Group 3 Project Report:
Social Media Analysis Using Apache Kafka

Surmeet Kaur: 54245972
Rohan Rajeev: 38249388
Aiswarya Manikandan: 94646410

Introduction

With the massive amounts of data being accumulated from various sources, analysis of Big Data is vastly important for decision making of truly any kind—whether it is for businesses, scientific study, or the improvement of technology as a few examples[1]. Data is mainly generated from social media, mobile devices, internet searches, sensor data and so on, and it only increases with the advent of technology. Today, it has become a valuable commodity for big technology companies, who use it for various research purposes.

Moreover, real-time applications rely upon instantaneous input and fast analysis to arrive at a decision or action within a short and very specific timeline. Originally, data analytics were performed after storing data on hard disks, which eventually have a fair amount of access latency. Recently, there has been a recent transition from hard disk drive storage to memory storage[2]. In-memory processing significantly decreases the amount of access latency, which will have a crucial role when real-time analytics is performed.

Dealing with social media data, including many different data types such as text messages, photos, and videos which are arriving in a large volume in every second, needs a proper framework which does not rely upon storing data on hard disks and can process data in memory, as it arrives.

In this project, we propose an intermediate “scheduler” to the already existing kafka architecture to manage incoming data efficiently to prevent huge losses, which also includes removing data which is not relevant anymore.

The following section talks about the related advancements in data streaming. Next we talk about the Apache Kafka architecture and how it is relevant to our project, followed by a section where we explain our methodology in detail. We then proceed to present our findings, and conclude our work in the last section.

Related work

As data streaming increases with the surge of social media and other platforms, big data analysis and it’s tools have the need to evolve to keep up with the pace of data generation. Tools usually analyse interactions as large interconnected networks and graphs and must make

use of optimal graph analysis algorithms [3]. Existing tools analyse data in 3 phases: Preprocessing and storing in appropriate data structures, analysing data with a graph kernel and storing results after post-processing. These methodologies work efficiently for static graphs, however with streaming data, we need dynamic processing assuming that properties change with time.

Although many algorithms have been proposed, but they are not enough. Big data processing is a complicated process and requires intricate design and architecture by connecting various modules. The frameworks that exist have been classified into 4 categories [4]. Data Mining frameworks such as GraphLab, FlexGP and Apache Mahout which are use distributed machines to “mine” datasets in parallel. Frameworks such as Apache Kafka which are used for messaging in distributed systems for real-time datastreams. Frameworks such as Apache Hadoop and Apache Storm which are used for parallel application development for data processing. Database frameworks such as MongoDB and Cassandra which store the processed data and allow it to be queried according to requirements. All these frameworks display properties of being well-structured, providing high-throughput with low-latency and they are able to efficiently handle high-speed, high-volume and varied data streams.

Kafka

We make use of Apache’s open-source stream-processing software for processing of heterogenous data incoming at different rates. Kafka provides a “unified, high-throughput, low-latency platform” [5] for handling real-time data feeds. It is a scalable and distributed publisher/subscriber messaging system which can easily be connected to external systems. The three main capabilities of the software [8] are:

- Publish and subscribe to streams of records, similar to a message queue or enterprise messaging system.
- Store streams of records in a fault-tolerant durable way.
- Process streams of records as they occur.

Kafka is mainly used for building data pipelines for reliable transmission between applications or systems and if required, transforming this data as required for further analysis.

The kafka architecture is as described:

- Records: immutable data structures that store data as key-value pairs, along with timestamps.
- Topics: records are sent as stream under a particular topic, eg: order or users
- Consumers: consumes the stream of records from the broker and
- Producers: Producers search for new brokers and send data to them asynchronously.
- Brokers: acts as the intermediate for load balancing data between producers and consumers. It takes data input from the producers and forwards it to the consumer at a rated pace.
- Logs: These are used to store records topic-wise on the storage disks. These can be broken into partitions and segments.

- Clusters: these consists of many brokers, which may be distributed across servers.

Figure 1 gives a representation of the Kafka architecture.

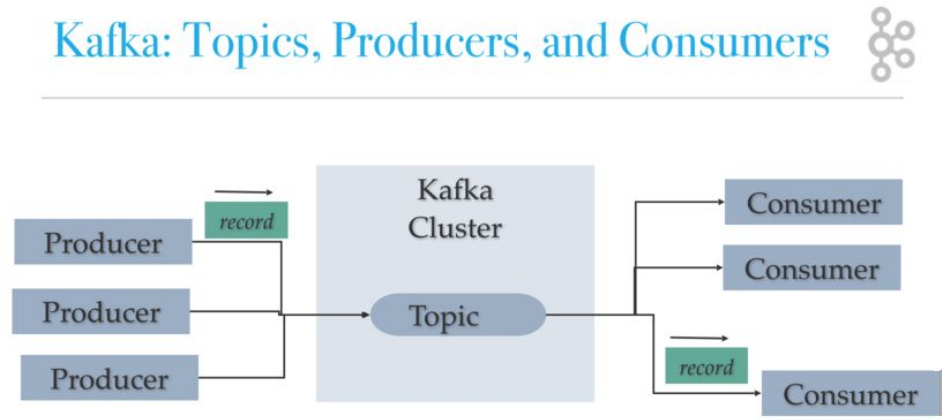

Figure 1: Kafka Architecture

We use Kafka as our primary system for distributing data. To the existent architecture, we add a “scheduler” module for the purpose of managing input rates efficiently.

Methodology

The main objective of our project is to define a scheduling middleware to determine the selection of processing logic based on data ingestion from multiple sources. We establish a data-pipeline which inputs live streaming data, tweets from twitter and posts from reddit, and processes it for analysis. We use Apache Kafka for the same. Figure 2 shows the architecture followed in this project. Input from Twitter and Reddit is fed to two Kafka producers into the same topic, and the Kafka consumer consumes from this topic. Based on the consumption rate, the scheduling middleware will decide the processing technique Pa or Pb, based on the rate of data received.

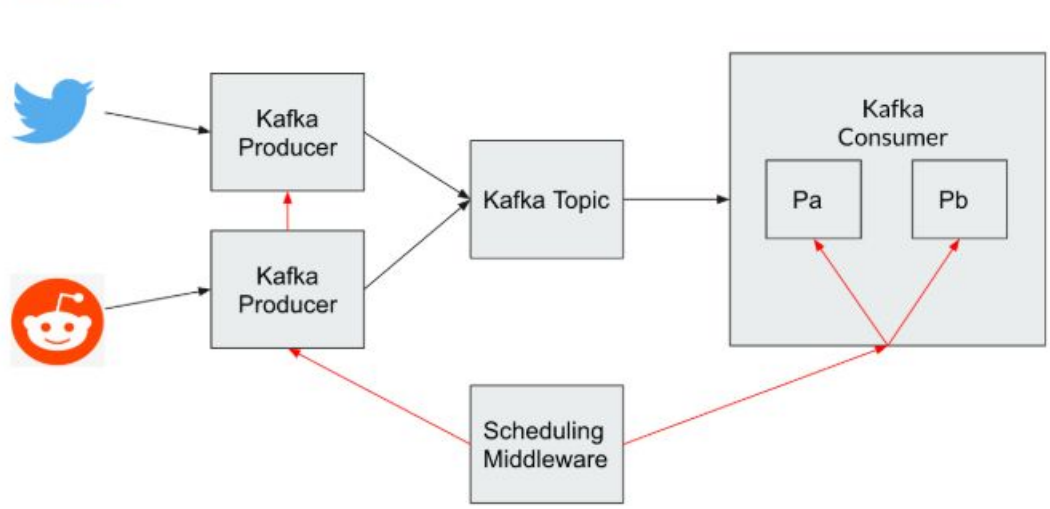

Figure 2: Architecture

Given below are the main steps followed to find the most trending results:

1. Create developer accounts on Twitter and Reddit in order to access their APIs.
2. Use the appropriate authentication techniques in order to read the tweets from twitter and posts from Reddit in real-time. We read tweets and posts with the words “dogs” and “cats” for the purpose of this project.
3. Streaming data input from Reddit and Twitter are fed into 2 respective Kafka producers using the aforementioned APIs.
4. The data is then sent, as it comes, from the two producers to the single consumer whose main goal is to find the most trending topic.
5. To handle variance in the rates of the incoming data, we use an intermediate “scheduler”.
6. The scheduler processes data using either “heavy-weight processing” or “light-weight processing”, depending on the input rate.
7. The rate of data received at the consumer is measured. The minimum rate and maximum rate is measured and from that the average rate is calculated. This calculation is done for a particular number of iterations to understand the trend.
8. Then this trend is extrapolated to the remaining iterations.
9. If the rate of data received is lesser than the average rate, it implies that we are receiving less data and hence, we can go for detailed processing technique. Hence, the heavy-weight processing is used. When the rate of data received is more than the average rate, it means that we are receiving more data and hence, we go for the simpler processing technique. Hence, the light-weight processing technique is used.
10. Given that topicPar1 and topicPar2 are the two topic parameters for which we want to find the most trending topic, the following is how we did the heavy-weight and light-weight processing.

11. For the light-weight processing: Counter1 was used to calculate the number of times topicPar1 appeared and Counter2 was used to calculate the number of times topicPar2 appeared. The higher counter value indicated the more trending topic parameter. So, if the Counter1 was more, then topicPar1 is more trending and if Counter2 was more, then topicPar2 is more trending.
12. For the heavy-weight processing: Here, more processing is done. First, the light-weight processing is done to understand which topic parameter is more trending. Then, for the most trending topic, the corresponding posts are processed to find the most frequent words belonging to that topic parameter. The results after processing are displayed.

Results

We conducted an experiment with two test keywords to find which word is more popular among users. The test keywords were “cat” and “dog”. More than 6000 posts of data pertaining to these keywords were ingested to our data pipeline governed by Kafka Producers and written to a Kafka Topic. The rate of this aperiodic ingestion was measured by a scheduler component. The scheduler component decides a processing framework based on the input data rate. In our experiment, it was found that “dog” was more popular with more than 3600 users talking about “dog” in their posts. At instants when the data rate was low, the scheduler component selects a processing framework that gave a much deeper information about the trending topic “dog”. This information is mostly about the most trending keywords associated with the trending topic. Figure 3 shows an instance when heavy processing is done because the input data rate is lesser than the average rate. It shows that “dog” is more trending, and associated with it, are the five words given below it, based on descending frequency rate.

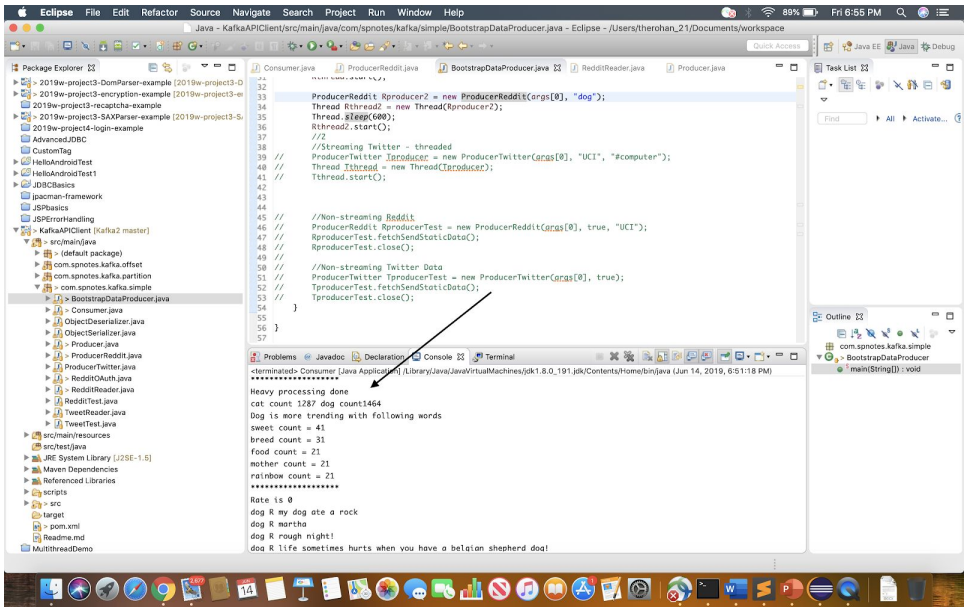

Fig 3: Heavy processing

Figure 4 shows an instance of light processing, which gives a count for both the topic parameters- “dog” and “cat”. It shows that “dog” is more trending with higher frequency than “cat”.

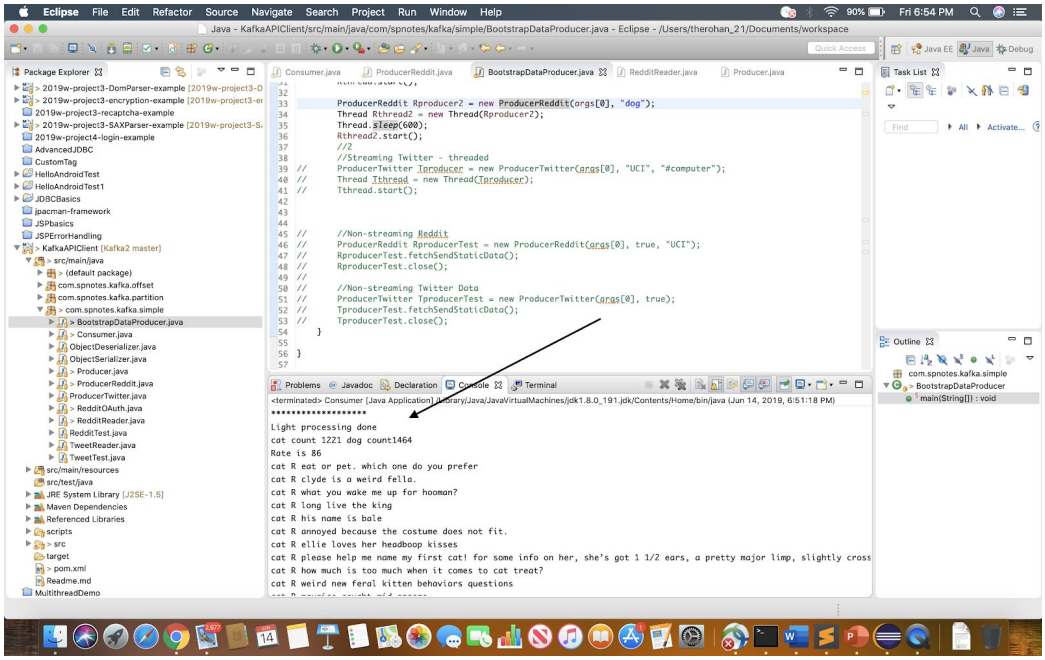

Fig 4: Light processing

Conclusion

With the bloom in social media data and their applications in all domains, it becomes extremely important to have quick processing and analysis of the same. Among the newer technologies for doing this, a popular one is Apache Kafka which is an open source, publisher/subscribe messaging system which connects multiple applications. Data is input using APIs of two popular social medias, Twitter and Reddit, and input into kafka producers which forward it to the consumer for further processing. However, input data rates change frequently and analysis needs to be scalable according to the incoming rates. To ensure this, we introduce a scheduler which defines two different processing schemes - heavy weight and light weight. Light weight Processing is used when the input rates are high and each record need not be analysed individually. Heavy weight processing is used for lower rate inputs where every individual record is considered in the analysis. Hence, we provide an efficient scheme to handle varying input rates for the analysis of social media data.

References

[1] K. Mani Chandy, Leslie Lamport, “Distributed Snapshots: Determining Global States of Distributed Systems”

[2] L Magnoni, “Modern Messaging For Distributed Systems”, Journal of Physics, Conference Series

[3] David Ediger, Karl Jiang , Jason Riedy, David A. Bader, “Massive streaming data analytics: A case study with clustering coefficients”, 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)

[4] Rajiv Ranjan, “Streaming Big Data Processing in Datacenter Clouds”, IEEE Cloud Computing (Volume: 1 , Issue: 1 , May 2014)

[5] Apache Kafka: wikipedia article: https://en.wikipedia.org/wiki/Apache_Kafka

[6] Hassan Nazeer, Waheed Iqbal, Fawaz Bokhari, Faisal Bukhari, “Real-time Text Analytics Pipeline Using Open-source Big Data Tools”

[7] Babak Yadranjiaghdam, Seyedfaraz Yasrobi, Nasseh Tabrizi, “Developing a Real-time Data Analytics Framework For Twitter Streaming Data”, 2017 IEEE 6th International Congress on Big Data

[8] Original Kafka website: https://kafka.apache.org/

[9] Guozhang Wang, Joel Koshy, “Building a replicated logging system with Apache Kafka”

[10] https://dzone.com/articles/running-apache-kafka-on-windows-os