MSC Basismodul : Study Stream Processing Platforms and develop a real time Data Analytics application

Bhargav J. Bhatt

January 21, 2019

Abstract

Stream processing is an ideal platform where data streams are produced continuously and requires real time processing. Over the past years such platforms have been developed which ensures distributed, high-performance, scalable, stateful and real time processing. Example of such platform are Apache Flink, Apache Spark

1 Introduction

Majority of today's data processing is done on data that is continuously produced.The reason is that often processing big volumes of data is not enough.Data has to be processed fast, so that a firm can react to changing business conditions in real time.This is required for trading, fraud detection, system monitoring, and many other examples. Before,data streaming technology was lacking in several areas, such as performance, latency, and operability, users were forced to run their own applications to ingest and analyze these continuous data streams, or use batch processing tools to process these continuous data.

This report will cover the brief overview of Apache Flink [1] and Apache Spark [2]. Followed by their respective architecture and comparison. This report describes on the code implementation of Stream processing using Apache Flink and Apache Kafka [3]. The data that is used is the New York City parking tickets generation from year 2014 to 2017, sourced from Kaggle.com [4]

2 Stream processing

Stream processing is the processing of continuous data. Stream Processing turns Typical Database Storage System around: The application logic, analytics, and queries exist continuously, and data flows through them continuously. Upon receiving an event from the stream, a stream processing application reacts to that event: it may trigger an action, update an aggregate or other statistic, or store that event for future reference. Streaming computations can also process multiple data streams jointly, and each computation over the event data stream may produce other event data streams. The systems that receive and send the data streams and execute the application or analytics logic are called stream

processors. The basic responsibilities of a stream processor are to ensure that data flows efficiently and the computation scales and is fault tolerant [5]. Stream processing deals with unbounded datasets which are infinite datasets which are added continuously and run continuously as long as data is received. Apache Flink is best example for Stream processing.

3 Apache Flink

3.1 Introduction

Apache Flink is an open source distributed platform for stream and batch processing.Flink is built on the philosophy that many classes of data processing applications, including real-time analytics, continuous data pipelines, historic data processing (batch), and iterative algorithms (machine learning, graph analysis) can be expressed and executed as pipelined fault-tolerant dataflows [6].

3.2 Features

The Following are the features of Apache Flink which make Flink stand out as a stream processing system in the open source.

- Support for event time and out of order streams.
- Expressive and easy-to-use APIs in Scala and Java.
- Support for sessions and unaligned windows.
- Consistency, fault tolerance, and high availability.
- Low latency and high throughput.
- Integration.
- Support for batch.

3.3 Architecture

A Flink runtime program compiles of core libraries that allows to process streaming data. It can run locally and also on cluster either on Flink standalone or hadoop. Flink can also be run on cloud either on Google compute engine or Amazon web service. On top of runtime lies two set of core APIs. One is datastream API which deals with the stream processing and dataset API deals with the batch processing . CEP (complex event processing) which is on top of data stream API allows you to match input streaming data against patterns. The table API allows you to build table and can run SQL like operations on streaming data. On top of data set API, Flink have FlinkML, a machine learning library which allows you to run machine learning algorithms. The Gelly Graph processing allows you to inter connecting entities in form of social graphs [6].

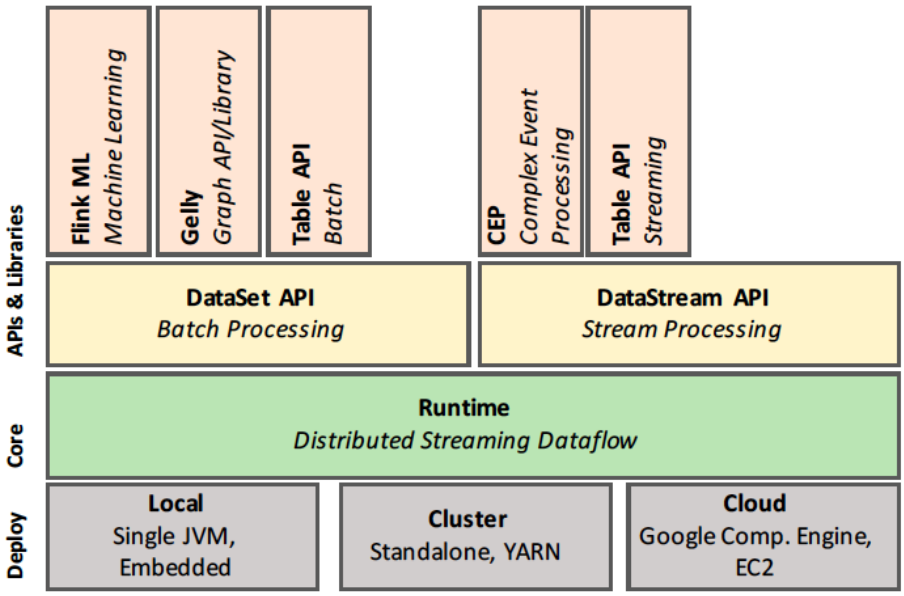

Figure 1: The Flink software stack.

3.4 Flink Programming Model

Flink programming model consist of 3 main components. Initially the Data source where the data is inputed. Transformation where the operation are performed on data. And Lastly the Data Sink where processed data is either stored, showed using representation or even act as input stream for another further transformation [7].

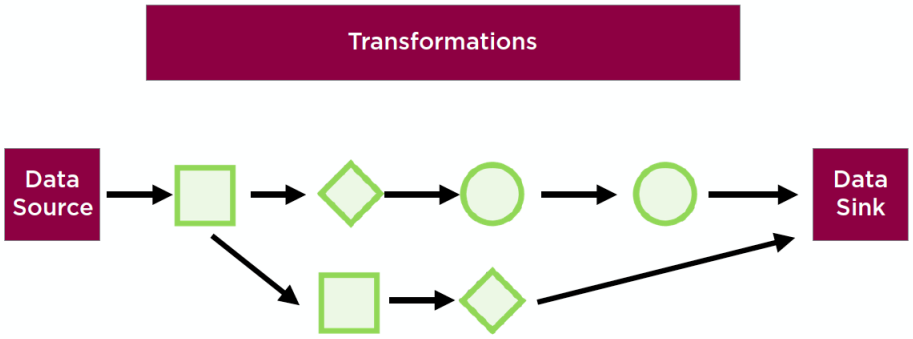

Figure 2: The Flink Programming Model.

4 Batch Processing

Batch Processing is the processing of Bounded dataset, which is finite unchanged dataset to be analyzed. Batch processing runs for a specific time, completes the

process and releases the resources once finished.

5 Apache Spark

5.1 Introduction

Scalable data processing is essential for computer applications dealing with large bounded dataset and typically involves a complex sequence of processing steps with different computing systems. Spark simplify this task by introducing a unified programming model and engine for big data applications. Spark is a generalized framework for distributed data processing providing functional API for manipulating data at scale, in-memory data caching and reuse across computations. It applies set of coarse-grained transformations over partitioned data and relies on dataset's lineage to recompute tasks in case of failures [8].

5.2 Spark Architecture

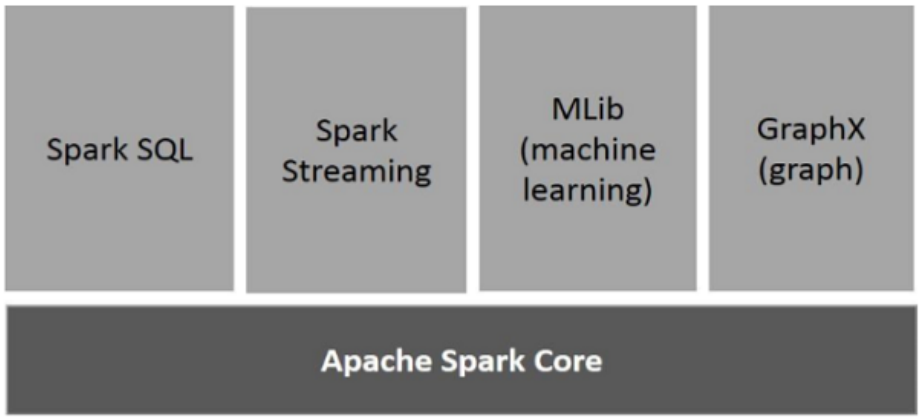

Figure 3: The Spark software stack.

Spark Core is the underlying general execution engine for spark platform that all other functionality is built upon. It provides In-Memory computing and referencing datasets in external storage systems. Spark SQL is a component on top of Spark Core that introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data. Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini-batches and performs RDD (Resilient Distributed Datasets) transformations on those mini-batches of data. MLlib is a distributed machine learning framework above Spark because of the distributed memory-based Spark architecture. GraphX is a distributed graph-processing framework on top of Spark. It provides an API for expressing graph computation that can model the user-defined graphs by using Pregel abstraction API [2].

5.3 Spark Programming model

Spark uses a master/worker architecture. There is a spark context that communicate to a single coordinator called master that manages workers in which executors run.Each application gets its own executor processes, which stay up for the duration of the whole application and run tasks in multiple threads.The system currently supports three cluster managers, namely Apache Mesos, Hadoop YARN and standalone [2].

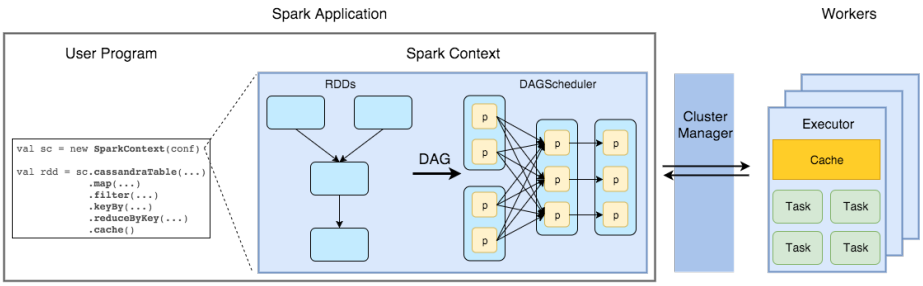

Figure 4: The Spark program model.

6 Comparison Batch and Stream Processing

Table 1: Batch and Stream Processing Comparison

Batch Processing	**Stream Processing**
Bounded finite datasets	Unbounded infinite datasets
Slow pipeline from data ingestion to analyses	Process immediately as data is received
periodic updates as job completes	Continuous updates as job runs constantly
Order of data received is not important	Order of data received is important
Single global state of world at any point of time	No global state, only history of events

7 Implementation

7.1 Data

The source of data is of NYC Parking Tickets from Kaggle [4]. The NYC Department of Finance collects data on every parking ticket issued in NYC (10M per year!). There are 55 fields, however for implementation purpose only 8 fields are selected and filter out. These fields are describe as below:

- Summons Number - unique number generated for each ticket
- Plate ID - number plate of vehicle
- Issue Date - date of issue of ticket
- Violation Code - type of violation

- Vehicle Body Type - type of vehicle
- Vehicle Make - manufacture of vehicle
- Violation Time - time when ticket was generated
- Violation County - state where the violation took place

Moreover Violation code can also be mapped with external source [9] which provides fine charges on each tickets. Thus it can also give economical perspective of Data.

7.2 Inspiration

Various results can be obtained from the dataset that is provided. Main Inspiration to get from data are:

- Tickets count per day, per month, per year.
- Where are tickets most commonly issued?
- What are the most common types of cars to be ticketed.
- Fine charged per month, year.

7.3 Architecture

The data is inputed through Apache Kafka medium and processed by Apache Flink and again output to Apache Kakfa (Multiple topics). The versions of Apache Flink and Apache Kafka used is Flink 1.3.2 and Kafka 1.0.0

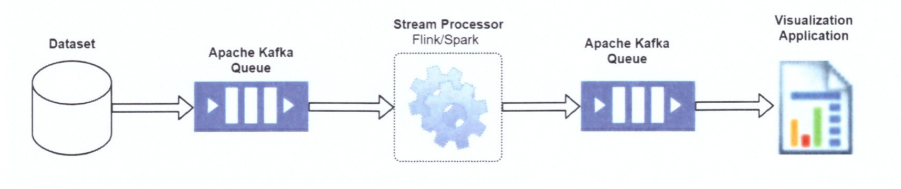

Figure 5: Streaming Pipeline

7.4 Process

The CSV files containing NYC parking tickets data is inputed as string from localhost as "dataStream". This string is processes through .flatmap(new RowSplitter) and break into Tuple. The fields which are relevant to these inspiration mentioned above, are only selected.

```
//Data inputed from Kafka
DataStream<String> dataStream = env.addSource(new
    FlinkKafkaConsumer082<>(parameterTool.getRequired("topic"),
    new SimpleStringSchema(), parameterTool.getProperties()));
```

```
//filtered datastream (only selected fields stored from string to tuple)
       DataStream<Tuple11<String, String, String, String, Integer,
           String, String, String, Integer, Integer, Integer>>
           dataStream2 = dataStream.flatMap(new RowSplitter());
```

The tuple is stored in table using Flink Table API and SQL. Various SQL queries which are supported by Flink Streaming process are executed.

```
//storing data streaming into table
tableEnv.registerDataStream("nycdata", dataStream2,
    "sNum,plateId,registrationState,issueDate,violationCode,
vehicleBodyType,vehicleMake,violationTime,Vday,Vmonth,Vyear");
```

```
//executing SQL query
Table query3 = tableEnv.sql("SELECT Vmonth, COUNT(Vmonth) as cnt Charge
    FROM nycdata GROUP BY Vmonth");
```

```
//storing back as dataStream from SQL result
DataStream<Tuple2<Boolean, Row>> retractStream =
              tableEnv.toRetractStream(query1, Row.class);
```

The Window functions are also applied to using Table SQL API. However it can also be applied to data stream to get results.

```
// compute Count using tumble window
Table query4 = tableEnv.sql(
  "SELECT Vmonth, " +
  "  TUMBLE_START(rowtime, INTERVAL '1' DAY) as wStart, " +
  "  Count(Vmonth) FROM Orders " +
  "GROUP BY TUMBLE(rowtime, INTERVAL '1' DAY), Vmonth");
```

8 Conclusion

Apache Flink can be used for both Stream as well as Batch Processing. The main advantage of Apache Flink is high throuhput, low latency and fault tolrance with low over head. Large data set can be processed even on local machine to generate respective output results.

9 Acknowledgement

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. A special gratitude I give to our Professor Dr. Michael Böhlen for giving me splendid opportunity to expose such developing platform. Furthermore I am highly indebted to Muhammad Saad for his guidance and constant supervision as well as for providing necessary information regarding Apache Flink and also for their support in developing the code.

References

[1] Apache Flink, *www.flink.apache.org.*

[2] Apache Spark, *www.spark.apache.org.*

[3] Apache Kafka, *www.kafka.apache.org.*

[4] NYC Parking Tickets, Kaggle *https://www.kaggle.com/new-york-city/nyc-parking-tickets.*

[5] What is stream processing? *https://data-artisans.com/what-is-stream-processing.*

[6] Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S. et al. (2015) *Apache flink: Stream and batch processing in a single engine.*

[7] Introduction to Apache Flink *https://flink.apache.org/introduction.html.*

[8] Apache Spark: core concepts, architecture and internals *http://datastrophic.io/core-concepts-architecture-and-internals-of-apache-spark/.*

[9] Violation Codes, Fines, Rules and Regulations*http://www1.nyc.gov/site/finance/vehicles/services-violation-codes.page.*