Big Data Analytics at Uber

Meet Patel 18bce164@nirmauni.ac.in

Poojan Patel 18bce166@nirmauni.ac.in

Abstract

Data is crucial in today's business and technology environment. There is a growing demand for Big Data applications to extract and evaluate information, which will provide the necessary knowledge that will help us make important rational decisions. These ideas emerged at the beginning of the 21st century, and every technological giant is now exploiting Big Data technologies. Big Data refers to huge and broad data collections that can be organized or unstructured. Big Data analytics is the method of analyzing massive data sets to highlight trends and patterns. Uber is using real-time Big Data to perfect its processes, from calculating Uber's pricing to finding the optimal positioning of taxis to maximize profits. Real-time data analysis is very challenging for the implementation because we need to process data in real-time. If we use Big Data, it is more complex than before. Implementation of real-time data analysis by Uber to identify their popular pickups would be advantageous in various ways. It will require a high-performance platform to run their application.

1. Introduction

Uber is one of the fastest-growing business companies in the world which has risen within a short time.Uber has more than 5 million drivers, 103 million Uber users monthly on average. Uber users use about 20 million rides within a day, across 900 cities in 93 countries around the world. Uber has resolved at least partly many tackling problems that taxi users have been experiencing, such as, inadequate transportation infrastructure in some cities, unsatisfactory customer experience, taxi delays, drivers who refuse to accept credit cards by the drivers and many more. Data is Uber's largest strength. Uber's Big Data is stored in a Hadoop data lake, and the data is analyzed using Spark and Hadoop. Uber's data comes from a range of data types and databases such as SOA database tables, schema lesser data, stores and Apache Kafka, the event message system. Using a massive driver database, their algorithms pair with a user for the most appropriate driver within a 15 seconds time frame to the closest driver as soon as a user asks for a vehicle. Uber's data science department also performs an in depth analysis of public transit networks in various cities so that they can concentrate on cities with weak transit systems and make effective use of the data to improve customer service experiences.

1.1 *Problem Formation*

What forms the core of businesses today? Huge volumes of data that flow in and out every day — and though it does matter, what really comes into play is the ability to use data and models to make better business decisions.

Uber is greedy about what data it collects and with many cheap relative storage options like Hadoop and Spark-it has got data about every single GPS point for every trip taken on Uber. Uber stores historic information about its system and capabilities to ease doing data science for its data scientists down the road. Keeping the change logs, versioning of database schemas helps data scientists answer every question on-hand. With the data Uber has, data scientists can answer questions like what did the Uber system look like at a particular point of time from a customer perspective, supply behaviour perspective, from inter-server communication perspective or even to the state of a database.

Some of the problems for which Uber uses Big Data Analytics are:
Surge Pricing, Matching Algorithms, Fare Estimates, Cutback in Expected Time Arrival (ETA), Route Optimization, Monitoring and bridging the gap between Supply and Demand etc. [1]

1.2 *Problem Definition*

- *Surge Pricing*

To create the most efficient market and maximize the number of rides it can provide –Uber uses surge pricing. You are running late and stressed enough to take the public transport, Uber could come to your rescue, and however you soon notice that they will charge you 1.5 times more than the usual rate.

Sometimes when you try to book an Uber, and what you thought would be a $10 ride is going to be 2 or 3 or even 4 times more – this is due to the surge pricing algorithms that Uber implements behind the scenes. Data Science is at the heart of Uber's surge pricing algorithm. Given a certain demand, what is the right price for a car based on the economic conditions. The king of ride sharing services maintains the surge pricing algorithm to ensure that their passengers always get a ride when they need one even if it comes at the cost of an inflated price. Uber has even applied for a patent on big data informed pricing i.e. surge pricing.

Most of the predictive models at Uber follow the business logic on how pricing decisions are made. For instance, the Geosurge (name for surge pricing or dynamic pricing model at Uber) looks at the data available and then compares theoretical ideals with what is actually implemented in the real world. Uber's surge pricing model is based on both geo-location and demand (for a ride) to position drivers efficiently. Data science methodologies are extensively

used to analyse the short term effects of surge pricing on customer demand and long term effects of surge pricing on retaining customers. Uber depends on regression analysis to find out which neighbourhoods will be the busiest so it can activate surge pricing to get more drivers on the roads.

Uber recently announced that it's going to limit the use of surge pricing through machine learning. The machine learning algorithms will take multiple data inputs and predict where the highest demand is going to be so that Uber drivers can be redirected there. This will ensure that there is no supply and demand shortage so that it does not have to actually implement surge pricing. Uber has not yet confirmed as to when this new system with smart machine learning algorithms would be rolled out to reduce surge pricing.

- *Matching Algorithms*

Timing is everything at Uber. Given a pickup location, drop off location and time of the day, predictive models developed at Uber predict how long it will take for a driver to cover the distance. Uber has sophisticated routing and matching algorithms that direct cars to people and people to places. Right from the time you open the uber app till you reach your destination, Uber's routing engine and matching algorithms are hard at work.

Uber follows a supplier pick map matching algorithm where the customer selects the variables associated with a service (in this case Uber app) and makes a match by sending requests to the most optimal list of service providers. Any Uber ride request is first sent to the nearest available Uber driver (the nearest available Uber driver is determined by comparing the customer location with the expected time of arrival of the driver). The Uber driver then accepts or rejects a ride request. This matching algorithm works well for Uber since the transaction is highly commoditized i.e. the number of variables that the customer has to decide before a match is made are minimal.

- *Fare Estimates*

Uber uses a mixture of internal and external data to estimate fares. Uber calculates fares automatically using street traffic data, GPS data and its own algorithms that make alterations based on the time of the journey. It also analyses external data like public transport routes to plan various services.

- *Cutback in Expected Time Arrival (ETA)*

"Expected time arrival" is mainly considered when you book a ride, it's extremely frustrating when much time is wasted in traffic jams, especially in urbanized areas. This situation gets worse when a cab takes longer than expected to reach a pickup location.

Specifying the machine learning specialties, it also addresses this issue which Uber utilizes. By forecasting demand and putting cabs ready, Uber can decrease the expected time arrival (ETA) when customers book rides.

Uber always makes the experience superior by reducing waiting time a lot. The flawless fusing of customer satisfaction, adhere schemes, and fruitful services leads to colossal augmentation in Uber.

- *Route Optimization*

Conventional ride-calling systems expect the drivers to choose routes based on presumption and availability which is not stable due to travel span might get changed due to huge traffic on the same route, weather conditions, road construction set up, etc.

But, twist here, the ML system of Uber upgrades the app with the possibilities in each route and recommends the agile route to drivers.

Through this process, Uber assists drivers to avert crowding areas and enables speedy rides. It not only makes customers pleased but also offers drivers extra time to conduct additional rides.

- *Monitoring and bridging the gap between Supply and Demand*

Looking at the archival data, the Uber team can estimate the time and location of demand. The system adopts these estimations to aware drivers of the particular region with the leads of demand. Through this, Uber ensures that there must be enough cabs in the demanded area and fill the gap amid route and supply.

Demand forecasting systems allow the app to hike the prices marginally midst peak hours that augment profit and demand ultimately.

For plausibility, customer retention is crucial for several services, sometimes customers don't want to delay their daily rides when cabs are not available instantly, they just book a cab from another service.

Even Though, getting new customers demands more effort than retaining existing customers. The supply-demand gap can affect customer retention to some extent that can be prevented via machine learning based forecasts, it also aids Uber from losing customers to its rivals.

1.3 Significance of research

Data science is an integral part of Uber's products and philosophy. Big data analysis spans across diverse functions at Uber – machine learning, data science, marketing, fraud detection and more.

Uber data consists of information about trips, billing, health of the infrastructure and other services behind its app. City operations teams use uber big data to calculate driver incentive payments and predict many other real time events. The complete process of data streaming is done through a Hadoop Hive based analytics platform which gives right people and services with required data at the right time.

On the product front, Uber's data team is behind all the predictive models powering the ride sharing cab service right from predicting that "Your driver will be here in 3 minutes." to estimating fares, showing surge prices and heat maps to the drivers on where to position themselves within the city.The business success of Uber depends on its ability to create a positive user experience through statistical data analysis. What makes Uber unique is that the data science driven insights don't just stay within the dashboards or company reports but they are implemented in real-time into it to create a positive user experience for customers and drivers.

Whether it's calculating Uber's "surge pricing, "helping drivers to avoid accidents, or finding the optimal positioning of cars to maximize profits, data is central to what Uber does.

1.4 3 V's of Big Data

- Volume

Today, 93 million customers use the Uber platform. 3.5 million drivers serve the growing user base. The service is available in 10,000 cities across 71 countries. Uber processed $26.61 billion worth of bookings in 2020. Uber has a 68% share of the US rideshare market. 5 cities alone contribute 22% of Uber's gross ridesharing bookings. For both of the problems mentioned above, the amount of data to be processed is very high.

- Velocity

Uber processes around 1 million rides per day.
1.44 billion rides are completed through Uber every quarter. The data of demand and supply, locations of the drivers change every second with this amount of data being processed everyday.

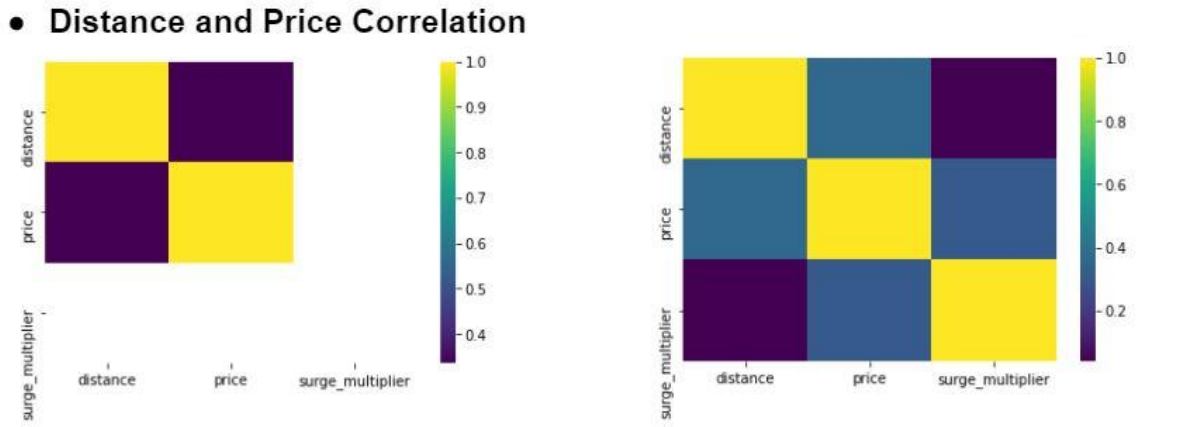
- Variety

Supply and demand data are not the same from city to city, so Uber engineers devised a way to map the "pulse" of a city to connect drivers and riders more efficiently. Also, Uber's data comes from a range of data types and databases such as SOA database tables, schema lesser data, stores and Apache Kafka, the event message system.[1]

2. Related Research work

2.1 Comparison - Uber v/s Lyft

1. Distance and Price Correlation



Uber Distance Vs. Price Correlation **Lyft Distance Vs. Price Correlation**

***In the graph above, the color yellow signifies a strong positive correlation, and the color purple indicates a low correlation**

The first parameter we explore is the distance and price correlation. We represent each unit with a colour in order to simplify the data. From the above visual, we can interpret the following points-

For Lyft: surge multiplier and distance are weakly correlated.

For Lyft: surge multiplier and price are more correlated.

For Uber: price and distance are weakly correlated.

Uber is riders' first choice irrespective of distance.[6]

2. Cloud Computing

Lyft uses Amazon Web Services and Uber has a classic hybrid cloud approach. Thus both companies are heavily invested in the cloud.

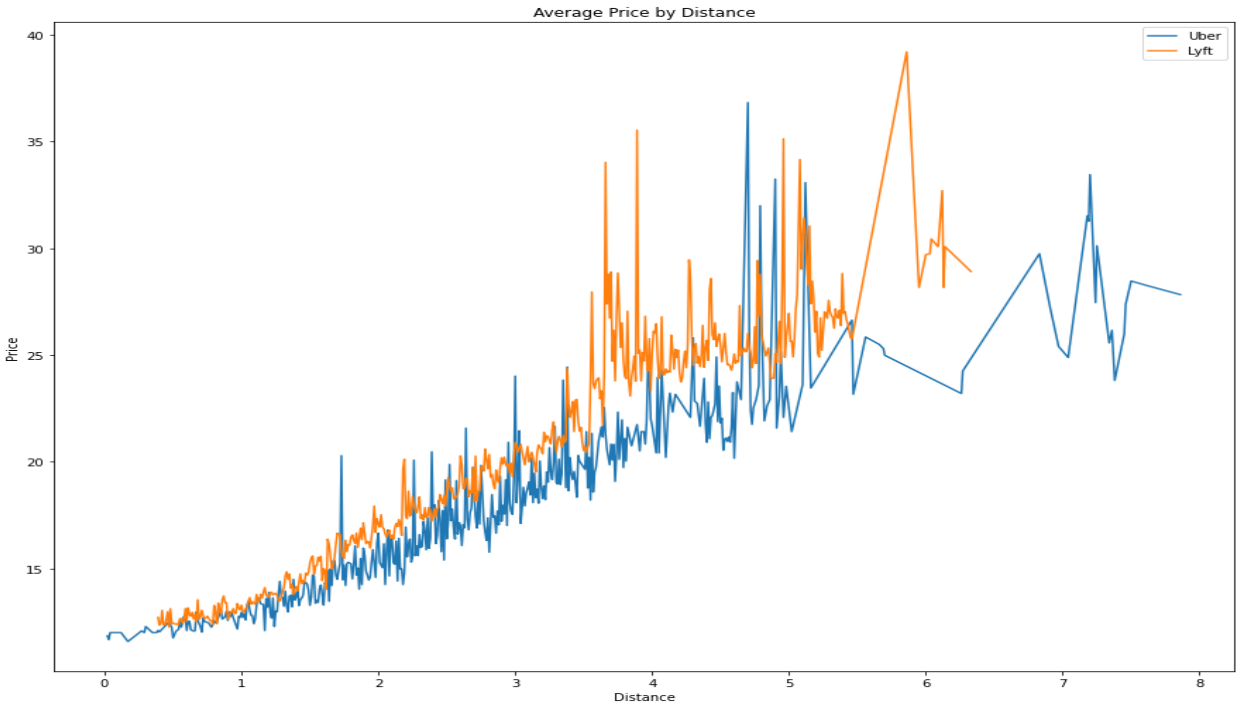
Lyft has agreed to spend a minimum of $300 million in the next few years. Amazon's architecture allows Lyft to withstand business surges. Uber's multi-cloud vendors include a highly automated infrastructure and co-located facilities, thus enabling efficient scaling without any upfront infrastructure expense.

3. AI and Machine Learning

Data science and algorithms are significant to Uber's marketplace technologies. Their business plan includes automation through artificial intelligence and machine learning. They have developed algorithms based on customer pain points ranging from menu (Optical Character

Recognition (OCR) to automatic driver license approval, crash detection to improved GPS, etc. By using real-time algorithms Uber has created its own proprietary payment, routing, and marketing technologies for Uber, Uber Eats, and Uber Freight. Thus the company has made technology its key driver to predict supply and demand, evaluate variables like distance, time, weather, traffic. This also stands true when it comes to its growth and future plans.

4. Distance vs. Price



Lyft uses data science tools for machine learning to use historical data efficiently. Data insights allow them to offer competitive prices during demand, predict driver availability, smooth ride experience for its customers, optimize routes, etc. Data science and algorithms will aid to inform service levels, facilitate product development, increase efficiency, and increase driver earnings.

Technology has really enabled both companies to evolve and grow in their preferred directions. It has been a key enabler. In order to keep tech and business coordinated Uber employs agile software and DevOps. Uber persistently experiments, tests, creates prototypes, and refines its products for better service for its platform users. It chooses small neighborhoods or cities to test its offers, gather feedback, improve its performance, and make necessary corrections before scaling the products to reach additional users.[4]

Lyft flawlessly integrates its product and development teams with engineering, analytics, data science, and design along with using third-party open-source software. It also follows a continuous deployment strategy.[6]

2.2 The Future

Autonomous vehicles are the future that both companies are looking towards however with different approaches.

With more than 1000 employees dedicated to next-gen transportation, Uber has it's Advanced Technologies Group focused on autonomous vehicles. It has partnered with manufacturers like Toyota and Volvo. Uber Elevate is another branch where Uber is exploring autonomous technologies via aerial ride-sharing.

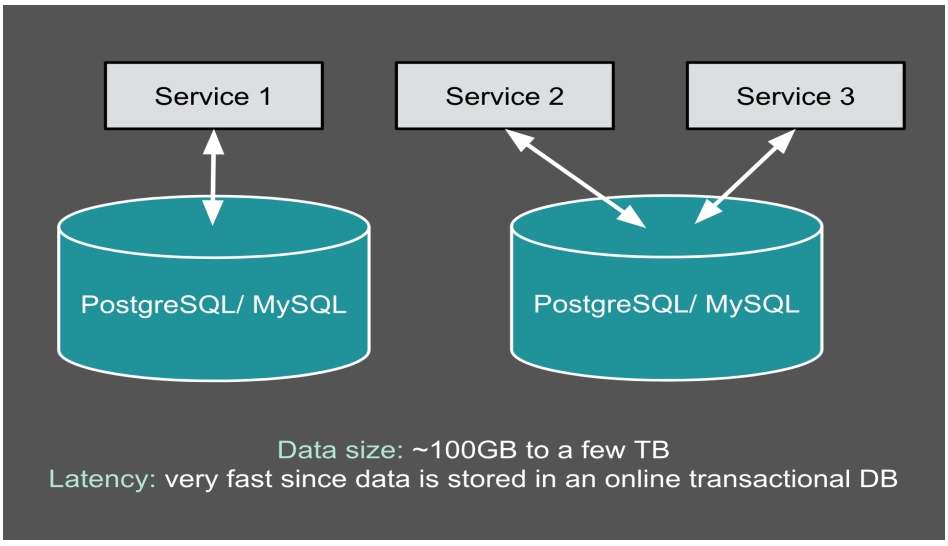
Lyft connects its network with partners by using an open autonomous vehicle platform. Lyft is test-driving its autonomous vehicles and the company soon will be able to compare its simulated testing to real driving. Nonetheless, the idea of autonomous vehicles completely replacing human drivers still remains far-fetched.

3. Proposed Research Work

The main problem was to handle big data, by means of Volume and Velocity. As Uber's grow had a exponential curve, We must think something bigger to handle these things

3.1 Generation 1

Uber has evolved their system time by time, to remain inline with the current need. When Uber Started, in 2013, they were using a very basic OLTP system for analytics and storing the data. Below is the System Diagram for the same [3]

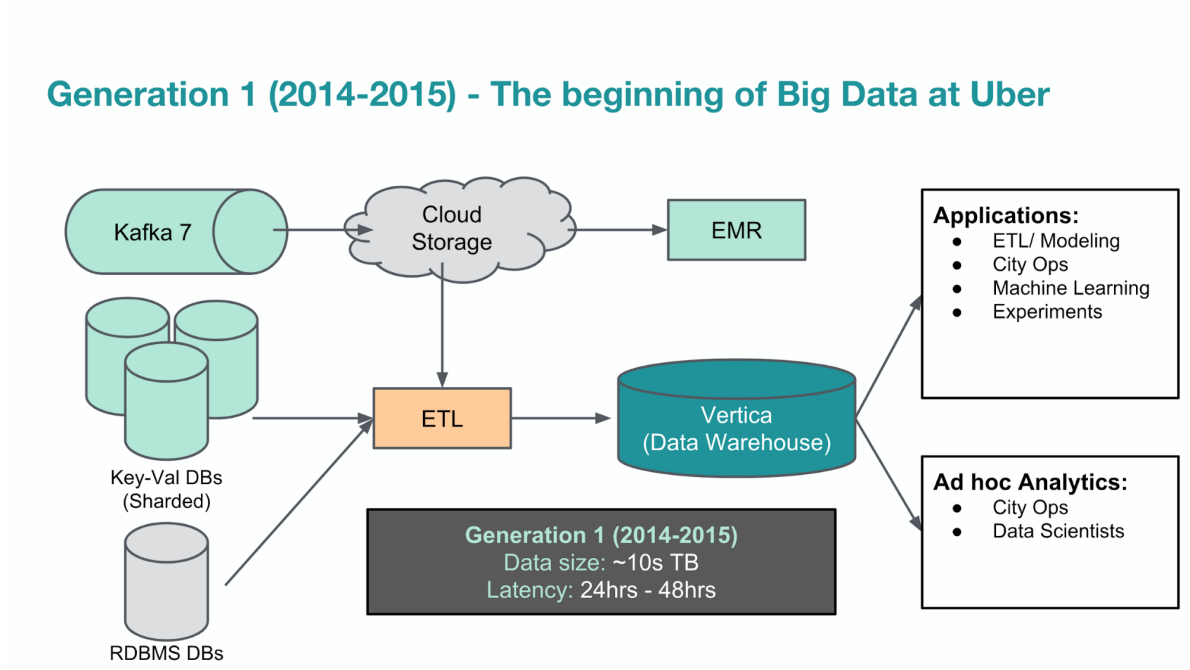


At that time, the data can fit easily into traditional systems like MySql, PostgreSQL. So this type of transitional databases were scattered over the regions to temporarily store the data. Kafka based broker systems were installed, to ETL the data into data warehouse, Vertica, globally.

All the analytics related mechanisms were set at Vertica.

This mechanism was sufficient at that time, where less amount of data was getting generated, but was not built to adapt.

The below one is the complete system diagram for Generation-I at Uber Big Data Analytics



Traditional DBs share data with ETL, which will reformat it and store it to Vertiva Warehouse. From Vertica, all the analytical Queries will be handled.

The data size was growing at its peak and the system started failing to handle such big data. Later on Hadoop was introduced

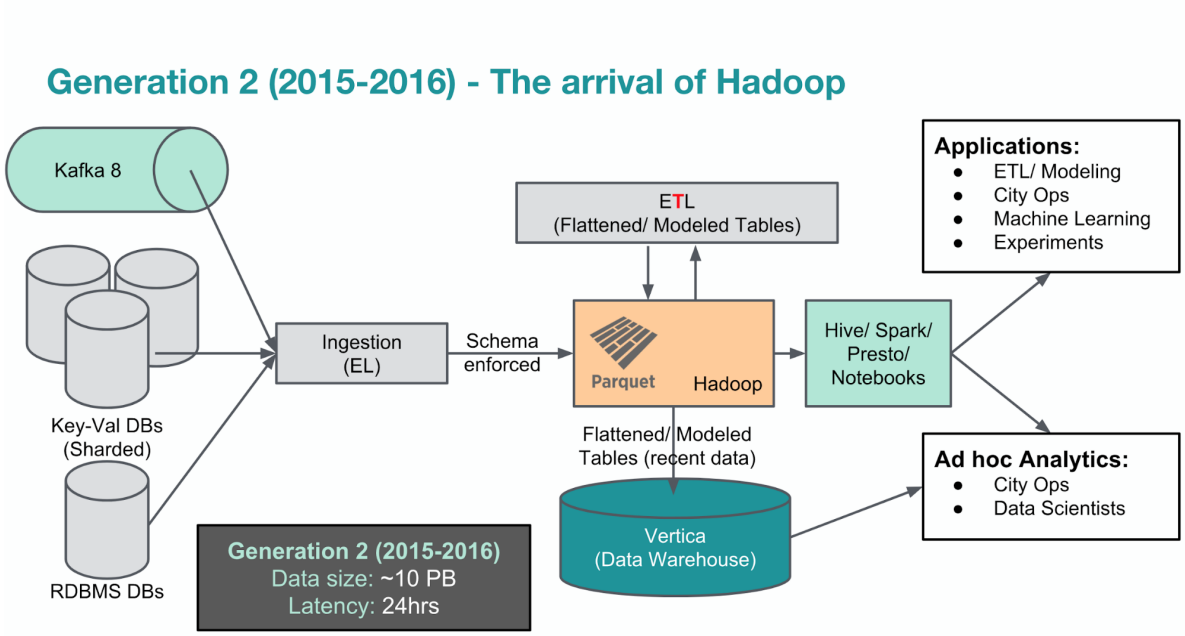
Hadoop is known for the scalability and OLAP dimension. They introduced a Hadoop data lake where all raw data was ingested from different online data stores, like MySQL and PostgreSQL, only once and with no transformation during ingestion. This design shift significantly lowered the pressure on our online datastores and allowed us to transition from ad hoc ingestion jobs to a scalable ingestion platform. In order for users to access data in Hadoop, They also introduced presto to enable interactive ad hoc user queries, as Hadoop's map-reduce does not provide the same, and Apache Spark to facilitate programmatic access to raw data (in both SQL and noSQL

formats), and Apache Hive to serve as the workhorse for extremely large queries. These different query engines allowed users to use the tools that best addressed their needs, making our platform more flexible and accessible.

They also used the standard columnar file format of Apache Parquet, resulting in storage savings given the improved compression ratio and compute resource gains given the columnar access for serving analytical queries. Moreover, Parquet's seamless integration with Apache Spark made this solution a popular choice for accessing Hadoop data.

3.2 Generation-2

The below one is Generation:2, Uber Big Data Analytics[3]



Here Hadoop along became mediatory for direct EL between Vertica and data stores.

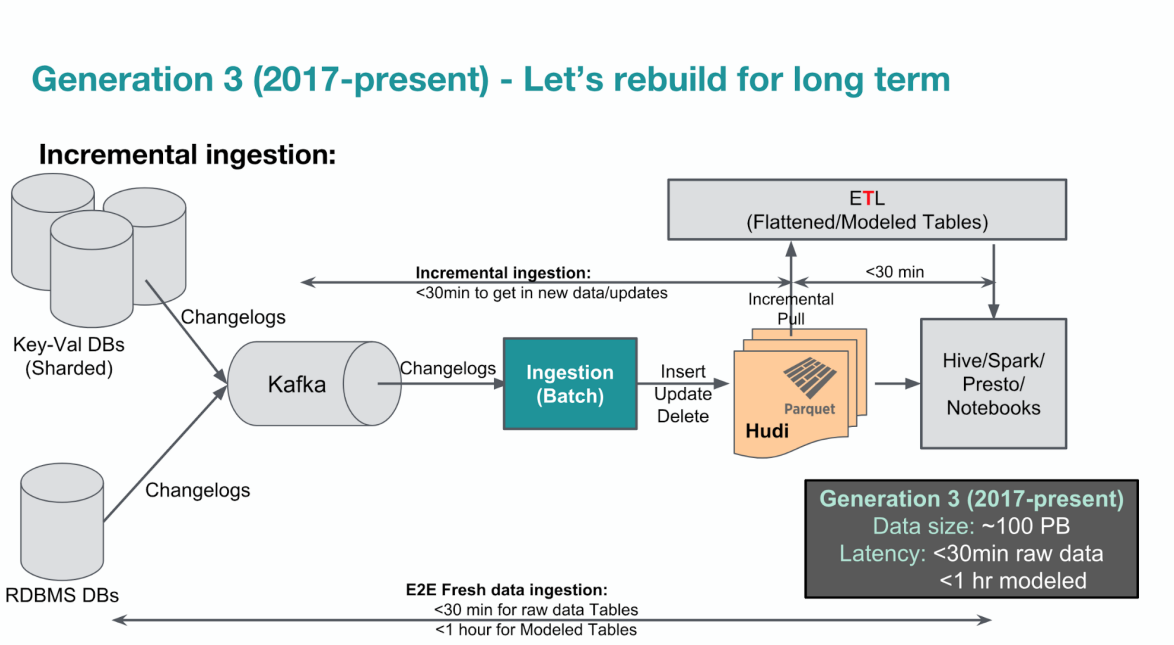
This was not the end, This architecture was also facing some bottlenecks, which are mentioned below

1. HDFS scalability limitations: We know that an ecosystem can only have one namenode which stores all the metadata related to the files stored in the datanodes, so a tremendous amount of continuous query can bottleneck the performance of Architecture.
2. Faster Data Access: latency to analyze the data which was newly generated was 24-hours which can be hard for some of the business logics.
3. Data Updation and Deletion: Uber's data contains a lot of updates, ranging in age from the past few days (e.g., a rider or driver-partner adjusting a recent trip fare) to a few

weeks (e.g., a rider rating their last trip the next time they take a new trip) or even a few months (e.g., backfilling or adjusting past data due to a business need). With snapshot-based ingestion of data, we ingest a fresh copy of the source data every 24 hours. In other words, we ingest all updates at one time, once per day. However, with the need for fresher data and incremental ingestion, our solution must be able to support update and delete operations for existing data. However, since our Big Data is stored in HDFS and Parquet, it is not possible to directly support update operations on the existing data. On the other hand, our data contains extremely wide tables (around 1,000 columns per table) with five or more levels of nesting while user queries usually only touch a few of these columns, preventing us from using non-columnar formats in a cost-efficient way.

3.3 Generation-3

To solve the above problems, Generation 3 introduced, which has Architectural diagram like below [9]



For the Bottleneck of NameNode, They introduced ViewFS and Hadoop namenode federation.
The View File System (ViewFs) provides a way to manage multiple Hadoop file system namespaces (or namespace volumes). It is particularly useful for clusters having multiple namenodes, and hence multiple namespaces, in HDFS federation. ViewFs is analogous to *client side mount tables* in some Unix/Linux systems.
So they created multiple namenodes and applied ViewFs to handle this problem.

They introduce "Hudi", Hadoop Upsert anD Incremental, to overcome the problem of Updation.

Hudi can manage, with high efficiency, to upsert the data already stored in HDFS. Furthermore, to access particular data and all the updates since the last checkpoint, Hudi has provided a functionality to find all the updates since given checkpoint time, otherwise searching the whole table for getting all the updates since checkpoint can be too expensive.[10]

4. Methodology and Concepts applied

4.1 Technology & Concepts

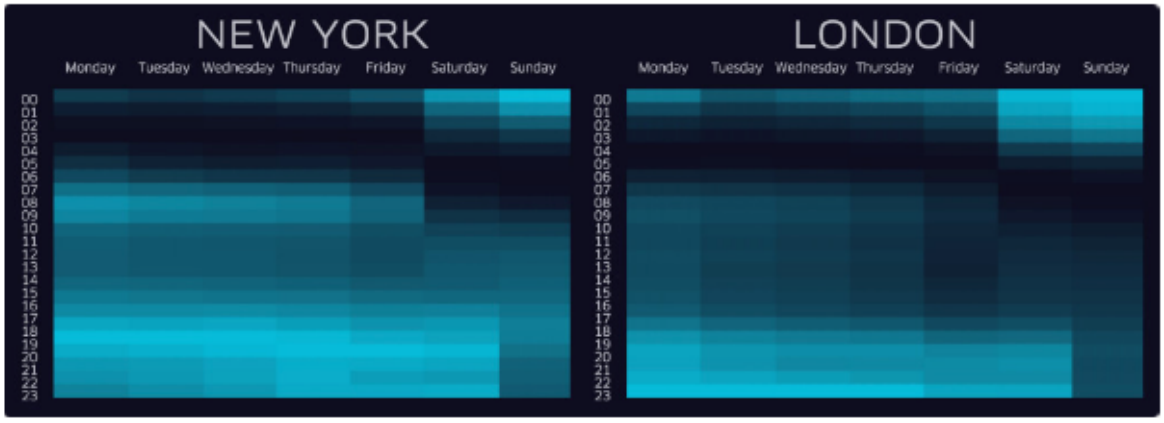
We will be taking the Surge Pricing algorithm to see how it works. Surge pricing is necessary for the welfare of Drive Sharing companies and its drivers. We all faced somewhere that during the days of festivals or in mid-night, prices are somewhat extra than how they are in general situations. It is a surge price. Surge Price basically works on the principles of Demand and Supply

4.2. Algorithm for surge pricing

1. Find the number of user requests from current area, say n
2. Find the number of drivers available from current area, say m
3. If m = n
   a. No surge price is needed as the demand and supply is at its equilibrium
   b. Determine the price of the ride based on the distance between a user and driver, current date, weekday, time, area
   c. Show it to user
4. Else if m < n
   a. Using the analytics, we can find the multiplier, surge multiplier
   b. Since the fare has been increased, what it should be, will motivate some of the inactive drivers to become active and some of the users to discard their request
   c. In ideal scenario it will become something like m+x = n-y where x,y > 0
5. Else
   a. We can find a negative surge multiplier, > 0 and < 1
   b. This will demotivate many drivers to become inactive. Users will rush to take the cheap rides.
   c. In ideal scenario it will become something like m-x = n+y whew x,y>0

Example:

Assume that in general situations where demand and supply is at its equilibrium, Ride price from the railway station to Airport is 40 dollars, for the distance 22 km.
The current surge multiplier is 1.0 [2]



Suppose, only 10 drivers are available in the opposite of 100 requests, ratio 0.1, which remains around 0.6-0.85 in general situations. By looking at the factors like weekday, time, festival we can query the hadoop system regarding the optimum value of multiplier. Suppose the query is Saturday, 7 PM in New York, by checking the visualization, the algorithm will describe a good surge multiplier.

Let's say the predicted surge multiplier is 1.84, so prices were increased from 40 to 70. This will motivate more 15 drivers to take the requests and since the price has surged, 100 requests are reduced to 50. Other 50 took the subway to reach the airport.

Prices will start to be subsidized after taking requests, as the ratio has increased from 0.1 to 0.5

5. Experimental Analysis

The classical k-means algorithm works as an iterative process in which at each iteration it computes the distance between the data points and the centroids that are randomly initialized at the beginning of the algorithm.

We decided to design such an algorithm as a MapReduce workflow. A single stage of MapReduce roughly corresponds to a single iteration of the classical algorithm. As in the

classical algorithm at the first stage the centroids are randomly sampled from the set of data points. The map function takes as input a data point and the list of centroids, computes the distance between the point and each centroid and emits the point and the closest centroid. The reduce function collects all the points belonging to a cluster and computes the new centroid and emits it. At the end of each stage, it finds a new approximation of the centroids that are used for the next iteration. The workflow continues until the distance from each centroid of a previous stage and the corresponding centroids of the current stage drops below a given threshold.

```
centroids = k random sampled points from the dataset.

do:
    Map:
        - Given a point and the set of centroids.
        - Calculate the distance between the point and each centroid.
        - Emit the point and the closest centroid.
    Reduce:
        - Given the centroid and the points belonging to its cluster.
        - Calculate the new centroid as the arithmetic mean position of the
points.
        - Emit the new centroid.
    prev_centroids = centroids.
    centroids = new_centroids.

while prev_centroids - centroids > threshold.
```

Mapper

The mapper calculates the distance between the data point and each centroid. Then emits the index of the closest centroid and the data point.[2]

```
class MAPPER
    method MAP(file_offset, point)
        min_distance = POSITIVE_INFINITY
        closest_centroid = -1
        for all centroid in list_of_centroids
            distance = distance(centroid, point)
            if (distance < min_distance)
                closest_centroid = index_of(centroid)
                min_distance = distance
        EMIT(closest_centroid, point)
```

Combiner

At each stage we need to sum the data points belonging to a cluster to calculate the centroid (arithmetic mean of points). Since the sum is an associative and commutative function, our

algorithm can benefit from the use of a combiner to reduce the amount of data to be transmitted to the reducers.

```
class COMBINER
    method COMBINER(centroid_index, list_of_points)
        point_sum.number_of_points = 0
        point_sum = 0
        for all point in list_of_points:
            point_sum += point
            point_sum.number_of_points += 1
        EMIT(centroid_index, point_sum)
```

Reducer

The reducer calculates the new approximation of the centroid and emits it. The result of the MapReduce stage will be the same even if the combiner is not called by the Hadoop framework.

```
class REDUCER
    method REDUCER(centroid_index, list_of_point_sums)
        number_of_points = partial_sum.number_of_points
        point_sum = 0
        for all partial_sum in list_of_partial_sums:
            point_sum += partial_sum
            point_sum.number_of_points += partial_sum.number_of_points
        centroid_value = point_sum / point_sum.number_of_points
        EMIT(centroid_index, centroid_value)
```

Dataset we have used contains datashape of 45,34,328 x 2, where columns are longitude and latitude for particular trip pickup location

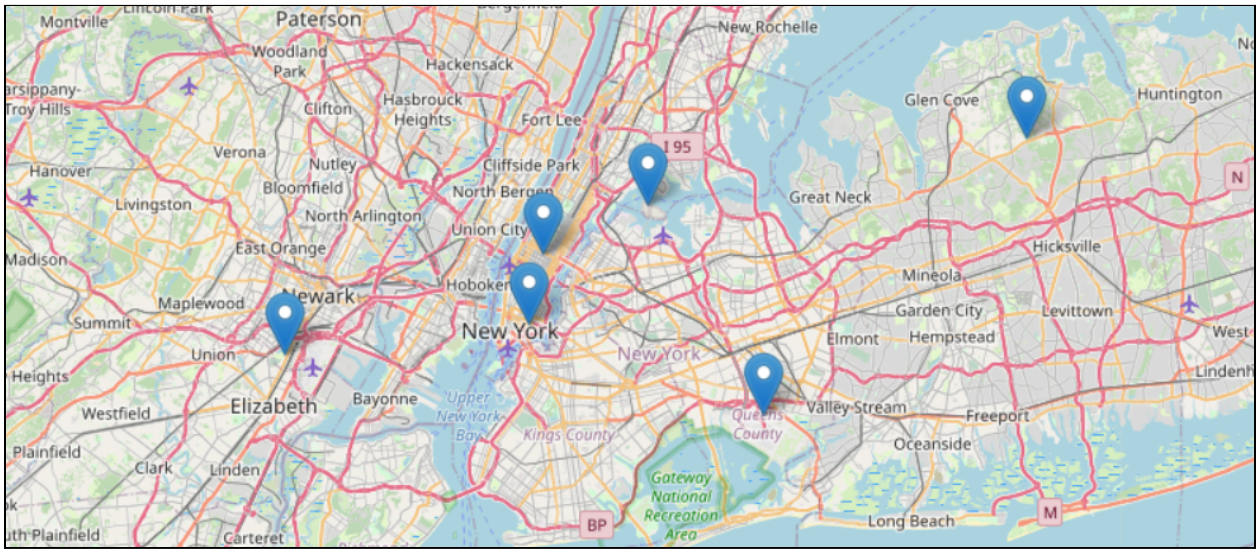
We have used PySpark for the implementation of K-Means Clustering Algorithm. For the visualization purpose, we have used Python's Folium Library to represent the clusters on the actual map of New York

Refer to this link for Code.

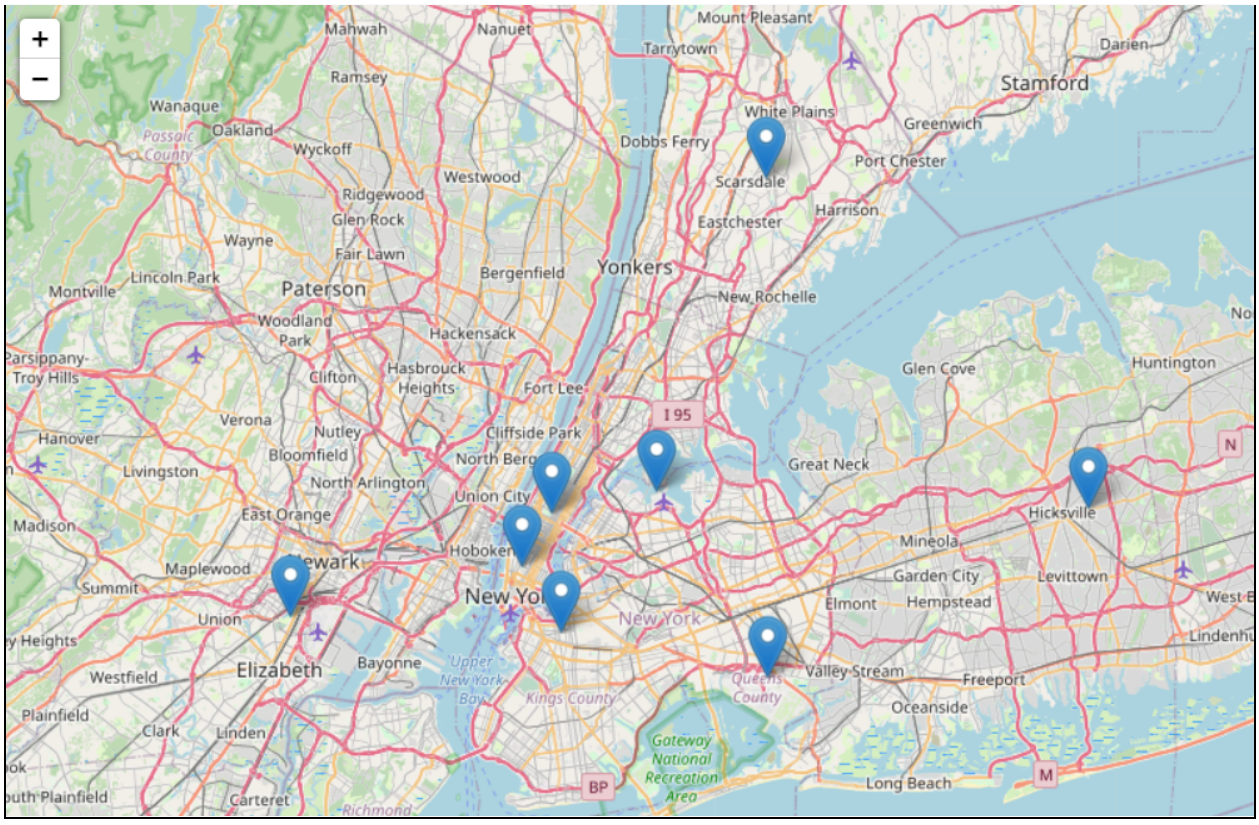
6. Results and Findings with Discussion

K-means Clustering on Uber Rides Data (NY: Apr 2014 - Sept 2014)
Dataset Size: *45,34,328*

6.1 K : 6



6.2 K : 8



7. Conclusion

Real-time Big Data analysis is still a challenging problem that needs to be addressed. There are many types of research addressing batch processing however only a few research done based on the stream processing analysis. But nowadays these techniques and technologies are very important for real-world business. In this paper, we have proposed Real-time Uber data analysis of Uber Rideshare Data with Apache Spark. From this we got a prediction, when we use Spark then we can gain our performance for the iterative algorithms, but not in Hadoop Mapreduce. As a future work, We would like to visualize popular Uber clusters in the google map in real-time.

8. References:

1. *Neilpatel's blog on how-uber-uses-data*
2. *Medium's Blog on how-uber-surge-price-works*
3. *Uber Engineering Blog: uber-big-data-platform*
4. *Lyft Engineering Blog: eng.lyft.com*
5. *https://www.productmanagementexercises.com/2273/design-dynamic-pricing-for-lyft*
6. *AnalyticsVidhya's blog on uber-and-lyft-cab-prices-data-analysis-and-visualization*
7. *Real-Time Uber Data Analysis of Popular Uber Locations in Kubernetes Environment, T.M Gunawardena, K.P.N Jayasena - IEEE*
8. *https://analyticsindiamag.com/uber-uses-data-analytics-supply-positioning-segmentation*
9. *Uber Engineering Blog: scaling-hdfs*
10. *Uber Engineering Blog: presto*