Data Pipeline for Business Review Analysis

Group 13
Nilay Kapadia, Priyance Mandlewala, Vishrut Patel
Data Intensive Computing
December 13th 2018

ABSTRACT

This paper aims to explain a data processing and analyzing pipeline for reviews given by users for a particular business and provide a dashboard to enable visual analysis as well as report generation of the analyzed data. This enables business owners to easily understand and apply business strategies more effectively. This paper utilizes the Yelp review dataset which consists of reviews of different restaurants obtained from Yelp. It also provides an explanation of the architecture used to develop the pipeline as well as the specific use of individual components in the pipeline. The paper also focuses on the advantages of using this architecture as well as issues faced during its development.

Introduction

Review data is one of the most important forms of data as it provides a feedback loop to business stakeholders. Performing analytics on this data provides crucial feedback to business owners about their product. Due to increase in broadband internet users over the past decade, review data about different products has expanded dramatically. Despite these increases in data, there doesn't exists a common infrastructure to analyze these reviews. According to a Crunchbase report[1], the failure rate of customer facing businesses is about 92% in the first few months. The third section of this paper describes the Yelp review data[2] used to develop the architecture. One of the reasons contributing to this failure rate is the lack of feedback mechanisms employed by the companies to obtain user feedback. Moreover, a simple feedback mechanism is not enough, businesses need to develop an analytics architecture to analyze this feedback from customers as well as have the ability to scale this architecture when the amount of data increases. The fourth and the fifth sections of this paper describe the architecture proposed to handle this data as well as detailed description of each component of the architecture and the reasons behind using them. It also explains how these different technologies come together to deliver a seamless interface to analyze this data.

Understanding review data

Customers are a great asset for many customer centric businesses like restaurants, software companies, online food delivery business, etc. Businesses need

reviews to get to know whether customers liked the product, service, software, etc. Businesses can take various kind of reviews like textual reviews where customer can write how they liked the product, ratings to give numerical value to the reviews which is easy to analyze and it depends on what kind of insights the company wants to know. Analysis on the review can be done in different ways. Businesses can analyze the textual review to understand the emotion of the customer like whether the product or service was good or bad. They can also analyze and compare with other competitors. The historical review data can be taken and machine learning model can be trained to predict the future of the business and which steps needs to be taken to attract more customers can be decided. We used Yelp review dataset which has more than million rows for each table like review, business, tip, users, etc from which some interesting analysis we made were the trend between the tip and the stars given by customers to restaurants which will always not be similar as it might happen that the customer liked the food but the service was not proper, the trend between the stars and the review text and the kind of business getting more 5 star reviews for finding the future correct opportunity for people trying to invest money in correct business.

Architecture

a) Initial Architecture:

Lambda architecture was initially proposed to utilize the streaming review data as well as the historical dataset. So the initial architecture involved querying the API's periodically to get reviews and store them into the Cloud SQL Database. The other historical data would be stored in AWS S3 storage and google cloud storage. A java application would fetch the data from S3 and google cloud storage and store it into Cloud SQL Database.

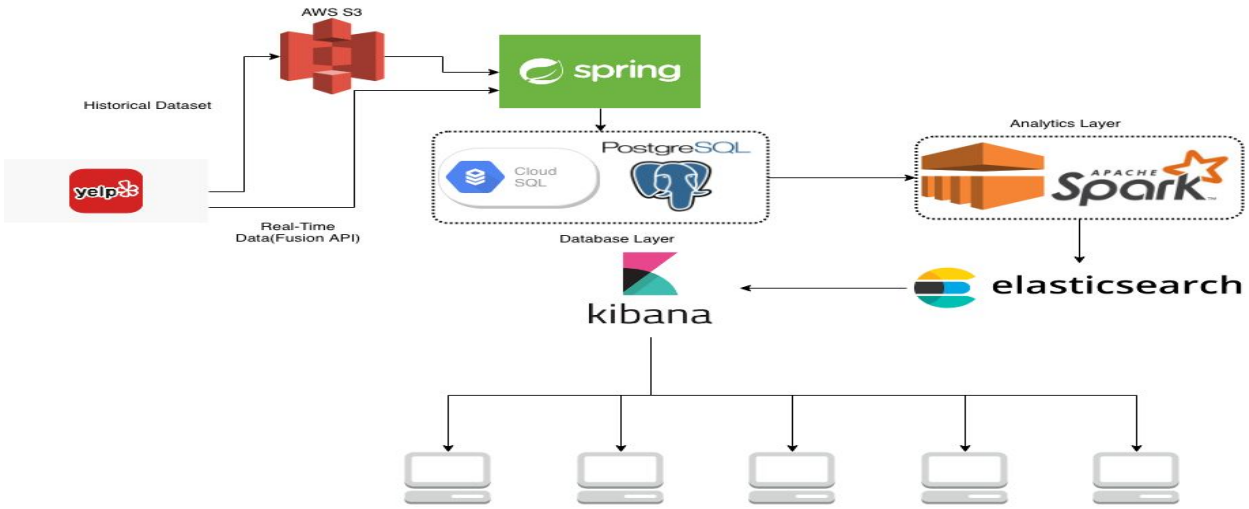

Figure 1: Initially proposed architecture

The architecture in Figure 1 was completely dependent on possibility of getting a good real time data source which can pass the real time review data at good frequency. We had examined Yelp Fusion API[3] and Webhose API[4] for getting review data but the free tier account of both had very harsh limitations. Webhose API allowed 1000 Req/month while Yelp Fusion API returned only 3 reviews per business on single request. Hence, handling the overhead for such less amount of data through which no significant information could be gained wasn't worth it. Also, it was later that we realised that in review analysis the real time data doesn't have much impact as compared to stock trade analysis, heart rate analysis etc.

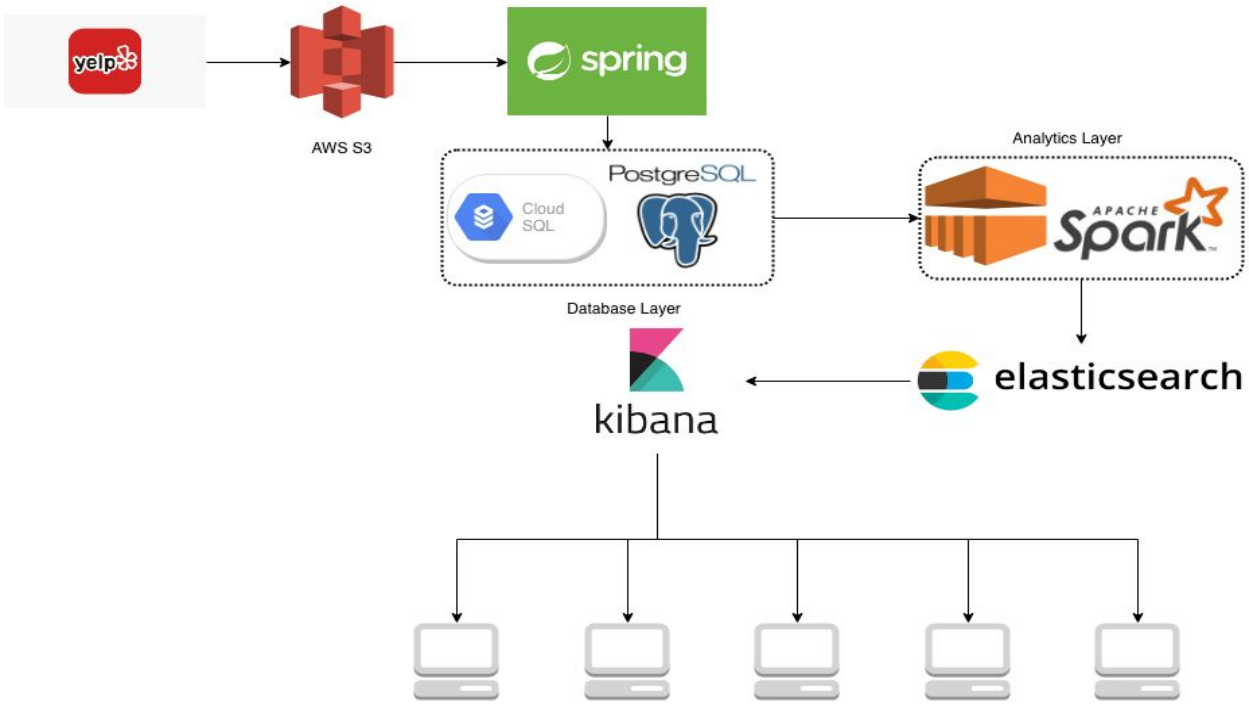

Figure 2: Final Architecture

b) Final Architecture:

The final architecture involved review analysis only on the historical/past review data. The data was available in form of dataset which was then stored into AWS S3 and Google Cloud Storage. A Spring boot application would transfer data from S3 to Google Cloud SQL. Spark was used for big data analysis. It would read data from Cloud SQL perform basic analysis on the review and control the data visibility for each user and ingest the analyzed data into elasticsearch. The graphical interface/dashboard was provided through kibana. Kibana reads data from the elastic search.

Technologies Used (S3, Cloud storage, Cloud SQL, Spark, Elasticsearch, Kibana)

a) AWS S3:

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can use it to store and protect any amount of data for a range of use cases, such as websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides easy-to-use management features so you can organize your data and configure finely-tuned access controls to meet your specific business, organizational, and compliance requirements. Amazon S3 is designed for 99.999999999% (11 9's)[5] of durability, and stores data for millions of applications for companies all around the world. The S3 had data as per following directory structure:

Bucket

Folder

1) Review.json
2) User.json
3) Photo.json
4) Checkin.json
5) Business.json
6) Tip.json

Primary reasons for storing the data into object storage was that it is possible that the platforms might be dumping data periodically into the files and not storing it into databases permanently and hence we need to accommodate that. Also, the businesses might be gathering review from the in-person visitors and this might be stored in files in either excel or csv sheets. We need to provide this support as well.

b) Cloud SQL

Cloud SQL is a fully-managed database service that makes it easy to set up, maintain, manage, and administer your relational databases on Google Cloud Platform. Our database had following tables and their structures are as below:

1) Review (review_id, user_id, business_id, stars, date, text, useful, funny, cool)
2) User (user_id, name, review_count, yelping_since, friends, useful, funny, cool, fans, elite, avg_stars)
3) Photo (photo_id, business_id, caption, label)
4) CheckIn (time, business_id)
5) Business (business_id, name, neighborhood, address, city, state, postal_code, latitude, longitude, stars, review_count, is_open)

6) Tip (text, date, likes, business_id, user_id)

The primary reason for using SQL database is that the legacy data might still be following a structured way of storage and it would be easier for the end user to interpret in that format. Our primary user base would be business who might have little to no knowledge related to computer science and hence it is important to present it in a way that is easily understandable.

c) Spark on EMR:

Spark[6] is an open source, scalable, massively parallel, in-memory execution environment for running analytics applications. It can be thought of as an in-memory layer that sits above multiple data stores, where data can be loaded into memory and analyzed in parallel across a cluster. We initially set up the cluster using 3 EC2 instances on AWS and manually configure the parameters. But when we executed our first Spark job, it ran out of memory for which the only solution was to increase the RAM of EC2 instances which was very costly on AWS. We finally decided to spin up EMR instances provided by AWS. Our cluster consists of 1 master, 2 core nodes and 2 task. This configuration helped us in running two parallel jobs on worker nodes as we used 2 core nodes. The primary reason for using Spark was that Spark can store more data in memory and can do computations on large sets which was the use case for us. We had to read data from SQL and do large joins on data top combine different tables for which Spark did a great job. Initially, we tried to dump data in Hadoop with Spark as the data store but latency was high as one of the reason we think is as we want to perform joins, everytime hadoop needs to do a read if some data is not present. But then removing Hadoop from between and directly reading data from SQL reduced the latency of executing the Spark jobs.

d) ElasticSearch:

Elasticsearch[7] is a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents. It helps in indexing texts very fast and has a very good API support. We choose Elasticsearch over other platforms available because it provides easy ways to scale it on multiple distributed nodes. Secondly, it supports searching indexed documents using REST API which can be integrated with different platforms. We used Kibana[8] for the analysis as it integrates easily with the Elasticsearch which is explained below.

e) Kibana:

Kibana is an open source analytics and visualization platform designed to work with Elasticsearch. Kibana is used to search, view, and interact with data stored in Elasticsearch indices. We used Kibana as it gives the power in the hand of the analyst

or the user to perform advanced analytics. Also, Kibana automatically reads data from Elasticsearch once port of Elasticsearch is provided in setup file of Kibana. Also, Kibana gives full independence to user to create custom dashboard based on the indexes to be used.

Conclusion and Future Work

This paper provides a comprehensive understanding of the architecture used to process, analyze and visual large scale review data. It highlights the shortcomings and advantages of using the above mentioned architecture for review data. The paper also demonstrates the versatility of cloud platforms, particularly Amazon Web Services, and how a distributed architecture for analyzing review data can be easily developed with a multitude of different softwares as well as their seamless integration when deployed. It also illustrates the reasons behind using a specific software or a specific combination of softwares and explains their effectiveness in this particular scenario. The architecture also has a large scope for improvement. One of the major point of improvement is creating an auto deployment infrastructure to introduce an element of automation in the system thereby reducing developer overhead. Another major improvement is developing a machine learning pipeline, to analyze historical reviews thereby enabling time-series analytics, in conjunction with the data pipeline. Lastly, improvements can be made in areas of user flexibility, particularly allowing users to upload any form of data that a user wishes to analyze and not specifically focusing on reviews or any form of text data while keeping the general architecture as it is.

References

[1]Crunchbase:(https://about.crunchbase.com/blog/failed-startups-and-lessons-learned/)
[2] Yelp Review Dataset:(https://www.yelp.com/dataset)
[3] Yelp Fusion API:(https://www.yelp.com/fusion)
[4] WebHose API:(https://webhose.io/)
[5] AWS S3:(https://d1.awsstatic.com/whitepapers/aws-overview.pdf)
[6] Spark:(https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final138.pdf)
[7] ElasticSearch: (https://thedudeabides.com/articles/the_future_of_compass)
[8] Kibana: (https://www.elastic.co/products/kibana)