YELP Data Analytics Using DataBricks and Power BI on Azure.

Bhargav Yagnik
Concordia University
Montreal, Canada
bhargav.yagnik@mail.concordia.ca

Dharan Thaker
Concordia University
Montreal, Canada
dharan.thaker@mail.concordia.ca

Heet Patel
Concordia University
Montreal, Canada
heet.patel@mail.concordia.ca

Shail Patel
Concordia University
Montreal, Canada
shail.patel@mail.concordia.ca

ABSTRACT

In this work, we leverage Databricks and PowerBI from the Microsoft Azure platform to analyze the YELP Dataset that contains 8M reviews of various businesses. We apply and analyze the practical implementation of various distributed systems concepts like Scalability, Concurrency, Fault Tolerance and Openness. Since the YELP dataset contains reviews, details of various businesses, we select some business questions and utilize the processing capabilities of Databricks to help us summarize the results that can be visualized then using PowerBI.

1 INTRODUCTION

With the advent of high influx data, processing it in real time as well transforming it for use in Analysis or Prediction becomes a tedious task. Each day, around 2.5 quintillion bytes of data is generated by internet users [1]. Big Data and Distributed services come into affect to help such processes. with developments from Spark, MapReduce and then to Databricks, real-time computation for large datasets has become very quick. With larger volumes of data generated each day, companies like Google, Netflix spend billions of dollars to process them real-time. Real-time processing also may help large organizations to save on storage costs and they could control on what data to store once its processed rather that storing all the unstructured data that may contain many unwanted bits. Platforms like Databricks has enabled organizations to compute queries on billions of rows of data efficiently and even use the power of other services like MLFlow, PowerBI, KubeFlow, Tensorflow on Google etc. to enable smart-decision making. This way companies can make many instantaneous decisions that would help them generate more revenue.

In this project, We are effectively making use of 4 significant characteristics of distributed systems which are Concurrency, Scalability, Openness, and Fault Tolerance. Each characteristics make the project highly unique to the platform and highlights the importance of Distributed systems. We also create a dashboard on PowerBI from the information that is transformed for better visualizations and understanding.

In section 2. we describe the dataset used in the project. We explain the architecture, the platforms involved in section 3. In Section 4, we discuss the characteristics achieved by our system. In section 5. we explain the overall methodology of the project with the detailed explanation of how the system components are connected with each other. Lastly we conclude our observation and results in section.

2 ABOUT THE DATASET

The dataset that we are using is the YELP dataset [3] and size of it is approximately 10.9 GB. Yelp Inc. develops, hosts, and markets the Yelp.com website and the Yelp mobile app, which publish crowdsourced reviews about businesses. It also operated Yelp Reservations, a table reservation service. It has got more than 178M users monthly and leveraging this information for business development can be game changing. Suppose you wish to open up a new restaurant in Montreal, you get on Yelp and see what the restaurants lack or what's trending. The dataset has been taken from Kaggle. The reason of choosing such heavy dataset is to extensively use and exploit various functionalities of databricks platform like clustering and jobs on a larger dataset in order to understand the ins and out of the platform.

3 ARCHITECTURE

3.1 Databricks

Databricks [4] is a fast, cloud based data-service that was developed by the creators of Apache Spark for efficient utilization of cloud service for processing data heavy tasks. Azure has an integrated support for Databricks that helps us to leverage this service on cloud easily. Not only for processing of Spark SQL or RDD but Databricks can even be used for Machine Learning processing and has easy support for MLLib [2] and MLFlow. In our project, we use the SAS tokens generated from the Azure Storage services to mount the dataset on Databricks cluster. This cluster is then responsible for achieving various functionalities on Databricks. They are explained in detail in the section 3.1. Databricks is then used to process millions of rows of data each containing multiple columns. The queries that would take a normal Python Pandas processor on Google cloud around

3.1.1 Clustering. Clustering in its most literal sense means, having a group of machines that are virtually or geographically separated and that work together to provide the same service or application to clients. There are in total three types of clusters in the databricks platform namely, the High-concurrency cluster, Standard cluster and Single node cluster. The high concurrency cluster is essentially a managed cloud resource and they are made to provide resource utilization and minimum query latencies. The Standard cluster is

meant for single user who wishes to parallelize the task and divide it into multiple CPUs and the single node cluster has only one node to run the program on. In our project, we are making use of Standard as well as single node cluster to implement and fulfill certain characteristics of distributed systems. The high concurrency cluster is out of the scope of this project as it required a premium service to be subscribed for. We are making use of 32 CPUs in the standard cluster to run our project. The number of CPUs to be assigned and utilized is entirely decided by the databricks control plane and is not allowed manually. It also depends on the resource server location and the time of the day. We did see that in spite of using a standard cluster we were assigned a single node as the resource server location was having a lot of traffic and limited amount of resources to be assigned. Clustering happens to be one of the most important part of this project as it is helped in fulfilling the characteristics of distributed systems mentioned above.

3.2 PowerBI

Today we are living in era of industrialization 4.0 where data is heart of it.For last few decades,analyzing data and drawing conclusions from data is extremely beneficial for business growth. PowerBI is one such tool that is very helpful for data visualizations and gathering meaning outputs from data that is rather complex to understand. PowerBI provides interactive visualizations and business intelligence capabilities with an interface so that users can create their own dashboards and charts. We used PowerBI that comes handy with Azure services to visualize the data very precisely.Mainly we created Geographical charts,WordCloud of reviews,line chart of average rating distribution and many more.

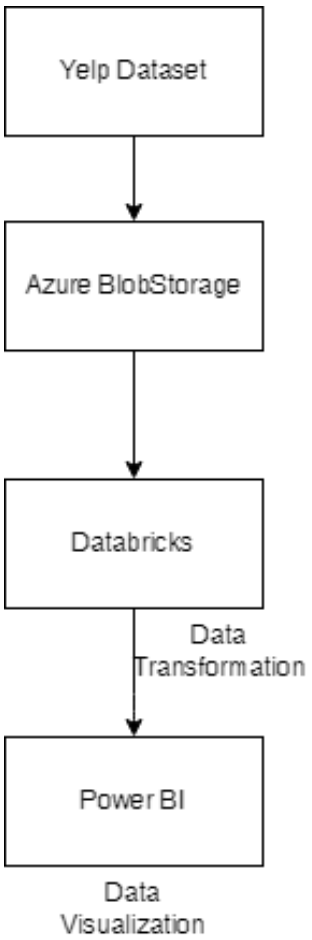

Figure 1: Overall process

4 CHARACTERISTICS OF DISTRIBUTED SYSTEMS IMPLEMENTED IN OUR SYSTEM

4.1 Concurrency

In distributed systems, Concurrency refers to the employing parallel compute to the system. With the help of clustering, Databricks enables the use of multiple cores depending on the task. This allows us the worker node to assign parallel jobs to the nodes making efficient use of the CPU cores for compute. Databricks also provides within the cluster concurrency by allowing to process multiple notebooks on share-data. This way it enables various fronts on computation and helps different departments to obtain feedback on data in the same time. On the same lines, if the data on Azure Blob storage is coming in real-time, it supports concurrency with 500 request per second to each blob. So for accessing 4kB of data, the server could allow 500 request for read or write. Azure storage also help in data concurrency by allowing Optimistic concurrency, Pessimistic concurrency and Last writer wins. In our project, we have used 4 cores as well as 8 cores for computation and questions 3 and 4 helped us observe multiple jobs that could utilize the cores efficiently. Without the presence of concurrency, faster computation of queries would be very difficult and real-time analytics would be faint.

4.2 Fault Tolerance

Fault tolerance in distributed system refers to the ability of a system to continue operating without interruption when one or more of its components fail. It basically makes use of the backup components that automatically take the place of failed components, ensuring no loss of service. Fault tolerance in our system is represented with the help of standard clustering where there is one driver node and multiple worker nodes. So, if by any chance, one of the worker nodes fail then the platform will discard that particular worker node and assign a new one on its one. Manual decision making about the assignment of new cores is out of the scope of the project, as for that we need to have premium subscription enabled. Here, if the driver node fails, all the executors which are running in a standalone cluster are killed as well, along with any data in their memory. The buffered data cannot be recovered even if the driver is restarted. To avoid this data loss, the databricks platform has got the functionality of write ahead logs. The main intent of this functionality is first written down into a log, and then the operation is applied to the data. If the system fails in the middle of applying the operation, it can recover by reading the log and reapplying the

operations it had intended to do. This is how the fault tolerance characteristics is implemented in our system.

4.3 Scalability

Scalability is the measure of a system's ability to increase or decrease in performance and cost in response to changes in application and system processing demands. While creating Azure Databricks clusters, there are two options available for the administrator to configure the clusters,

- Provide a Fixed Number of workers : In this case, Azure Databricks ensures that the cluster has the specified number of workers.
- Provide Minimum and Maximum number of workers: Databricks chooses the appropriate number of workers required to run your job.

Scalability can be achieved in the second case where the Databricks cluster configures into autoscale mode based on the load of workers. With autoscaling, Azure Databricks dynamically reallocates workers to account for the characteristics of your job. Certain parts of your pipeline may be more computationally demanding than others, and Databricks automatically adds additional workers during these phases of your job (and removes them when they're no longer needed). Autoscaling makes it easier to achieve high cluster utilization, because you don't need to provision the cluster to match a workload. This applies especially to workloads whose requirements change over time (like exploring a dataset during the course of a day), but it can also apply to a one-time shorter workload whose provisioning requirements are unknown.

4.4 Openness

The openness of the distributed system is defined as the difficulty involved to extend or improve an existing system or functionality. This characteristic allows us to reuse a distributed system for multiple functions or to process varying sets of data. While traditional systems are highly depedent on other components, the major benefit that distributed cloud systems provide is interoperability without interdependency. Since our components are loosely coupled from each other and the pipeline models gives us the flexibility to alter and update any component very easily without affecting the system as a whole. For example, if the requirements of the dataset or type of data changes, we can easily change the settings on Azure Blob storage as it is highly supportive for unstructured data so even structured data can be easily compiled . Even we can shift the storage to other services like DataLake, DataFlake or Azure File storage etc. Databricks has support to multiple languages and versions so it is flexible to support development environment in Scala, Python, R and SQL. So these languages are themselves very rich in the functionalities that they can provide. With support from the opensource community, developments in new libraries makes the existing algorithm more efficient and that changes can be effortlessly implemented in the Databricks pipeline. On the end node, if the requirements shift from a dashboard to a ML model. Shifting them is very smooth thus enabling multiple facets of the project.

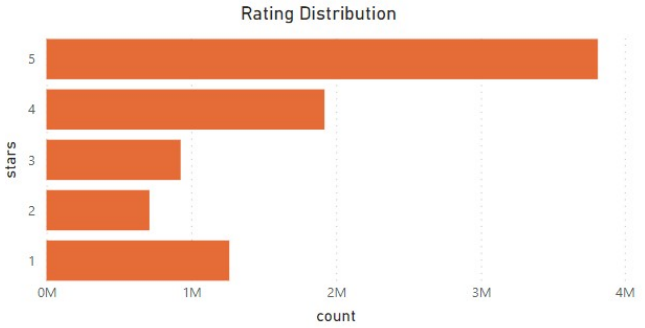

Figure 2: Rating Distribution

Figure 3: Restaurant Count over various cities

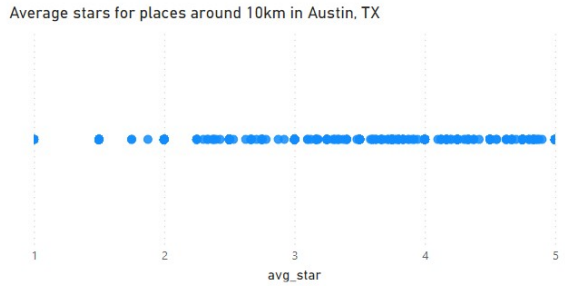

Figure 4: Distribution of business ratings in 10km around Austin

Figure 5: Wordcloud of reviews

5 METHODOLOGY

In this section, the overall approach followed to construct the whole pipeline is explained. Firstly, we create a resource group on Azure Cloud where the Azure Blob storage is mounted as well as the Databricks service is connected to the same resource group. Then we upload the YELP dataset on Blob storage where each json file is stored in 4kb blocks that are capable to scale upto 195GB. The storage is then mounted to Databricks using the SAS tokens generated from the Storage services. We also create a cluster on Databricks according to our requirements. On student version, we can create a single node cluster with 8GB memory and 4 cores. While on paid versions, we also tried 32 GB workers 8 cores which support concurrency and Scalability to greater extent. Once the data is mounted, we create workspaces where we access the dataset and create transformation and analysis on the dataset. Using multiple queries in Scala, Python and SQL, we solve the following 4 questions that were identified in the beginning.

- "What is the Rating distribution on the platform ? " - This allows us to see the general trend of user satisfaction with the platform and how overall ratings are distributed. Refer Fig 2.
- "Number of restaurants distributed over various cities " - This allows us to understand the presence of various restaurants across the country and we can also solve complex question like the presence of 4-5 star businesses in a specific city or presence of good Sushi restaurants in USA for a new Sushi brand that is opening a chain in USA. Refer Fig 3.
- "Distribution of ratings for a various business in 10km radius in Austin Texas" - Suppose we want to open a new business or a new housing scheme is looking to find a high profile area in a specific city or area, we can use the ratings and reviews of businesses around to understand how good the area is. Fig 4 explains the distribution of ratings of various businesses around 10km radius in Austin, Texas.
- "Wordcloud of reviews" Suppose a business is looking why their reviews are bad and want to improve their services, they can easily understand the scope on improvements from it. Fig 5 shows the wordcloud of all the reviews.

6 CONCLUSION

In this work, we leveraged the functionalities like Scalability, Concurrency, Openness and Fault Tolerance on Azure cloud service. We analyzed and visualized the YELP Dataset which was 11 GB in size. Using various queries, we visualized the outputs on the PowerBi dashboard. We also compared Databricks with Google Colab for comparining the scale of computation and other distributed features to appreciate Databricks and observed that while only one of the 4 queries could execute on Colab with 12GB memory, the execution time also varied by 17 times. While the same query took 0.17 seconds on Databricks, it took 2.21 second on Colab. So even with more RAM on colab, factors like scalability, concurrency are highly critical that come into place on Databricks. Thus, for large data or high influx of data, platforms like Databricks are highly impactful.

REFERENCES

[1] Marr Bernard. 2018. How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read. https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/?sh=4d4ae2b160ba

[2] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, D B Tsai, Manish Amde, Sean Owen, Doris Xin, Reynold Xin, Michael J Franklin, Reza Zadeh, Matei Zaharia, and Ameet Talwalkar. 2016. MLlib: Machine Learning in Apache Spark. *Journal of Machine Learning Research* 17 (2016), 1–7.

[3] YELP Inc. 2021. Yelp Dataset. https://www.yelp.com/dataset

[4] Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. 2016. Apache spark: A unified engine for big data processing. *Commun. ACM* 59, 11 (nov 2016), 56–65. https://doi.org/10.1145/2934664