Real-time Analytics and Visualization from Multiple Sources

Sachin Saligram (ssaligr) **Siddharth Sharma (ssharm24)** **Akanksha Singh (asingh27)**

1 Abstract

Real-time data analytics is the use of, or the capacity to use, data and related resources as soon as the data enters the system. It has many applications such as CRM (customer relations management), and real-time analytics can provide up-to-the-minute information about an enterprise's customers and present it so that better and quicker business decisions can be made, perhaps even within the time span of a customer interaction. By performing real-time analytics on data from multiple sources, we not only provide the user more variety in terms of information and insights but also increase the reliability of our data.

In this paper, we have built a data pipeline that consumes data from Twitter [1], and Reddit [2], generates insights from data through data processing, creates a persistent storage of this data and provides search and visualization all in real-time.

2 Introduction

Creating a real-time, multi-sourced pipeline that provides analytics and insights on the fly presents a couple of major infrastructure challenges. The pipeline needs to be built such that it can handle high velocity of incoming data on one hand, and on the other hand, generate new insights from this data at an equal velocity. We used Kafka [3] and Spark [4] to stream data and perform analytics in real-time. Another major challenge is combining data from multiple sources. For our pipeline, we used Reddit and Twitter APIs. However, on analyzing the data, it was found that keeping the two sources separate and creating two parallel pipelines generated much more insights as opposed to combining the two sources. This can be attributed to the fact that Reddit and Twitter data are very different. Reddit is anonymized whereas Twitter data have real identities. This implies information, such as location and user details to name a few, are not available through the Reddit API. Reddit comments can be very verbose, whereas tweets are limited to 140 characters. Reddit comments are specific to a post whereas tweets are free-flowing.

Another major design decision was to create a persistent storage of data in MongoDB [5] after streaming and processing in Kafka and Spark. Although this adds a small latency in the pipeline, it is a small tradeoff for ensuring there is no loss of information in the pipeline. Finally, Elasticsearch [6] and Kibana [7] were used to provide search and analytic capabilities on the data. Elasticsearch and Kibana seamlessly integrate with our pipeline and provide an intuitive yet powerful interface for generating insights from our data. Finally, we used Amazon Web Services [8] to create a distributed pipeline which provided a flexible, easy to use, secure and reliable cloud computing environment for our pipeline.

Our pipeline was found to be fast and scalable. It can handle 1,000,000 tweets at the rate of 56,000 tweets per hour or 5.6 tweets per second and can easily scale to handle much larger volumes of data without compromising on speed and throughput. Our search and query interface along with real-time visualization dashboard gave us sub-second query response time.

In this paper we discuss the architecture of our pipeline in depth along with the reasoning behind our design decisions. We talk about real-time analytics performed on our data sources and what information we were able to generate. Also, we talk about how we created a distributed infrastructure

and the capabilities of our infrastructure. We finally conclude with the lessons we learned along the way along with results that we obtained.

3 Data Sources

3.1 Twitter API

We used python's Tweepy [1] library for accessing the Twitter API. Data obtained was streamed to a topic in Kafka and subsequently to Spark where tweets were filtered for relevant information such as text, number of favorites, number of re-tweets, number of replies, location to name a few.

3.2 Reddit API

We used Python's Reddit wrapper PRAW [2] to stream data from Reddit. Data obtained was streamed to a topic in Kafka and subsequently to Spark where tweets were filtered for information relevant to a subreddit (eg- Black Friday) and streamed all the comments made on all topics under that subreddit. Each comment comes with a score assigned to it.

4 Architecture and Design Choices

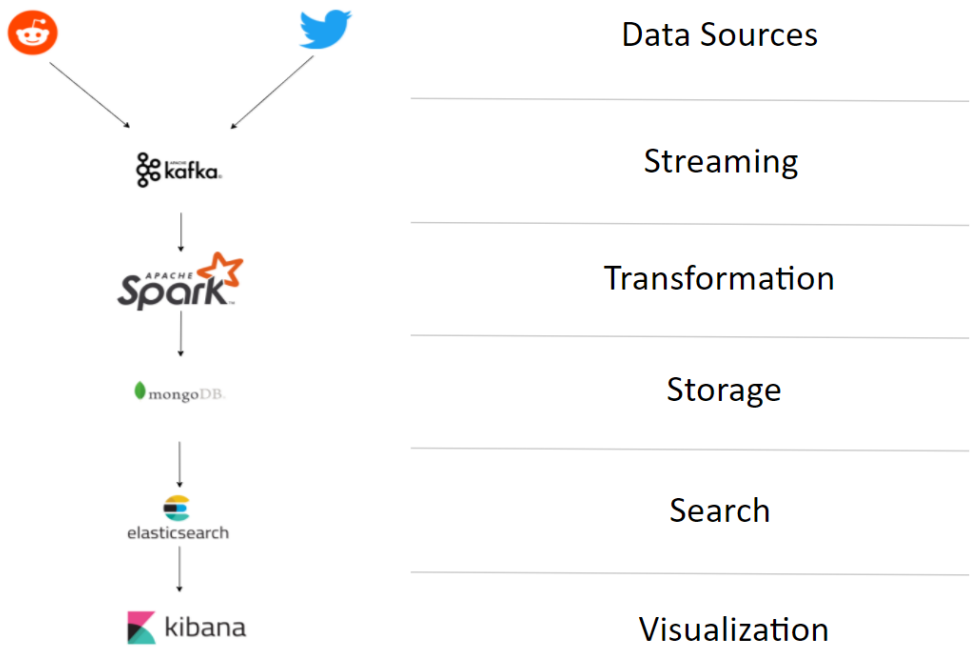

Figure 1: High-level Pipeline Architecture

Figure 1 shows the high-level architecture of our pipeline. We stream real-time data (Twitter API or Reddit API) into Kafka which is then passed to Spark for processing so that the data can be transformed to to generate new insights. Post transformation in Spark, data is persistently stored in Mongo DB. Finally, Elasticsearch is fed with data from Mongo DB to provide search and visualization through Kibana.

4.1 Streaming through Kafka

We use Kafka to buffer the incoming data from the APIs as Spark cannot transform the data at the high velocity at which streaming APIs feed data into the pipeline. We have created two topics in Kafka, one for Reddit and the other for Twitter. We then intitialize the kafka client that buffers data for every ten seconds or thousand data objects.

4.2 Processing in Spark

Apache Spark is a distributed data processing platform allowing for fast, scalable, real-time processing and data transformation and for. Spark streaming subscribes to a Kafka topic and acts as the consumer. It receives data in batches every ten seconds. We perform filtering and processing for sentiment analysis, and keyword identification in Spark. We have used the Python package Afinn [8] to perform sentiment analysis on the incoming to data. Afinn assigns a sentiment score to every data object.

4.3 Persistent Storage in Mongo DB

We use MongoDB as our database for persistent storage. Providing a persistent storage offers two advantages at the cost of adding a small latency in our pipeline -

- It ensures that there is no loss of data
- It provides a historical database that can be referenced anytime. MongoDB was the ideal choice for our project as it provides flexible and fast support for Json and is easy to integrate with Elasticsearch.

4.4 Search and Visualization through Elasticsearch and Kibana

Elasticsearch and Kibana are used for search and visualization. This combination provides us with fast indexing and search, along with strong visualization power. One of the strongest features of Elasticsearch that made it an easy design choice for this project was its ability to incrementally index on the data. As we are continuously ingesting data into our pipeline, we eventually end up with huge volumes of data in our persistent storage. However, our visualization needs to show only the latest real-time analytics (without loosing track of historical data, thus the persistent storage). Since Elasticssearch provides the ability to incrementally index on the data instead of re-indexing on the entire data store every-time we decide to generate a visualization, it made sense to use only one instance of Elasticsearch for both historical and real-time data. Moreover, Kibana in addition to integrating seamlessly with Elasticsearch, also provides the ability to specify a time window for the visualizations. So we can easily specify the time window in Kibana to generate not only real-time analytics, but also analytics on historical data. This makes our architecture much more flexible as a single pipeline provides the capability of doing analytics on real-time and historical data.

4.5 AWS and Ansible

We used Amazon Web Services for creating a distributed pipeline on the cloud. We used a VPC (Virtual Privade Cloud) on AWS along with EC2 instances. AWS is flexible, secure, reliable, salable and is easy to use and economical. We also used Ansible for infrastructure automation and to easily handle configuration.

5 Results

We created an interactive data analysis platform that can provide historical as well as real time data analytics on varied data sources. Our pipeline is distributed, scalable and fast. We were able to process tweets at the rate of 5.6 tweets per second and obtain fast response time for our search queries, and data visualizations.

Figure 2 shows a sample dashboard that refreshes every minute.The dashboard displayed is queried using the twitter_all index. On the top-left we see a line graph showing the number of tweets per

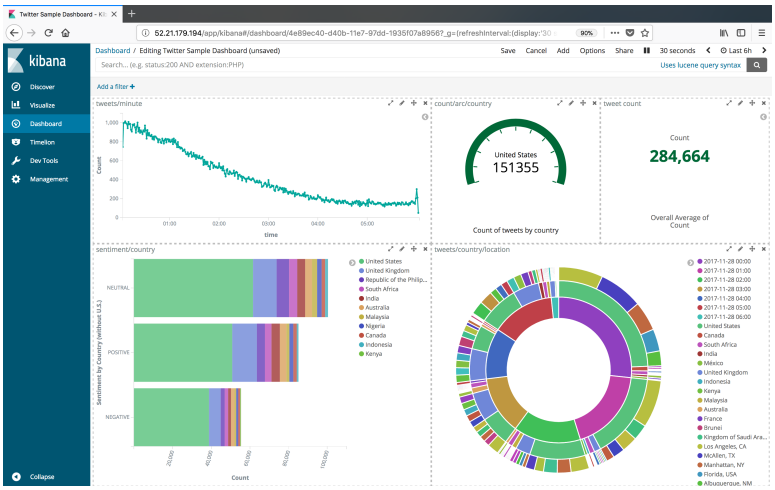

Figure 2: Dashboard

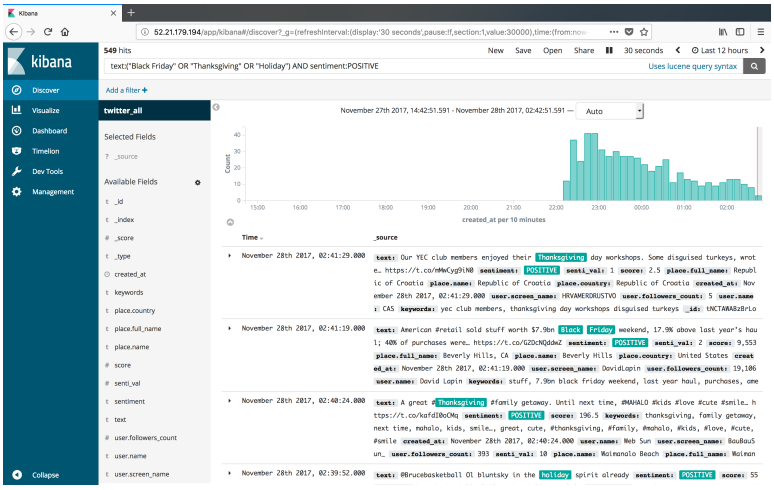

Figure 3: Query Interface

minute in the United States. The top-middle plot gives a metric about the number of tweets in the united states. The top-right plot gives us information about the number of tweets in the last hour

Kibana provides a query interface to search for any relevant information. This can be used to filter data based on a certain text or even a particular range of values of a date range. As an example, we can obtain information where thanksgiving was mentioned in a tweet and the user had a positive sentiment about it.

6 Architecture Trade-offs

- We chose to use Kafka instead of hosted services like Kinesis because Kafka provided us the flexibility of using plugins and third party integration tools which in turn made it easier to integrate with Spark.
- Feeding first into MongoDB to persistently store data gives us that extra security, and fault tolerance at the cost of adding a small latency in the pipeline.
- We explored options like Druid, and creating a separate index in Elasticsearch for processing latest real time data. However, eventually we discovered that you can keep adding

data to Elasticsearch with just the new data getting indexed. This incremental indexing capability makes it efficient for our purposes of getting a real-time view.

7 Lessons in Architecture

Although we were able to successfully create a powerful, scalable pipeline, we did not have sufficient data flowing into the pipeline to fully test the limits of our architecture. You truly need big data to complete utilize all the resources.

Elasticsearch with kibana is a very mature platform for search and visualization. Kibana provides a broad range of toolkits. Elasticsearch has lucene syntax querying which is flexible and powerful. The Elasticsearch and MongoDB combination is simple yet effective.

Big components in the pipeline can add big latency in the realm of real-time analysis. Therefore, one must make sure that every component of the pipeline is essential, critical and provides value to the final result. Also, instead of creating a generic pipeline, we can optimize for specific use cases.

8 Lessons in Data Gathering

Social media data is varied. Thus, when combining data from different sources it makes more sense to combine data from similar platforms. For example, one can combine tweets and Facebook status updates or combine Reddit and Quora comments.

Social media data may not always be a hundred percent reliable as there are people writing fake reviews or bots putting content on the platform. Also, sometimes extreme views or outrage generates the most buzz, which may not represent reality.

9 Conclusion

We claimed that we would combine multiple data sources into a unified schema and perform search and analytics on the schema in a rel-time, distributed fashion. It was soon realized that combining our data sources (Reddit and Twitter) did not give us many meaningful insights. Thus, we created a generic pipeline that can consume any type of data (Reddit, Twitter or any other social media platform with streaming APIs) and generate real-time insights on it.

Our pipeline is not only distributed, scalable and fast, but also highly flexible. It provides both real-time and historical data analytics without compromising on speed.

10 References

[1] Tweepy: An easy to use Python library for accessing the Twitter API *http://docs.tweepy.org/en/v3.5.0/*.

[2] PRAW: The Python Reddit API Wrapper *https://praw.readthedocs.io/en/latest/*.

[3] Jay Kreps, Neha Narkhede, Jun Rao (2011) Kafka: a Distributed Messaging System for Log Processing *publication descriptionVery Large Data Base Endowment Inc. (VLDB Endowment)*.

[4] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin and others (2010) Spark: Cluster Computing with Working Sets *HotCloud 10 (10-10), 95*.

[5] MongoDB Whitepapers *https://www.mongodb.com/white-papers*.

[6] Elasticsearch: Open Source Search and Analytics *https://www.elastic.co/*.

[7] Kibana: A Window into Elastic Stack *https://www.elastic.co/guide/en/kibana/6.0/index.html*.

[8] Amazon Web Services (AWS): Cloud Computing Services *https://aws.amazon.com/whitepapers/*.

[8] AFINN: Sentiment Analysis in Python *https://pypi.python.org/pypi/afinn*.