

Stream Computing & Analytics At Uber

Sudhir Tonse, Uber Engineering

@stonse

Mar 7, 2016



UBER

UBER

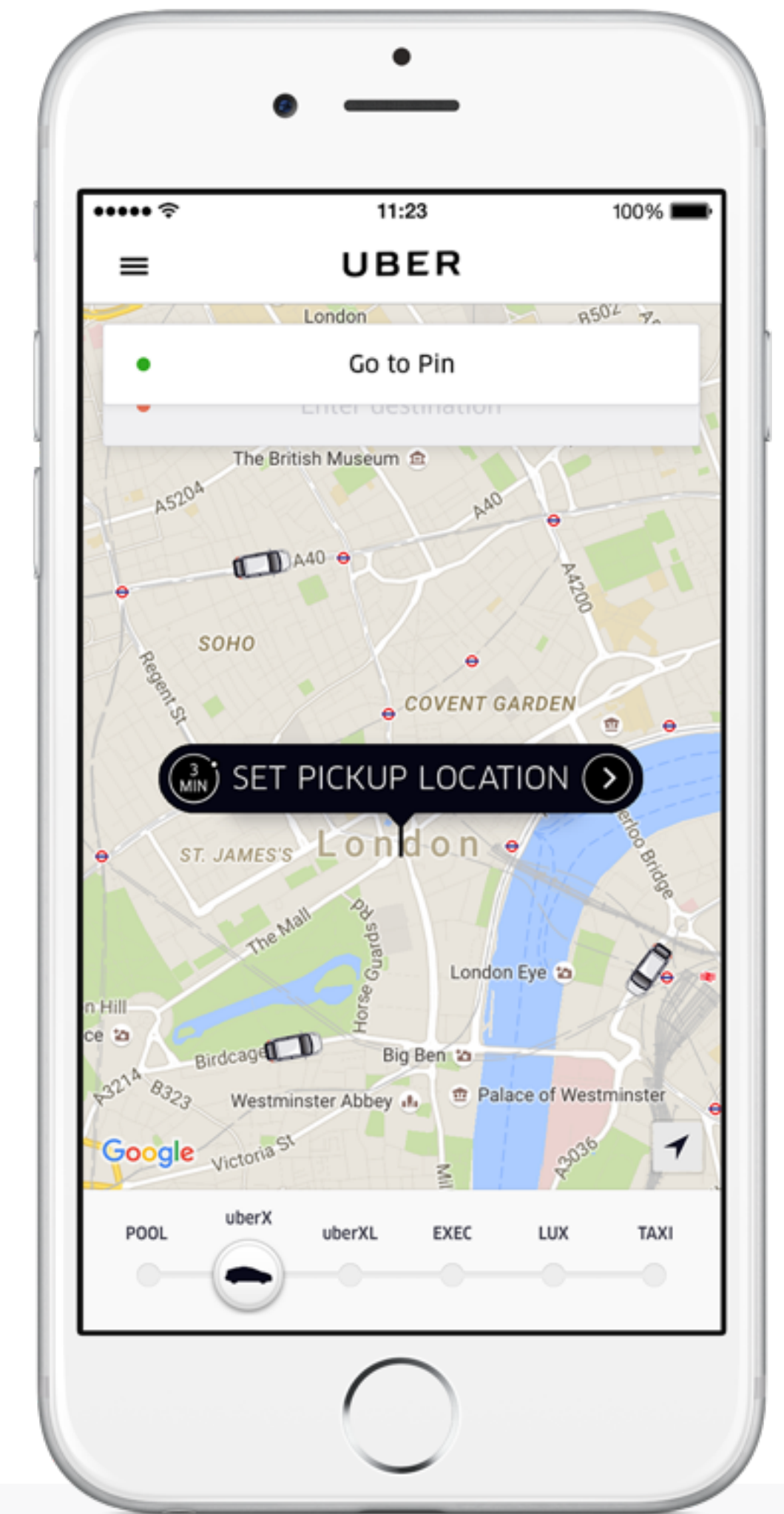


UBER

Get There

Your day belongs to you

- ~ 68 countries / 350+ cities
- Transportation as reliable as running water, everywhere, for everyone



Who am I

Engineering Leader, Marketplace Data at Uber

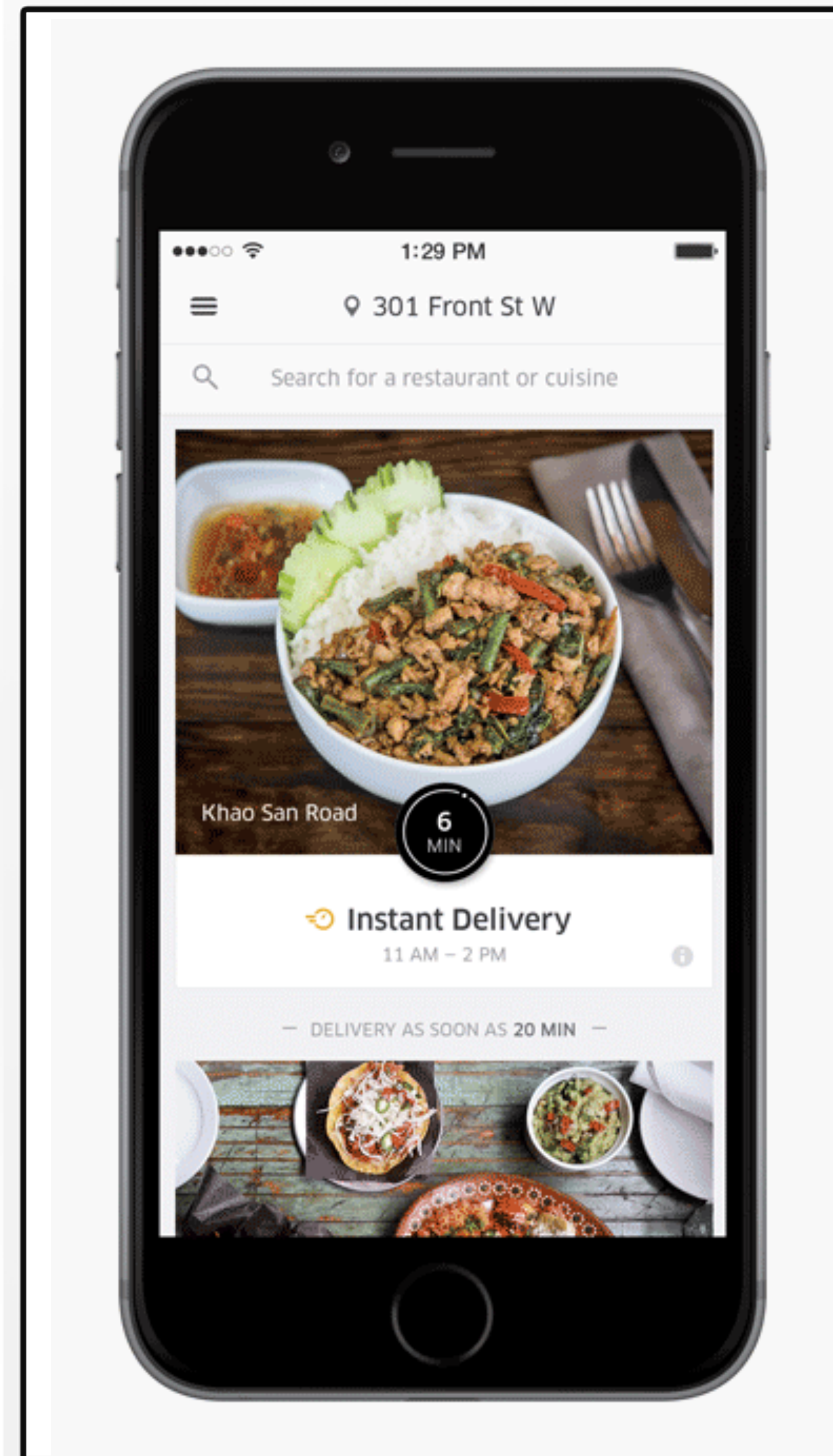
- Marketplace Dynamics
 - Realtime Data Processing
 - Analytics
 - Forecasting
- Previously managed Cloud Platform at Netflix
- Twitter @stonse



Agenda

What's on the menu?

- Use Cases
- Problem Space
- Overall Architecture
- Choices & Tradeoffs
- Q & A

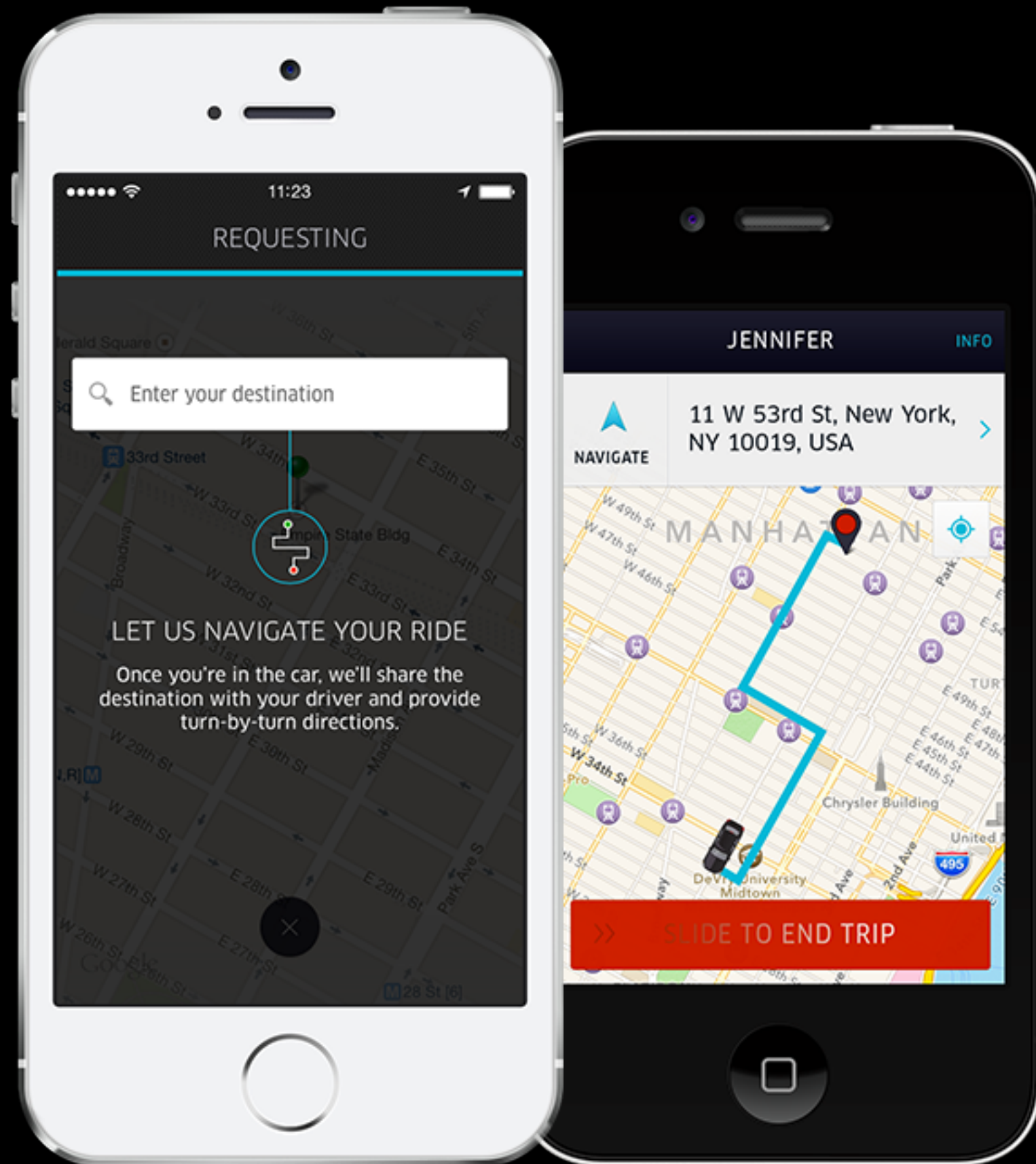


Use Cases

Some examples of what we work on



Stream Processing ...



Request Event



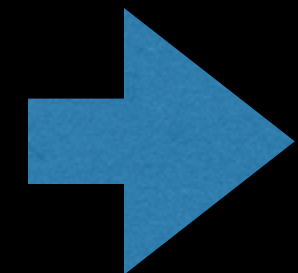
Driver Accept Event



Trip Started Event



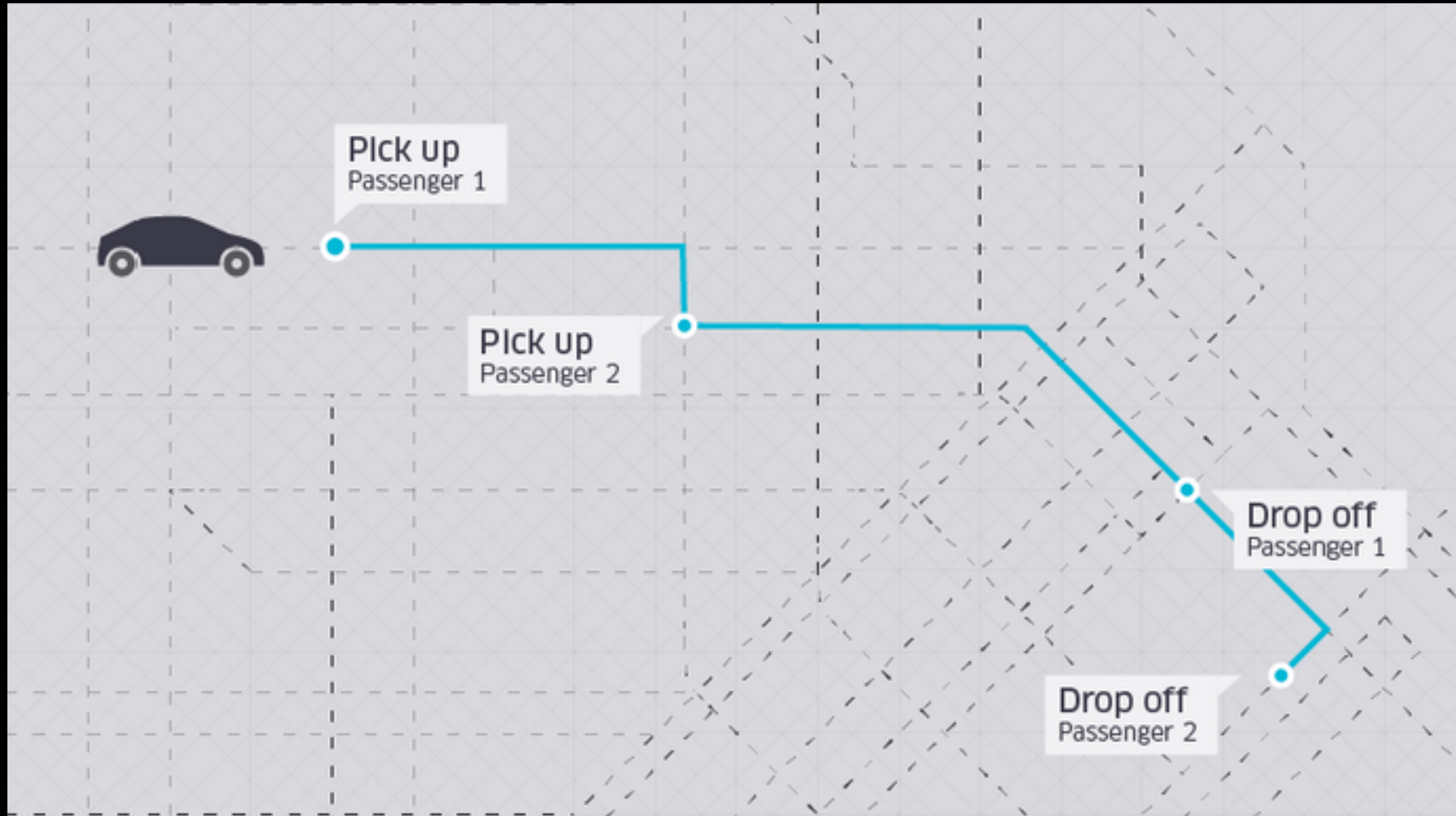
more events ...



M
a
t
c
h

S
e
r
v
i
c
e
s

Trip States



Realtime OLAP/Exploration

There is always a need for quick exploration

How many open cars in London, right **NOW**?



**Estimated Pickup time, Driving Time and etc over time by
geographic area**

KPIs over time by hexagon area

SAN FRANCISCO

ALLEN KEY



CITY: San Francisco Bay Area

PRODUCT: UberX

DATE FROM: 12/1/15 2:00pm (UTC-8)

DATE TO: 12/2/15 2:00pm (UTC-8)

INTERVAL: 1 hour

QUERY

PICK METRICS

SELECTION RADIUS

0 1 2 3 4 5 6 7

Driver Driving Client (trips)



Load Heatmap

Pre request ETA (min)



Load Heatmap

Post request ETA (min)



Load Heatmap

Eyeballs count (views)



Load Heatmap

Completed trips (trips)



Load Heatmap

Requested trips (trips)



Load Heatmap

Average time from request to dispatch (sec)



Load Heatmap

Driver Pickup Time (min)



Load Heatmap

Driver Enroute Time (min)



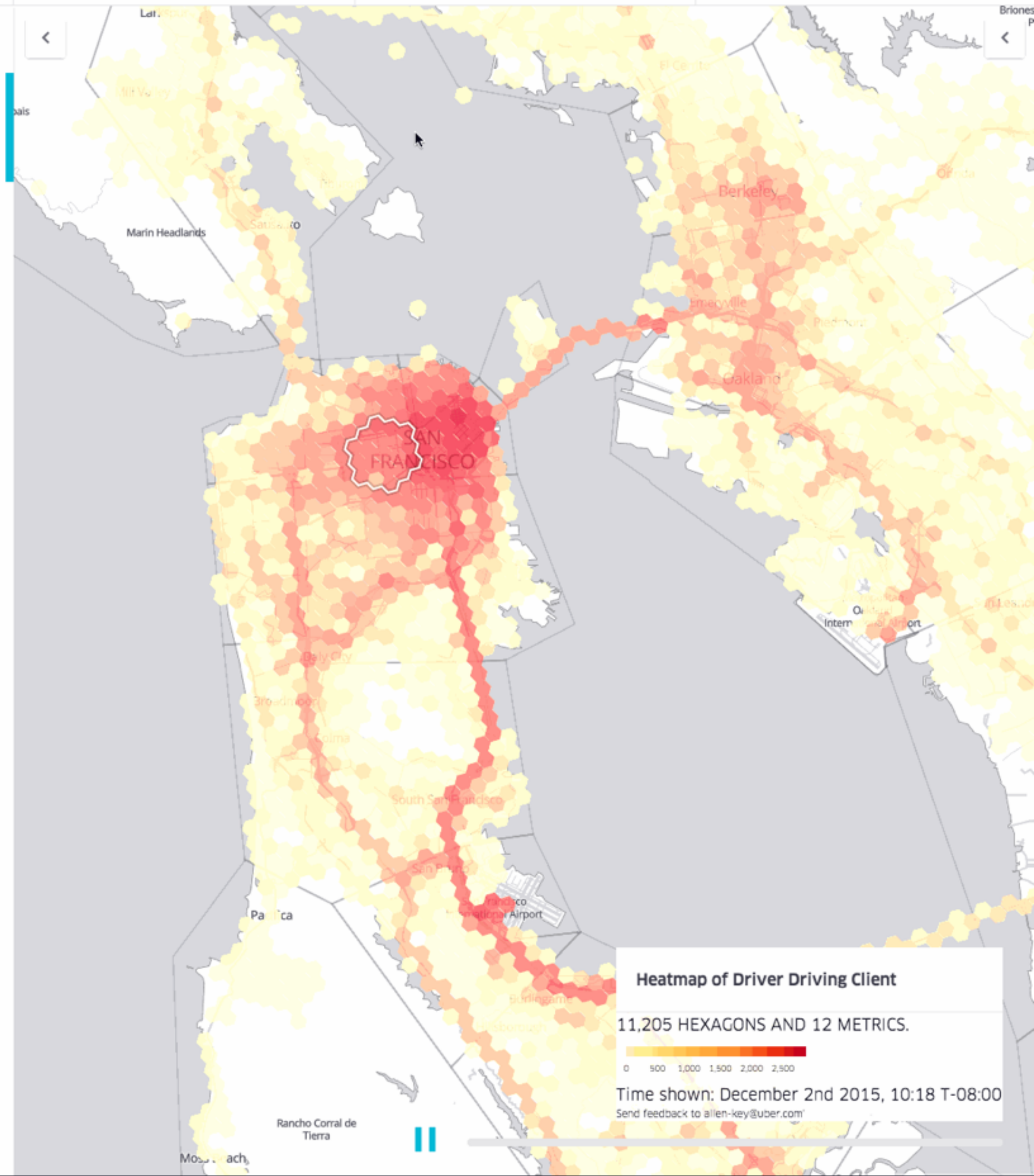
Load Heatmap

Drivers time in Open status (min)



Load Heatmap

Drivers time in On Trip status (min)



Heatmap of Driver Driving Client

11,205 HEXAGONS AND 12 METRICS.



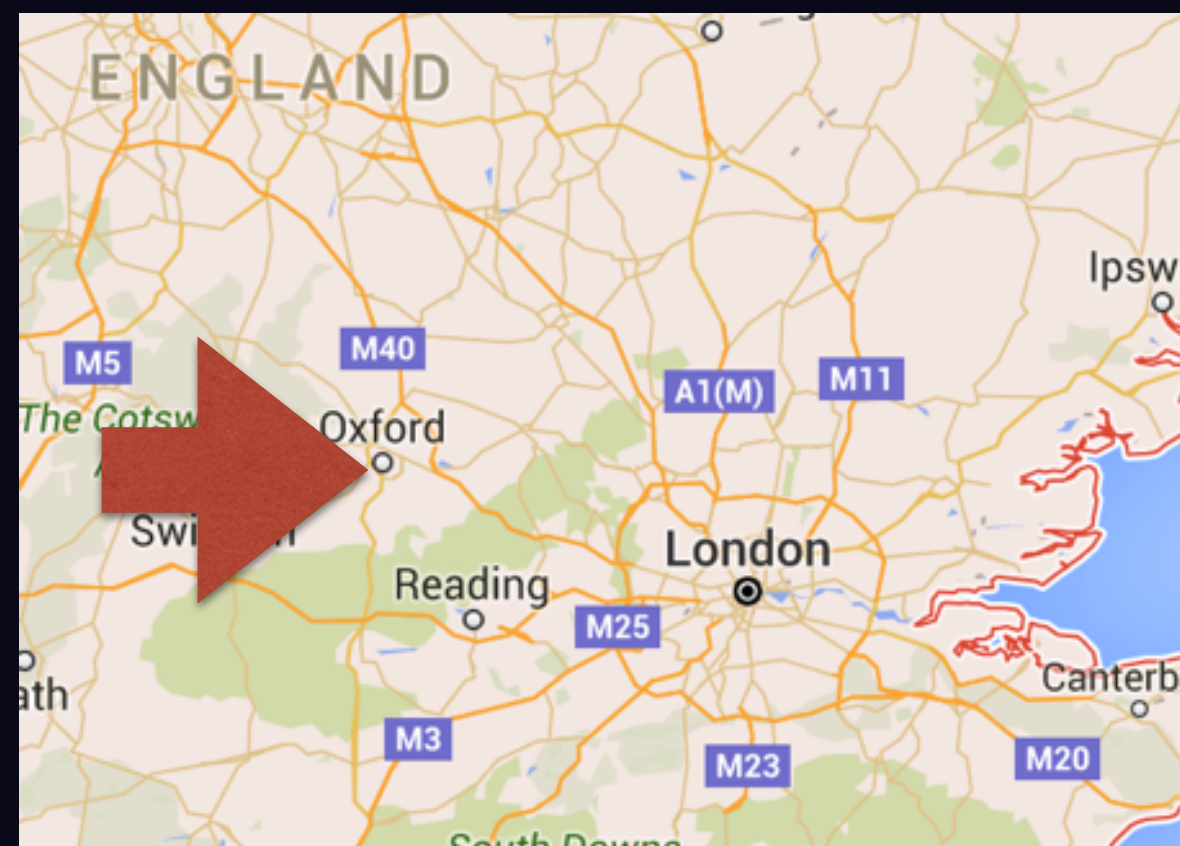
Time shown: December 2nd 2015, 10:18 T-08:00

Send feedback to allen-key@uber.com

Behaviour/Gaming/Fraud

How many drivers **cancel** a
request **> 3 times** in a row
within a **10-minute** window?

Detect riders requesting a pickup 100 miles apart within a half hour window?



Complex Event Processing

IF
This ->

```
FROM driver_canceled#window.time(10 min)
SELECT clientUUID, count(clientUUID) as cancelCount
GROUP BY clientUUID HAVING cancelCount > 3
INSERT INTO hipchat(room);
```

Then that ->

If This Then That

A simple SQL-like
syntax!

that can take **ACTIONS!!**

In **Real Time!**

Actions

HipChat Action

Topic
driver_rejection_repeatedly_SF

HipChat Room
SF cancellation realtime detection by Mystique

HipChat

SF cancellation realtime detection by Mystique
This is the room topic. Double click to change it.

CAG Bot driver ([redacted]) reject 4 trips in the last 10 minutes

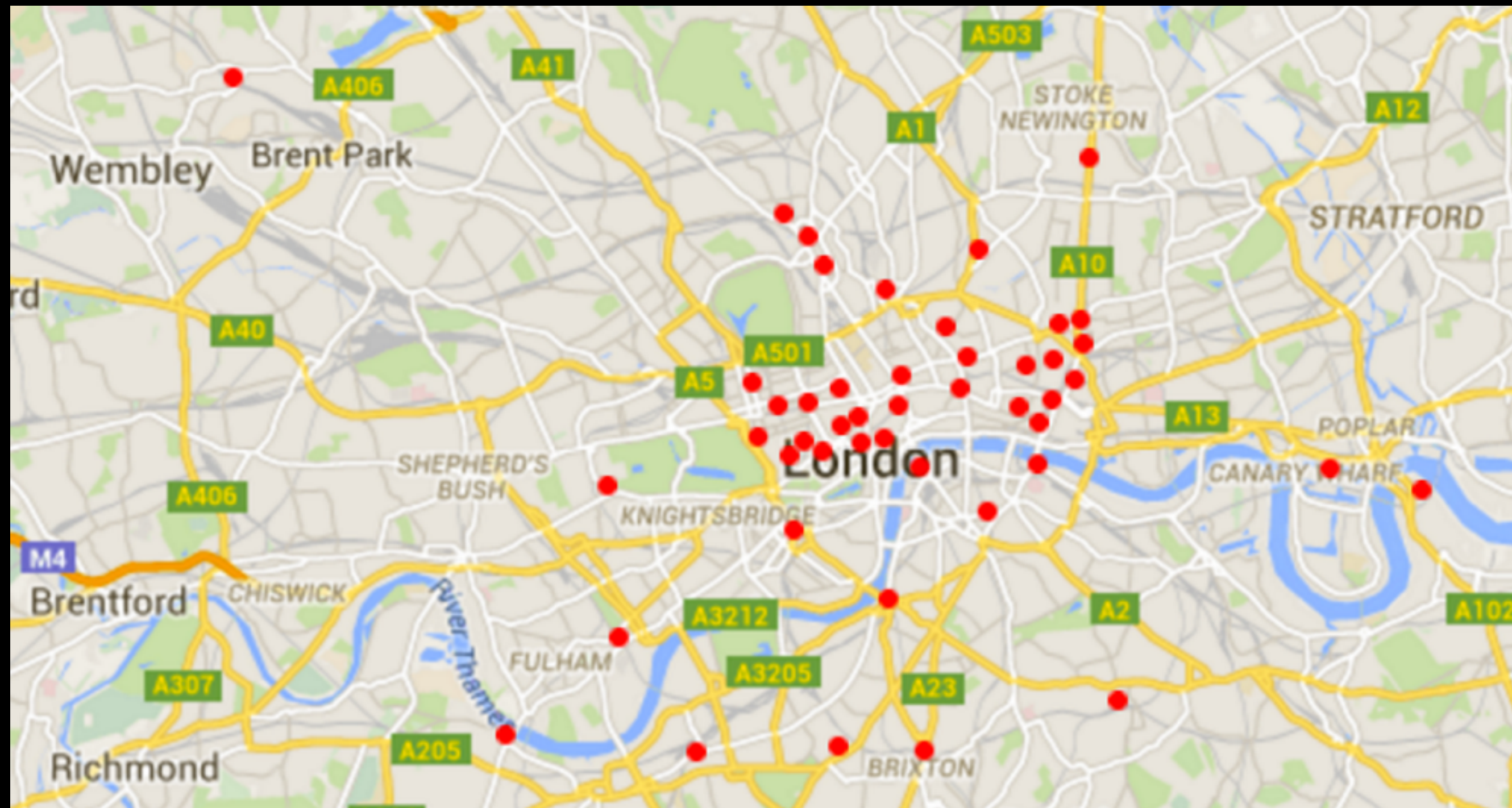
CAG Bot driver ([redacted]) has been repeatedly canceled by clients 2 times in the last 10 minutes

Supply Positioning

Clusters Of Supply & Demand



Near Term Forecasting



Airports, Stadiums, Arenas, Business districts, Transit stations, Malls, Dining

...

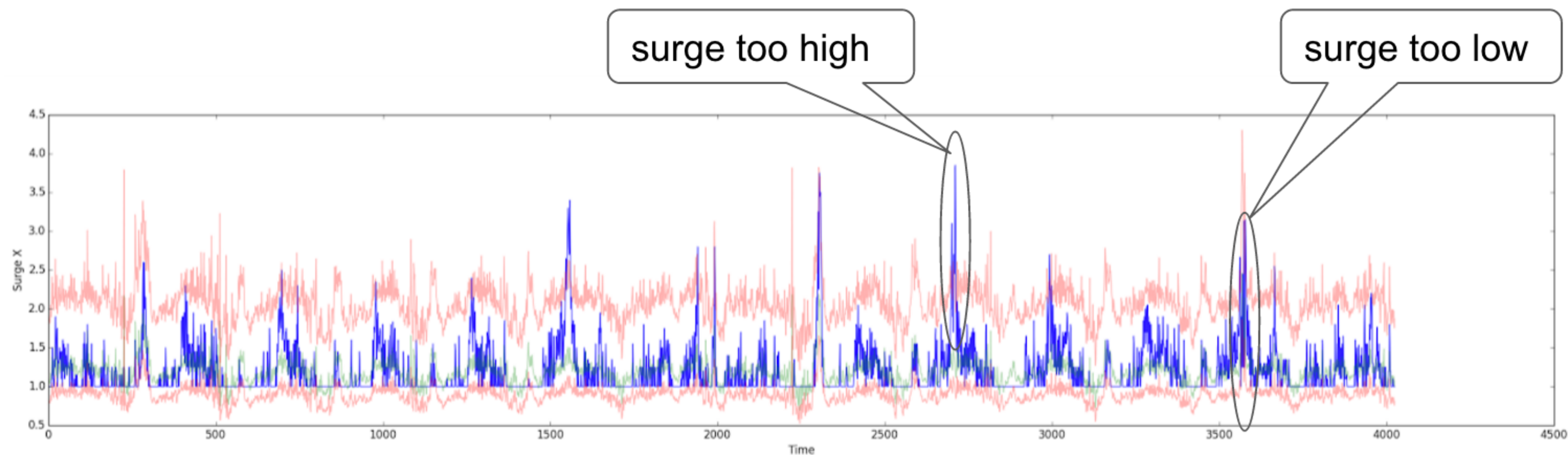
Monitoring Business Metrics

Realtime Monitoring of Business Metrics

Blue line: production surge x ;

Green line: model estimated surge x ;

Red line: error bounding surge x



Ops & Data Scientists

Ops & Data Scientists (Dashboards & Analytics)

Gairos - Realtime Events & Data Solutions

Uber Realtime Data Intelligence

Data Sources | User Datasets | Process Data | Curated Queries | User Queries | Data Visualization | Data Tools | Help

```

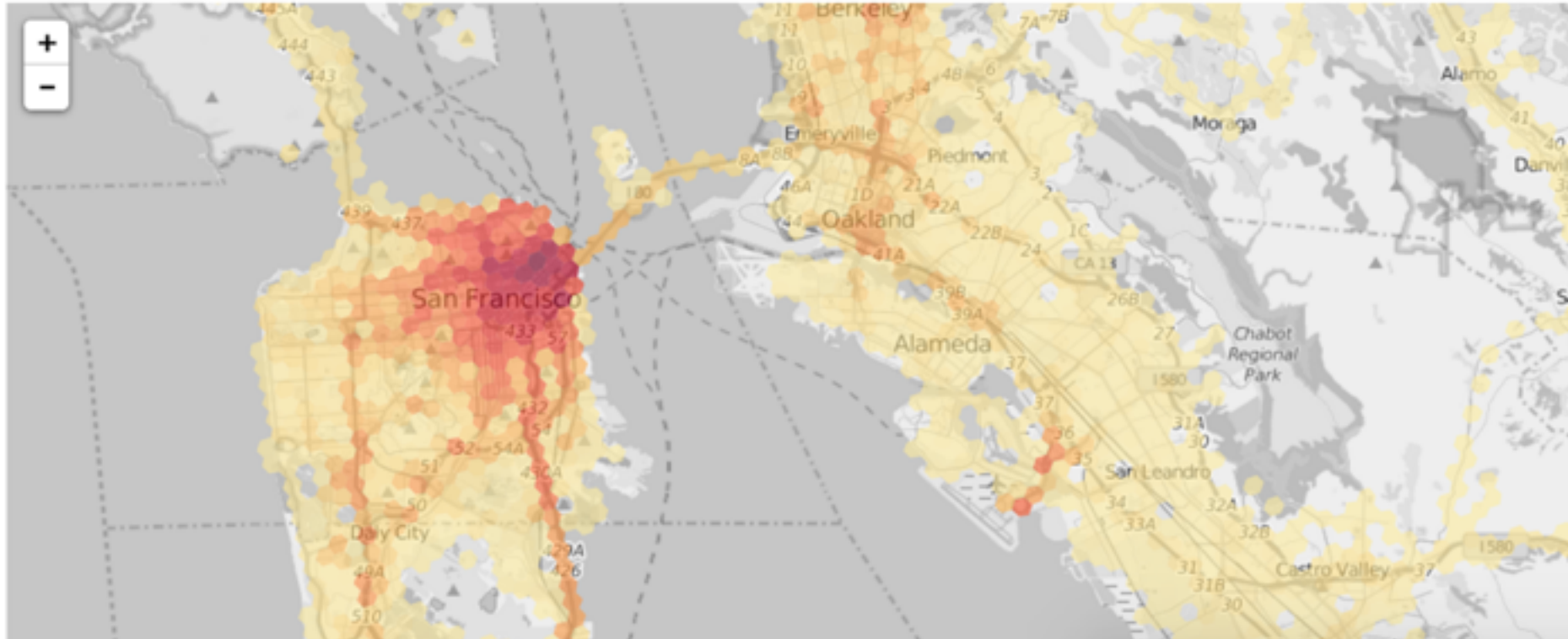
1 {
2   "by": [
3     "hexagon_id"
4   ],
5   "filter": {
6     "type": "and",
7     "fields": [
8       {
9         "type": "eq",
10        "dimension": "city",
11        "value": "1"
12      },
13      {
14        "type": "eq",
15        "dimension": "vvids",

```

Data Source: Provides aggregated driver information. Dimensions: @timestamp, driverUUID, city, hexagon_id, geofence, status, vvids Metrics:

Query | Download as JSON | Download as CSV

Table | Heatmap | Stats




jupyter marketplace_experience-Copy1 Last Checkpoint: 02/08/2016 (unsaved changes) Control Panel Logout

File Edit View Insert Cell Kernel Help Python 2

Code	dispatching
2 dispatch_accept	dispatching
3 dispatch_accept	offline
4 dispatch_accept	on_trip
5 dispatch_accept	open
6 dispatch_secondary_accept	None
7 dispatch_secondary_accept	accepted
8 dispatch_secondary_accept	arrived
9 dispatch_secondary_accept	dispatching
10 dispatch_secondary_accept	on_trip

Logistic Regression

```
In [197]: import statsmodels.api as sm
logit = sm.Logit(data['label'], data.drop('label', axis=1))
result = logit.fit()
print(result.summary())
```

Optimization terminated successfully.
Current function value: 0.197133
Iterations 7

Logit Regression Results

Dep. Variable:	label	No. Observations:	129080		
Model:	Logit	Df Residuals:	129075		
Method:	MLE	Df Model:	4		
Date:	Mon, 08 Feb 2016	Pseudo R-squ.:	-0.01496		
Time:	17:10:30	Log-Likelihood:	-25446.		
converged:	True	LL-Null:	-25071.		
		LLR p-value:	1.000		
	coef	std err	z	P> z	[95.0% Conf. Int.]
predicted_eta	-0.0012	5.96e-05	-19.894	0.000	-0.001 -0.001
surge	2.9637	0.025	117.685	0.000	2.914 3.013
speed	-0.0057	0.002	-3.437	0.001	-0.009 -0.002
horizontal_accuracy	-0.0018	0.000	-5.897	0.000	-0.002 -0.001
fare	0.0189	0.001	15.509	0.000	0.017 0.021

Other Data

```
In [228]: from shapely import speedups
from shapely import wkt
speedups.enable()
wkt.loads(geofences.filter(geofences.name.isin(['East Bay'])).select('shape').collect()[0].shape)
```

Out[228]:



What's not covered

to keep this focused

ETL Pipeline

Offline/Batch Analytics

Business Intelligence

Stream Processing fundamentals ..

...

Problem space

What are the challenges?

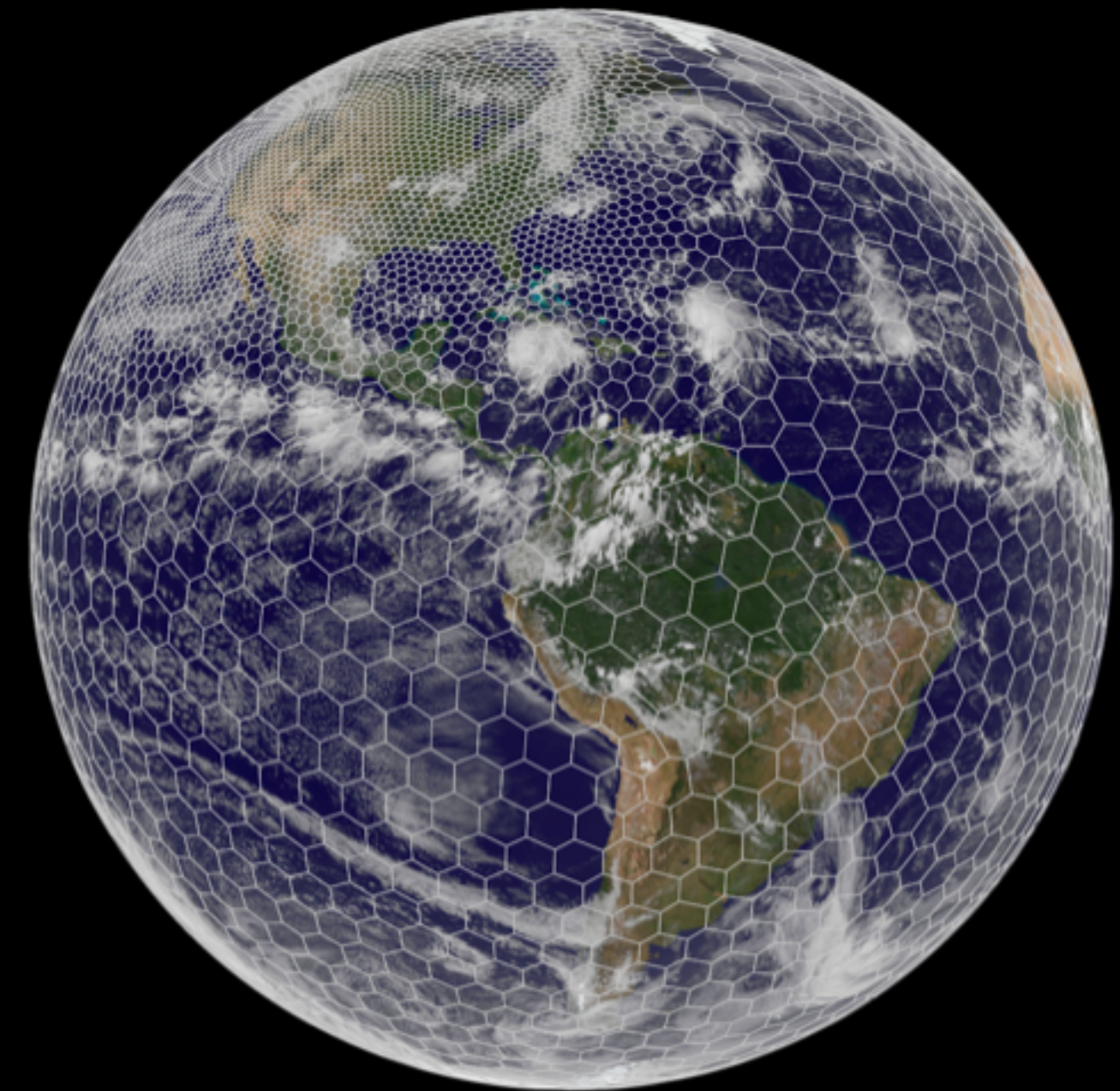
OLAP of Spacio-Temporal data

Large Scale Data

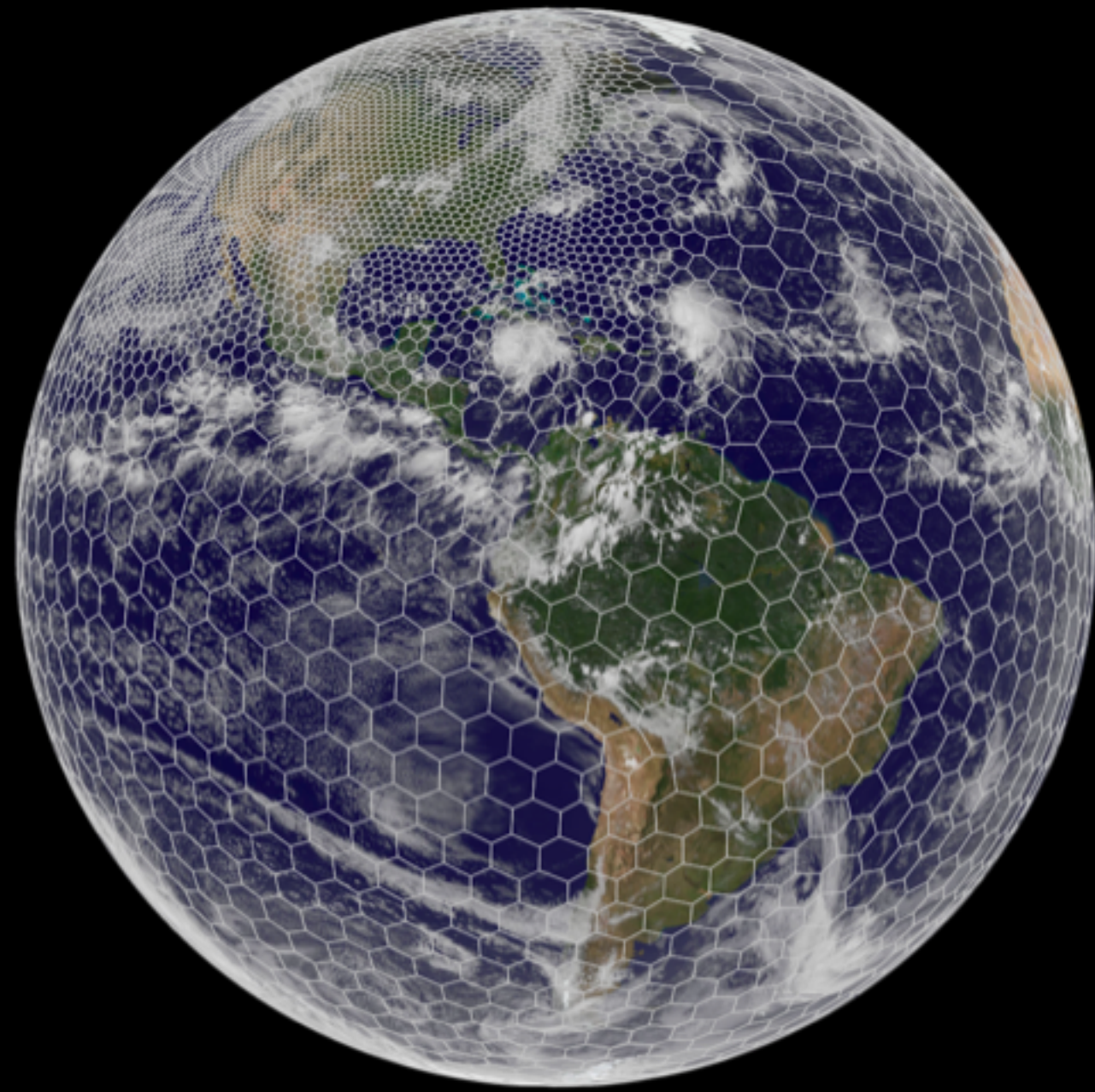
Near Real Time

Hexagons

- Indexing, Lookup, Rendering
- Symmetric Neighbors
- Convex & Compact Regions
- Equal Areas
- Equal Shape



Scale



Geo Space

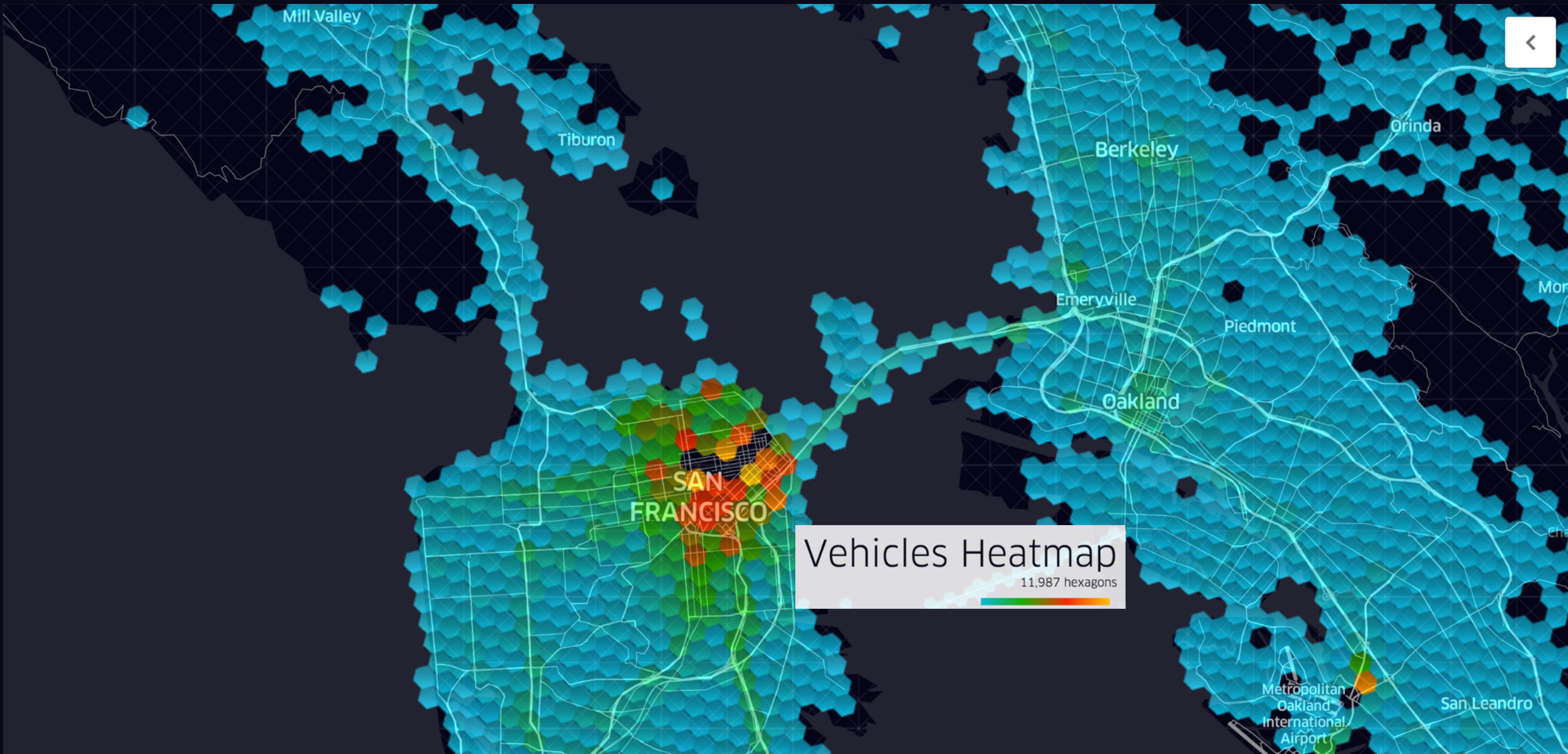


Vehicle Types

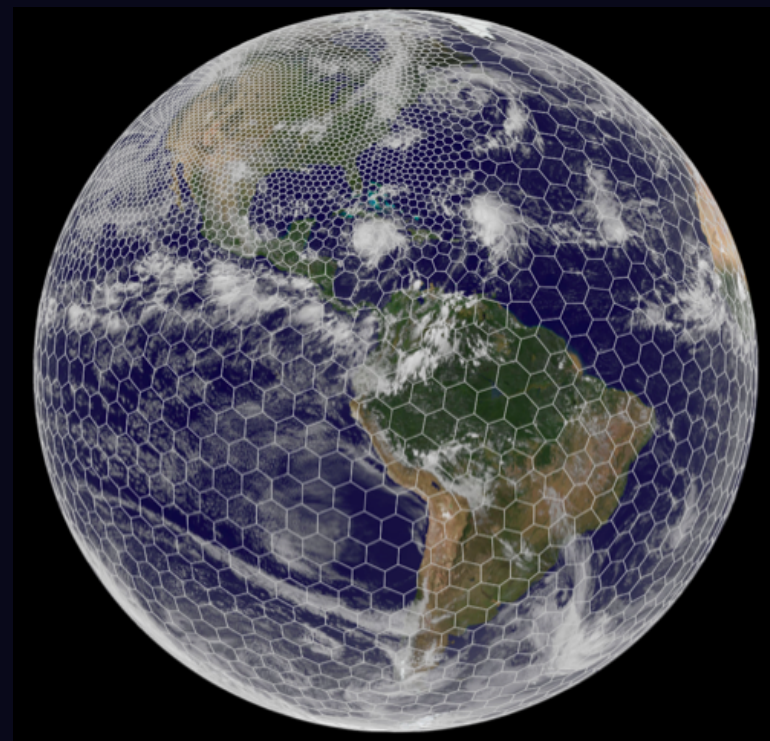


Time

Granular Data



Granular Data



Over **10,000** hexagons in the city



Granular Data

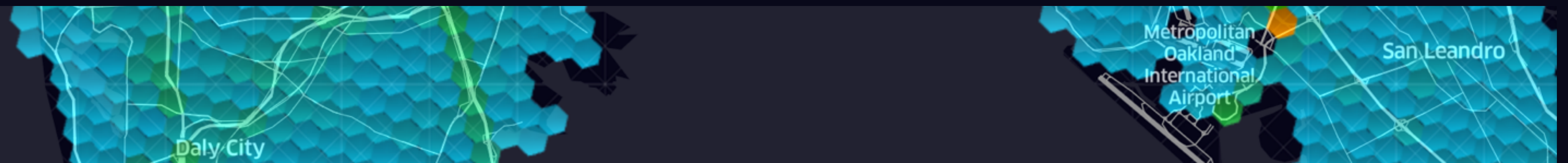
7 vehicle types



Granular Data



1440 minutes in a day



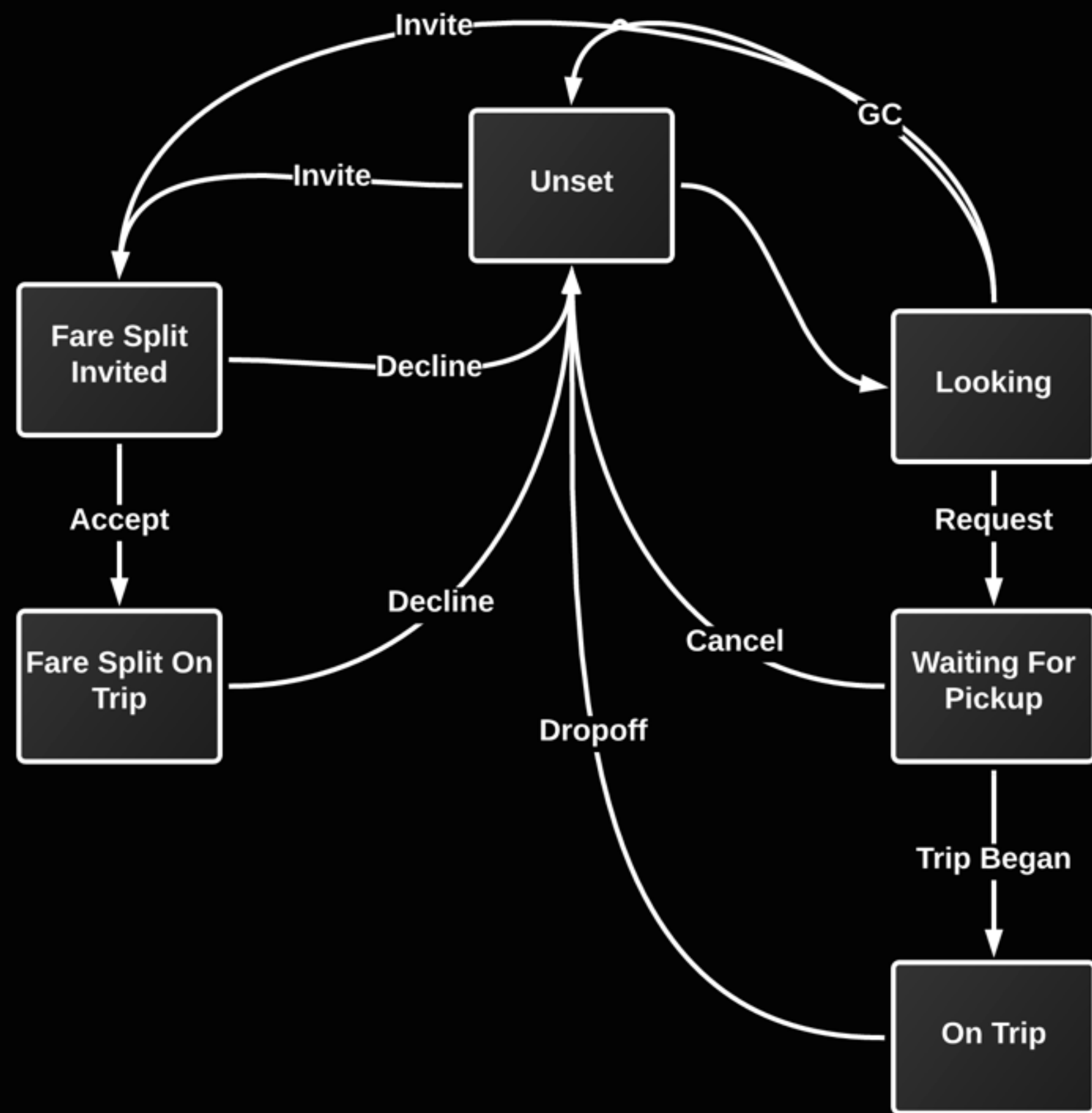
Granular Data

13 driver states

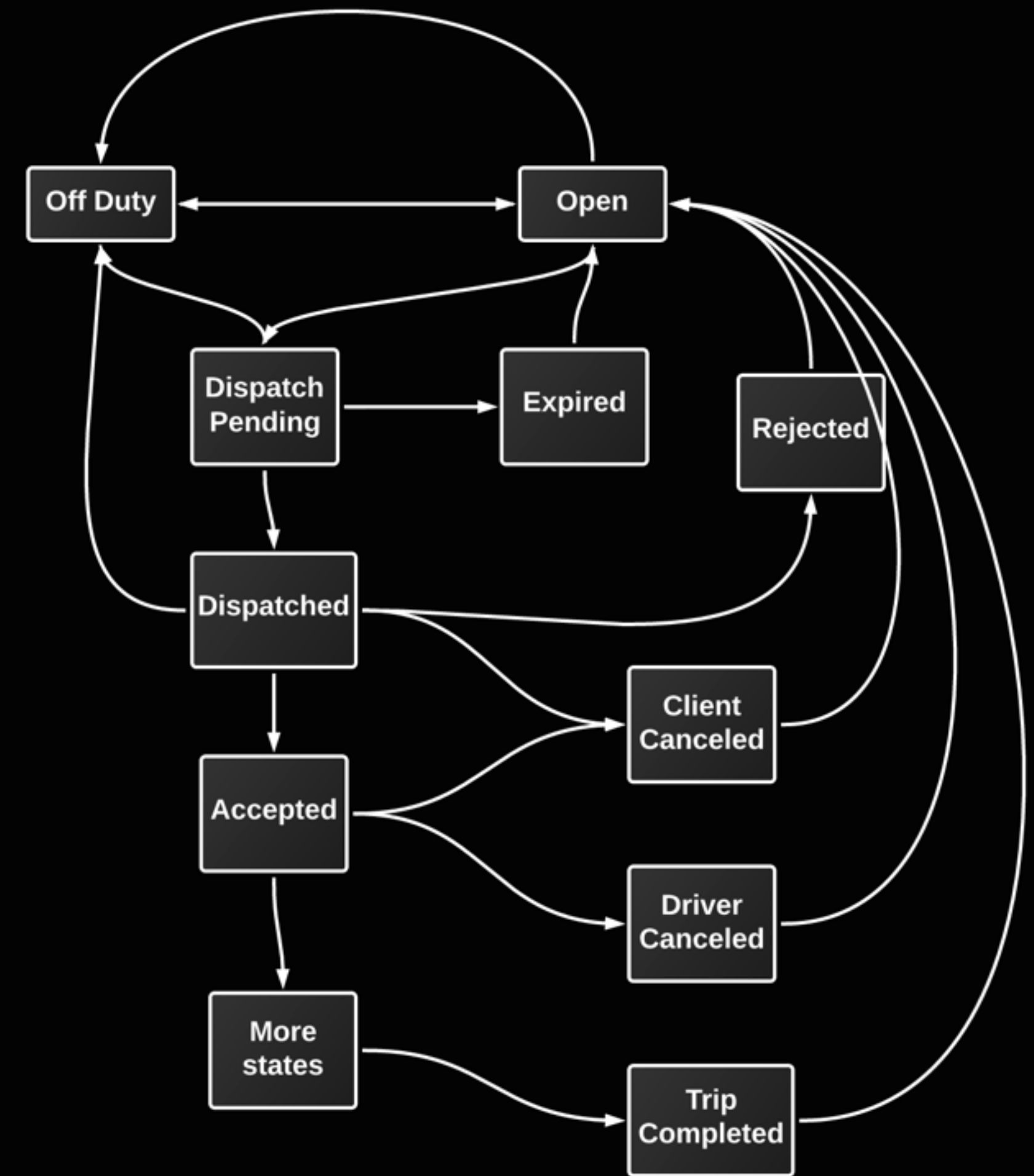


Events - for each action/state

Rider States



Driver States



Granular Data

300 cities



Sample Data Scale

1 day of data: $300 \times 10,000 \times 7 \times 1440 \times 13 = 393 \text{ billion}$
possible combinations



Unknown Query Patterns

Any combination of dimensions

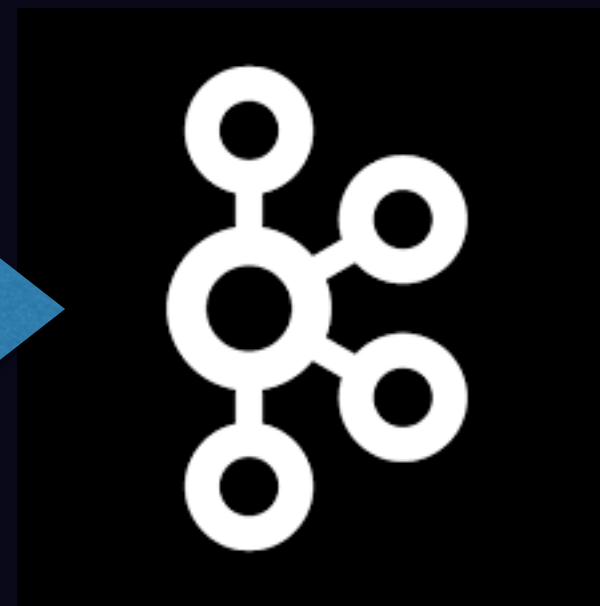
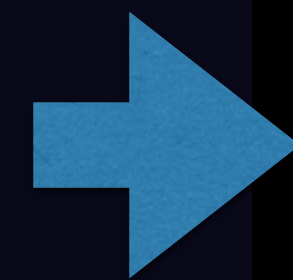
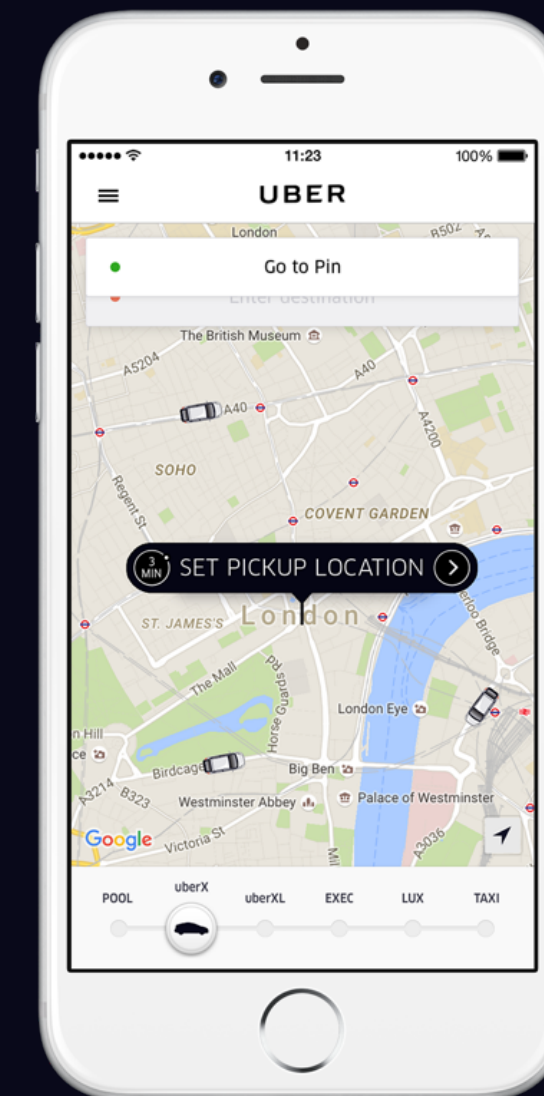
Talk about an example

Variety of Aggregations

- Heatmap
- Top N
- Histogram
- `count()`, `avg()`, `sum()`, `percent()`, `geo`

Large Data Volume

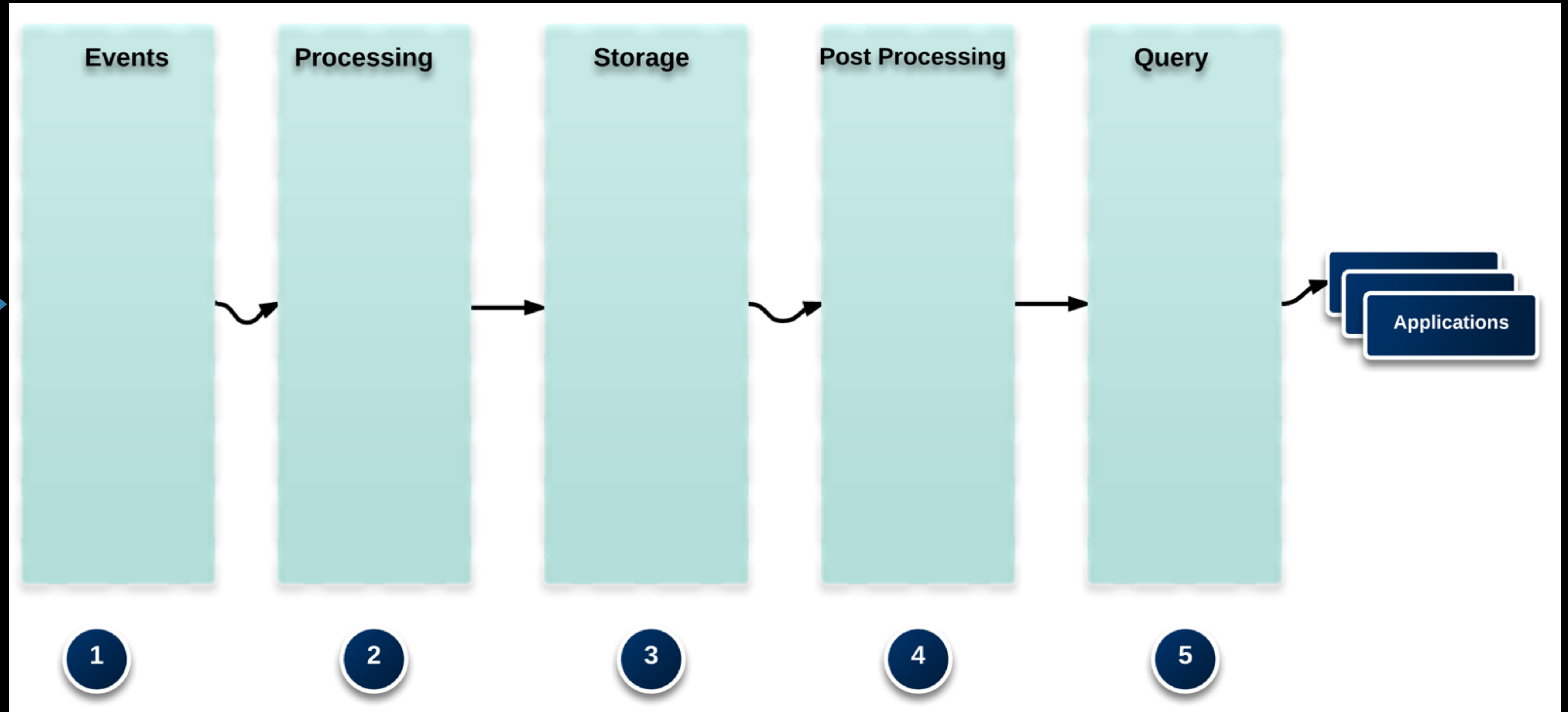
- Hundreds of thousands of events per second, or billions of **events** per day
- At least dozens of fields in each **event**



```
{
  "query": {
    "filtered": {
      "query": {
        "match_all": {}
      },
      "filter": {
        "and": [
          {
            "or": [
              {
                "term": {
                  "dispatch.tags": "driver_accepted"
                }
              },
              {
                "term": {
                  "dispatch.tags": "pickup_requested"
                }
              }
            ]
          },
          {
            "range": {
              "@timestamp": {
                "gte": "2015-01-20T02:52:45.582Z",
                "lte": "2015-01-20T04:59:45.582Z"
              }
            }
          },
          {
            "geo_distance": {
              "distance": "10km",
              "geo": {
                "lat": 37,
                "lon": -122
              }
            }
          }
        ]
      }
    }
  },
  "aggs": {
    "pick_up_counts": {
      "terms": {
        "field": "tags"
      }
    }
  }
}
```


Lets Build a Stream Processing System!

Skeleton Of A System



Event Producing/Consuming

**Match (Dispatch) Services Emit
Billions Of Events Per Topic**

High Scale/Throughput

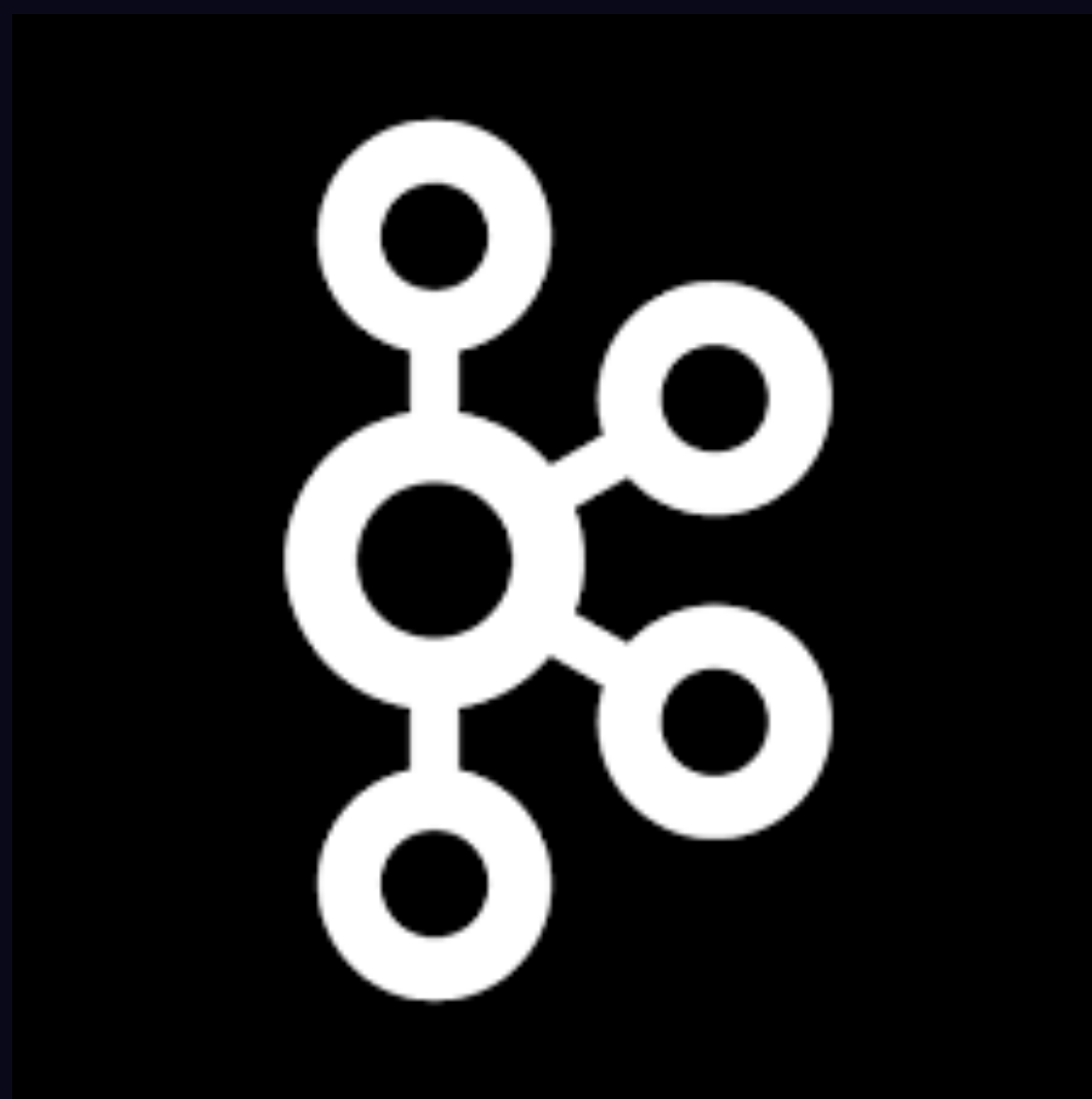
Events Should Be Available In
m-Seconds

Low Latency

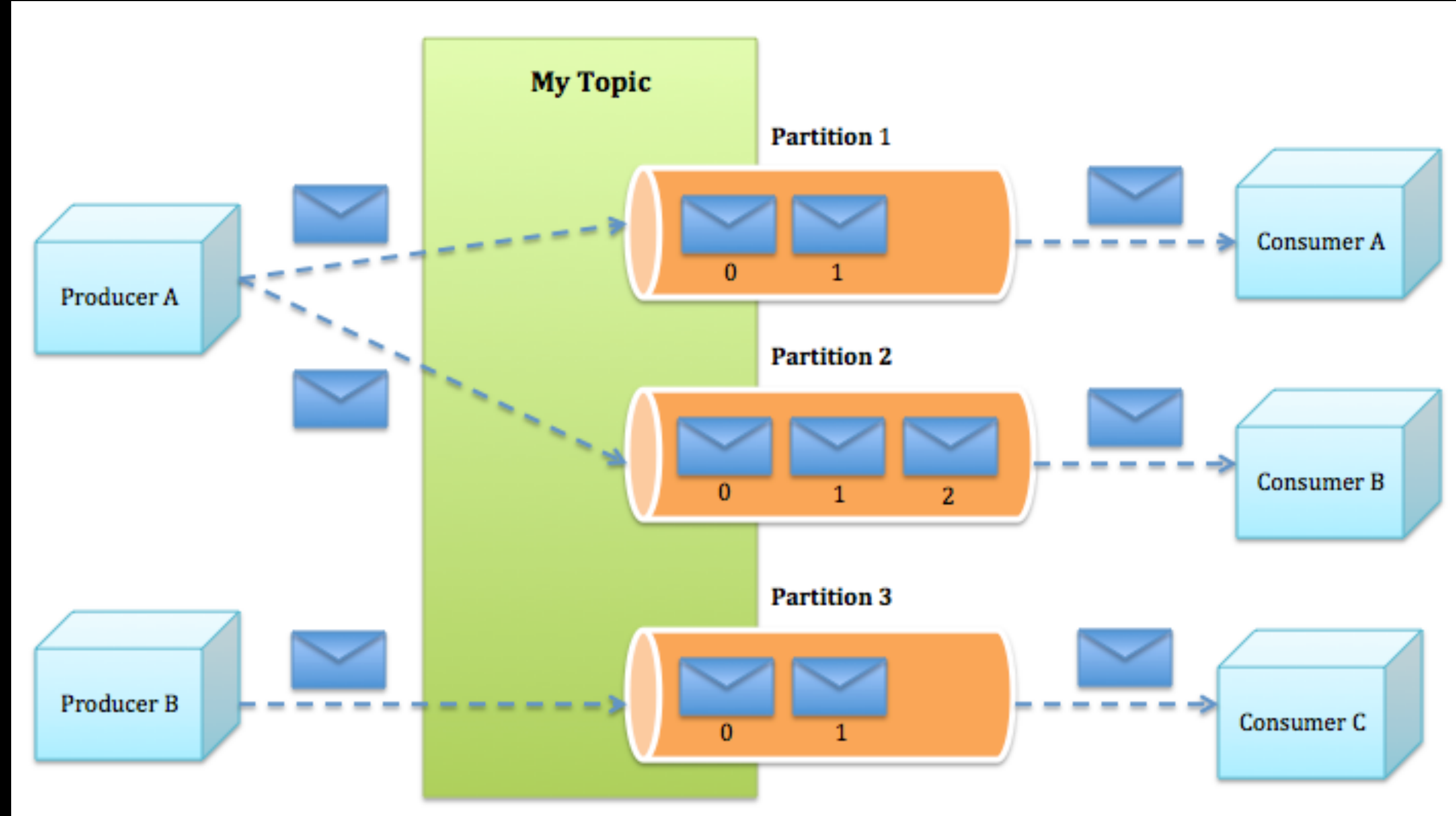
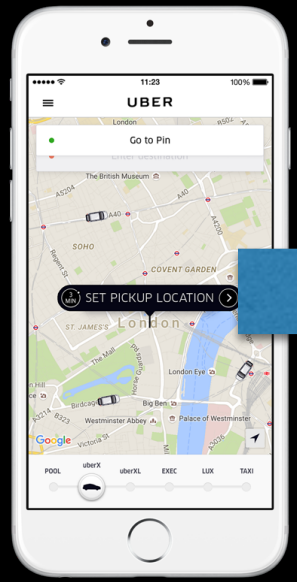
Events Should Rarely Ever Get Lost

Durability

Events Should Be Consumable By Many Consumers



Apache
Kafka



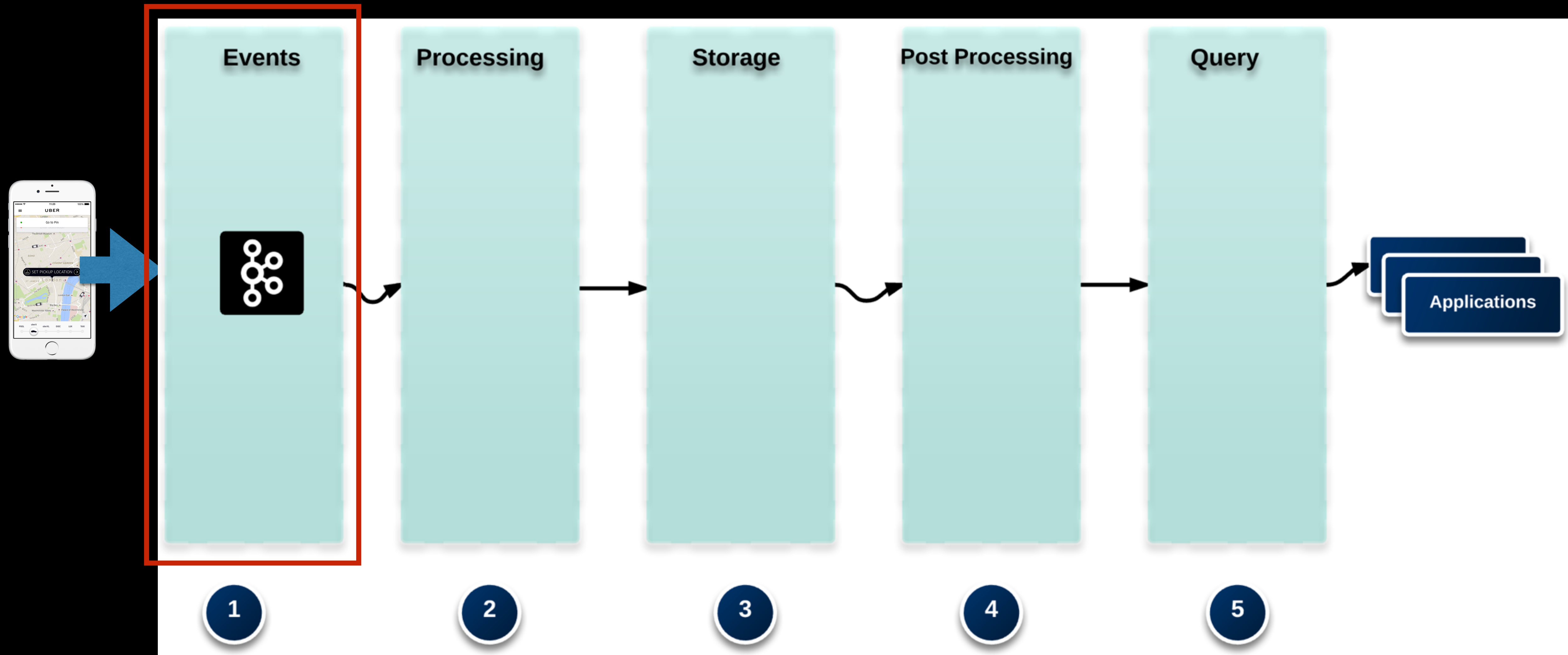


Apache
Kafka

- ✓ High Scalability (Billions of event per day)
- ✓ Durability (no loss)
- ✓ Multiple Consumers
- ✓ Very efficient & low latency

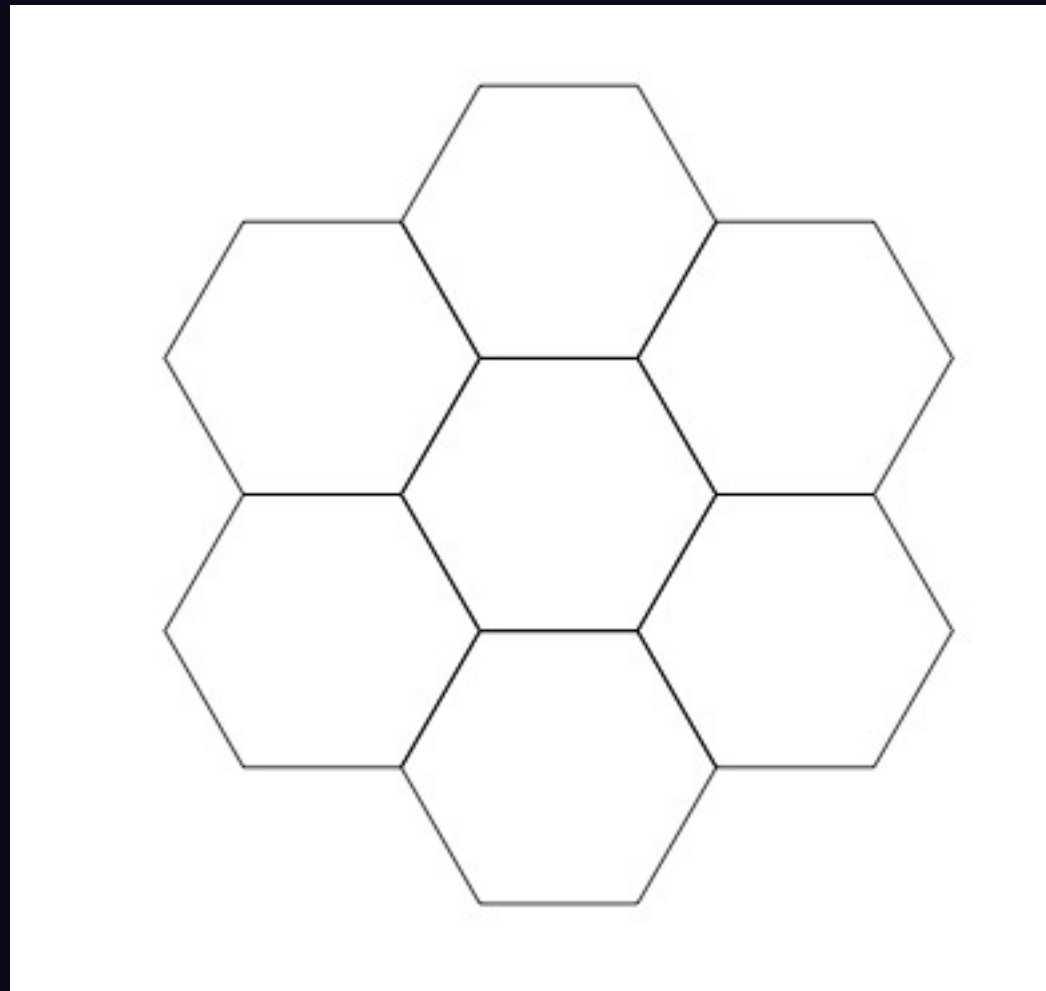
...

Stream Processing System

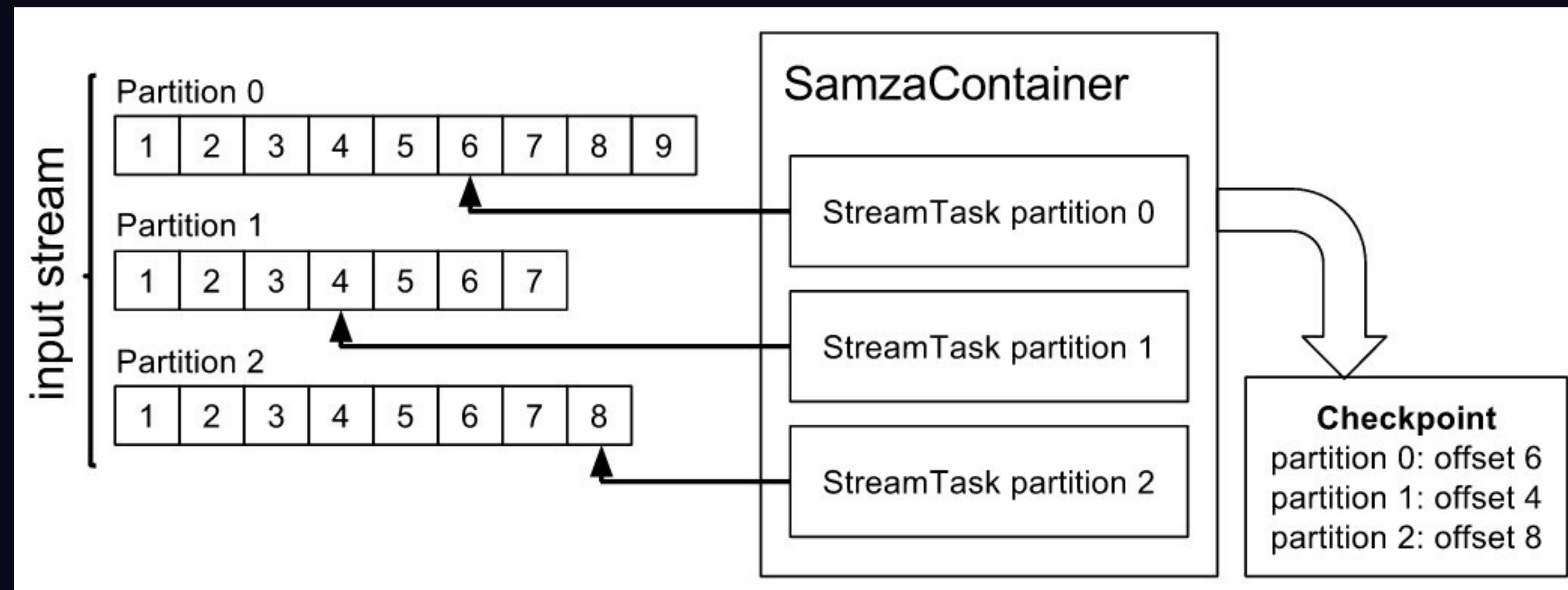


EVENT PROCESSING

Pre-aggregation



Checkpointing



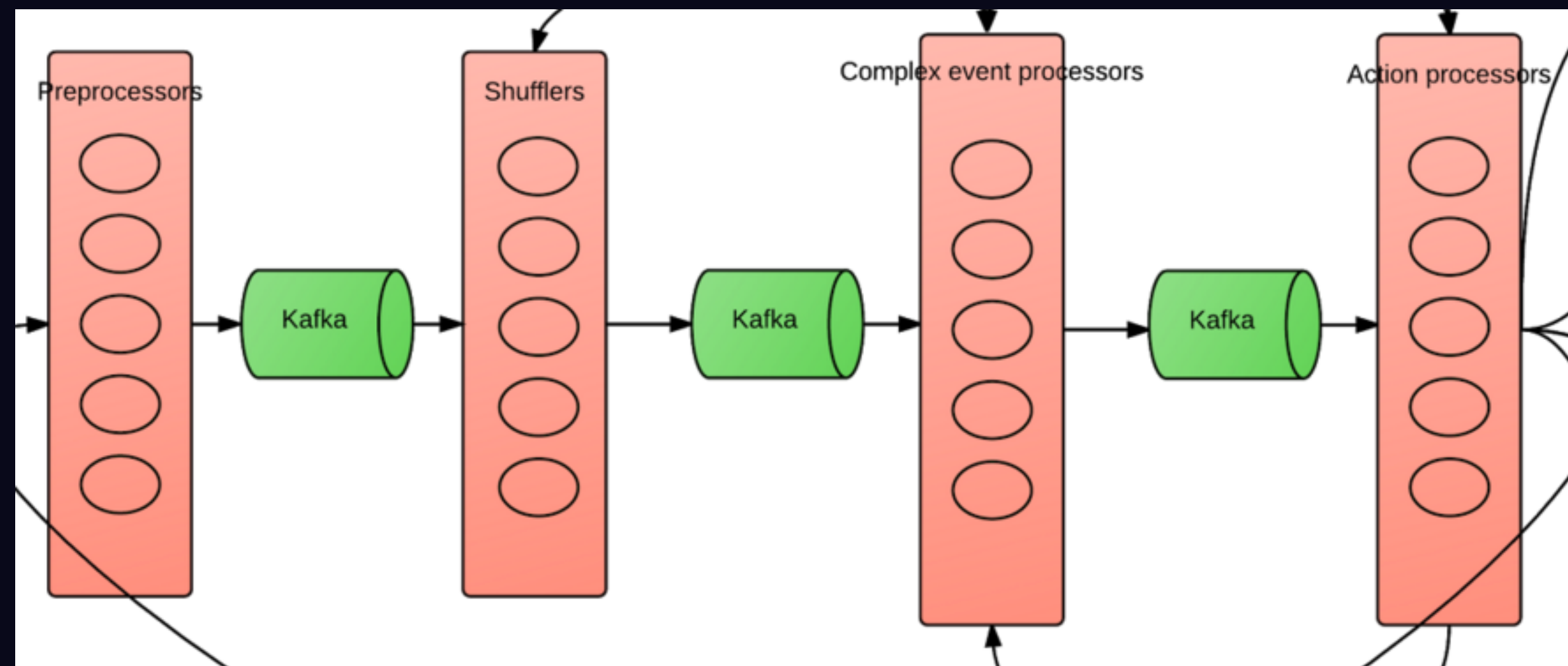
Joining Multiple Streams

Sessionization

Trips on Uber can take from few **minutes** to a few **hours**

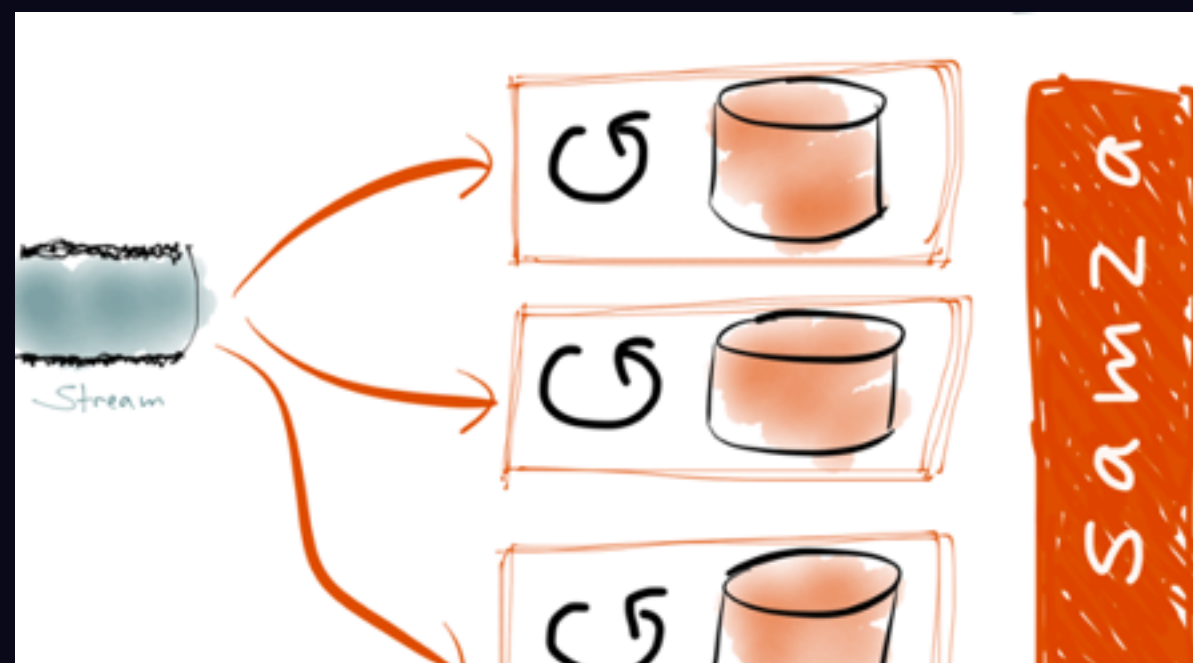
Driver Partners can be “online” from few mins to hours

Multi-Staged Processing

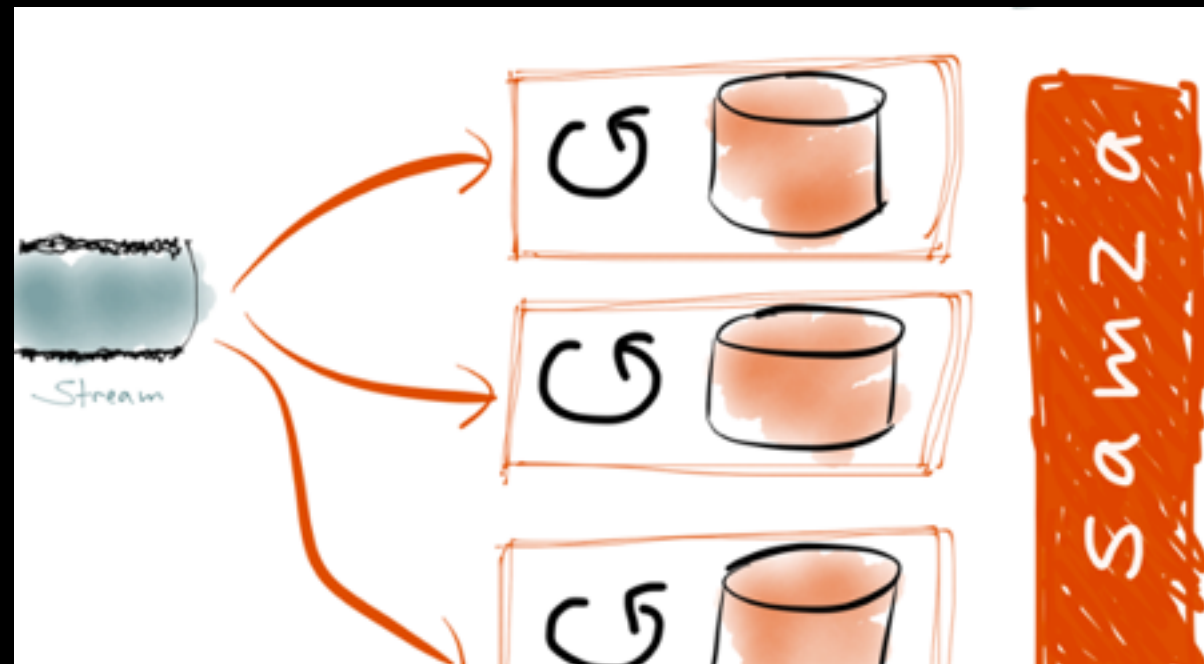


State Management

Apache Samza

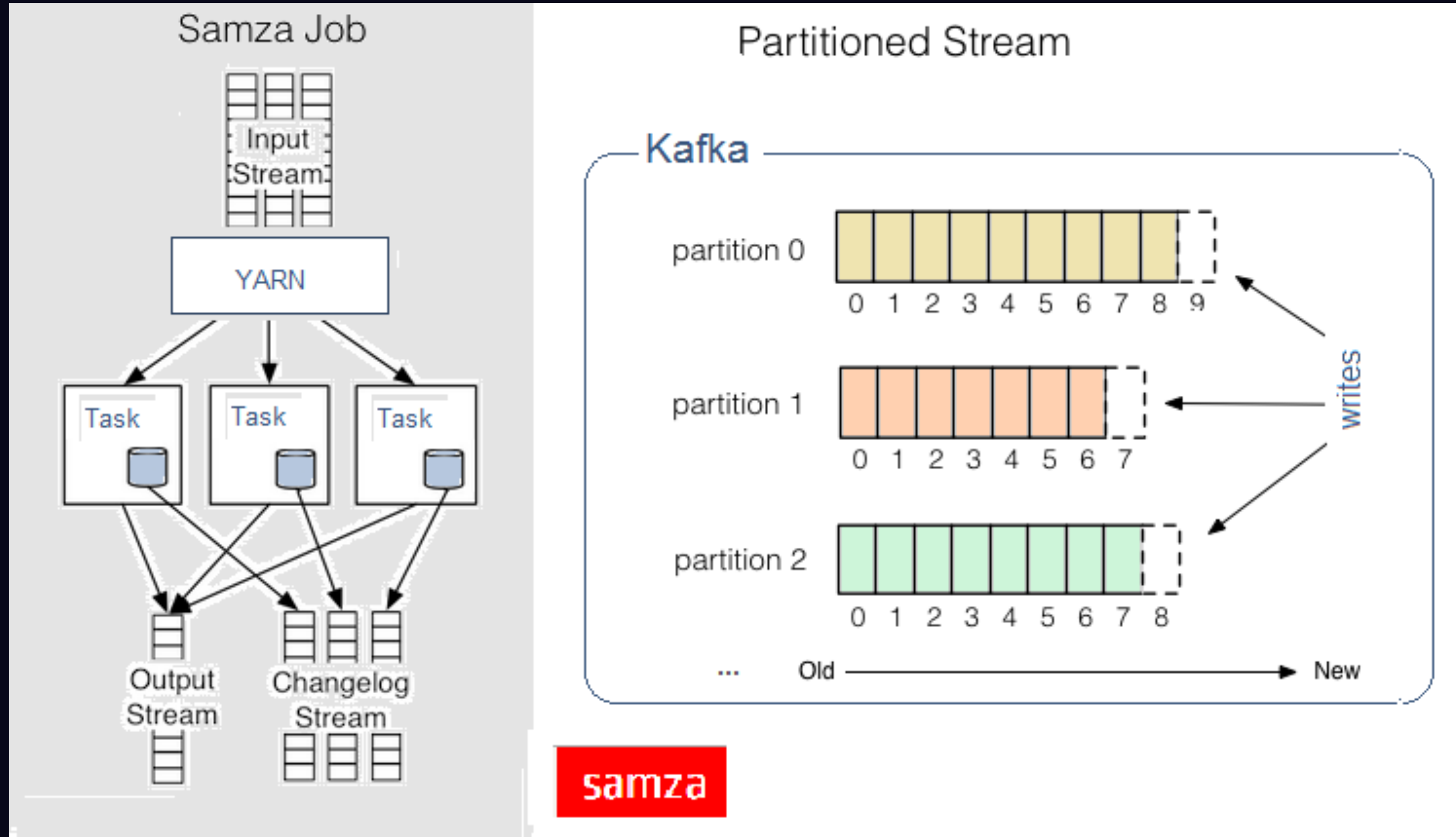


Why Apache Samza?

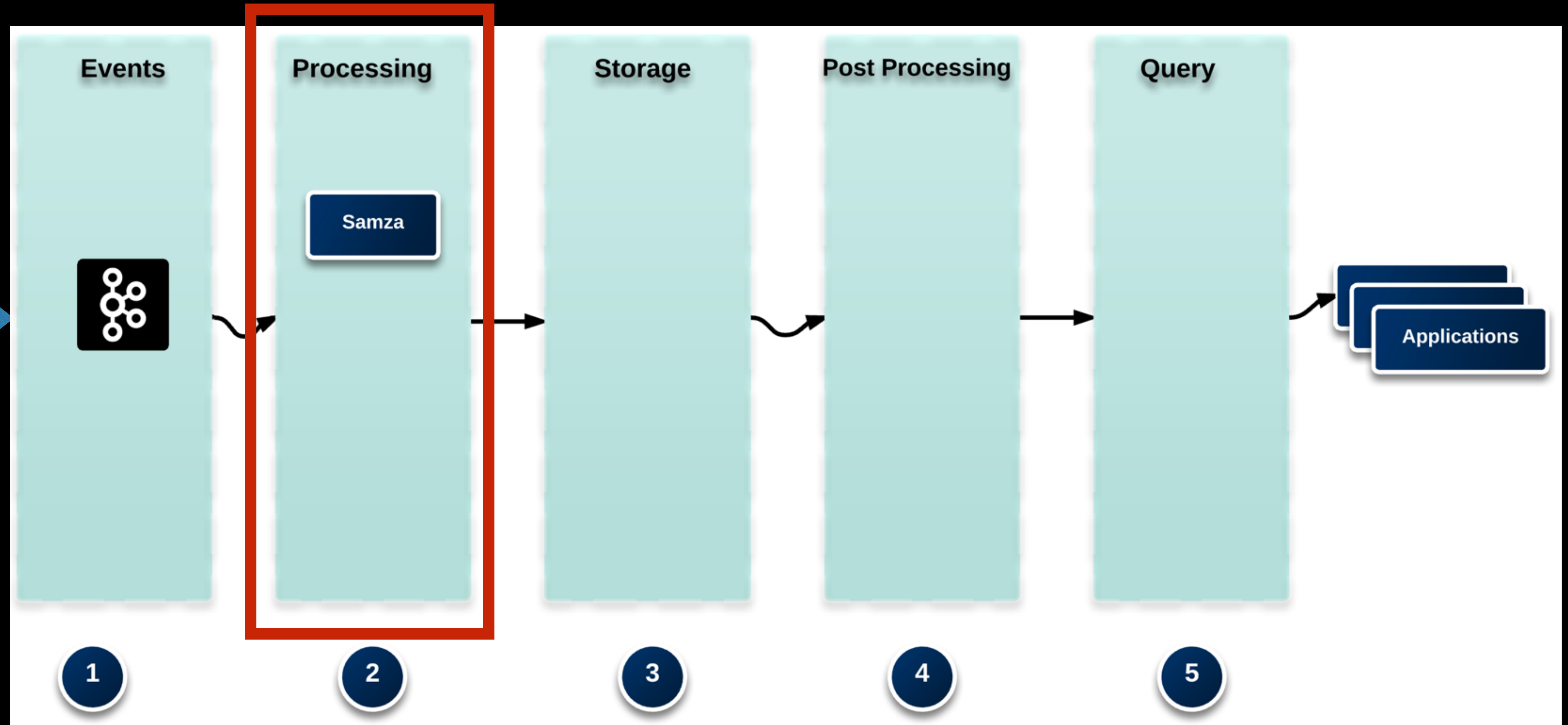


- ✓ DAG on Kafka
- ✓ Excellent integration with Kafka
- ✓ Built in checkpointing
- ✓ Built in state management
- ✓ Highly Scalable
- ✓ Fault tolerant
- ...

Why Apache Samza?



Skeleton Of A System



WAIT!

aka

What About Complex Event Processing?

Continuous Queries

IF

This ->

```

FROM driver_canceled#window.time(10 min)
SELECT clientUUID, count(clientUUID) as cancelCount
GROUP BY clientUUID HAVING cancelCount > 3
INSERT INTO hipchat(room);

```

Then that ->

Actions

HipChat Action

Topic
driver_rejection_repeatedly_SF

HipChat Room
SF cancellation realtime detection by Mystique

If This Then That

A simple SQL-like syntax!

that can take **ACTIONS!!**

In **Real Time!**

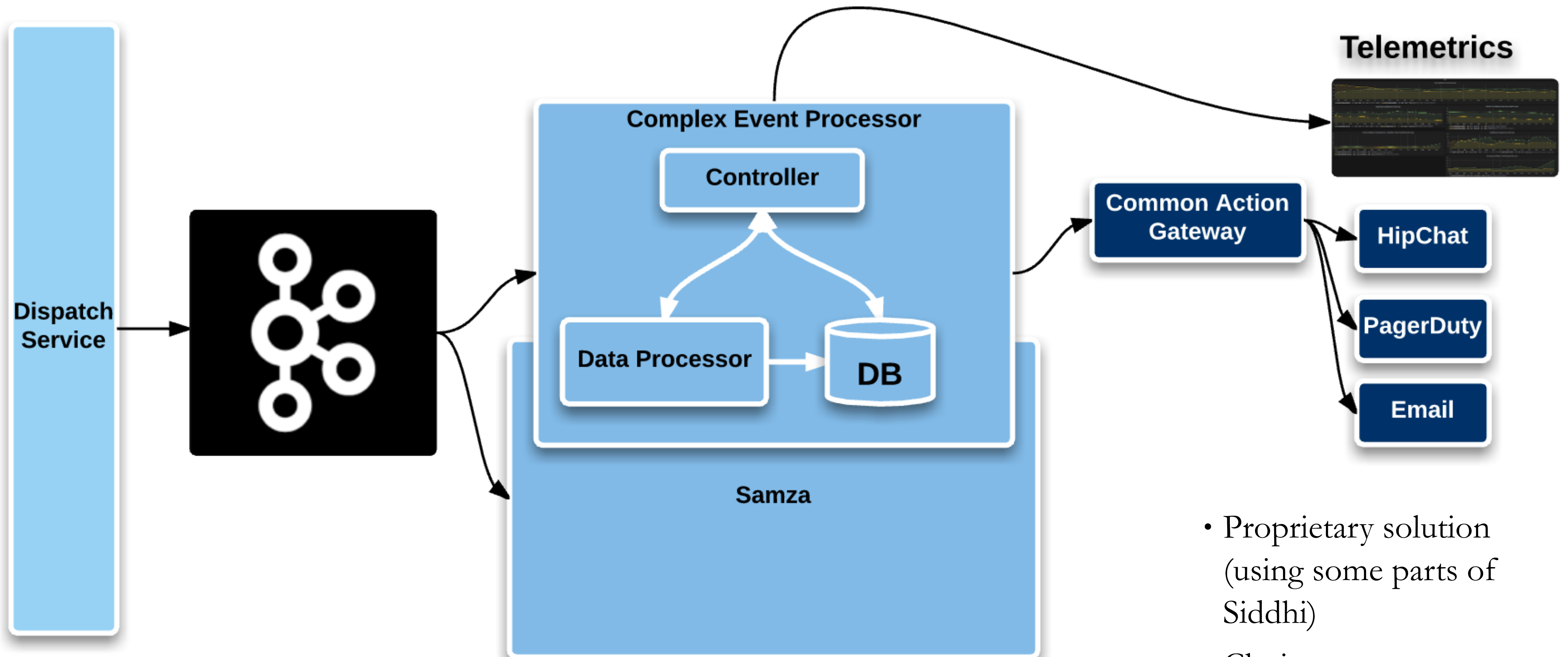
HipChat

SF cancellation realtime detection by Mystique
This is the room topic. Double click to change it.

CAG Bot driver ([redacted]) reject 4 trips in the last 10 minutes

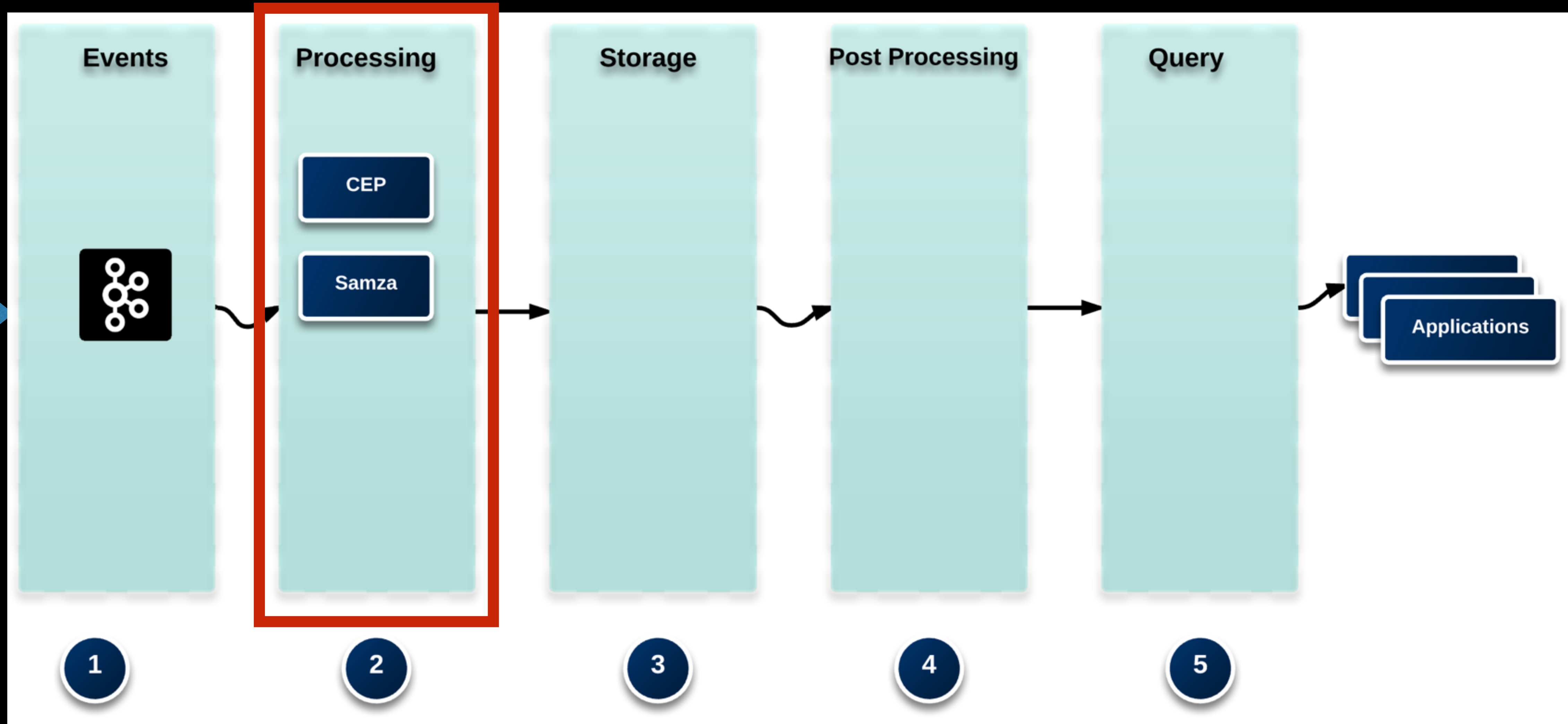
CAG Bot driver ([redacted]) has been repeatedly canceled by clients 2 times in the last 10 minutes

Complex Event Processing



- Proprietary solution (using some parts of Siddhi)
- Choices
 - Esper
 - Siddhi
 - ...

Skeleton Of A System



STORAGE

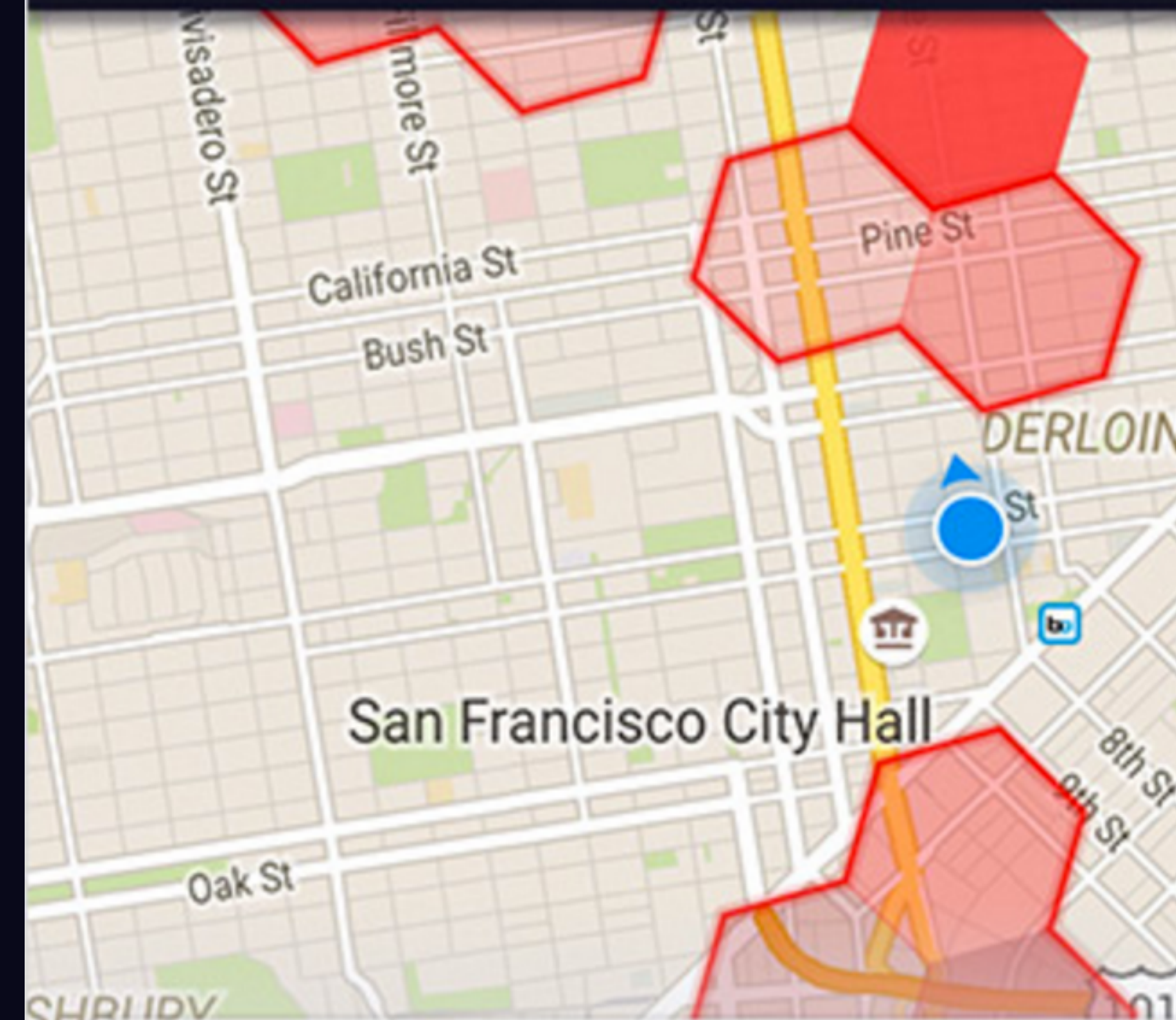
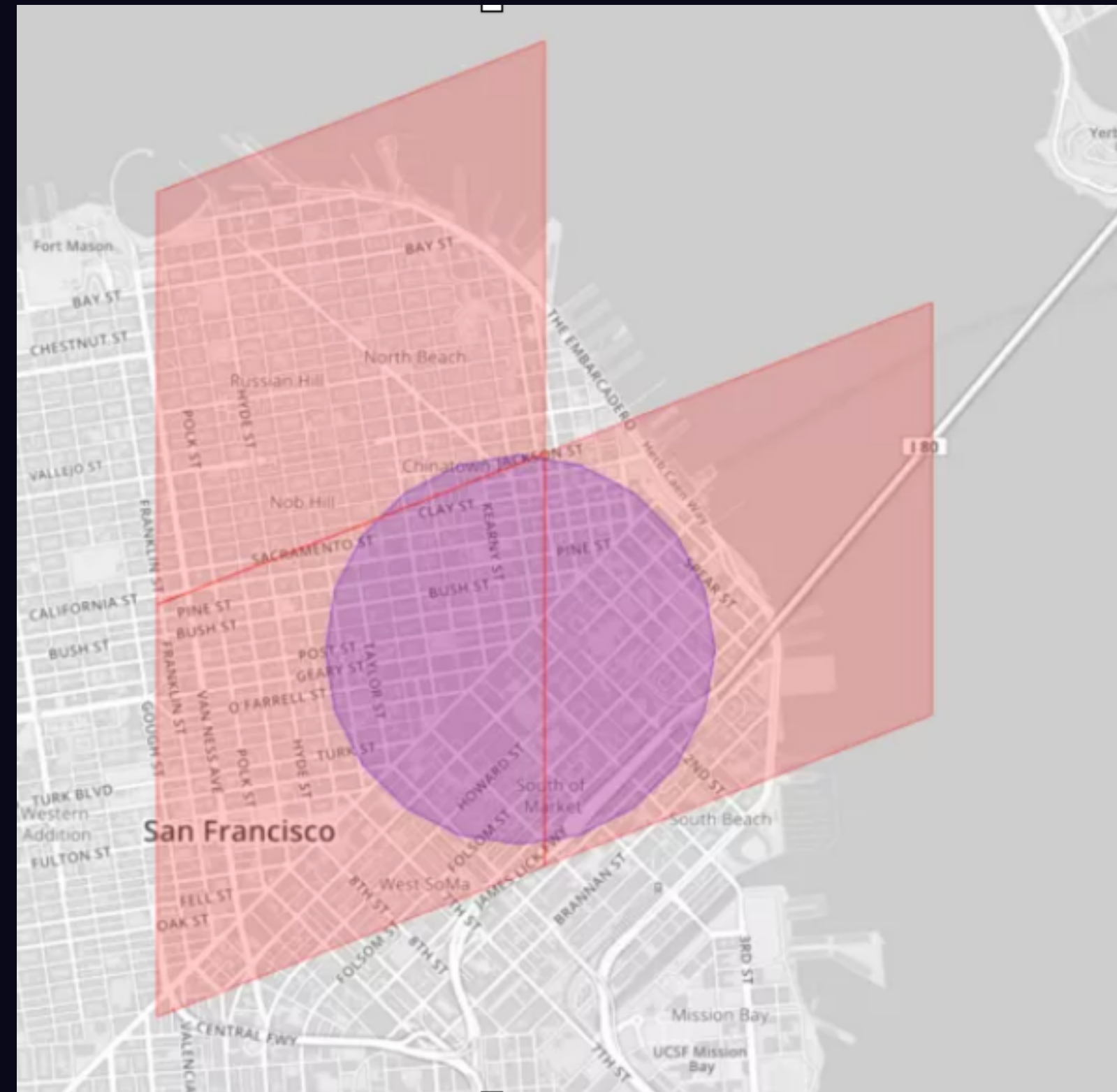
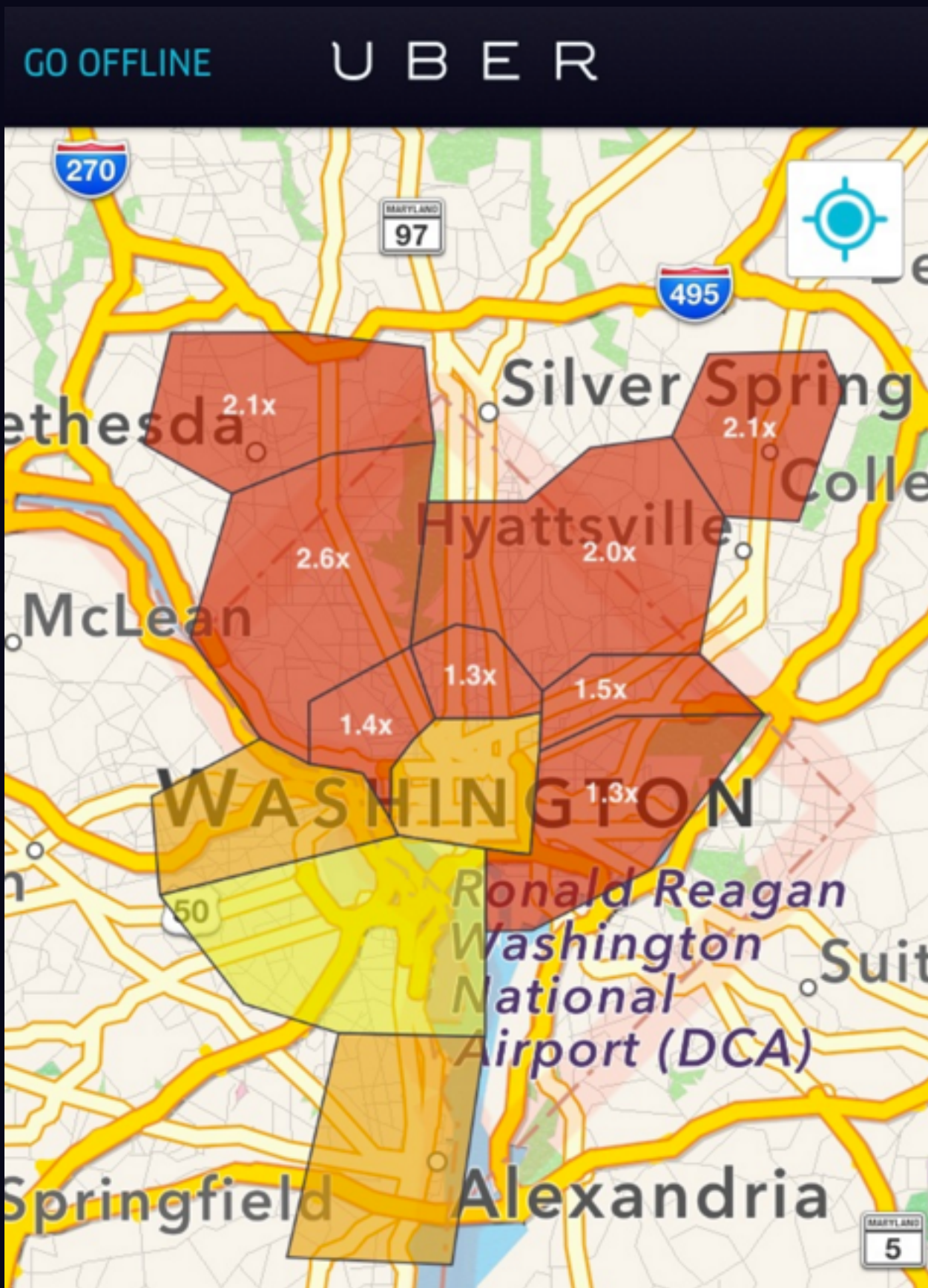
Where are the challenges?

Many Dimensions

Dozens of fields per event

```
{
  "query": {
    "filtered": {
      "query": {
        "match_all": {}
      },
      "filter": {
        "and": [
          {
            "or": [
              {
                "terms": {
                  "dispatch.tags": "driver_accepted"
                }
              },
              {
                "terms": {
                  "dispatch.tags": "pickup_requested"
                }
              }
            ]
          },
          {
            "range": {
              "@timestamp": {
                "gte": "2015-01-20T02:52:45.582Z",
                "lte": "2015-01-20T04:59:45.582Z"
              }
            }
          },
          {
            "geo_distance": {
              "distance": "10km",
              "geo": {
                "lat": 37,
                "lon": -122
              }
            }
          }
        ]
      }
    }
  },
  "aggs": {
    "pick_up_counts": {
      "terms": {
        "field": "tags"
      }
    }
  }
}
```


Different Geo Aggregation



Data Type

- Spatio-Temporal Data

Dimension	Value
state	driver_arrived
vehicle type	uber X
timestamp	13244323342
latitude	12,23
longitude	30,00

Data Query

- OLAP on single-table spatio-temporal data

```
SELECT <agg functions>, <dimensions>  
FROM <data_source>  
WHERE <boolean filter>  
GROUP BY <dimensions>  
HAVING <boolean filter>  
ORDER BY <sorting criterial>  
LIMIT <n>  
DO <post aggregation>
```

Data Query

- OLAP on single-table temporal-spatial data

```
SELECT <agg functions>, <dimensions>  
FROM <data_source>  
WHERE <boolean filter>  
GROUP BY <dimensions>  
HAVING <boolean filter>  
ORDER BY <sorting criterial>  
LIMIT <n>  
DO <post aggregation>
```



```
/driverAcceptanceRate?  
geo_dist(10, [37, 22])&  
time_range(2015-02-04, 2015-03-06)&  
aggregate(timeseries(7d))&  
eq(msg.driverId, 1)
```

Finding the Right Storage System

Minimum Requirements

- OLAP with geospatial and time series support
- Support large amount of data
- Sub-second response time
- Query of raw data

It can't be a KV store

How many keys?

Dimension	Value
A	a
B	b

How many keys?

- All boolean operators: **AND, OR, NOT**

Dimension	Value
A	a
B	b

How many keys?

Dimension	Value
A	a
B	b

- All boolean operators: **AND, OR, NOT**
 - A and (not B)
 - B and (not A)
 - A or B
 - not (A or B)

How many keys?

Dimension
A
B

- $\{A\}$
- $\{B\}$
- $\{A, B\}$
- $\{\}$

Challenges to KV Store

Pre-computing all keys is $O(2^n)$ for both space and time

e.g. $2^{10} = 1024$

Sure, K-V Stores Are Fast

Being Fast Is Not Enough

Number of cars per hexagon in a city => **18,000** lookups

Mean latency: **1ms**

99.99%-ile latency: **2s**

Failure rate: **0.001%**

Being Fast Is Not Enough

Probability that a request will experience 99.99%-ile: $(1 - 0.9999^{18000}) \times =$
83%

Probability that a single query will succeed: $(1 - 0.00001)^{18000} =$ **84%**

Lesson: Don't play the probability game

Can we use a relational database?

Challenges to Relational DB

- Managing multiple indices is painful
- Scaling Is Hard

We Need A System That Supports

- Fast scan
- Arbitrary boolean queries
- Raw data
- Wide range of aggregations

A System That Optimizes

- Data segmentations
- Parallel queries
- Bitset-based set operations
- Index compressions
- Fast range queries

Is there such a system?



elasticsearch.

Elasticsearch



elasticsearch.

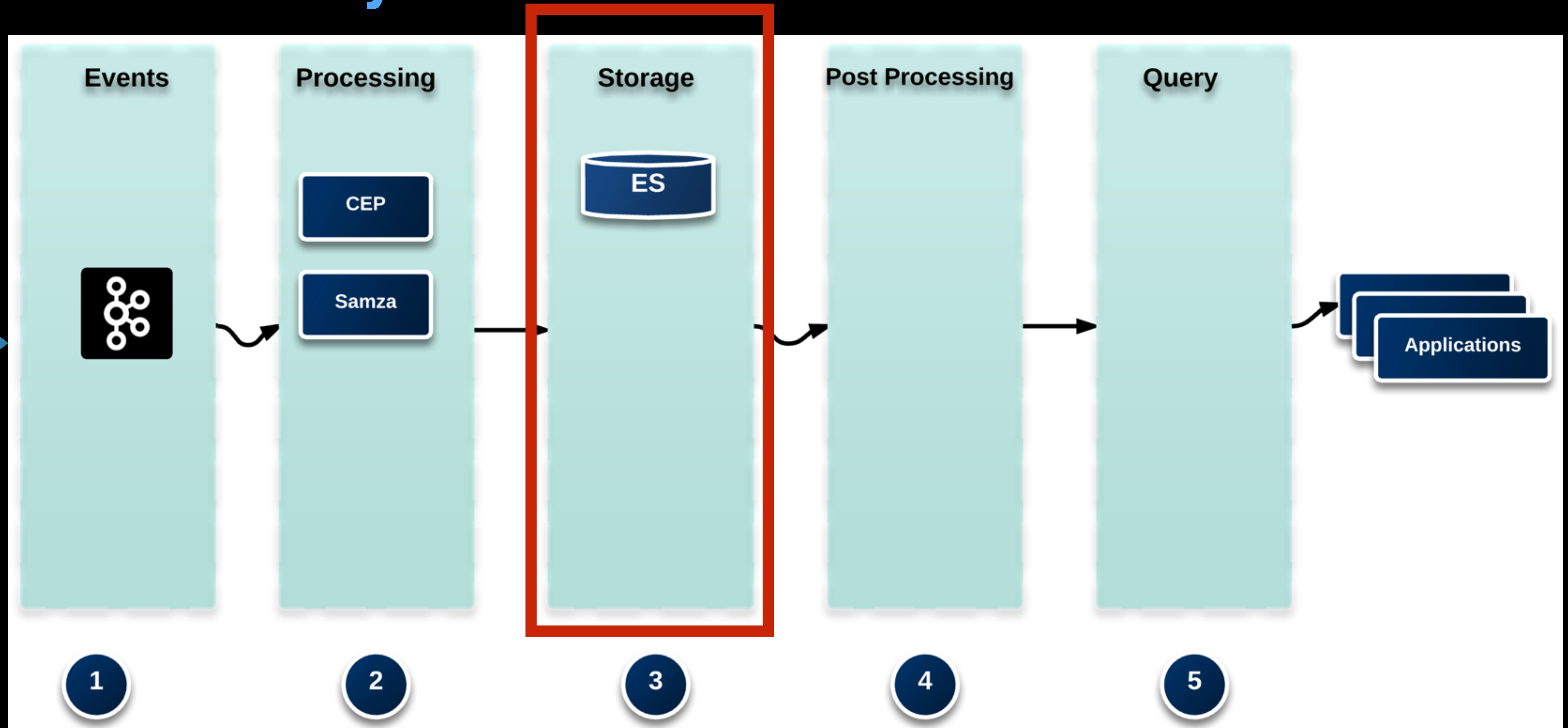
```
{
  "query": {
    "filtered": {
      "query": {
        "match_all": {}
      },
      "filter": {
        "and": [
          {
            "or": [
              {
                "term": {
                  "dispatch.tags": "driver_accepted"
                }
              },
              {
                "term": {
                  "dispatch.tags": "pickup_requested"
                }
              }
            ]
          },
          {
            "range": {
              "@timestamp": {
                "gte": "2015-01-20T02:52:45.582Z",
                "lte": "2015-01-20T04:59:45.582Z"
              }
            }
          },
          {
            "geo_distance": {
              "distance": "10km",
              "geo": {
                "lat": 37,
                "lon": -122
              }
            }
          }
        ]
      }
    },
    "aggs": {
      "pick_up_counts": {
        "terms": {
          "field": "tags"
        }
      }
    }
  }
}
```

Highly Efficient Inverted-Index For Boolean Query

Built-in Distributed Query

Fast Scan with Flexible Aggregations

Skeleton Of A System

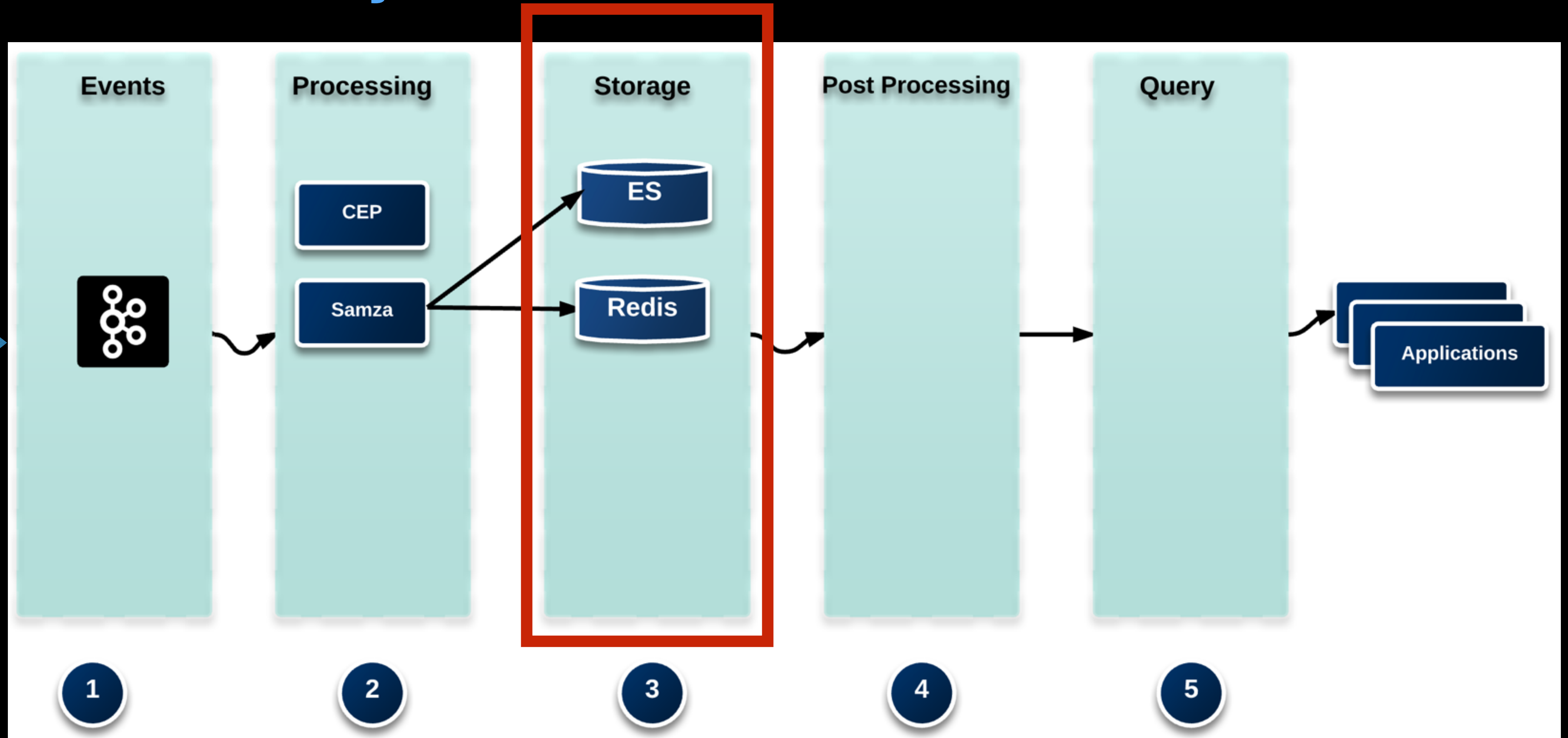


What About Really Fast Lookups?



redis

Skeleton Of A System



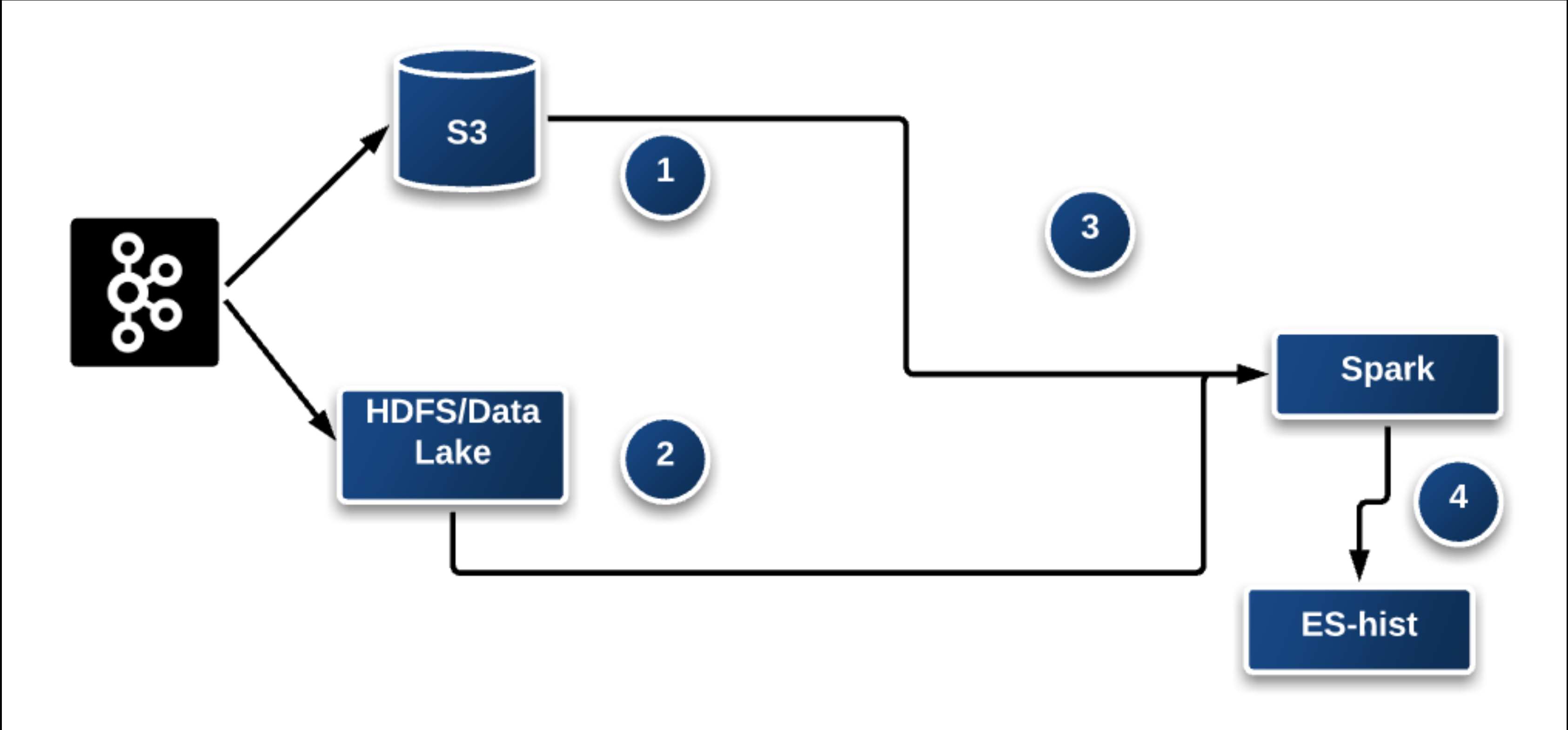
What If there is data corruption?

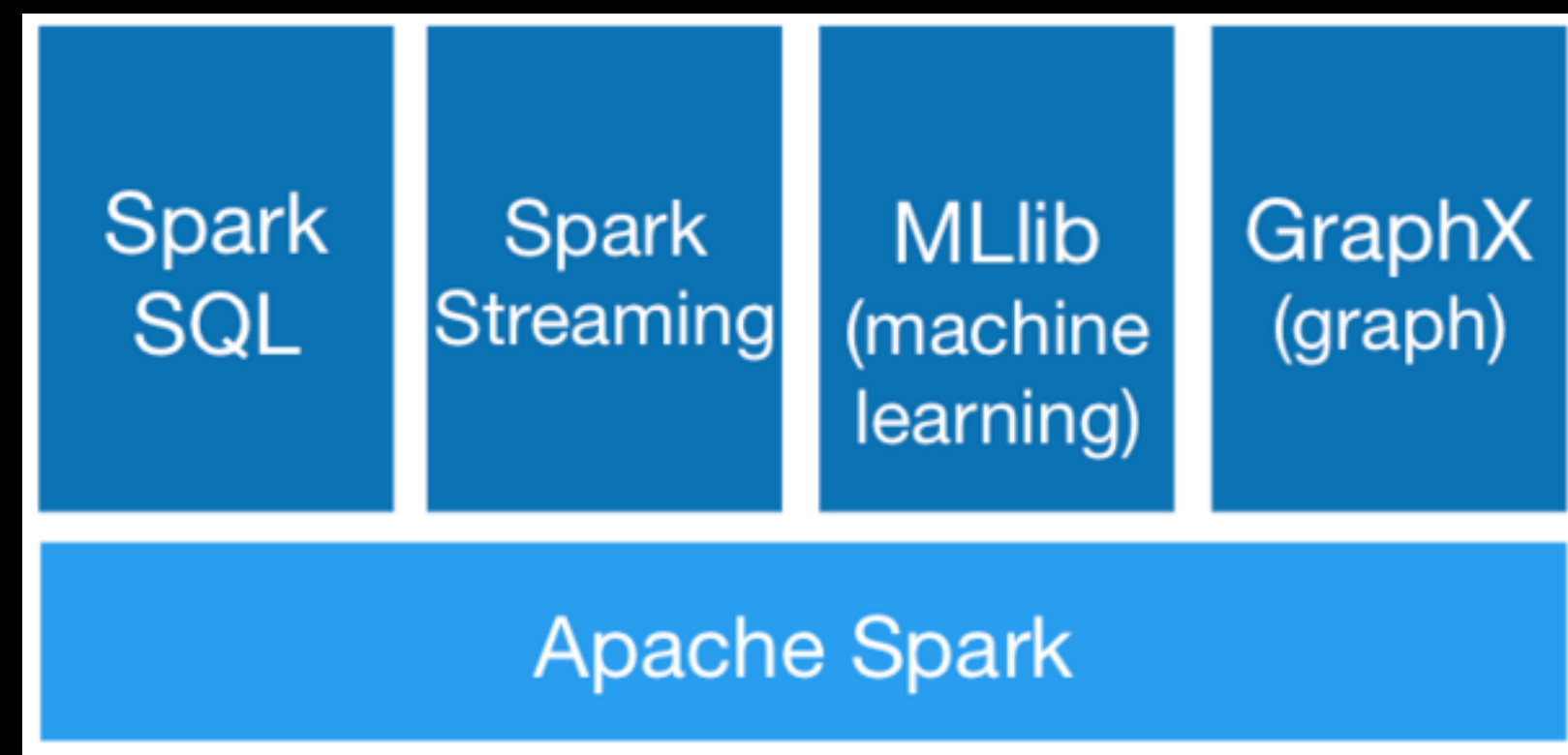
or

**There was a bug in the Event
Processing Job?**

We Would Want To Backfill Data!

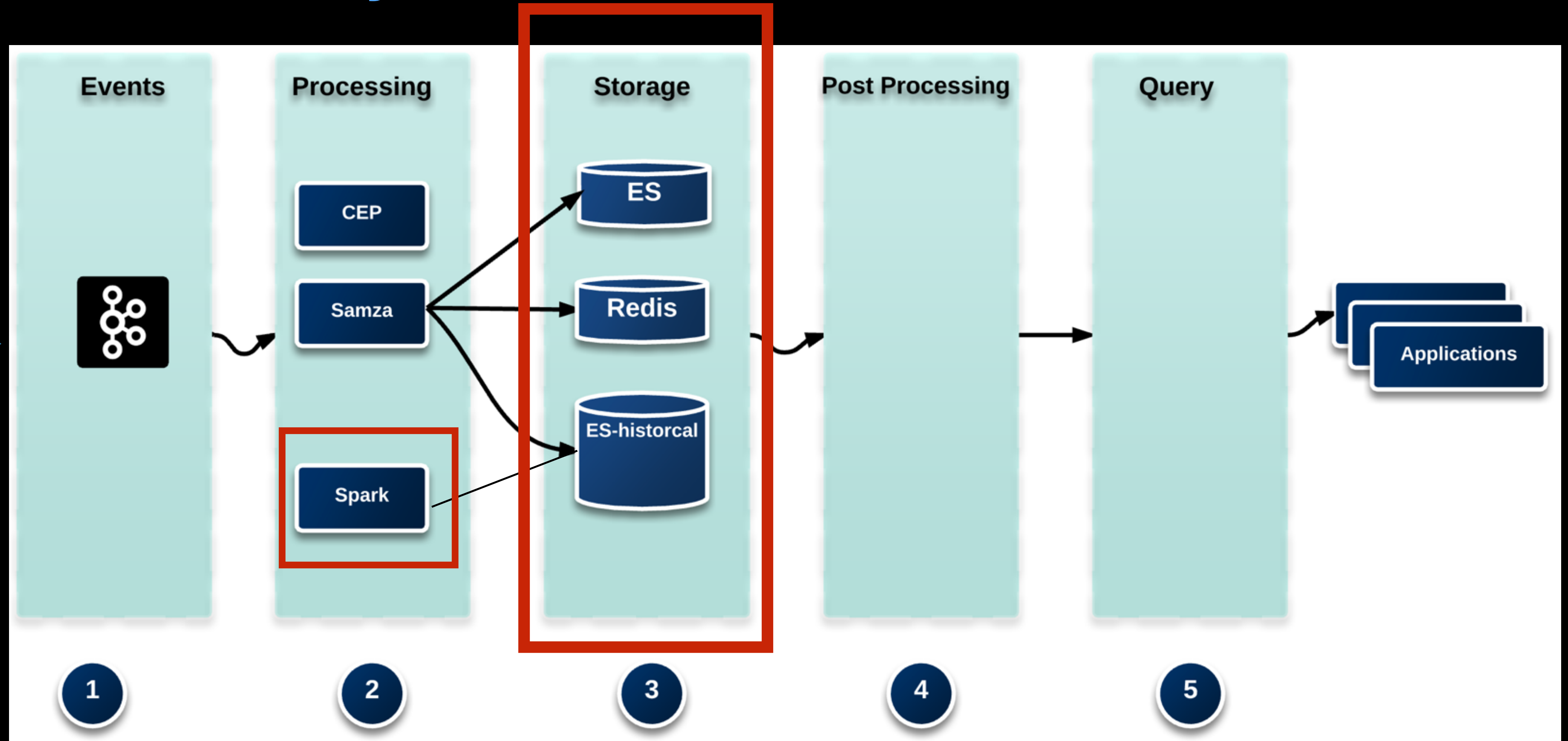
Backfill Data





- ✓ HDFS or S3 ..
- ✓ “exactly once” processing**
- ✓ ML support (for our Data Scientists)
- ✓ Batch and Streaming (well, micro batching) support
- ...

Skeleton Of A System

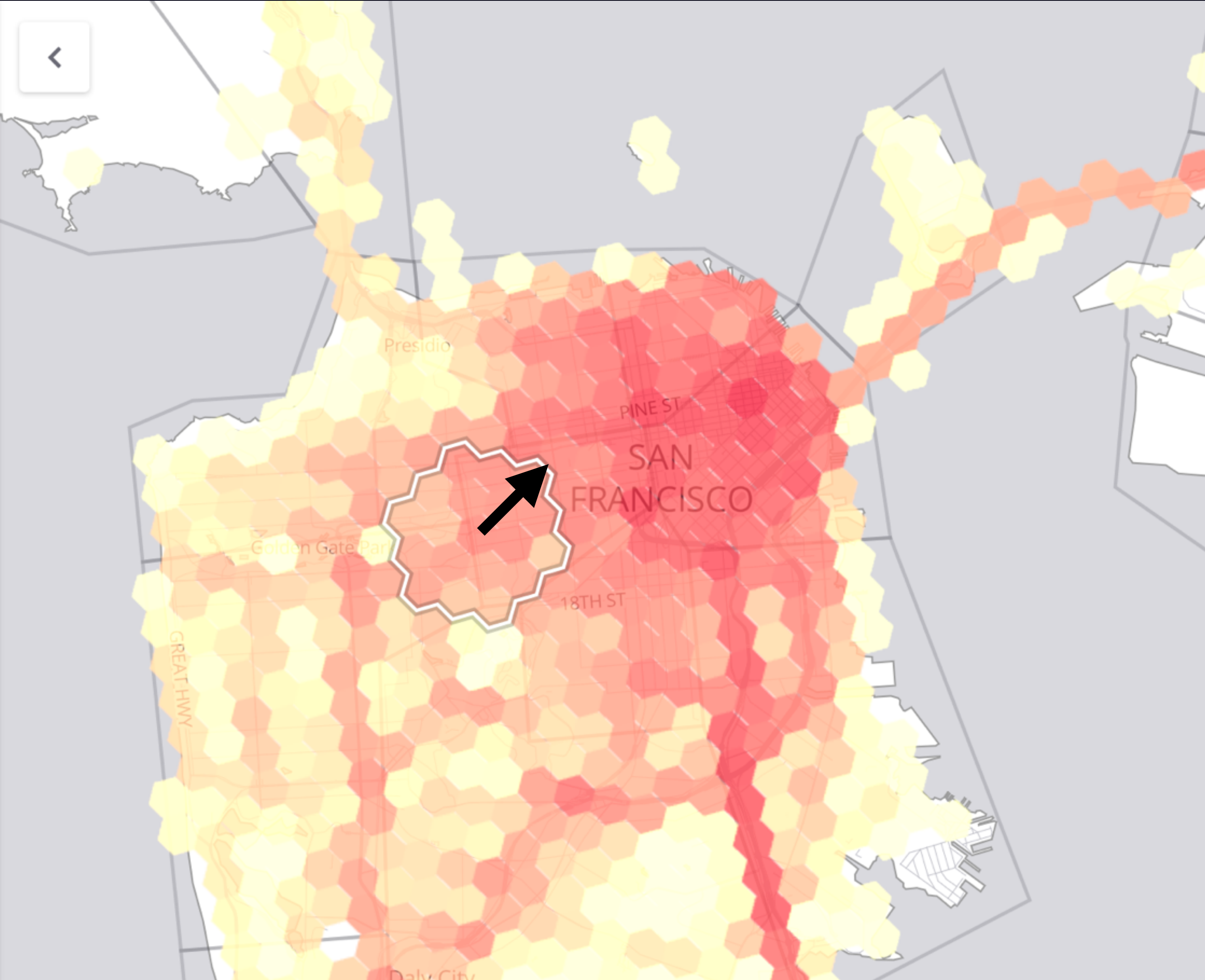
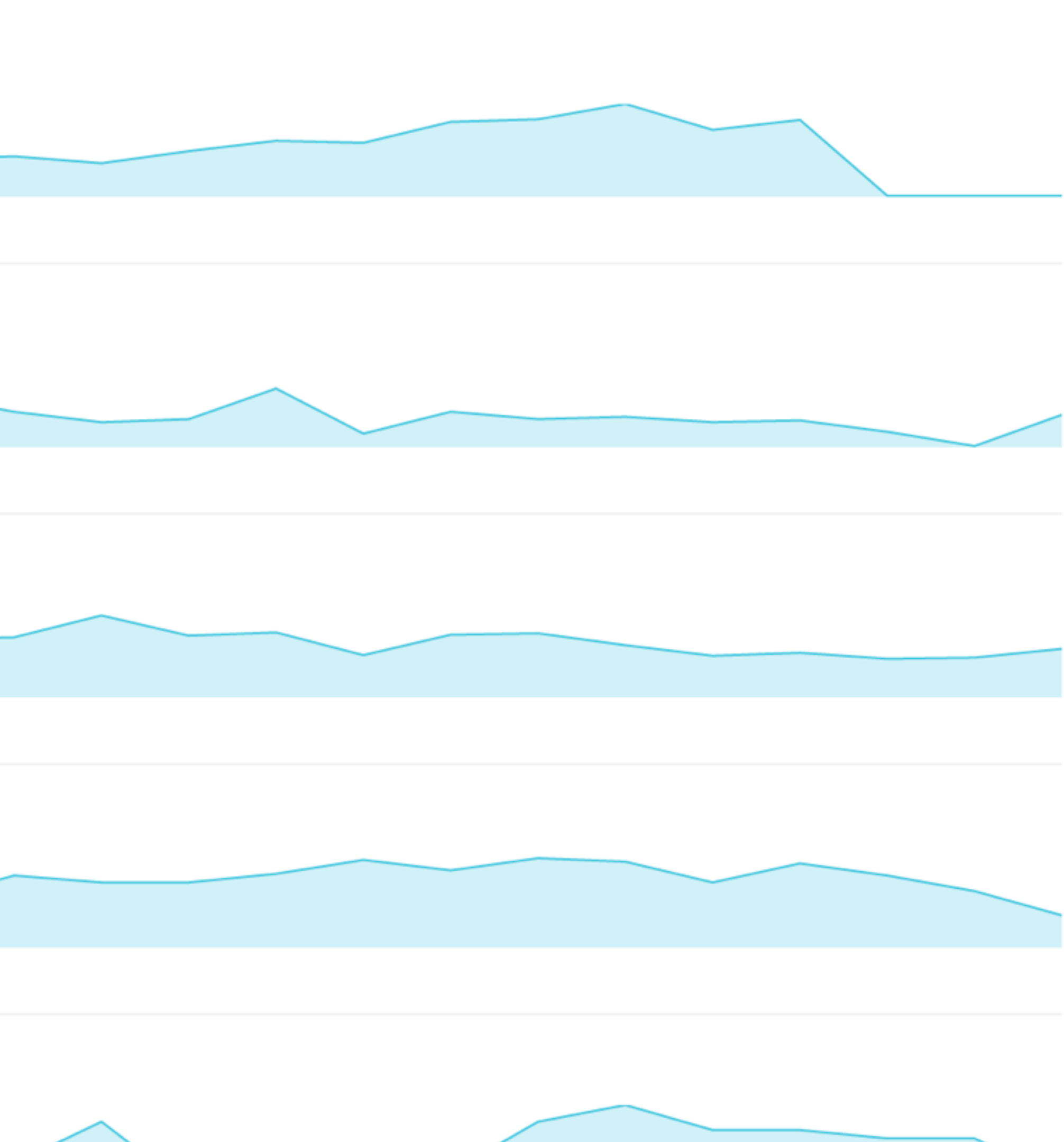


Query Pipelining

Aggregation By Ring Size (Hexagons)

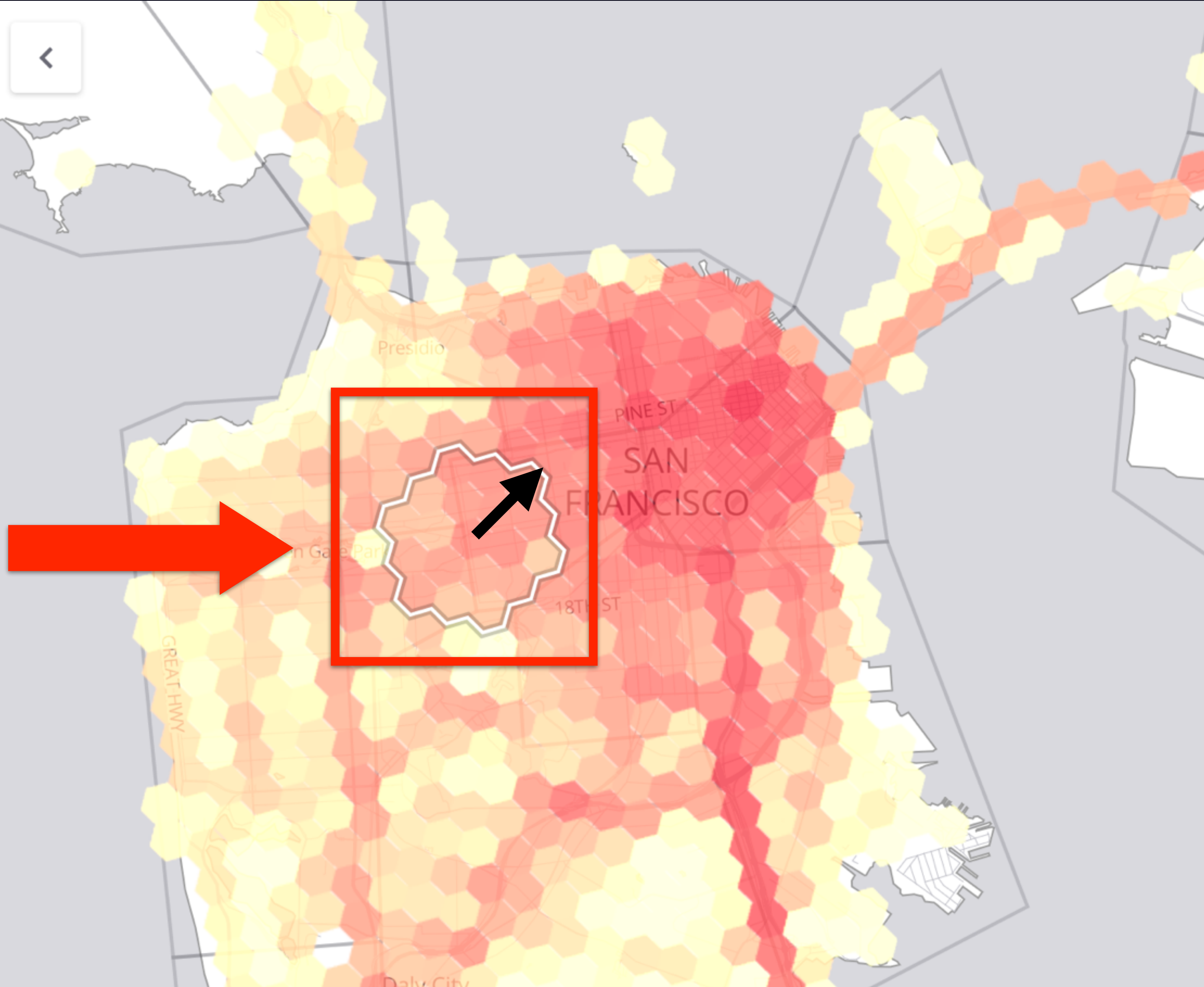
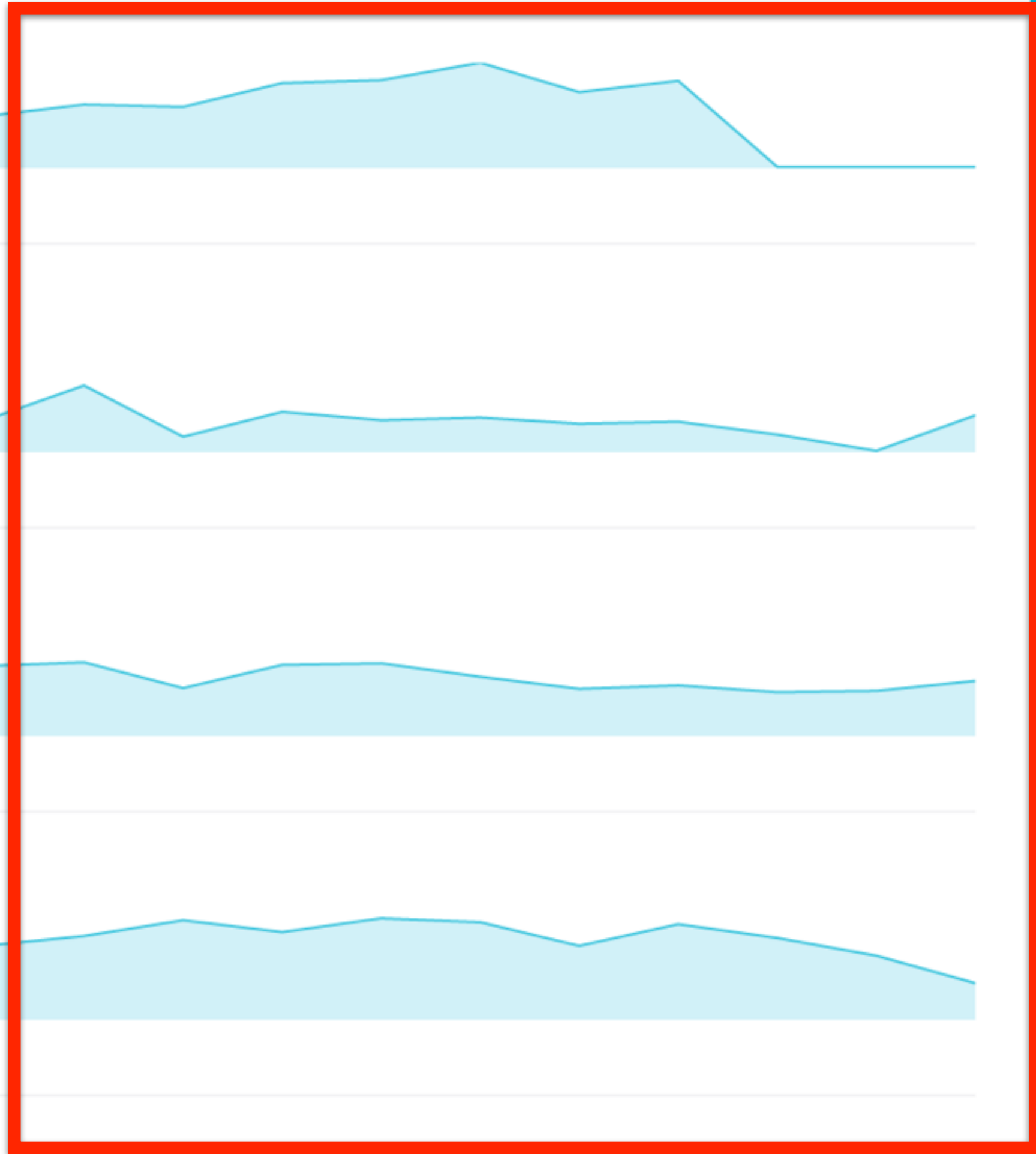
SELECTION RADIUS

0 1 2 3 4 5 6 7



SELECTION RADIUS

0 1 2 3 4 5 6 7



Results Transformation and Smoothing



Scale

10,000 hexagons in a city

Scale

331* neighboring hexagons to look at

*For a ring size of 9

Scale

$331 \times 10,000 = 3.1$ Million Hexagons to
Process for a Single Query

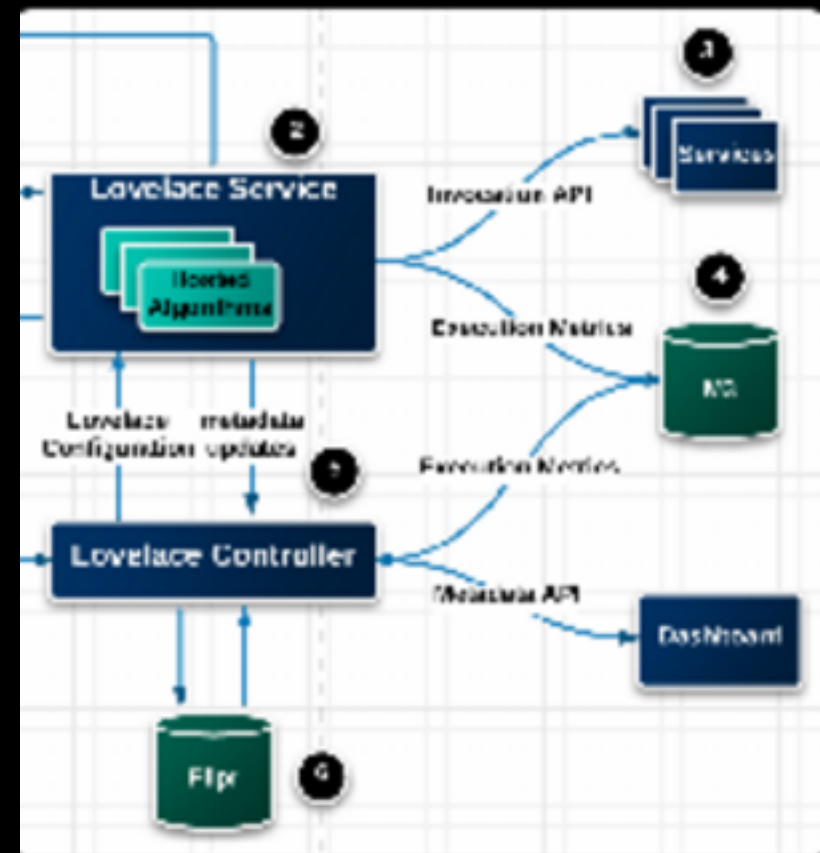
Scale

99%-ile Processing Time: 100ms

Requirements

- Highly parallelized execution
- Pipelining

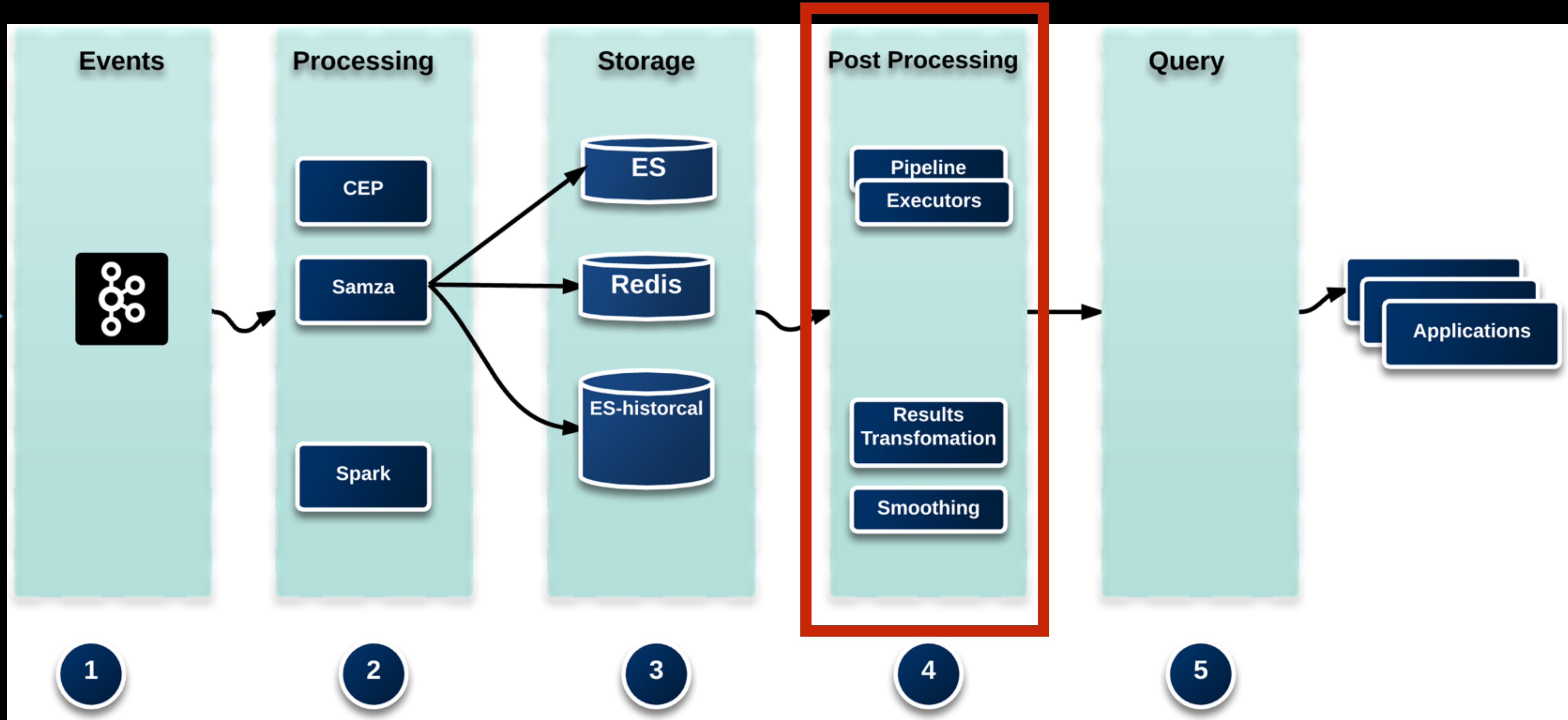
Is there an Open source solution ? :-)



✓ any out-of-box solution?

...

Stream Processing Flow



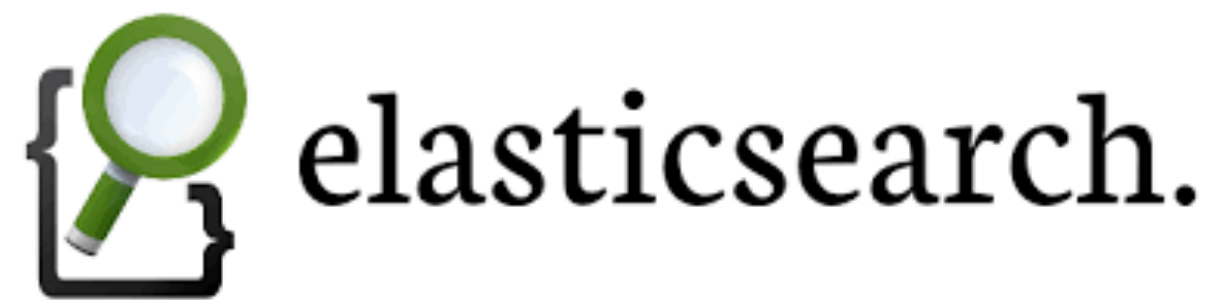
Querying

Elasticsearch Query Can Be Complex

```
/driverAcceptanceRate?  
geo_dist(10, [37, 22])&  
time_range(2015-02-04, 2015-03-06)&  
aggregate(timeseries(7d))&  
eq(msg.driverId, 1)
```

```
    },  
    },  
    "aggs": {  
      "pick_up_counts": {  
        "terms": {  
          "field": "tags"  
        }  
      }  
    }  
  }  
}
```

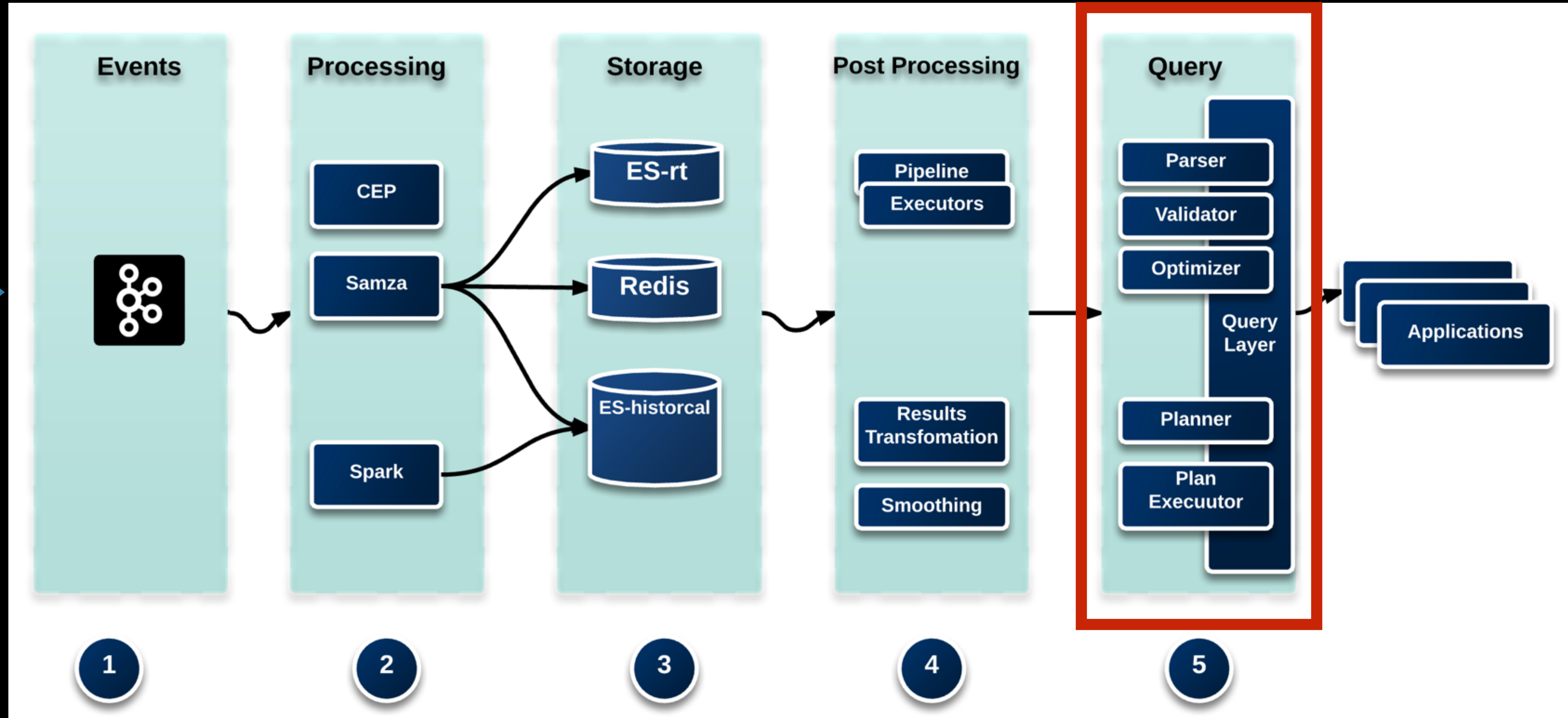
Also, we need to stitch data from
ES Realtime, Redis, ES Historical &
any other DBs we add in the future



Optimizing Queries

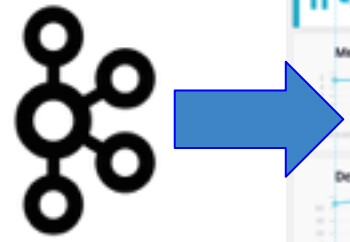
- Pipelining
- Validation
- Throttling

Skeleton Of A System



Applications that use the Query Engine

Uber Marketplace Data Query Applications



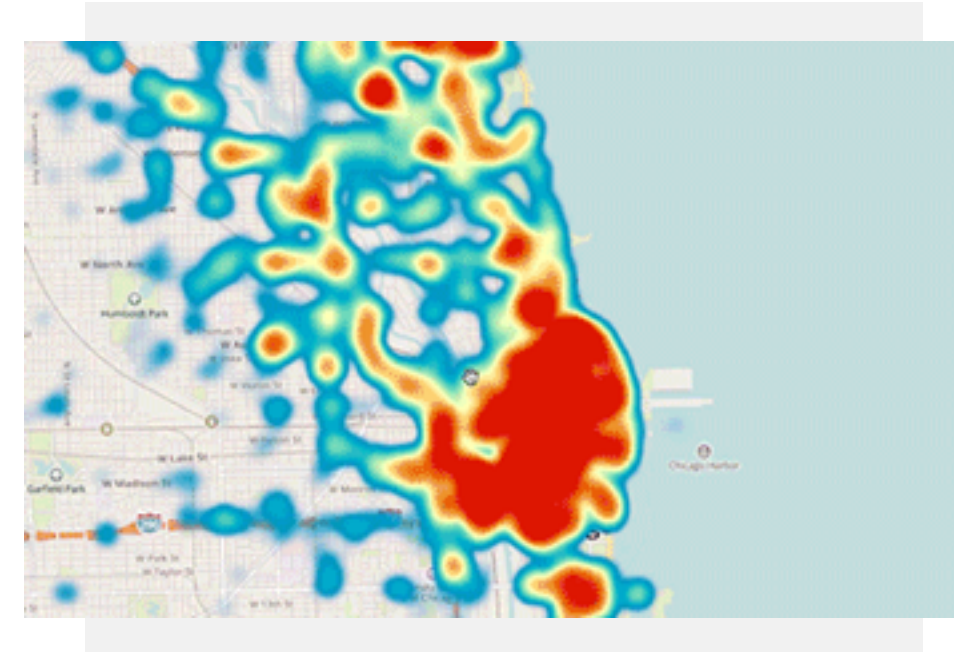
Dashboards

Business Metrics
Dashboards



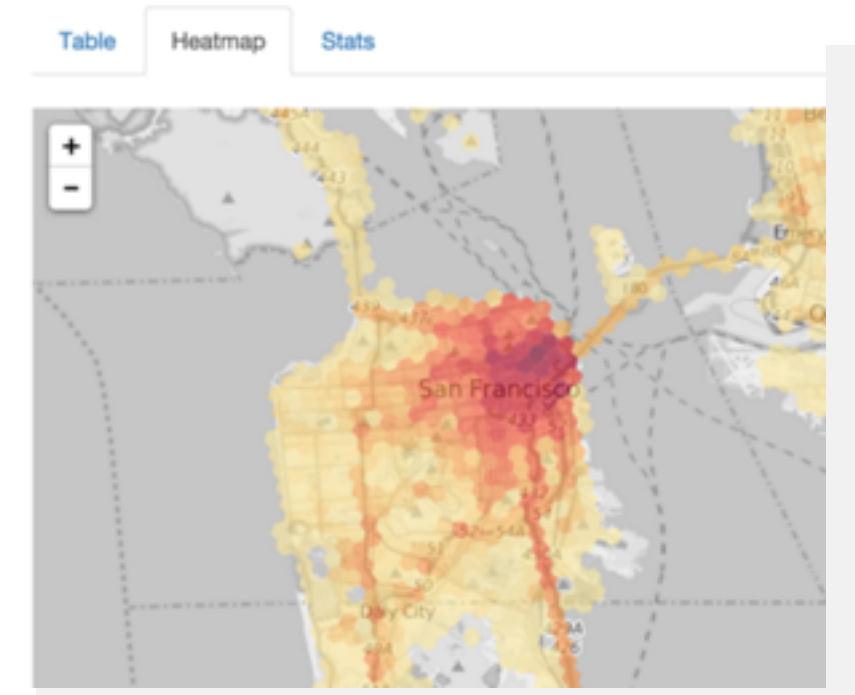
State Transitions/Raw Query

Querying data in flexible ways



Streaming

Seeing what's happening now, continuously



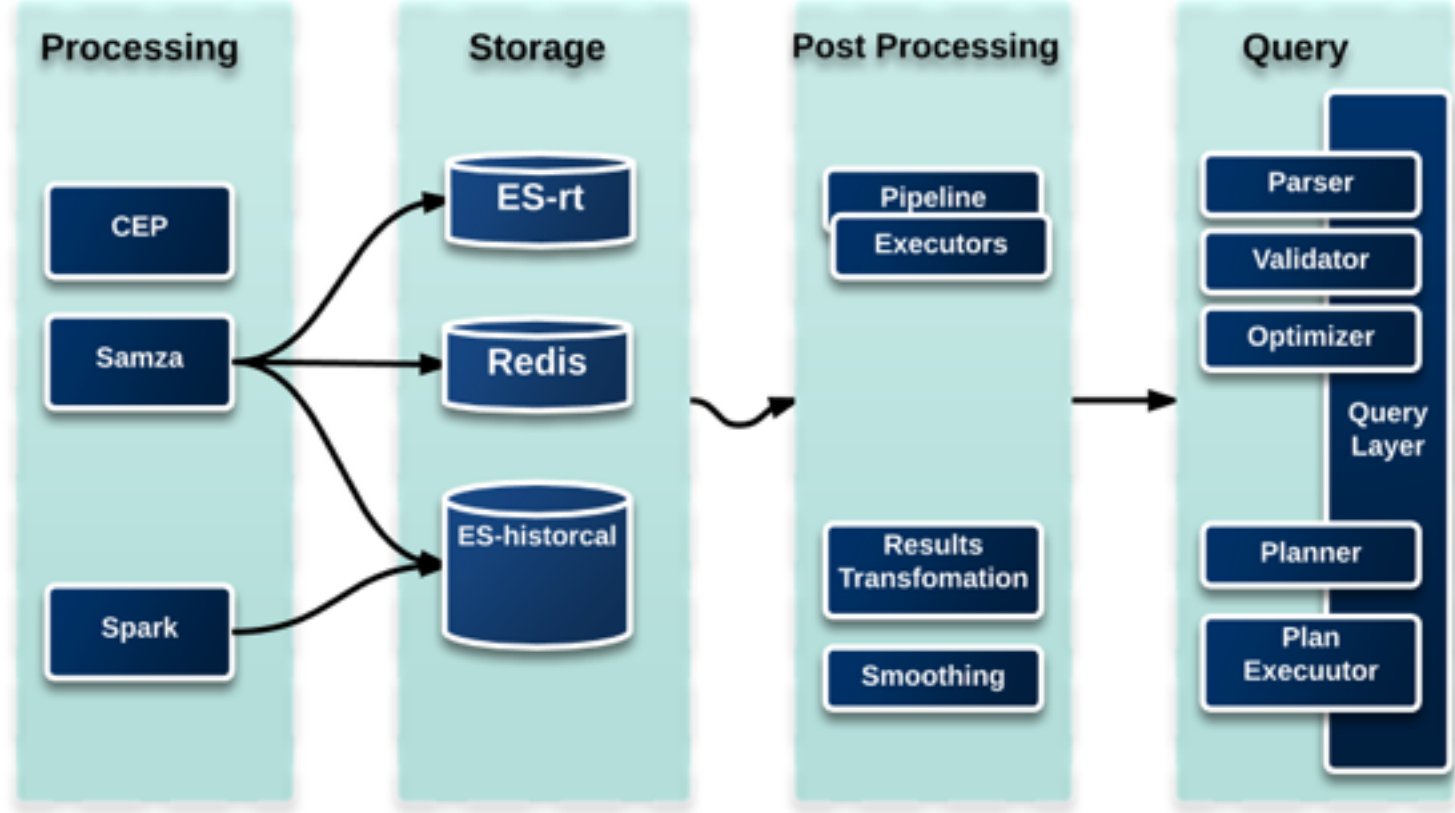
Visual Exploration

Explore your data via Geo Visualization tools

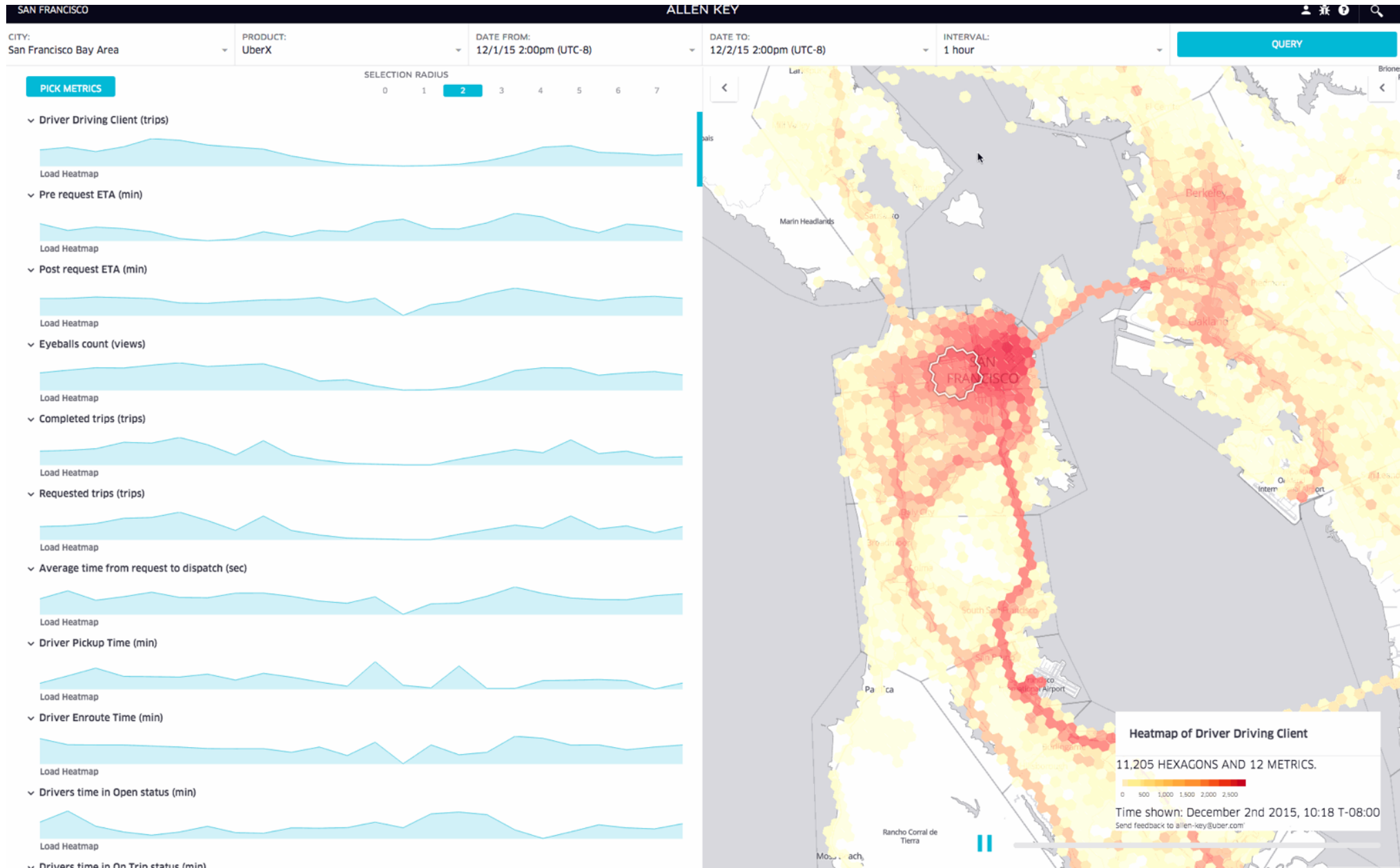
Business Metrics Dashboards



Kafka



Realtime Analytics



Explore Business Metrics Per City/Vehicle Type

Query

Driver Efficiency Timeseries
Driver Efficiency Per City
Time To Request
Dynamic K-Ring Heatmap
Trip Ended Heatmap
Weighted K-Ring Heatmap
K-Ring Status Change Heatmap
Surge Calculation

End time: 2016-02-25T16:44:51.834-08:00

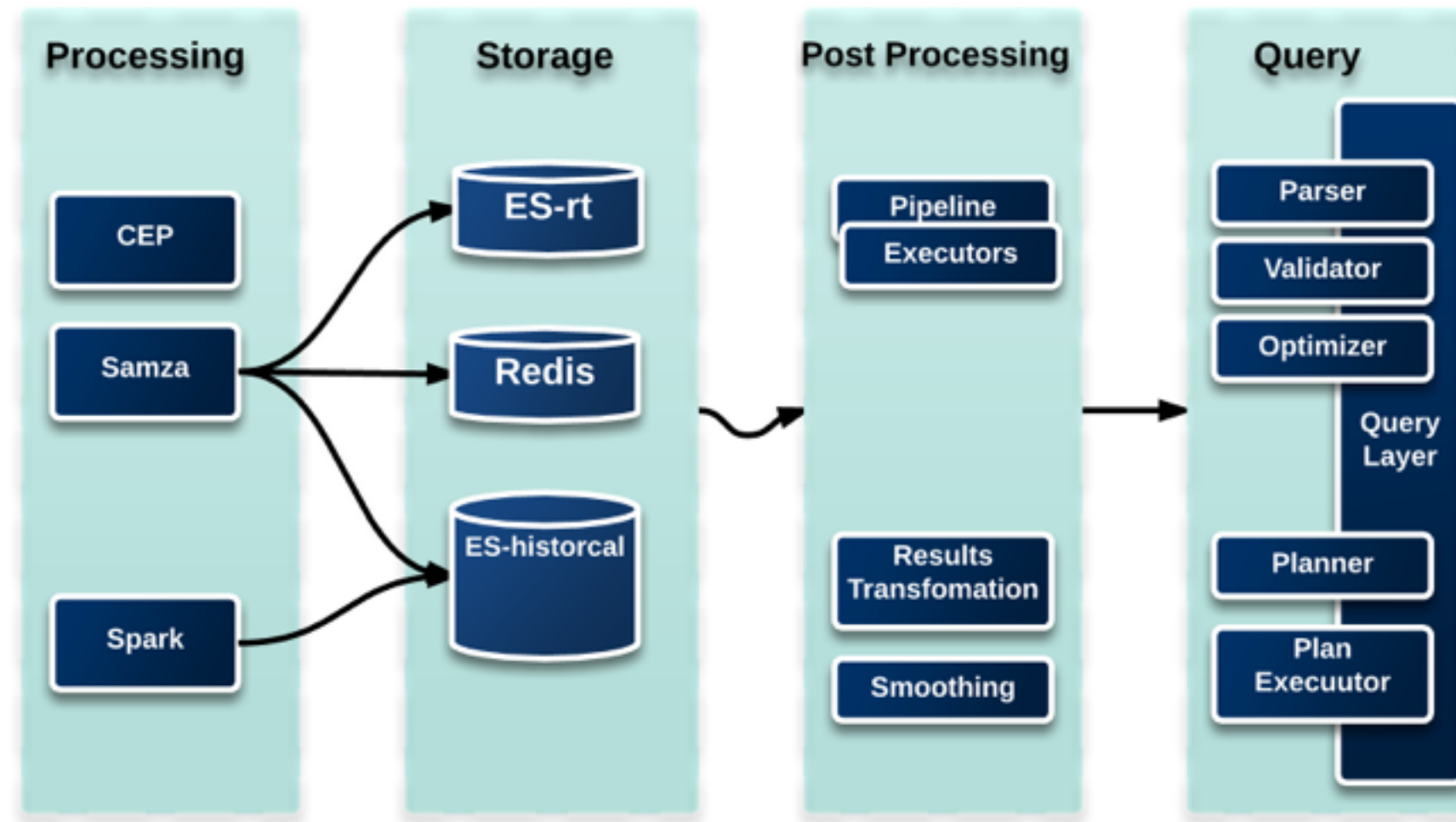
Interval: 10m

City: (all)

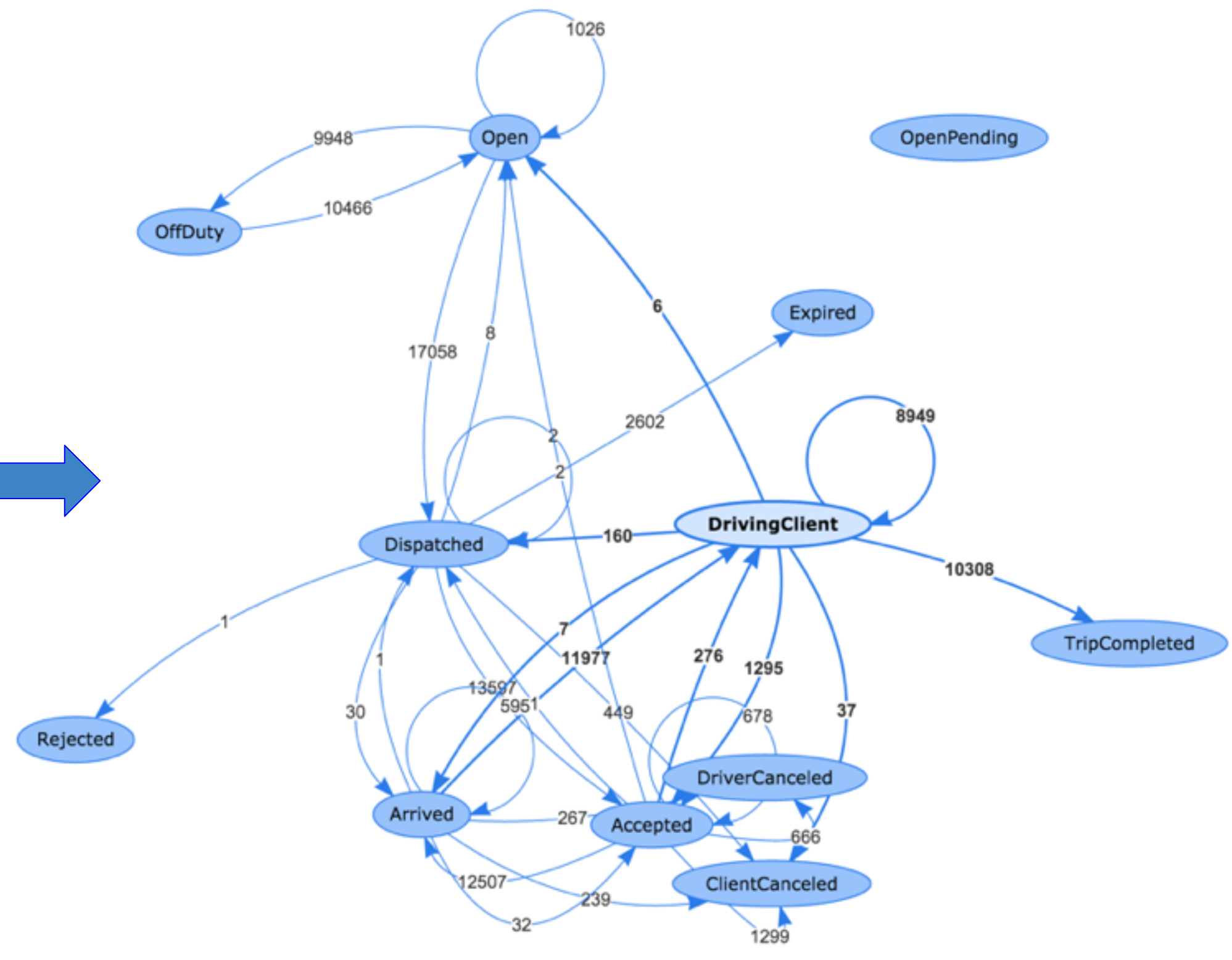
Query



Kafka



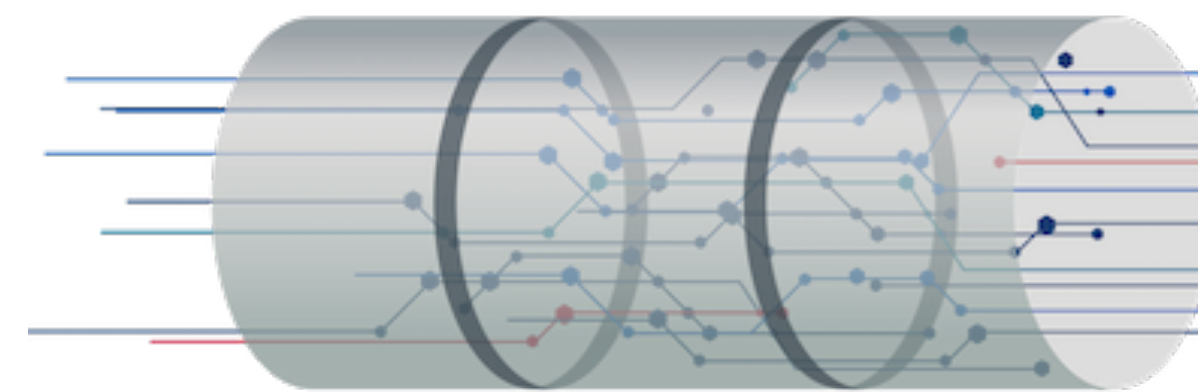
Gairos Processing



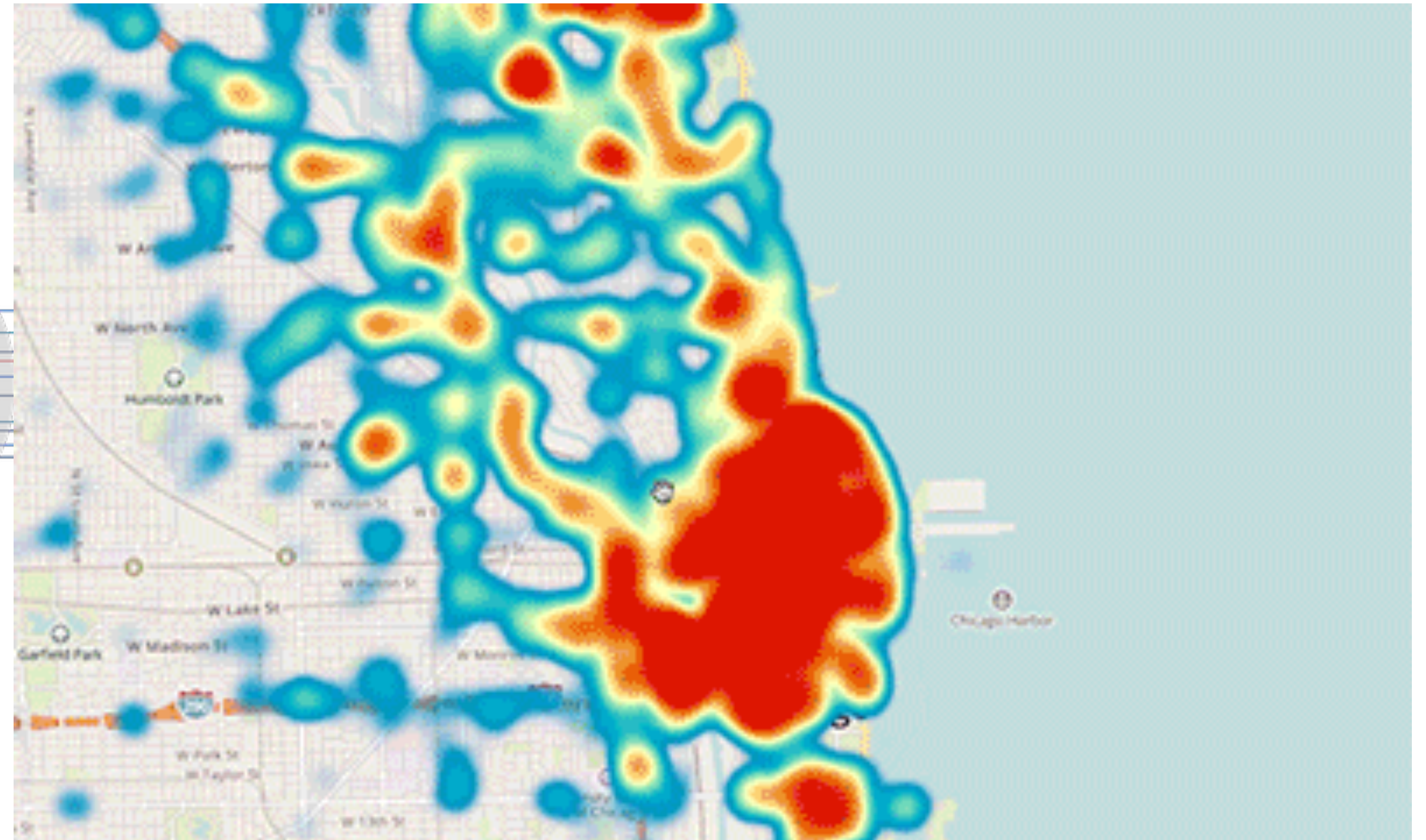
Streaming



Kafka

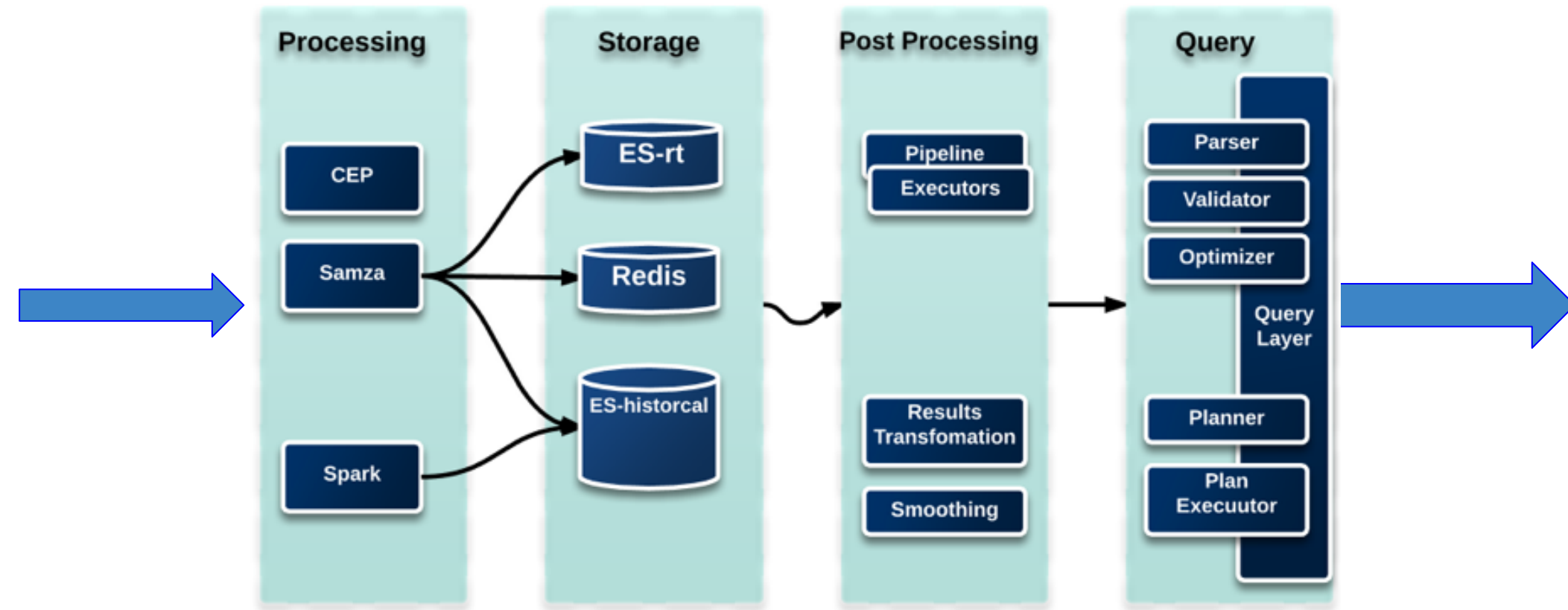
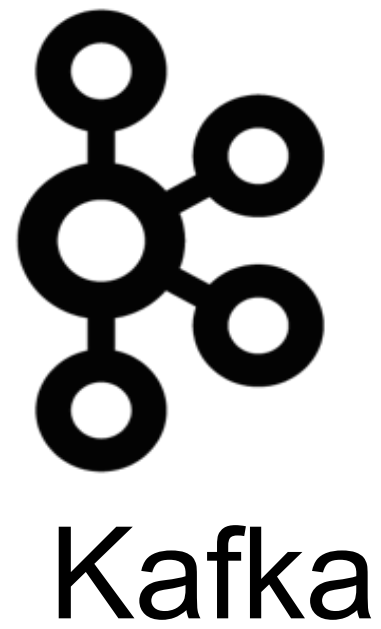


Query /
Streaming



Realtime
Visualization

Exploration



Realtime Analytics



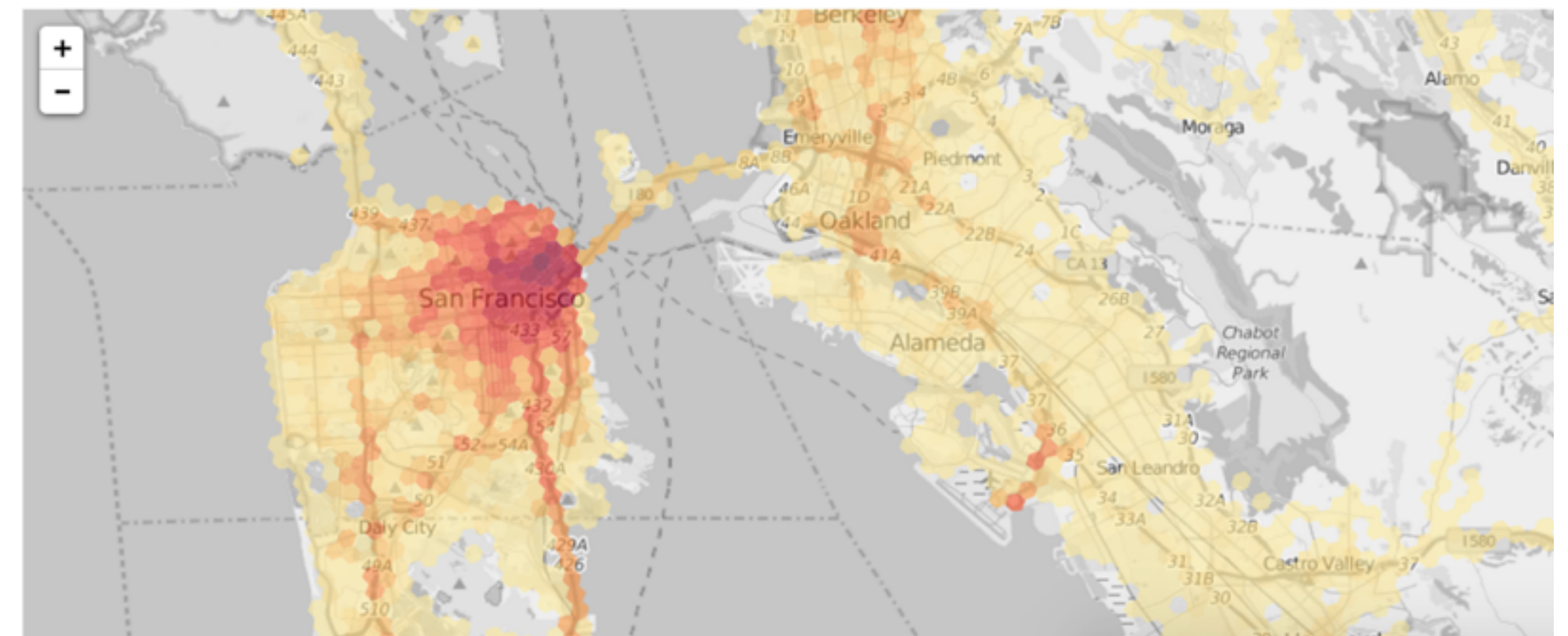
```
1 {
2   "by": [
3     "hexagon_id"
4   ],
5   "filter": {
6     "type": "and",
7     "fields": [
8       {
9         "type": "eq",
10        "dimension": "city",
11        "value": "1"
12      },
13      {
14        "type": "eq",
15        "dimension": "vvids",
```

Data Source

Provides aggregated driver information.
Dimensions: @timestamp, driverUUID, city, hexagon_id, geofence, status, vvids
Metrics:

Query Download as JSON Download as CSV

Table Heatmap Stats



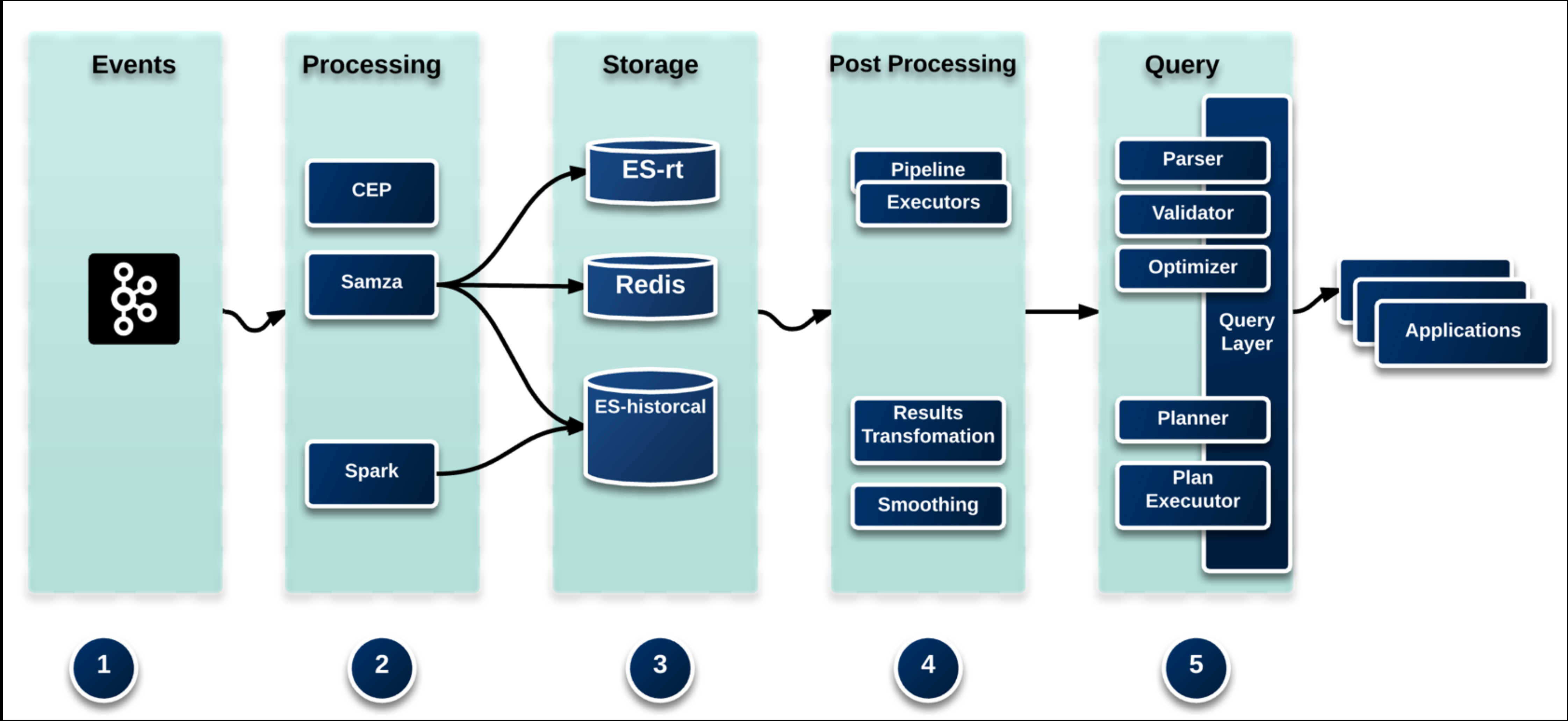
Gairos Dashboard

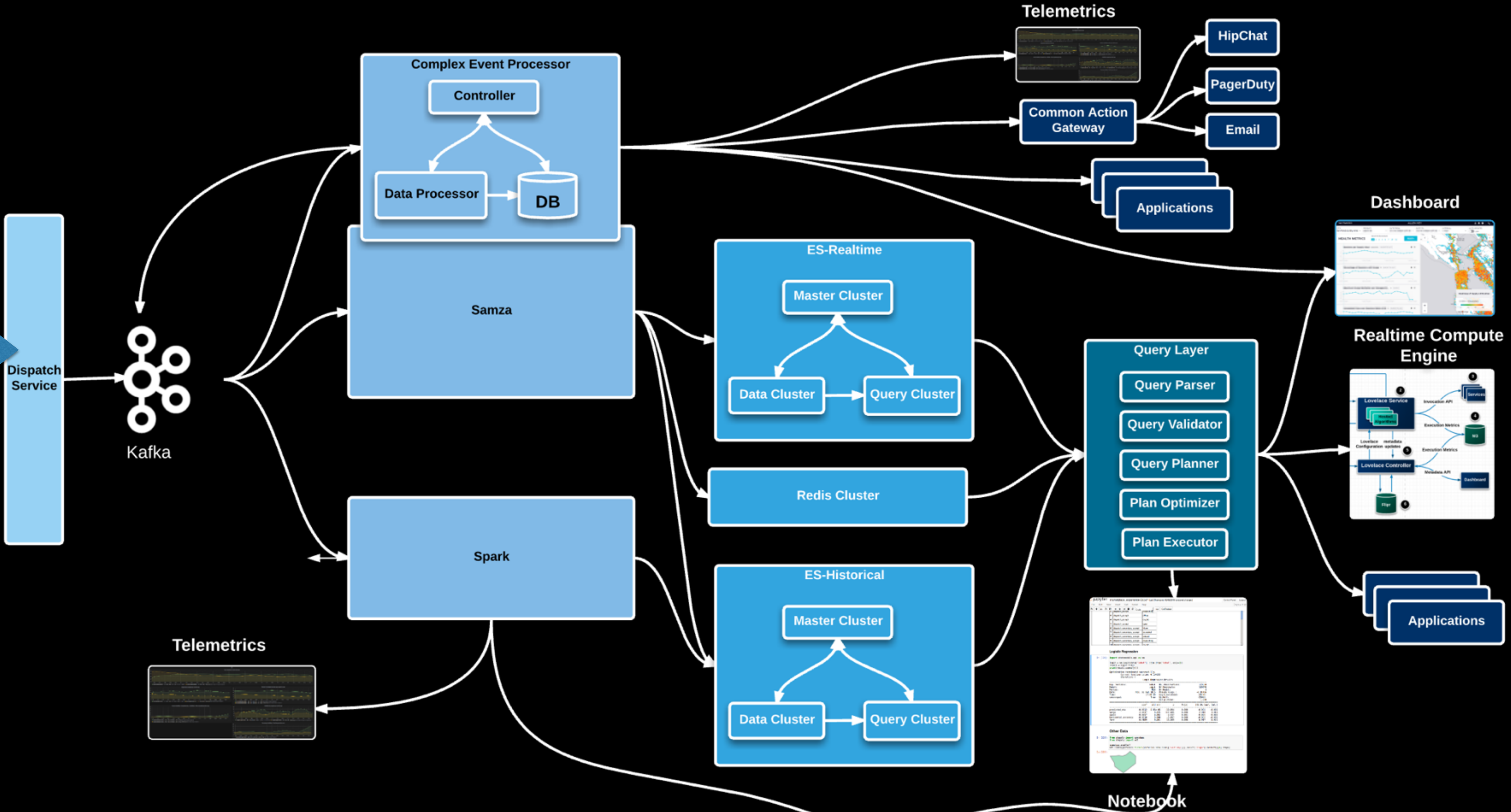
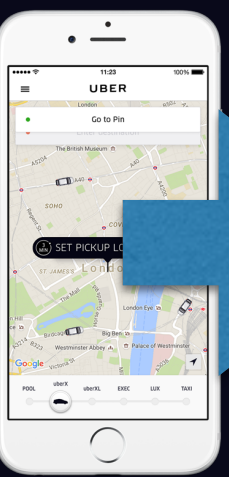
Overall Architecture

To facilitate exploring, real-time analytics, backfilling, monitoring, ...



Overall Analytics System





Choices/Tradeoffs

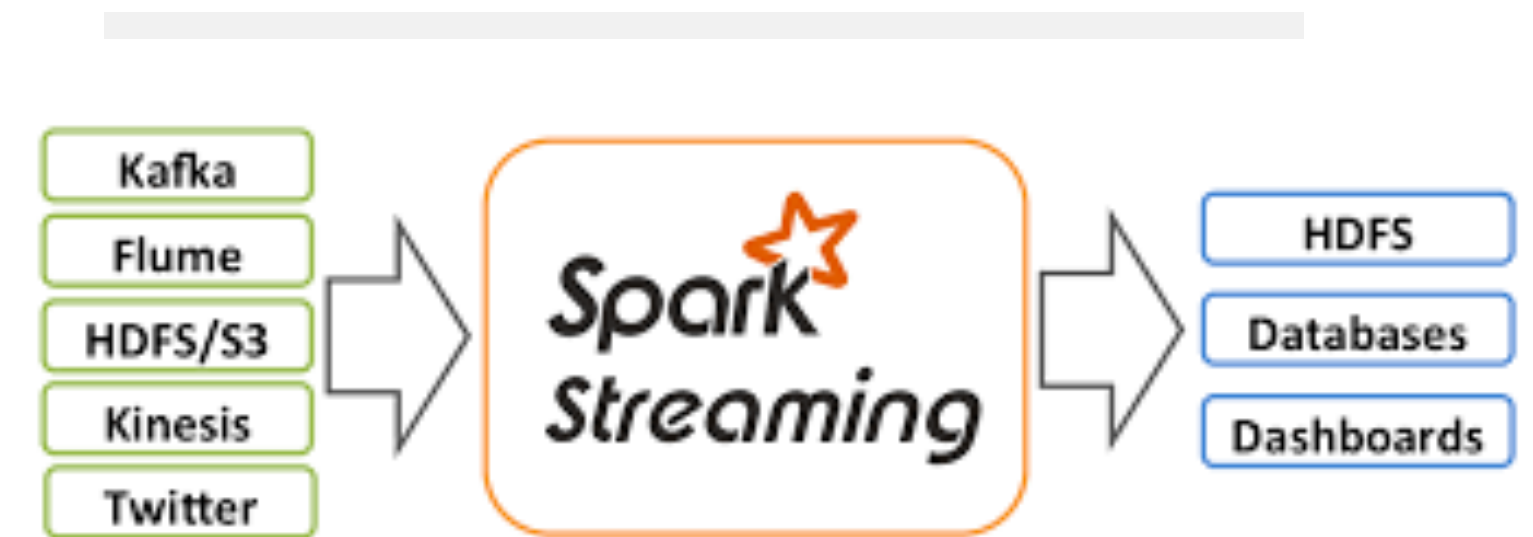
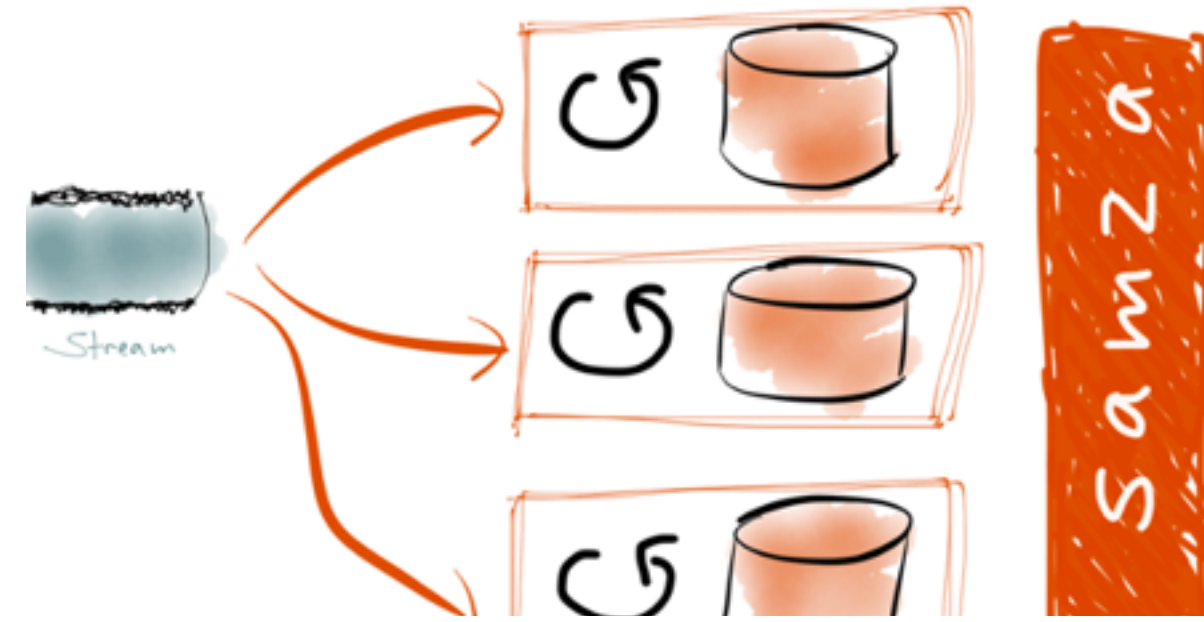
What were some of the choices considered?

How did we settle down on the final choice?



Stream Processing

Some Choices



Storm

Was our original choice

Initial systems built on Storm

However

Twitter moving away from Storm

Unbalanced topologies were problematic

Operational complexities

Samza

Our current choice

Well integrated with Kafka

Built in State Management

Built in Checkpointing

Spark Streaming

Looking at this actively

Micro Batch based processing

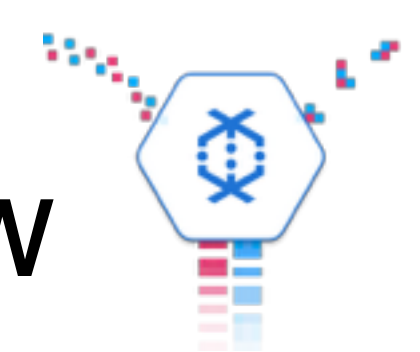
Good integration with HDFS & S3

Exactly once semantics

Kinesis



Dataflow



Persistence

Some Choices



Elasticsearch

Distributed Indexes & Queries
Versatile aggregations



memsql

In-memory database
Fast Analytic Engine



Druid

Highly scalable
Designed for Realtime OLAP
However
Operationally Complex

Analytics/Dashboards etc.

Some Choices

IP[y]: IPython
Interactive Computing

Jupyter/IPython

Great community support

Data Scientists familiar with Python

```
jupyter marketplace_experience-Copy1 Last Checkpoint: 02/08/2016 (unsaved changes) Control Panel Logout
File Edit View Insert Cell Kernel Help Python 2.0
+ + + + + Code Cell Toolbar
1 dispatch_accept ondispatching
2 dispatch_accept offline
3 dispatch_accept on_trip
4 dispatch_accept open
5 dispatch_accept open
6 dispatch_secondary_accept None
7 dispatch_secondary_accept accepted
8 dispatch_secondary_accept arrived
9 dispatch_secondary_accept dispatching
10 dispatch_secondary_accept not_trip

Logistic Regression
In [197]: import statsmodels.api as sm
logit = sm.Logit(data['label'], data.drop('label', axis=1))
result = logit.fit()
print(result.summary())
Optimization terminated successfully.
Current function value: 0.197133
Iterations: 7
Logit Regression Results
Dep. Variable: label No. Observations: 129080
Model: Logit DF Residuals: 129075
Method: MLE DF Model: 4
Date: Mon, 08 Feb 2016 Pseudo R-squ.: -0.01496
Time: 17:19:30 Log-Likelihood: -25446.
Converged: True LL-Null: -25071.
LLR p-value: 1.000
coef std err z P>|z| [95.0% Conf. Int.]
-----
predicted_eta -0.0012 5.06e-05 -19.894 0.000 -0.001 -0.001
surge 2.9637 0.025 117.885 0.000 2.914 3.013
speed -0.0027 0.002 -1.417 0.001 -0.009 0.002
horizontal_accuracy -0.0018 0.000 -5.897 0.000 -0.002 -0.001
fare 0.0189 0.001 15.569 0.000 0.017 0.021

Other Data
In [228]: from shapely import speedups
from shapely import wkt
speedups.enable()
wkt.loads(geofences.filter(geofences.name.isin(['East Bay'])).select('shape')).collect()[0].shape
Out[228]:
```



Zeppelin

Integrated with Spark

Offer many language support (Python, Scala, ..)



Kibana

Kibana

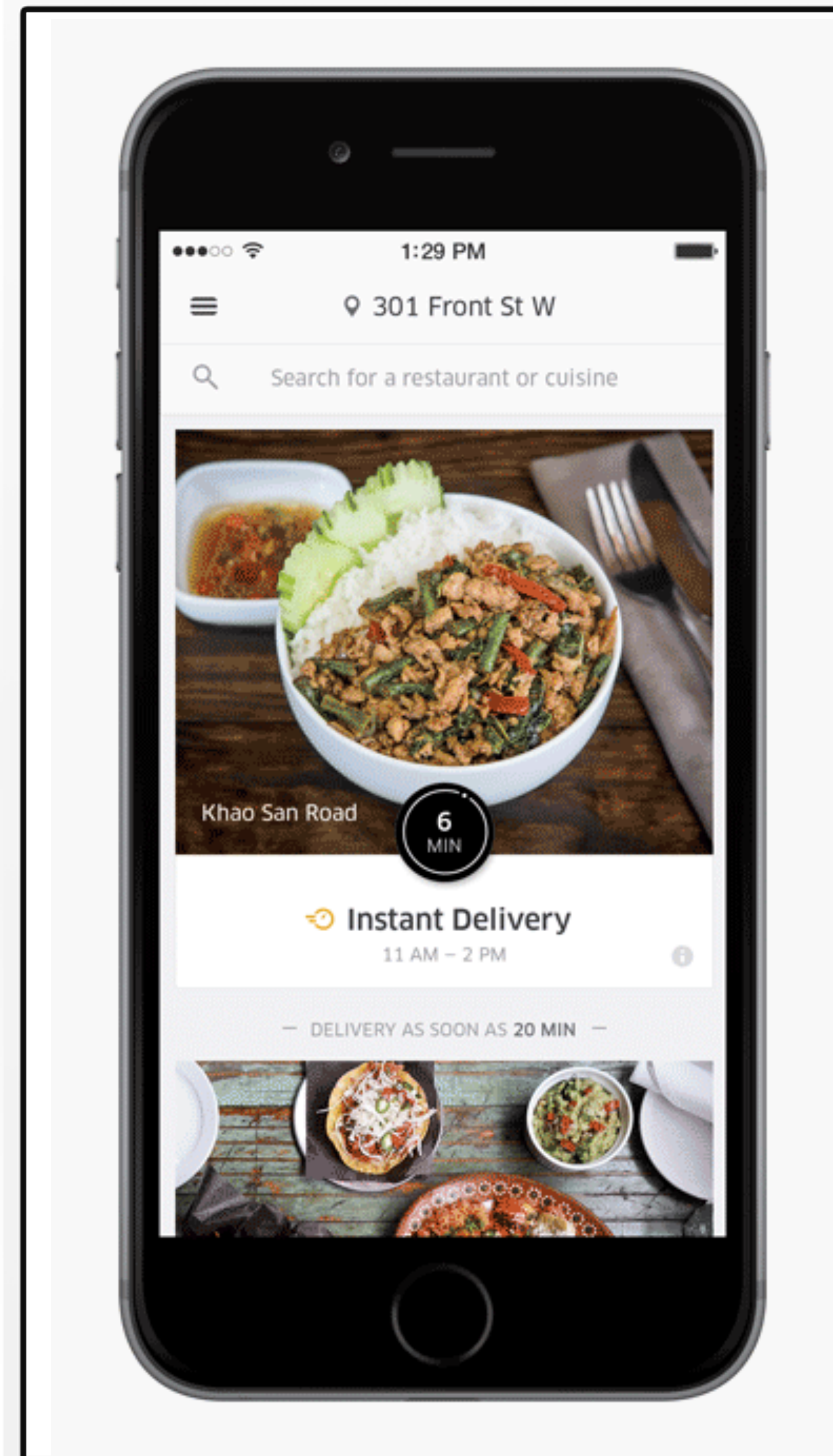
Integration with ElasticSearch



Links

Thank you!

- Realtime Monitoring with Uber's Argos (<https://eng.uber.com/argos/>)
- Spark at Uber (<http://www.slideshare.net/databricks/spark-meetup-at-uber>)
- Career at Uber (<https://www.uber.com/careers/>)



Q & A

Happy to discuss design/architecture

No product/business questions please :-)

@stonse



Thank you

Sudhir Tonse

@stonse

Proprietary and confidential © 2016 Uber Technologies, Inc. All rights reserved. No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval systems, without permission in writing from Uber. This document is intended only for the use of the individual or entity to whom it is addressed and contains information that is privileged, confidential or otherwise exempt from disclosure under applicable law. All recipients of this document are notified that the information contained herein includes proprietary and confidential information of Uber, and recipient may not make use of, disseminate, or in any way disclose this document or any of the enclosed information to any person other than employees of addressee to the extent necessary for consultations with authorized personnel of Uber.

The Uber logo, consisting of the word "UBER" in white, uppercase, sans-serif font, centered on a black rectangular background. This logo is positioned in the lower-left corner of the overall image, overlapping the white text area and the photograph of the woman.

UBER

