Optimizing Latency Issues in Real-time Streaming Data in Big Data using Spark Stream Processing

Karan Shah and Kalpana Mudaliar

December 2, 2019

Reduce Latency Issues in Real-time Streaming Data in Big Data using Spark Stream Processing

Karan Shah[1] and Prof Kalpana Mudaliar[2]

[1] PG Scholar, Gandhinagar Institute of Technology, Gujarat, India
[2] Assistant Professor, Gandhinagar Institute of Technology, Gujarat, India

Abstract. Real-time data streaming is the process by which big volumes of data are processed quickly such that a firm extracting the info from that data can react to changing conditions in real time. Real-time streaming data is used in E-commerce, Network monitoring, Risk management, Fraud detection, Pricing and analytics. Apache Spark has become one of the most popular open source frameworks in the world also known as key cluster-computing frameworks. Spark is deployed in many ways like in Machine Learning, streaming data, and graph processing. Stream processing means each time we will process each data is processed when it arrives.

Keywords: Data, Big Data, Apache Spark, Stream Processing

1 Introduction

The term "Data" means collection of many types of data likes facts, such as numbers, words, measurements, observations or even just descriptions of things. Data can be found in two types qualitative or quantitative. Qualitative data is means descriptive information (it describes something). Quantitative data is means numerical information (numbers)[4]. Big Data is also known as data but with a large amount of data. Big Data is a term used to describe a collection of different types of data that is huge in size and yet growing exponentially with time. In short such data is so large and complex that none of the traditional data management tools are able to store it or process it efficiently [5].

Fig. 1. Big Data, Spark

Big data is primarily defined by the volume of a data set. Big data sets are generally huge – measuring tens of terabytes – and sometimes crossing the threshold of petabytes. The term big data was preceded by very large databases (VLDBs). Let's consider example of Big Data the New York Stock Exchange generates about one terabyte of new trade data per day. The statistic shows that 500+terabytes of new data get ingested into the databases of social media site Facebook, every day. This data is mainly generated in terms of photo and video uploads, message exchanges, putting comments etc. A single Jet engine can generate 10+terabytes of data in 30 minutes of flight time. With many thousand flights per day, generation of data reaches up too many Petabytes**.** Apache Spark has become one of the most popular open source framework in the world. Spark is deployed in many ways like in Machine Learning, streaming data, and graph processing. Stream Processing is belonging to big data technology. It is used to query continuous data stream and detect conditions, quickly, within a small time period from the time of receiving the data. The detection time period varies from few milliseconds to minutes.

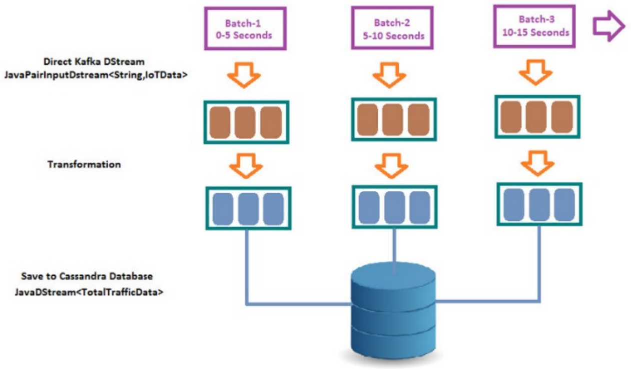
2 What Is Data Streaming?

Data streaming is the process of transferring a stream of data from one place to another place, to a sender and recipient or through some network connections [6]. Data streaming is processed in multiple ways with various types of protocols and tools that help provide security, efficient delivery and other data results. Real-time data streaming is the process by which big volumes of data are processed quickly such that a firm extracting the info from that data can react to changing conditions in real time. Real-time streaming data is used in E-commerce, Network monitoring, Risk management, Fraud detection, Pricing and analytics.

Fig. 2. Data Streaming

3 Problems in Data Streaming

Traditional methods include usage of tools such as spark which are mainly based on batch processing, newly arriving data elements are collected into a group. The whole group is then processed at a future time (as a batch, hence the term "batch processing"). Exactly when each group is processed can be determined in a number of ways–for example, it can be based on a scheduled time interval (e.g. every five minutes, process whatever new data has been collected) or on some triggered condition (e.g. process the group as soon as it contains five data elements or as soon as it has more than 1MB of data). This concept is currently being used in oil companies, which uses this concept to detect the oil rigs and End to End Application for Monitoring Real-Time Uber Data. In Oil companies, Sensors are set-up in oil rigs to generate streaming data, which is processed by Spark and divided into batch and then stored in HBase, for use by various analytical and reporting tools. We want to store every batch and single event in HBase as it streams in [7]. We also want to filter for, and store alarms. Daily Spark batch processing will store aggregated summary statistics. While any user is trying to book a cab on Uber it fetches their latitude and longitude to find the nearest ride, if the nearest driver does not accept or reject the ride it starts looking for the cabs that are little bit far from the user's location and it continues this process till it finds a ride for the user, hence the algorithm performs this step in form of batches which is the exact concept of batch processing [8]. So, there are some of the reasons for not using Batch Processing. **Reason 1**: Some data naturally comes as a never-ending stream of events. To do batch processing, you need to store it, stop data collection at some time and processes the data. Then you have to do the next batch and then worry about aggregating across multiple batches. **Reason 2:** Batch processing lets the data build up and try to process them at once over time. **Reason 3:** Sometimes data is huge and it is not even possible to store it.


Batch-1
0-5 Seconds
Batch-2
5-10 Seconds
Batch-3
10-15 Seconds
Direct Kafka DStream
JavaPairInputDstream<String,IoTData>
Transformation
Save to Cassandra Database
JavaDStream<TotalTrafficData>

Powered by infoq.com

Fig. 3. Batch-processing

4 Proposed Solution

In batch processing we have seen there are many reasons to not use the batch processing in large data and application. Because it has taken huge time to process data and store that data. So, we are using the stream processing in data streaming. So, there are some of the reasons for using Stream Processing. Reason 1: Stream processing naturally fit with time series data and detecting patterns over time. For example, if you are trying to detect the length of a web session in a never-ending stream (this is an example of trying to detect a sequence). It is very hard to do it with batches as some session will fall into two batches. Stream processing can handle this easily. Reason 2: while stream processing process data as they come in hence spread the processing over time. Hence stream processing can work with a lot less hardware than batch processing. Reason 3: Stream processing let you handle large fire horse style data and retain only useful bits

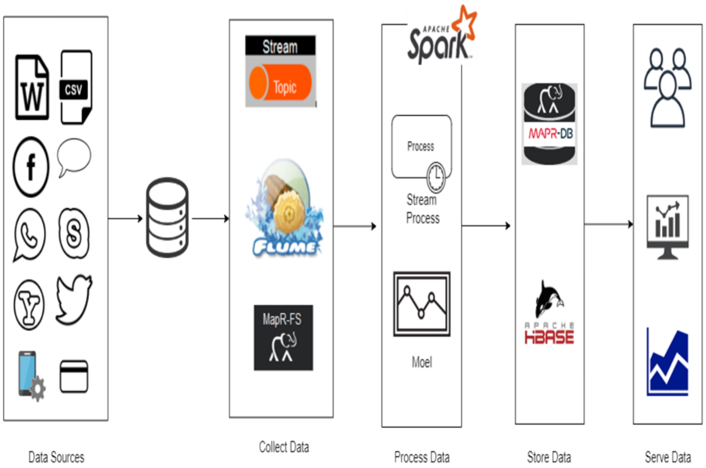


Fig. 4. stream processing In Spark

As shown in the above diagram there are 5 components we are using. In the first components is for Data Source. Data source means in this component we input the different types the files like csv format, Word Document, Social media Etc. Second components id for collect that input data from Data source components in this component we are using the three tools for collecting the data. First tool is Stream Topic this is basically is list. Lists the topics that are in a stream, as well as the number of partitions in each topic. There are two parameters in the Stream topic 1: Path,2: Topic. The path and name of the stream for which you want to display information about topics. The name of the topic for which you want to display information. Second tool is Apache Flume. Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flows. It is robust and fault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms. It uses a simple extensible data model that allows for online analytic application. Third tool is MapR-FS. MapR is a company that offers a Distributed Data Platform to store and analyze data of any size (typically big data) in a distributed fashion which is also linearly scalable.

In third components we are using the stream processing for process the Data. So, in this model we process the whole of the data at once and not divided into batches. After the process, we create the models for that data. In fourth component, we display the two databases. But we are using one of the databases to store the streaming data. MapR-DB is an enterprise-grade, high-performance, in-Hadoop, NoSQL ("Not Only SQL") database management system. You can use it to add real-time, operational analytics capabilities to Hadoop. Apache HBase™ when you need random, real-time read/write access to your Big Data. This project's goal is the hosting of very large tables -- billions of rows X millions of columns -- atop clusters of commodity hardware. In fifth components we serve the data to customer, business dash boards and share the analytics to data to customer's dash board in against the clock to deliver data to where it needs to be as quickly as possible to ensure that the insight extracted is as fresh as possible.

5 Conclusion

The proposed technique will be focused on database should be able to store hundreds of terabytes of data, handle billions of requests per day and have a 100% uptime and better efficiency.

6 Future Work

In batch processing, we have seen there are many reasons to not use the batch processing in large data and application. Because it takes so much time to process, store and retrieve the data. So, we will optimize time with the use of the stream processing for streaming data. This technique will be used for all data from business that is growing explosively, changing data types and sources and processing in real-time, with a more robust ability to deliver the right data at the right time.

References

1. Umit demirbaga, Devki Nandan Jha.: A Batch and Real-time Data Analytics Framework for Healthcare Applications(2018).

2. K. Tawsif, Jakir Hossen, Joseph Emerson Raja, Md Arif Hossain.: A Review on Complex Event Processing Systems for Big Data(2018).

3. Muttipati Appala Srinuvasu, Egala Bhaskara Santhosh.: Big Data: Challenges and Solutions(2017).

4. Data, https://www.mathsisfun.com/data/data.html.

5. Big Data, https://www.guru99.com/what-is-big-data.html.

6. Data Streaming, https://www.techopedia.com/definition/31747/real-time-data-streaming.

7. Batch Processing in Oil Pipelines,https://mapr.com/blog/real-time-streaming-data-pipelines-apache-apis-kafka-spark-streaming-and-hbase/.

8. Batch Processing in Uber,https://mapr.com/blog/monitoring-uber-pt4/