance</strong></td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Out-of-order processing</strong></td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Primarily written in</strong></td>
<td>Java</td>
<td>Scala</td>
</tr>
<tr>
<td><strong>Windowing</strong></td>
<td>Time and count based</td>
<td>Time based</td>
</tr>
<tr>
<td><strong>Resource Management</strong></td>
<td>YARN and Mesos</td>
<td>YARN and Mesos</td>
</tr>
<tr>
<td><strong>Auto-scaling</strong></td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
References

