Big Data: Uber Case Study

Agenda

- Learning Big Data with Uber Case Study
- Uber’s Big Data Platform: 100+ Petabytes with Minute Latency
- Uber Data-set
- Uber Data Analysis in Map reduce(preliminaries)

Generation 1: The beginning of Big Data at Uber

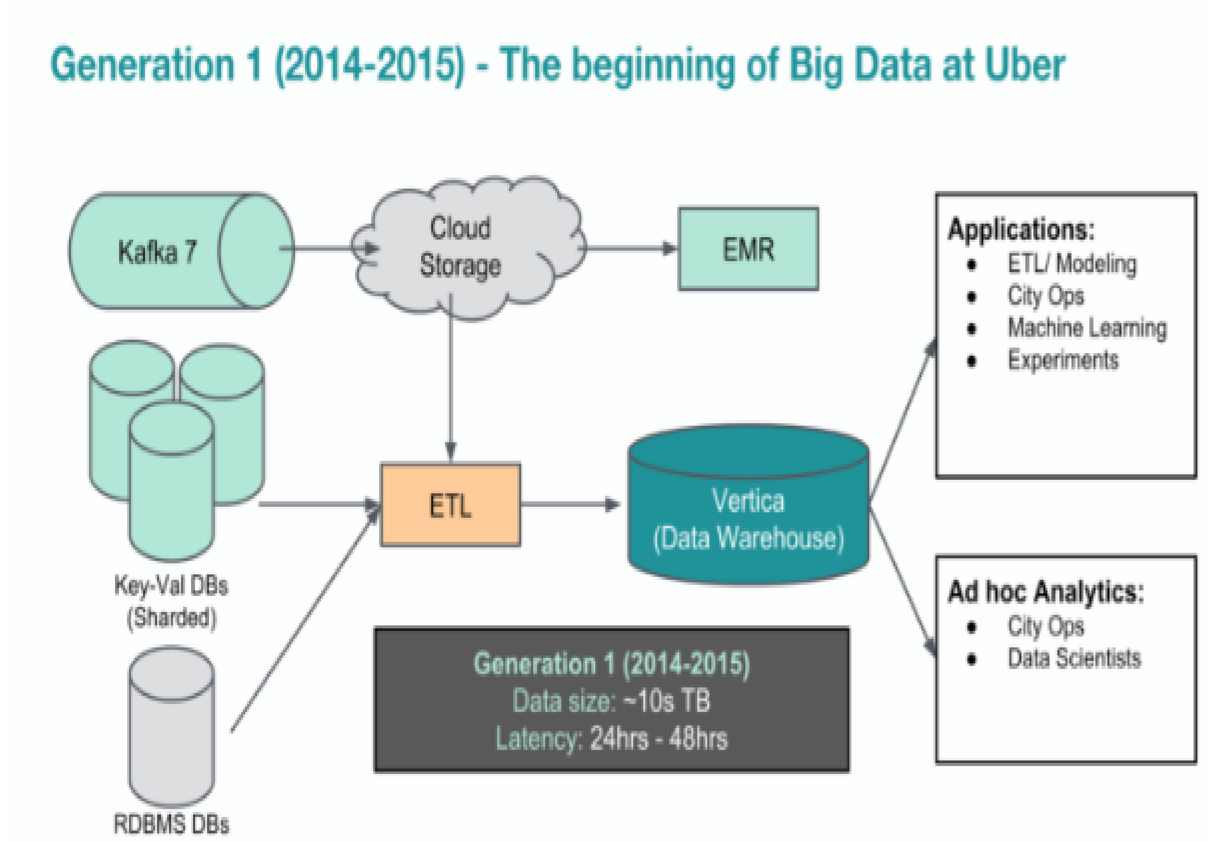


Figure 2: The first generation of Uber's Big Data platform allowed us to aggregate all of Uber's data in one place and provide standard SQL interface for users to access data.

City operations teams (thousands of users): These on-the-ground crews manage and scale Uber's transportation network in each market. With our business expanding to new cities, there are thousands of city operations teams accessing this data on a regular basis to respond to driver-and-rider-specific issues.

Data scientists and analysts (hundreds of users): These are the analysts and scientists spread across different functional groups that need data to help deliver the best possible transportation and delivery experiences to our users, for instance, when forecasting rider demand to future-proof our services.

Engineering teams (hundreds of users): Engineers across the company focused on building automated data applications, such as our Fraud Detection and Driver Onboarding platforms.

Generation 2: The arrival of Hadoop

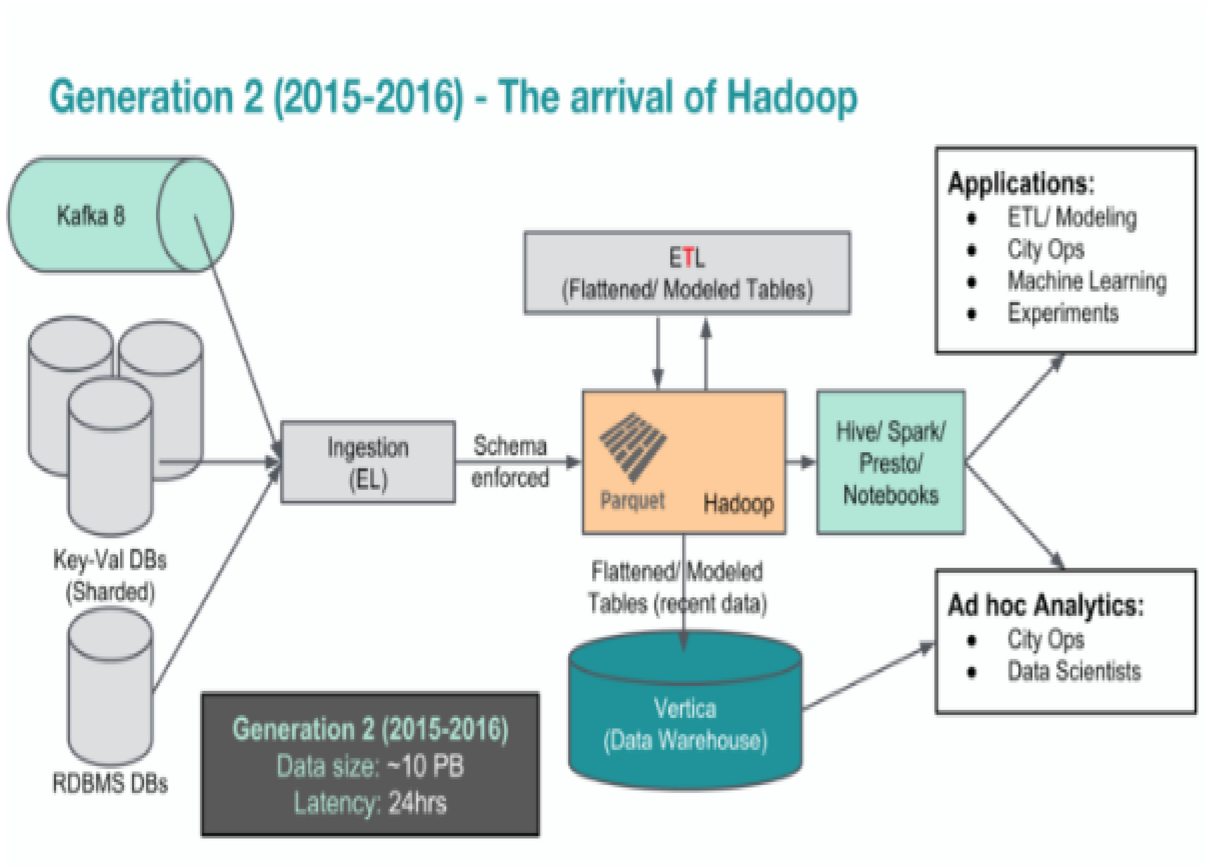


Figure 3: The second generation of our Big Data platform leveraged Hadoop to enable horizontal scaling. Incorporating technologies such as Parquet, Spark, and Hive, tens of petabytes of data was ingested, stored, and served.

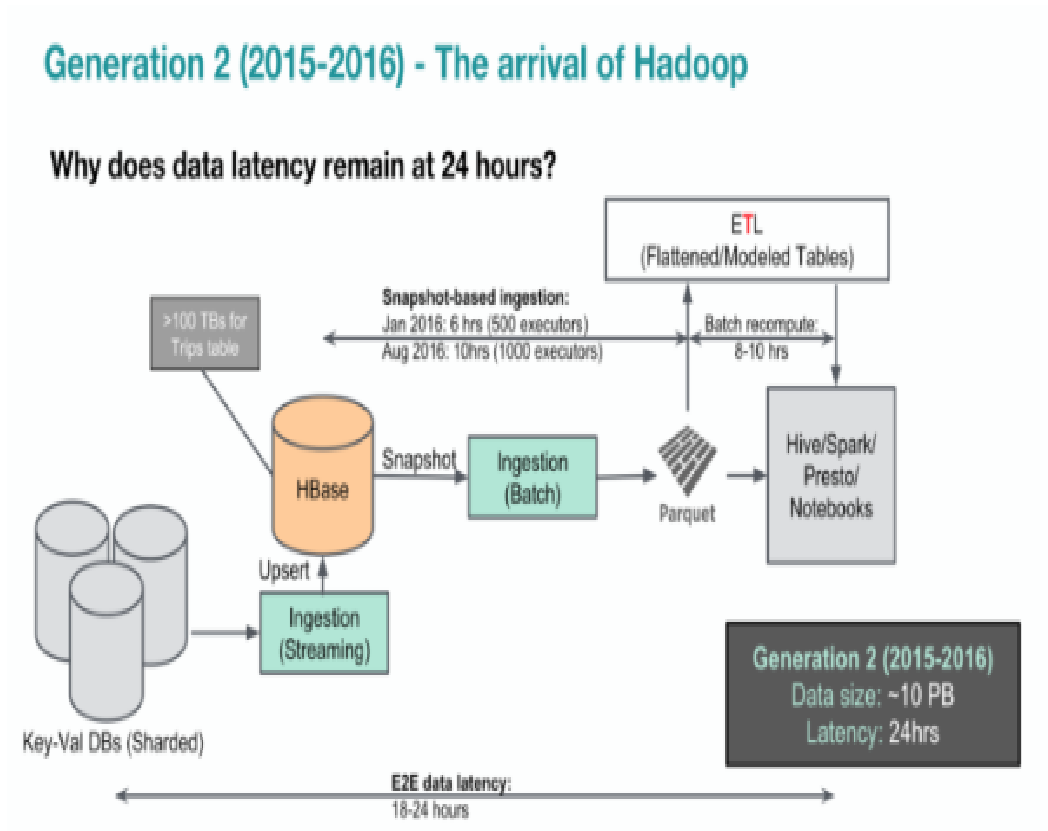

Generation 2 (2015-2016) - The arrival of Hadoop
Why does data latency remain at 24 hours?
ETL (Flattened/Modeled Tables)
>100 TBs for Trips table
Snapshot-based ingestion:
Jan 2016: 6 hrs (500 executors)
Aug 2016: 10hrs (1000 executors)
Batch recompute: 8-10 hrs
HBase
Snapshot
Ingestion (Batch)
Parquet
Hive/Spark/ Presto/ Notebooks
Upsert
Ingestion (Streaming)
Key-Val DBs (Sharded)
Generation 2 (2015-2016)
Data size: ~10 PB
Latency: 24hrs
E2E data latency: 18-24 hours

Figure 4: While Hadoop enabled the storage of several petabytes of data in our Big Data platform, the latency for new data was still over one day, a lag due to the snapshot-based ingestion of large, upstream source tables that take several hours to process.

Generation 3: Rebuilding our Big Data platform for the long term

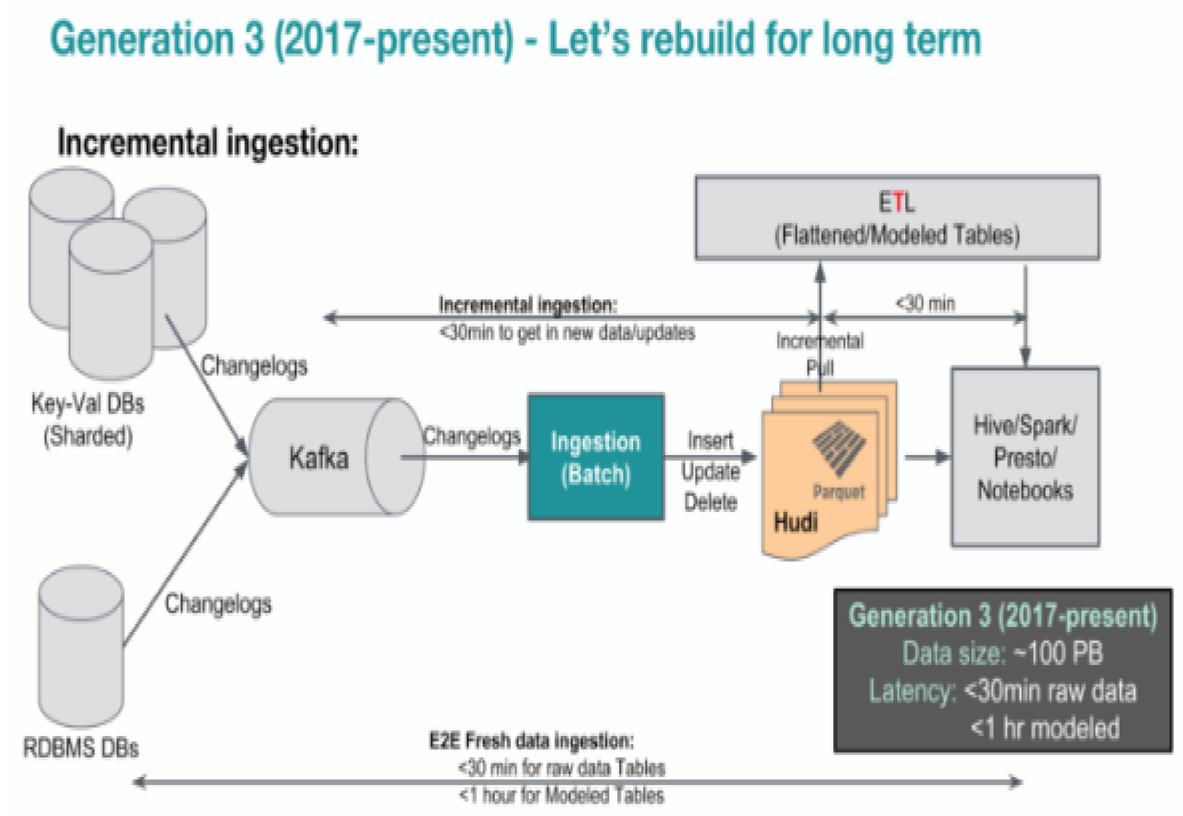

Generation 3 (2017-present) - Let's rebuild for long term
Incremental ingestion:
Key-Val DBs (Sharded)
RDBMS DBs
Changelogs
Changelogs
Changelogs
Kafka
Ingestion (Batch)
Insert
Update
Delete
Parquet
Hudi
ETL (Flattened/Modeled Tables)
Incremental ingestion: <30min to get in new data/updates
<30 min
Incremental Pull
Hive/Spark/ Presto/ Notebooks
Generation 3 (2017-present)
Data size: ~100 PB
Latency: <30min raw data
<1 hr modeled
E2E Fresh data ingestion:
<30 min for raw data Tables
<1 hour for Modeled Tables

Figure 5: The third generation of our Big Data platform incorporates faster, incremental data ingestion (using our open source Marmaray framework), as well as more efficient storage and serving of data via our open source Hudi library.

Relationship between different components of Big Data platform

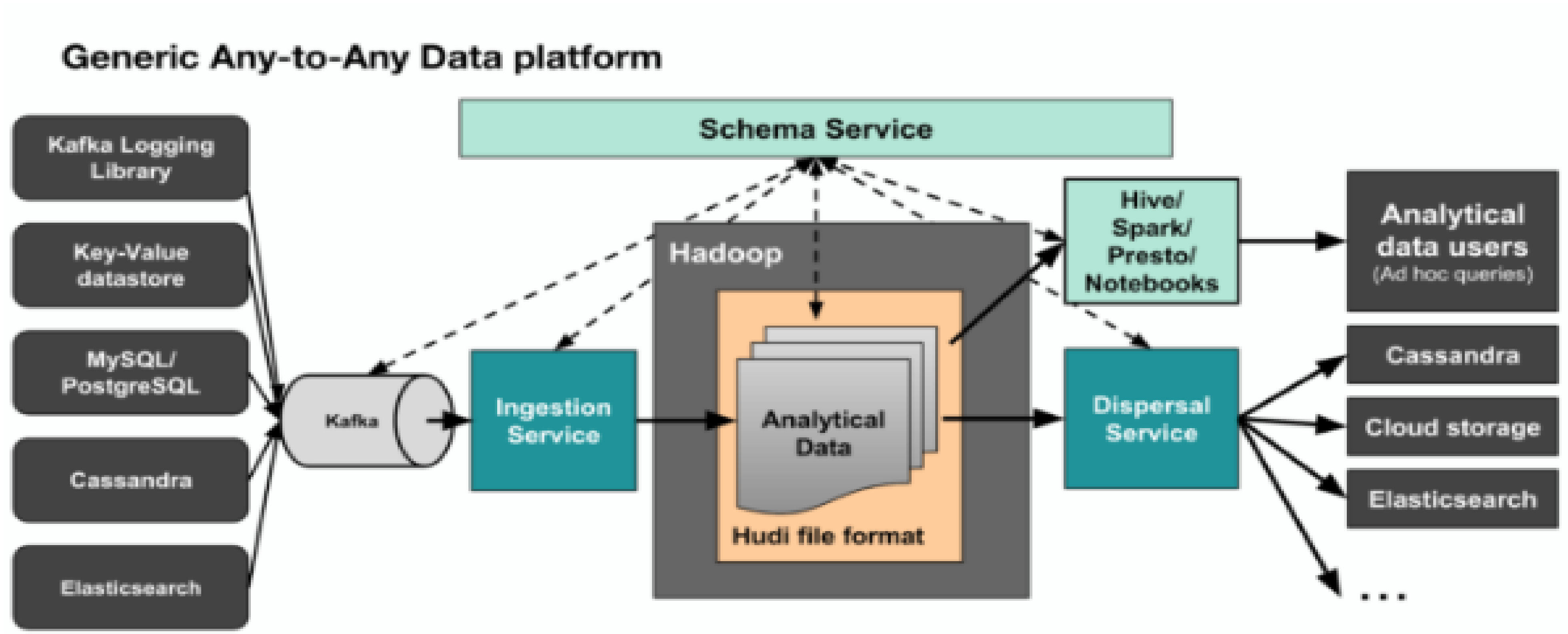
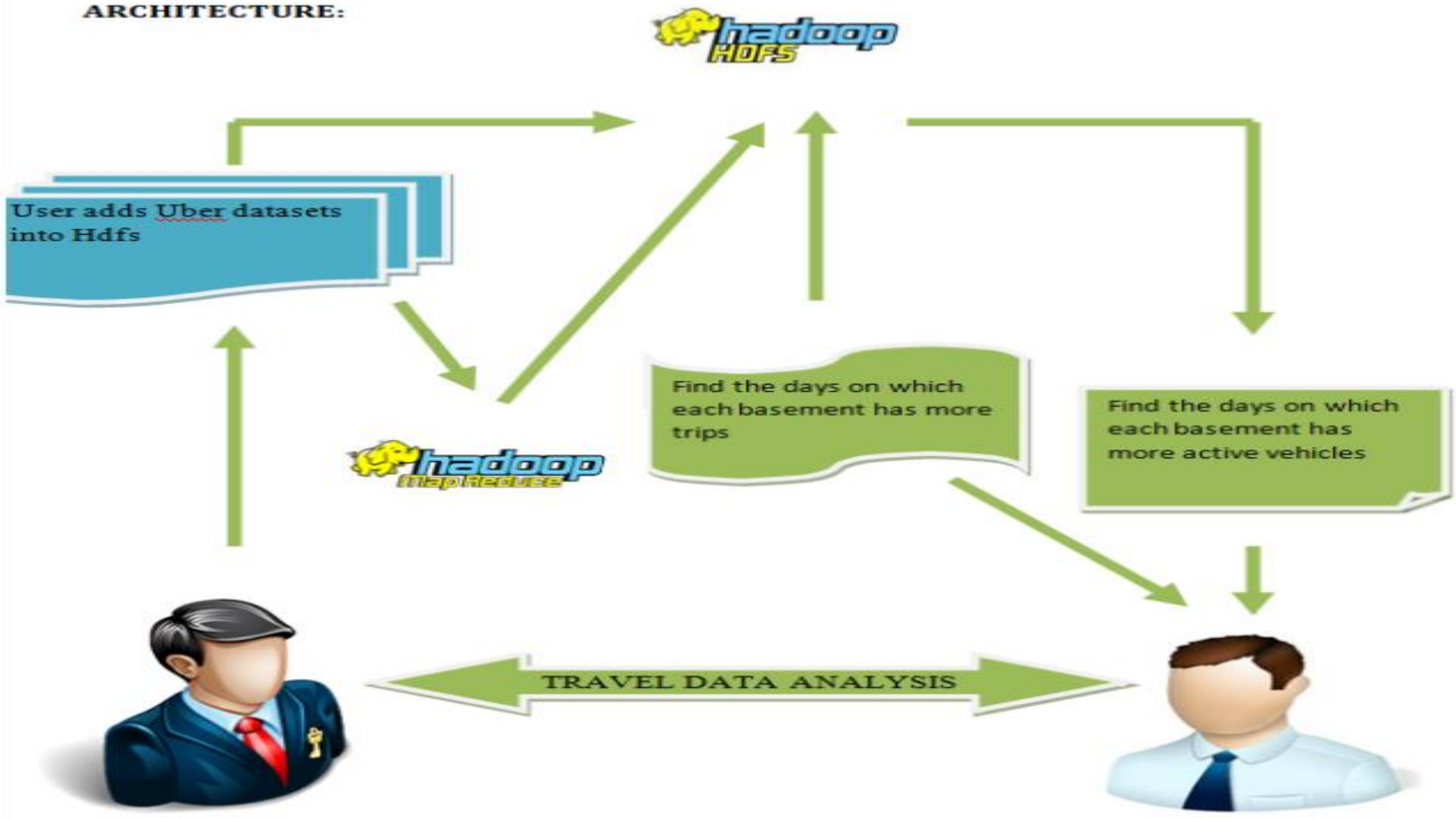
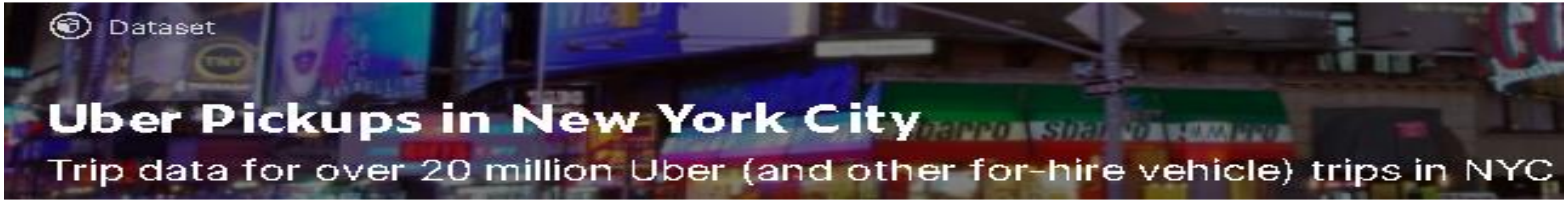


Figure 8: Building a more extensible data transfer platform allowed us to easily aggregate all data pipelines in a standard way under one service as well as support any-to-any connectivity between any data source and data sink.

ARCHITECTURE:


hadoop HDFS
User adds Uber datasets into Hdfs
hadoop map Reduce
Find the days on which each basement has more trips
Find the days on which each basement has more active vehicles
TRAVEL DATA ANALYSIS


Dataset

Uber Pickups in New York City

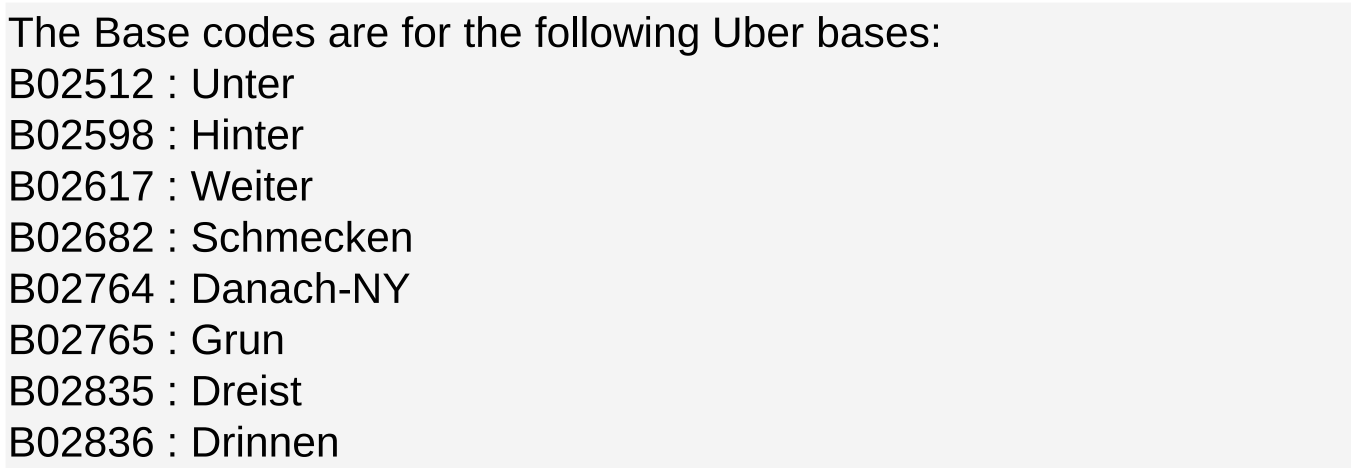
Trip data for over 20 million Uber (and other for-hire vehicle) trips in NYC

- This explains data on over 4.5 million Uber pickups in New York City from April to September 2014, and 14.3 million more Uber pickups from January to June 2015.
- Trip-level data on 10 other for-hire vehicle (FHV) companies, as well as aggregated data for 329 FHV companies, is also included.

- FiveThirtyEight obtained the data from the NYC Taxi & Limousine Commission (TLC) by submitting a Freedom of Information Law request on July 20, 2015.
- This data was used for four FiveThirtyEight stories: Uber Is Serving New York's Outer Boroughs More Than Taxis Are, Public Transit Should Be Uber's New Best Friend, Uber Is Taking Millions Of Manhattan Rides Away From Taxis, and Is Uber Making NYC Rush-Hour Traffic Worse?.

Uber trip data

- There are six files of raw data on Uber pickups in New York City from April to September 2014. The files are separated by month and each has the following columns:
- Date/Time : The date and time of the Uber pickup
- Lat : The latitude of the Uber pickup
- Lon : The longitude of the Uber pickup
- Base : The TLC base company code affiliated with the Uber pickup



The Base codes are for the following Uber bases:
B02512 : Unter
B02598 : Hinter
B02617 : Weiter
B02682 : Schmecken
B02764 : Danach-NY
B02765 : Grun
B02835 : Dreist
B02836 : Drinnen

Uber Data Analysis using MapReduce

- The Uber dataset consists of four columns; they are dispatching_base_number, date, active_vehicles and trips
- **In thie problem statement-1, we will find the days on which each basement has more trips.**

```
public void map(Object key, Text value, Context context
) throws IOException, InterruptedException {
String line = value.toString();
String[] splits = line.split(“,”);
basement.set(splits[0]);
date = format.parse(splits[1]);
} trips = new Integer(splits[3]);
String keys = basement.toString()+ ” “+days[date.getDay()];
context.write(new Text(keys), new IntWritable(trips));
}
```

From the Mapper, we will take the combination of the basement and the day of the week as key and the number of trips as value.
using **SimpleDateFormat** class in Java. Now, to take out the day of the date, we will use
the **getDay()** which will return an integer value with the day of the week’s number. So, we have created an array which consists of all the days from Sunday to Monday and have passed the value returned
by **getDay()** into the array in order to get the day of the week.
Now, after this operation, we have returned the combination of **Basement_number+Day of the week** as keyand the **number of trips** as value.

Reducer Class:

- In the reducer, we will calculate the sum of trips for each basement and for each particular day, by using the below lines of code.

```
public static class IntSumReducer
extends Reducer<Text,IntWritable,Text,IntWritable> {
private IntWritable result = new IntWritable();
public void reduce(Text key, Iterable<IntWritable> values,
Context context
) throws IOException, InterruptedException {
int sum = 0;
for (IntWritable val : values) {
sum += val.get();
}
result.set(sum);
context.write(key, result);
}
}
```

Output:

```
B02512 Sat 15026
B02512 Sun 10487
B02512 Thu 15809
B02512 Tue 12041
B02512 Wed 12691
B02598 Fri 93126
B02598 Mon 60882
B02598 Sat 94588
B02598 Sun 66477
B02598 Thu 90333
B02598 Tue 63429
B02598 Wed 71956
B02617 Fri 125067
```

Problem Statement-2

- In this problem statement, we will find the days on which each basement has more number of active vehicles.

```
public void map(Object key, Text value, Context context
) throws IOException, InterruptedException {
String line = value.toString();
String[] splits = line.split(",");
basement.set(splits[0]);
try {
date = format.parse(splits[1]);
} catch (ParseException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
active_vehicles = new Integer(splits[3]);
String keys = basement.toString()+ " "+days[date.getDay()];
context.write(new Text(keys), new IntWritable(active_vehicles));
}
}
```

From the Mapper, we will take the combination of the basement and the day of the week as key and the number of active vehicles as value.
Now, to take out the day of the date, we will use the getDay(), which will return an integer value with the day of the week's number. So, we have created an array which consists of all the days from Sunday to Monday and have passed the value returned by getDay(), into the array in order to get the day of the week.

Now, after this operation, we have returned the combination of Basement_number+Day of the week as key and the number of active vehicles as value.

Reducer Class:

- Now, in the reducer, we will calculate the sum of active vehicles for each basement and for each particular day, using the below lines of code.

```
public static class IntSumReducer
extends Reducer<Text,IntWritable,Text,IntWritable> {
private IntWritable result = new IntWritable();
public void reduce(Text key, Iterable<IntWritable> values,
Context context
) throws IOException, InterruptedException {
int sum = 0;
for (IntWritable val : values) {
sum += val.get();
}
result.set(sum);
context.write(key, result);
}
}
```

Output:

```
B02512 Fri 2221
B02512 Mon 1676
B02512 Sat 1931
B02512 Sun 1447
B02512 Thu 2187
B02512 Tue 1763
B02512 Wed 1900
B02598 Fri 9830
B02598 Mon 7252
B02598 Sat 8893
B02598 Sun 7072
B02598 Thu 9782
B02598 Tue 7433
B02598 Wed 8391
B02617 Mon 9758
```

- **“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. All these data problems…are really crystalized on this one math with people all over the world trying to get where they want to go. That’s made data extremely exciting here, it’s made engaging with Spark extremely exciting.”- said Uber’s Head of Data Aaron Schildkrout.**
- **Data Products at Uber - Surge Pricing**
- Matching Algorithms at Uber
- Fare Estimates