Spark Streaming

Introduction

- Streaming transmits data as a continuous flow, which allows the recipients to start processing almost immediately. The data could be coming from multiple sources. The data could be going to multiple sinks.
- A stream represents a continuous sequence of events that goes from producers to consumers, where an event is defined as a key-value pair.
- Common examples: Audio, Video (for example Netflix, iTunes, Hulu, YouTube etc)
- More interesting examples!
  - The analysis of GPS car data can allow cities to optimize traffic flows based on real-time traffic information.
  - Uber uses Spark streaming to detect and visualize popular Uber locations
  - Smart cities will be using about 1.39 billion connected cars, IoT sensors, and devices by 2020. The analysis of location and behavior patterns within cities will allow optimization of traffic, better planning decisions, and smarter advertising
  - Real time data in manufacturing can be used for anomaly detection

Streaming in Spark

- Spark provides two ways of processing stream data:
- Spark Streaming: a separate library in Spark to process continuously flowing streaming data. It provides us with the `DStream` API, which is powered by Spark RDDs. DStreams provide us data divided into chunks as RDDs received from the source of streaming to be processed and, after processing, sends it to the destination.
- Structured Streaming: Built on the Spark SQL library, Structured Streaming is is based on `DataFrame` and `Dataset` APIs. Hence, with this library, we can easily apply any SQL query (using the `DataFrame` API) or Scala/Java operations (using `Dataset` API) on streaming data.
- We can express our streaming computation the same way we would express a batch computation on static data. The Spark SQL engine will take care of running it incrementally and continuously and updating the final result as streaming data continues to arrive.

More on Structured Streaming in Spark

- The system ensures end-to-end exactly-once fault-tolerance guarantees through checkpointing and Write-Ahead Logs.
- Internally, by default, Structured Streaming queries are processed using a micro-batch processing engine, which processes data streams as a series of small batch jobs thereby achieving end-to-end latencies as low as 100 milliseconds and exactly-once fault-tolerance guarantees.
- Since Spark 2.3, a new low-latency processing mode called Continuous Processing has been introduced that can achieve end-to-end latencies as low as 1 millisecond with at-least-once guarantees.