Spark: Interactive to Production

Dara Adib (Marketplace Data)

Spark Summit 2016
June 7, 2016

Who

- Uber
  - –70+ countries. 450+ cities.
- Marketplace Data
  - –Realtime Data Processing
  - –Analytics
  - –Forecasting
- Spark

Who

- Uber
  - –70+ countries. 450+ cities.
  - Marketplace Data
    - –**Realtime Data Processing**
    - –**Analytics**
    - –Forecasting
  - Spark

Marketplace Data

Dashboards

Business Metrics Dashboards

State Transitions/Raw Query

Querying data in flexible ways

Streaming

Seeing what's happening now, continuously

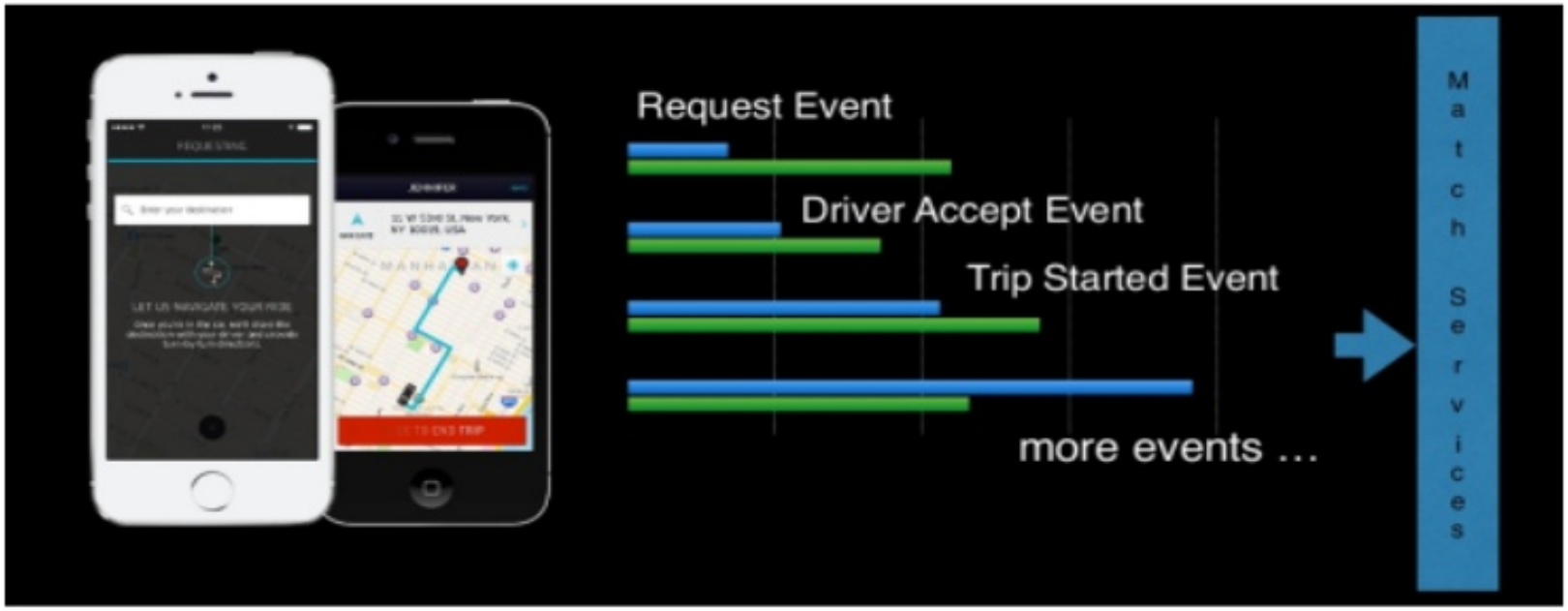
Visual Exploration

Explore your data via Geo Visualization tools

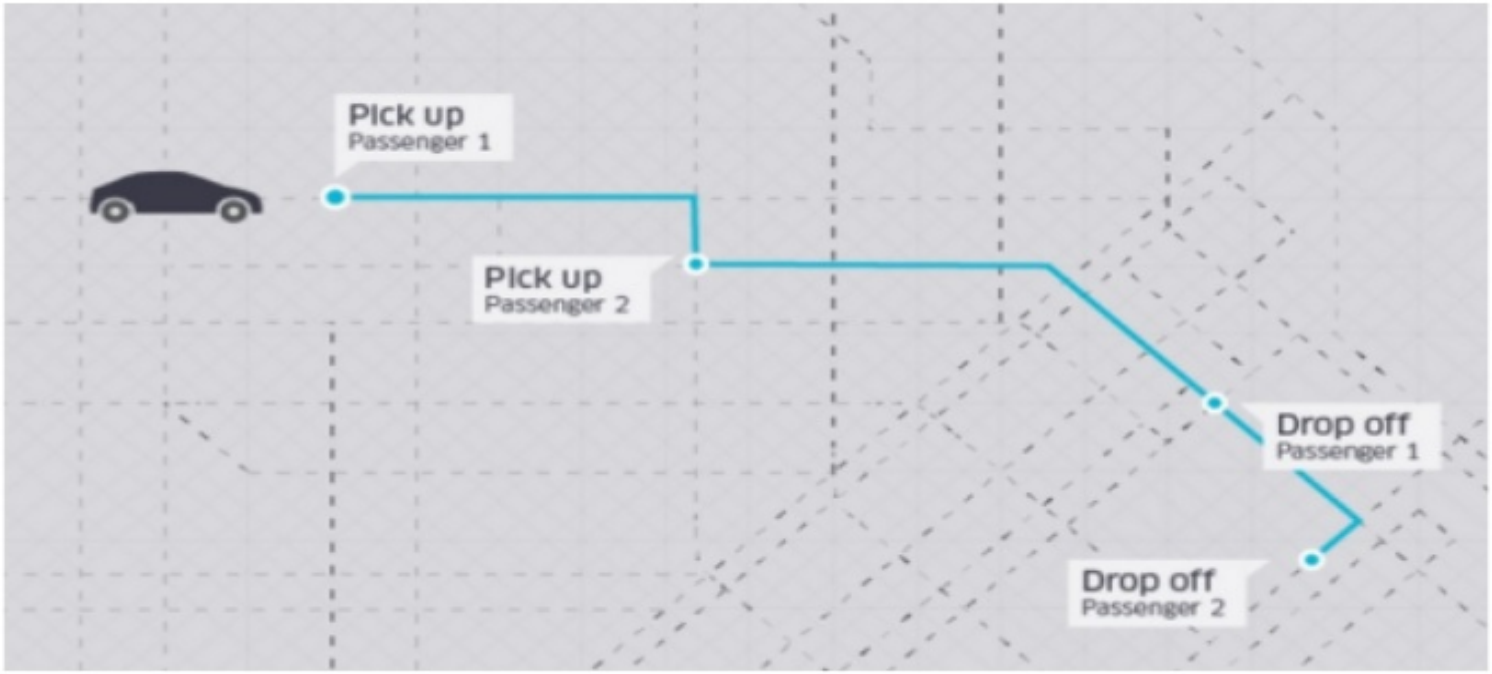
Relational Data

- Traditionally data is stored in a RDBMS.
- This works well for row lookups and joins.
- But what about events and windowing?

Stream Processing



Trip States

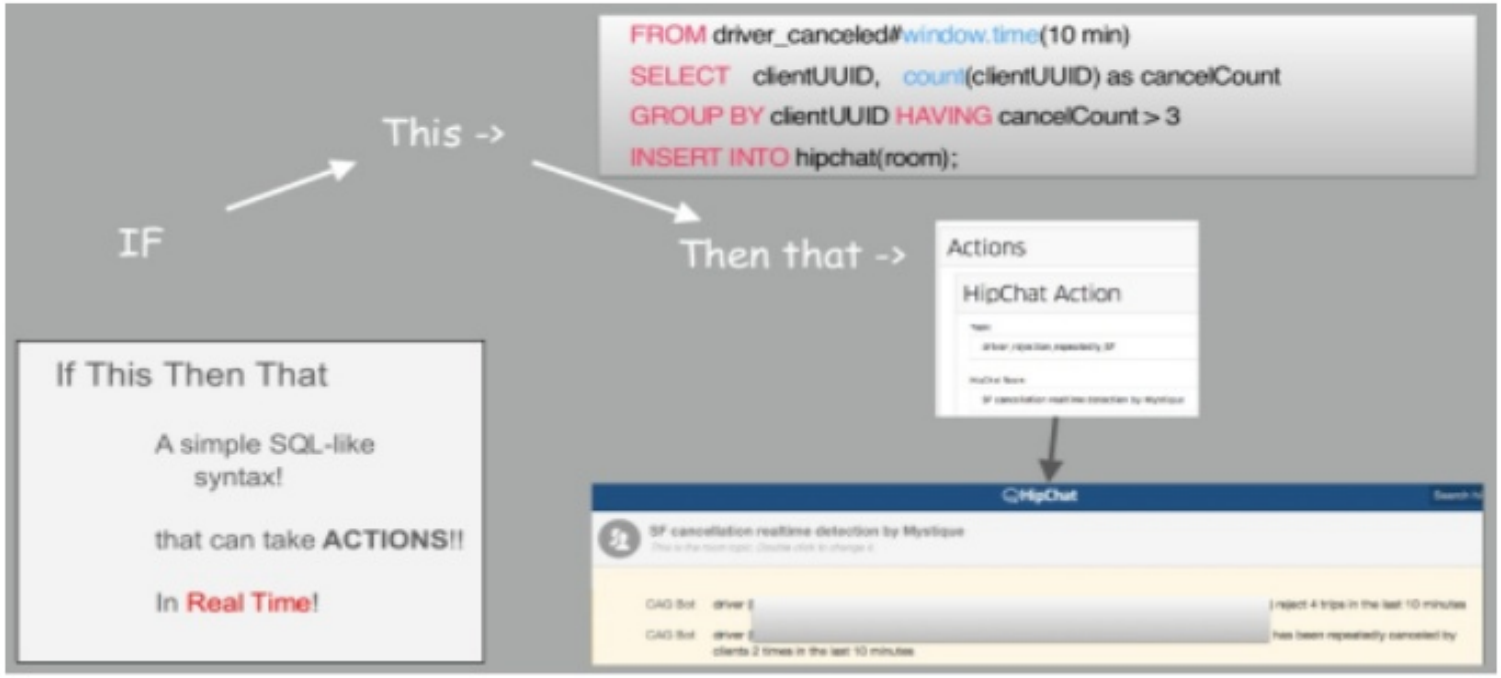

Pick up
Passenger 1
Pick up
Passenger 2
Drop off
Passenger 1
Drop off
Passenger 2

OLAP Queries

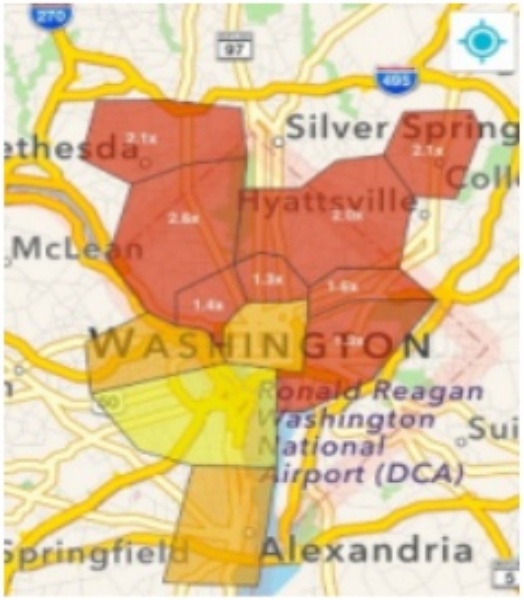
- How many **open cars** are in London now?
- What is the **driving time** in New York's Financial District, by time of day and day of week?
- What is the **conversion rate of requests into trips** on Friday evenings in San Francisco?

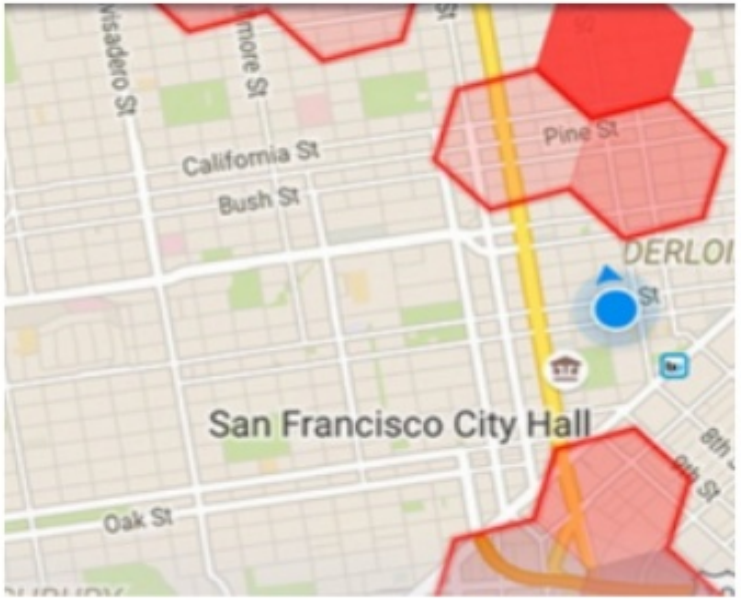
How many drivers cancel a request > 3 times in a row within a 10-minute window?

Complex Event Processing



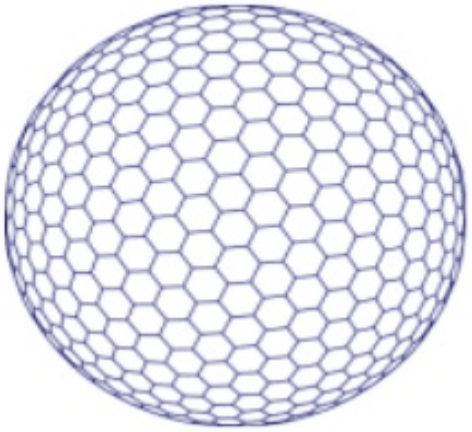
Geo Aggregation

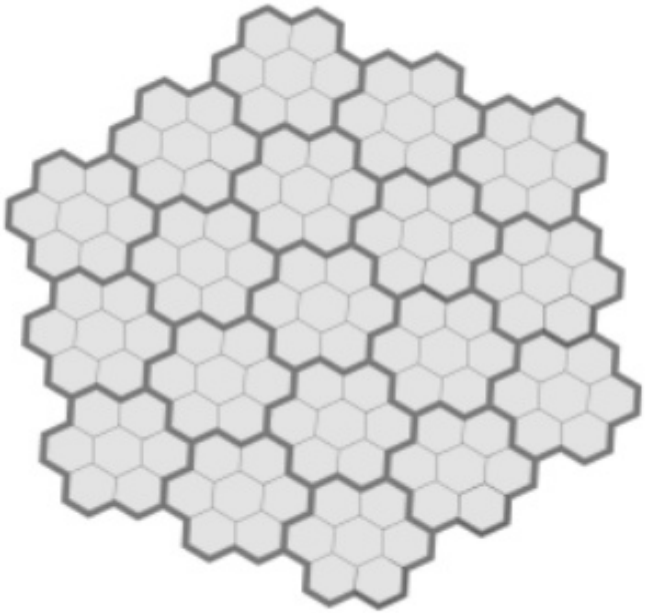




Hexagons

- Indexing, Lookup, Rendering
- Symmetric Neighbors
- Convex Regions
- ~Equal Area
- ~Equal Shape





No magic bullet, yet?

- Empower users. Democratize data.
  - –Services want reliability and consistent performance.
  - –Data Scientists want Pandas and flexibility.
- Spark is not a database.
  - –Data too big to fit in memory?
  - –Checkpointing UPDATEs.
- Spark 2.0? Alluxio?

A Tale of Two Cities

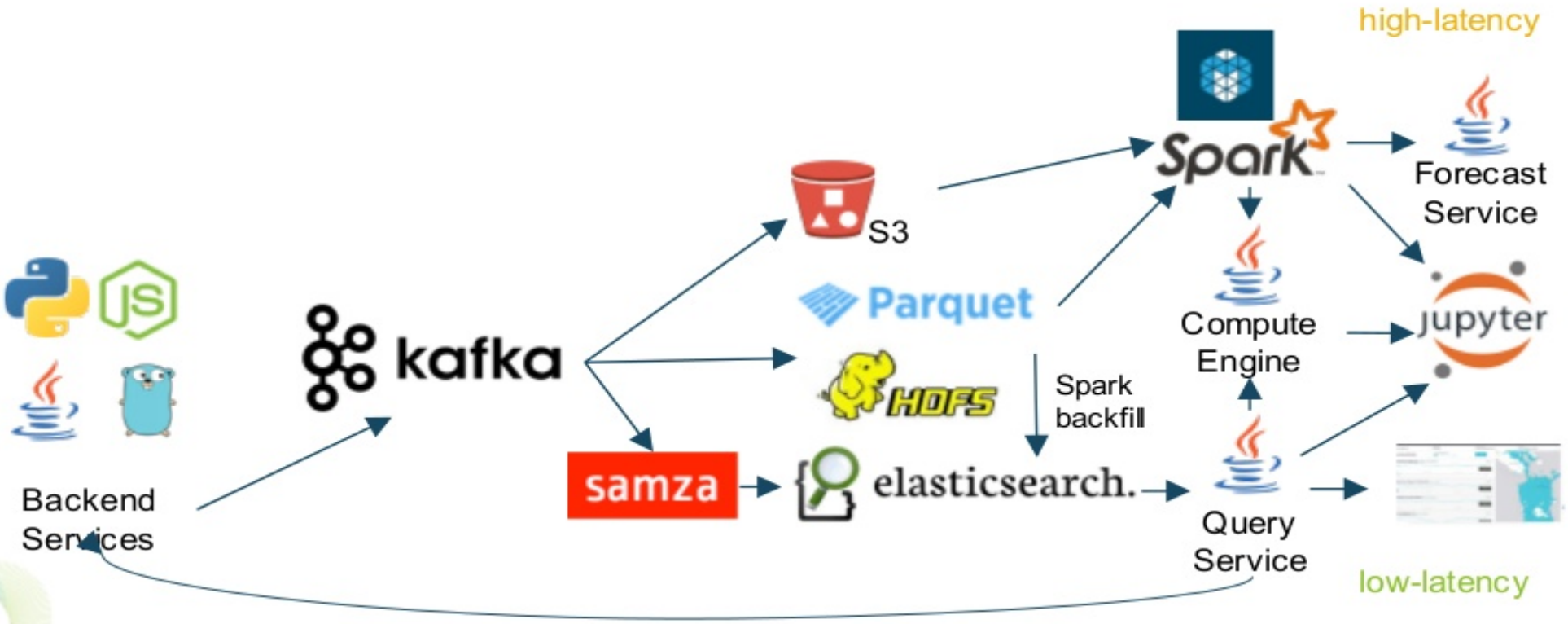
- Extensibility vs. Reliability
- Months of Data vs. Minutes of Data
- Batch v.s. Streaming
- Development v.s. Production
- HDFS v.s. Relational Database
- YARN v.s. Mesos

"Data scientists don't know how to code."
-Software Engineer

Other Challenges

- Data discoverability
- Data freshness
- Query latency
- Debuggability
- Isolation
  - –CPU, memory, disk space, disk I/O, network I/O
  - –“Bad queries”

Service Oriented Architecture

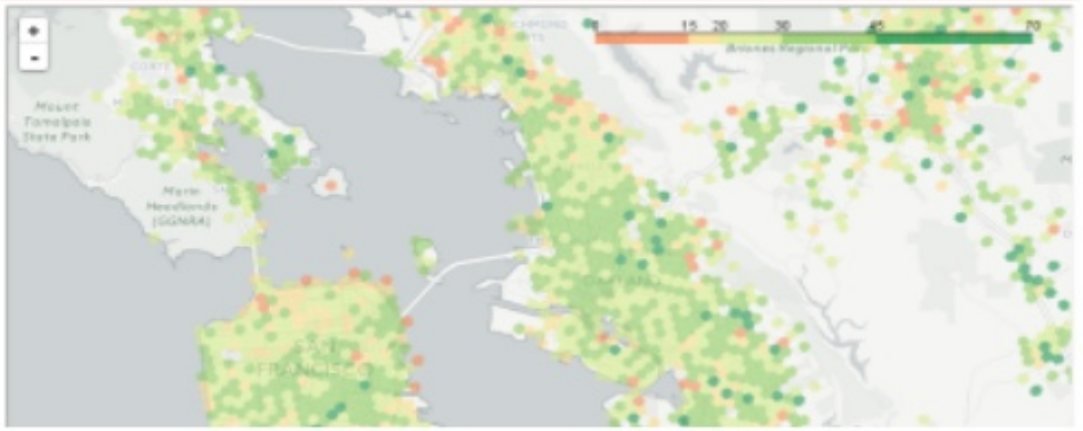


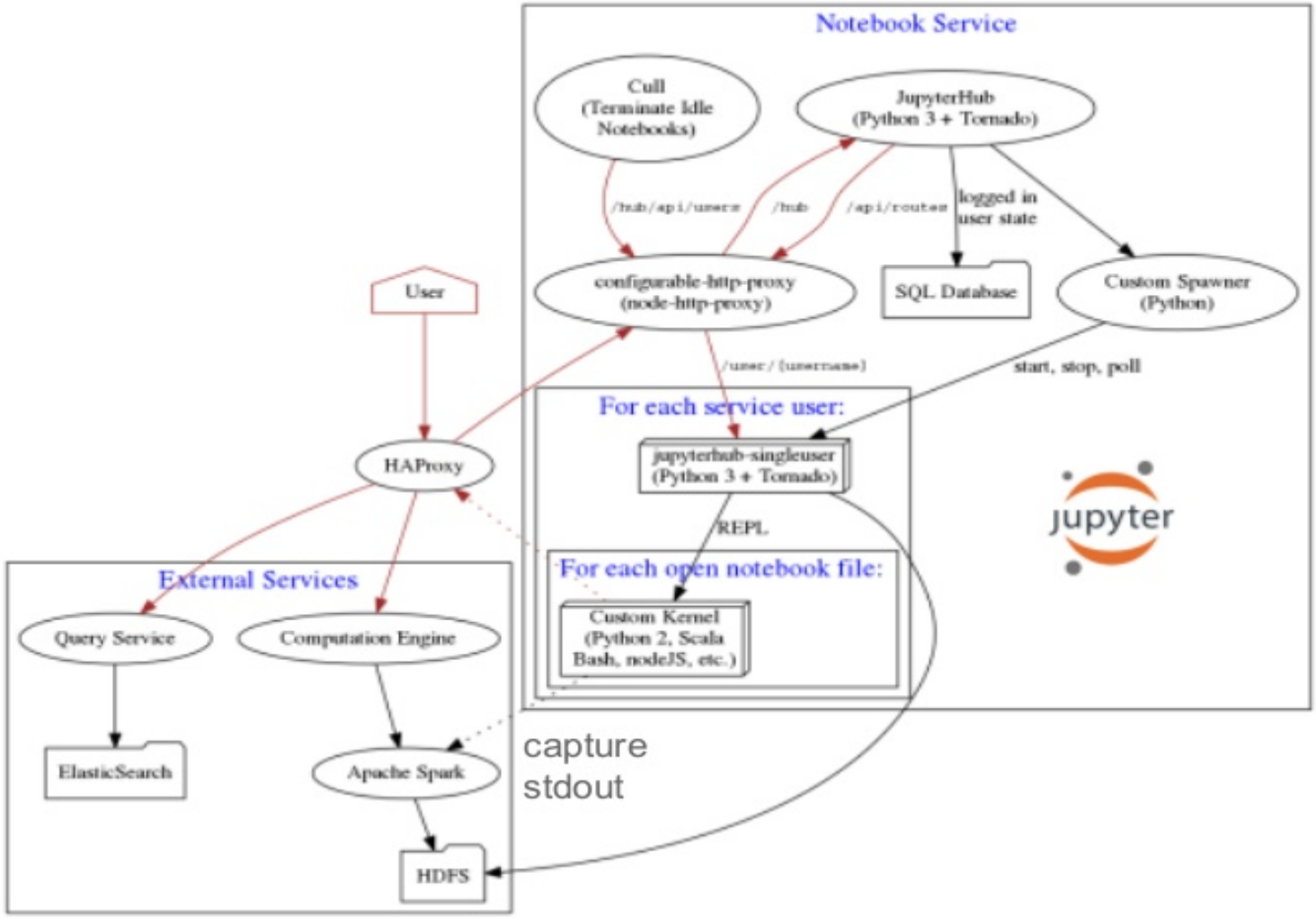
Why Jupyter

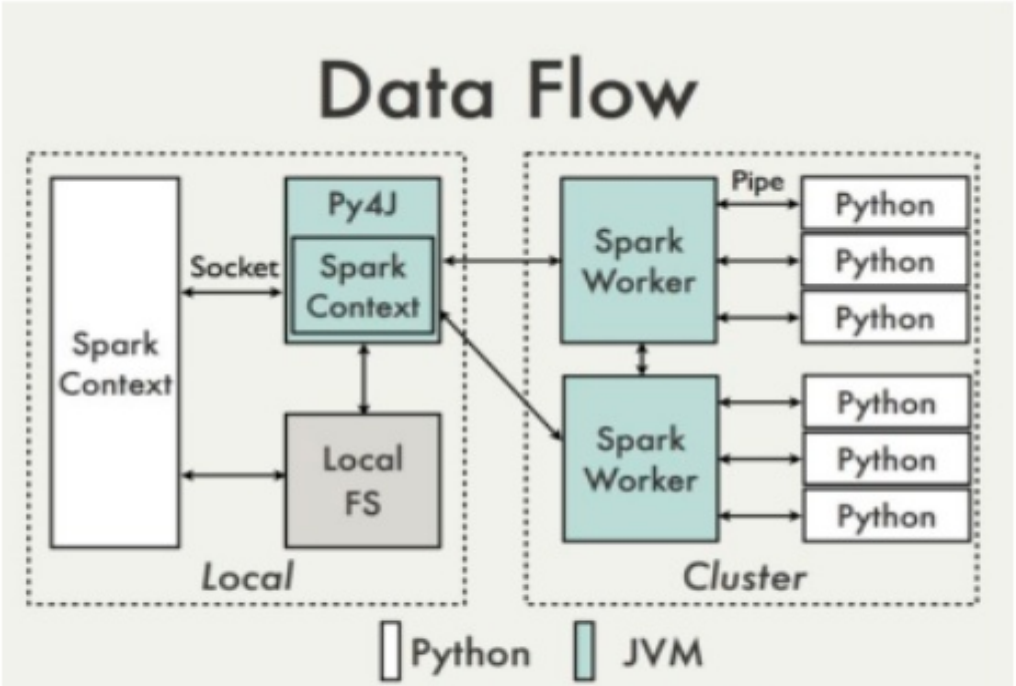
- Ease-of-Use
- Extensibility
  - –Python and JavaScript libraries
- Alternatives
  - –Apache Zeppelin
  - –Databricks

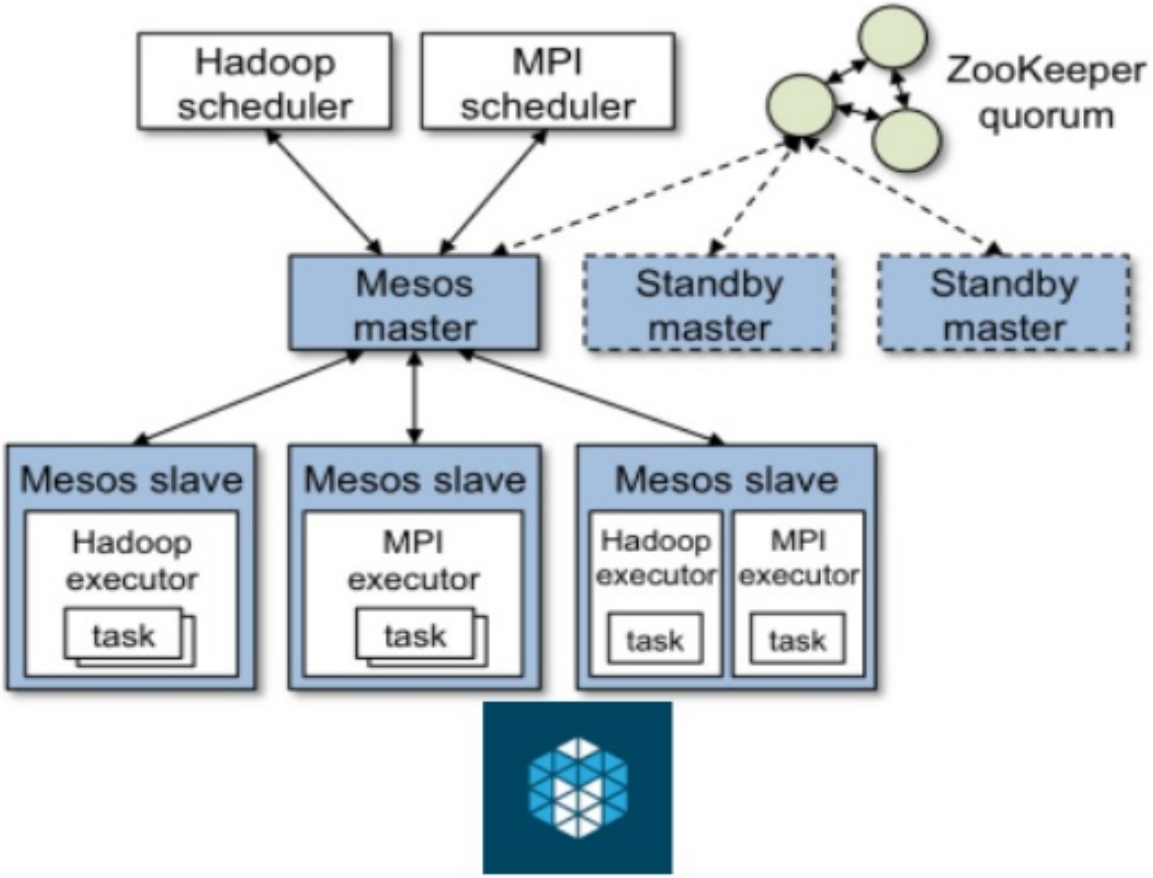
```
data_df = hex_trips.filter("vvid in (1,2)").select('hex', 'speed').toPandas()
geojson = {'type': 'FeatureCollection', 'features': []}
for hex_id in data_df['hex']:
    geojson['features'].append({
        'type': 'Feature',
        'id': hex_id,
        'properties': {},
        'geometry': {'type':'Polygon', 'coordinates': [h.get_hex_vertices(hex_id)]}
    })

m = folium.Map([37.8, -122.3], zoom_start=11, tiles='cartodbpositron')
m.choropleth(geo_str=json.dumps(geojson), data=data_df, columns=['hex', 'speed'],
             key_on='feature.id', threshold_scale=[0, 15, 20, 30, 45, 70],
             fill_color='RdYlGn', line_weight=0)
m
```

Out[3]:




Notebook Service
Cull
(Terminate Idle
Notebooks)
JupyterHub
(Python 3 + Tornado)
/hub/api/users
/hub
/api/routes
logged in
user state
User
configurable-http-proxy
(node-http-proxy)
SQL Database
Custom Spawner
(Python)
/user/[username]
start, stop, poll
For each service user:
HAProxy
jupyterhub-singleuser
(Python 3 + Tornado)
jupyter
REPL
For each open notebook file:
Custom Kernel
(Python 2, Scala
Bash, nodeJS, etc.)
External Services
Query Service
Computation Engine
ElasticSearch
Apache Spark
HDFS
capture
stdout


Data Flow
Spark Context
Socket
Py4J
Spark Context
Local FS
Local
Pipe
Spark Worker
Spark Worker
Python
Python
Python
Python
Python
Python
Cluster
Python
JVM


Hadoop scheduler
MPI scheduler
ZooKeeper quorum
Mesos master
Standby master
Standby master
Mesos slave
Hadoop executor
task
Mesos slave
MPI executor
task
Mesos slave
Hadoop executor
task
MPI executor
task


pySpark

Mesos

Frameworks

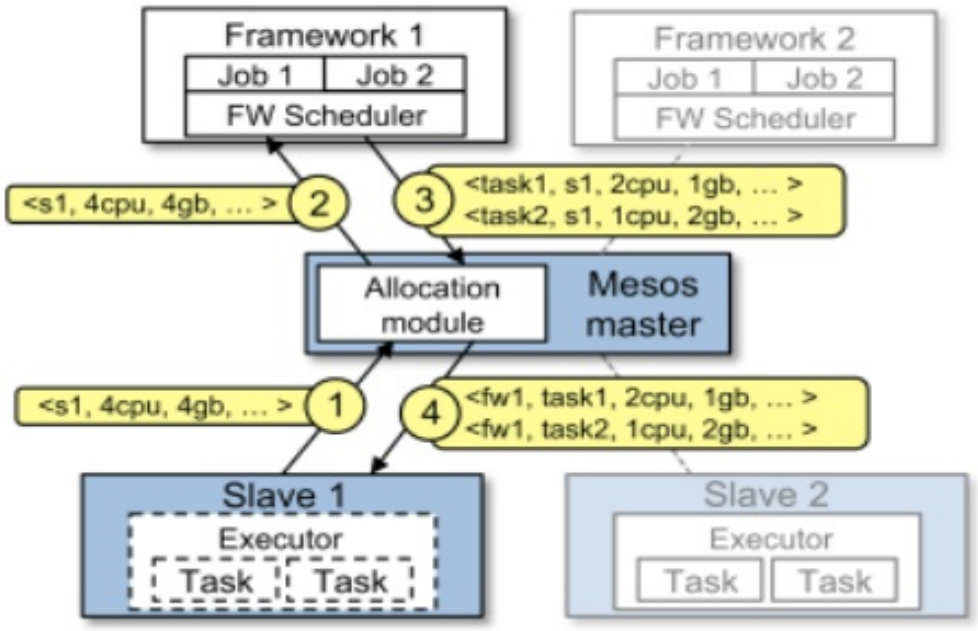
- Scheduler
  - Connects to Mesos master.
  - Accepts or declines resources.
  - Contains delay scheduling logic for rack locality, etc.
- Executor
  - Connects to local Mesos slave.
  - Runs framework tasks.
- Examples
  - Aurora, Marathon, Chronos
  - Spark, Storm, Myriad (Hadoop)

Masters and Agents

- Master
  - Shares resources between frameworks.
  - Keeps state (frameworks, agents, tasks) in memory.
  - HA: 1 master elected.
- Agent
  - Runs on each cluster node.
  - Specifies resources and attributes.
  - Starts executors.
  - Communicates with master and executors to run tasks.

Mesos Resource Offers

1. Slave reports available resources to the master.
2. Master sends a resource offer to the framework scheduler.
3. Framework scheduler requests two tasks on the slave.
4. Master sends the tasks to the slave which allocates resources to the framework's executor, which in turn launches the two tasks.



Mesos Resources

Types

- cpu: CPU share
  - optional CFS for fixed
- mem: memory limit
- disk space: disk limit
- ports: integer port range
- bandwidth

Custom resources: k,v pairs

Isolation

- Linux container
  - control groups (cgroup)
  - namespaces
- Docker container
- External container

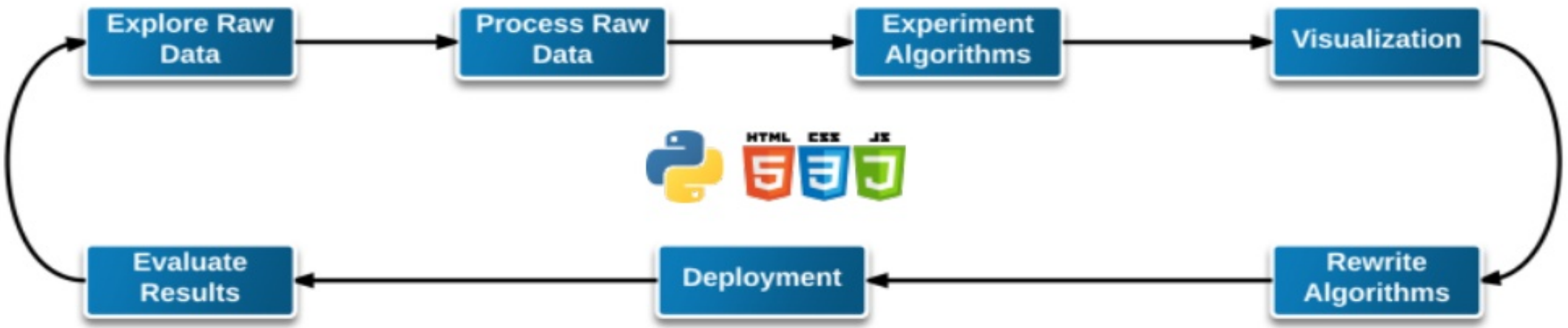
Other features

- Reserved resources by role
- Oversubscription
- Persistent volumes

```
from shutil import rmtree
from tempfile import mkdtemp

from pyspark import SparkFiles
from pyspark.sql.types import StringType
from rtree import index
from shapely.geometry import asPoint

def build_point_in_poly(polygons):

    def point_in_poly(x, y):
        idx = index.Index(SparkFiles.get('index.idx')[:-4])
        for fid in idx.intersection([x, y]):
            speedups.enable()
            point = asPoint([x, y])
            (name, polygon) = polygons_broadcast.value[fid]
            if point.intersects(polygon):
                # Assume non-overlapping polygons, so return.
                return name

    tempdir = mkdtemp()
    basename = os.path.join(tempdir, 'index')
    try:
        idx = index.Index(basename, ((fid, polygon.bounds, None)
                          for (fid, (name, polygon)) in enumerate(polygons)))
        idx.close()
        sc.addPyFile(basename + '.idx')
        sc.addPyFile(basename + '.dat')
    finally:
        rmtree(tempdir)

    polygons_broadcast = sc.broadcast(polygons)
    return point_in_poly

# Build a UDF to find the airport code of a lat-lng poit.
sqlContext.registerFunction('getAirportCode', build_point_in_poly(airports),
                            StringType())
```

Spark doesn't have built-in GIS support but we can leverage Shapely and rtree, Python libraries based on libgeos (used by PostGIS) and libspatialindex, respectively.

Workflow


Explore Raw Data
Process Raw Data
Experiment Algorithms
Visualization
HTML
CSS
JS
Evaluate Results
Deployment
Rewrite Algorithms

Technical Workarounds

- Use Mesos coarse-grained mode with dynamic allocation and the external shuffler.
  - –Backported 13 commits from Spark master branch to fix dynamic allocation and launch multiple executors per slave.
- Deploy Python virtualenvs to Spark executors.
  - –Managed with requirements.txt files and pip-compile.
- Stitch Parquet files together (SPARK-11441).

Other Issues

- Spark SQL scans all partitions despite LIMIT (SPARK-12843)
- Mesos checkpoints (SPARK-4899)
  - Restart Mesos agent without killing Spark executors.
- Mesos oversubscription (SPARK-10293)
  - "Steal" idle but allocated resources.

Locality Sensitive Hashing by Spark

Tomorrow, Wednesday, June 8
5:25 PM – 5:55 PM
Room: Imperial

Alain Rodriguez, Fraud Platform, Uber
Kelvin Chu, Hadoop Platform, Uber

Other Resources

- Stream Computing & Analytics at Uber
  - http://www.slideshare.net/stonse/stream-computing-analytics-at-uber
- Spark at Uber
  - http://www.slideshare.net/databricks/spark-meetup-at-uber
- Career at Uber
  - https://www.uber.com/careers/

THANK YOU.

Feedback? Dara Adib <dara@uber.com>
Happy to discuss technical details.
No product/business questions please.

SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO