Jupyter and Spark on Mesos: Best Practices

June 21st, 2017

scrapinghub

Agenda

- About me
- What is Spark & Jupyter
- Demo
- How Spark+Mesos+Jupyter work together
- Experience
- Q & A

About me

- Graduated from EE @ Tsinghua Univ.
- Infrastructure Engineer @ Scrapinghub
- Contributor @ Apache Mesos & Apache Spark

Apache Spark

- Fast and general purpose cluster computing system
- Provides high level APIs in Java/Scala/Python/R
- Integration with Hadoop ecosystem

Why Spark

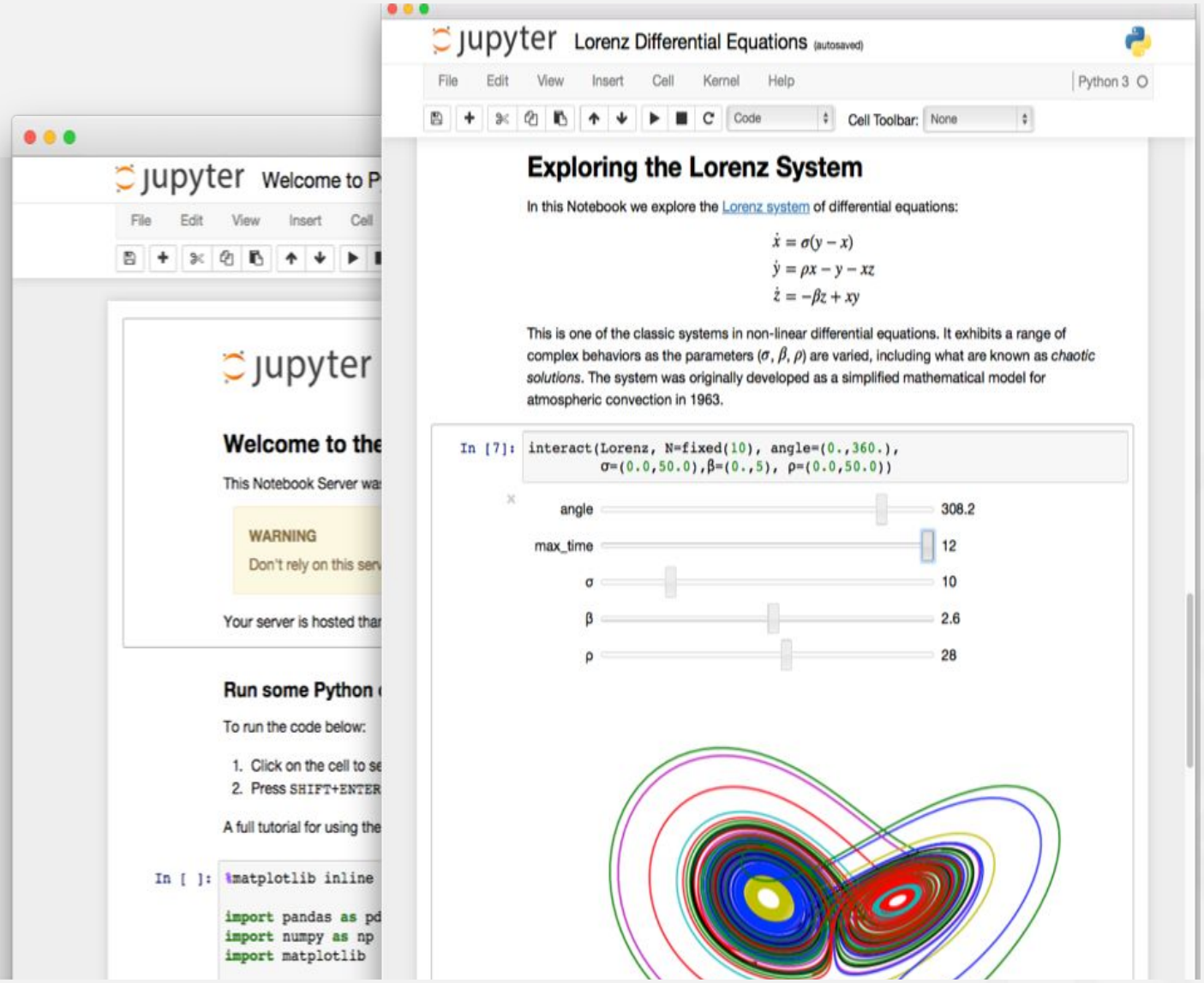
- Expressive API for distributed computing
- Support both streaming & batch
- Low level API (RDD) & High level DataFrame/SQL
- First-class Python/R/Java/Scala Support
- Rich integration with external data sources: JDBC, HBase, Cassandra, etc.

Jupyter Notebook Server

- IPython shell running in the web browser
- Not only code, also markdown & charts
- Interactive
- Ideal for demonstration & scratching

http://jupyter-notebook.readthedocs.io/en/latest/notebook.html

Jupyter Notebook Server



Recap

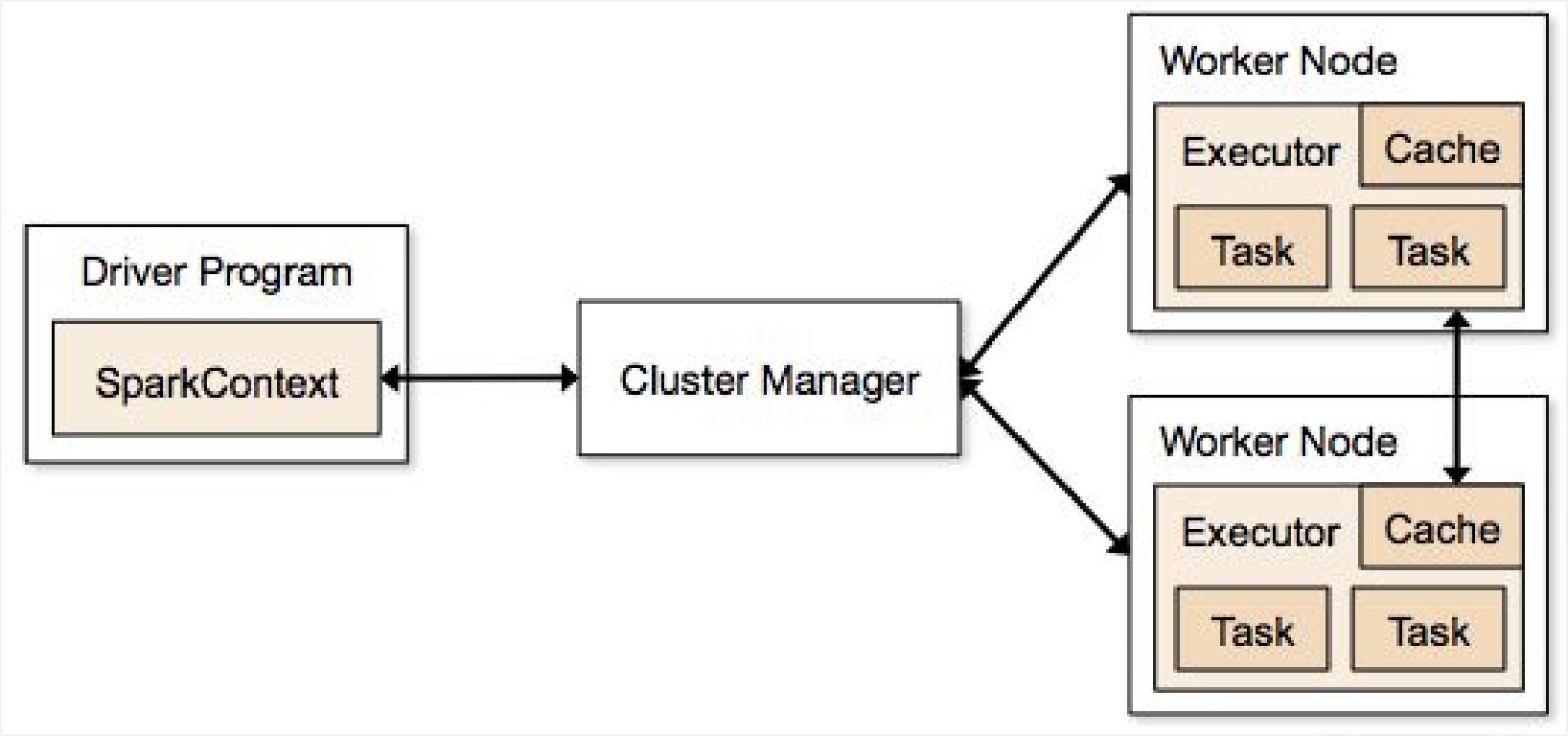
Prev:

- Introduction to Spark
- Introduction to Jupyter Notebook Server

Next:

- Why Spark on Mesos
- Why Spark+Mesos+Jupyter

Why Spark on Mesos



Why Spark on Mesos

- Run Spark drivers and executors in docker containers

  (avoid python dependency hell)

- Run any version of spark!

- Making use of our existing mesos cluster

- Reuse the monitoring system built for mesos

Why Spark + Jupyter Notebook

- Run in Local computer
  - Not enough storage capacity for large datasets
  - Not enough compute power to process them
- Run in company cluster
  - takes too long to set up
  - Hard to debug (only through logs)

Why Spark + Jupyter Notebook

- Run in Notebook
  - No need to set up - just on click
  - Easy to debug
  - Full access to the cluster’s compute power

Recap

Prev:

- Why Spark on Mesos
- Why Spark+Mesos+Jupyter

Next:

- Demo

DEMO

Recap

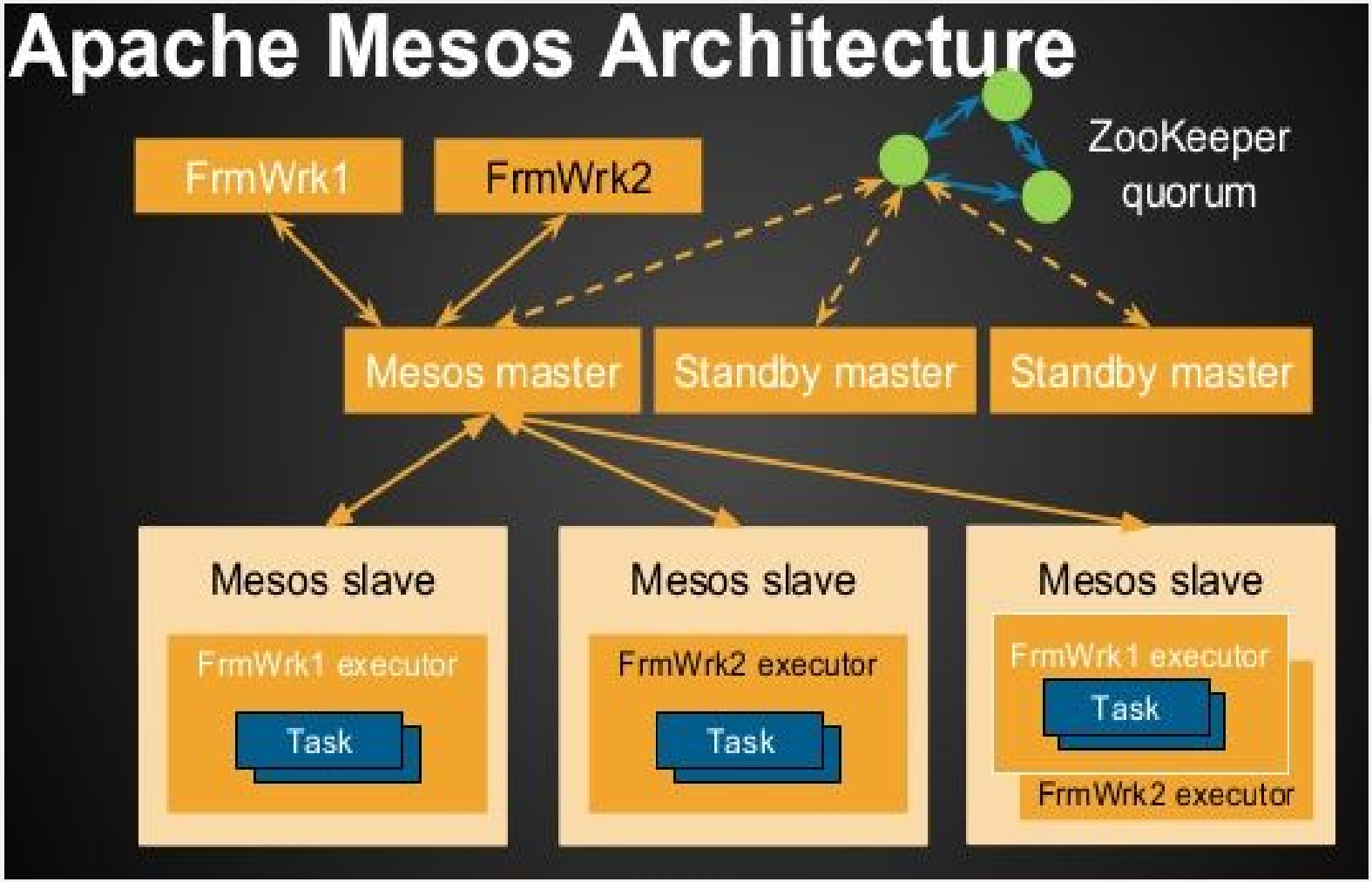
Prev:

- Demo

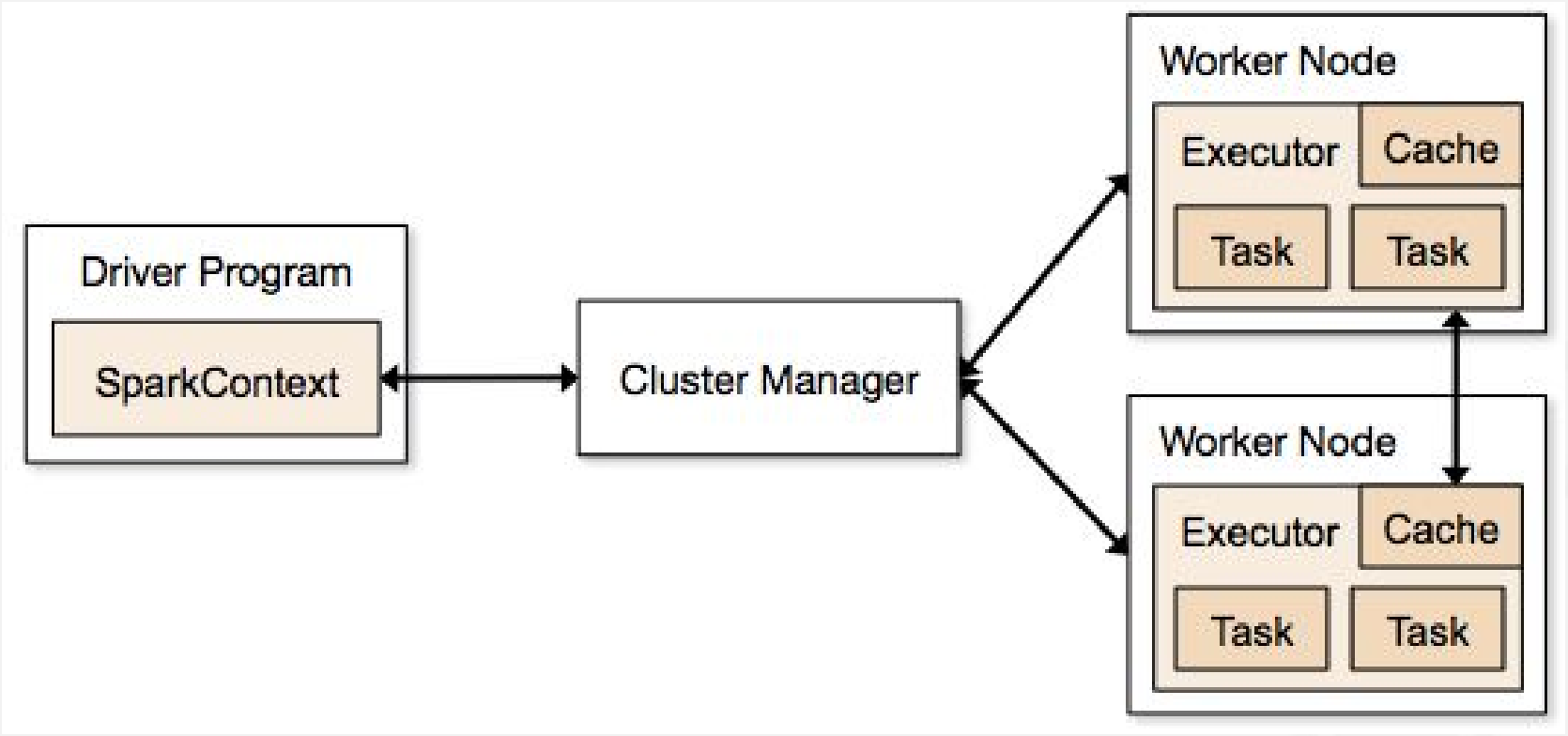
Next:

- How Spark and Mesos work together
- Experience & Caveats

Mesos & Spark: Mesos Architecture



Mesos & Spark: Spark Architecture



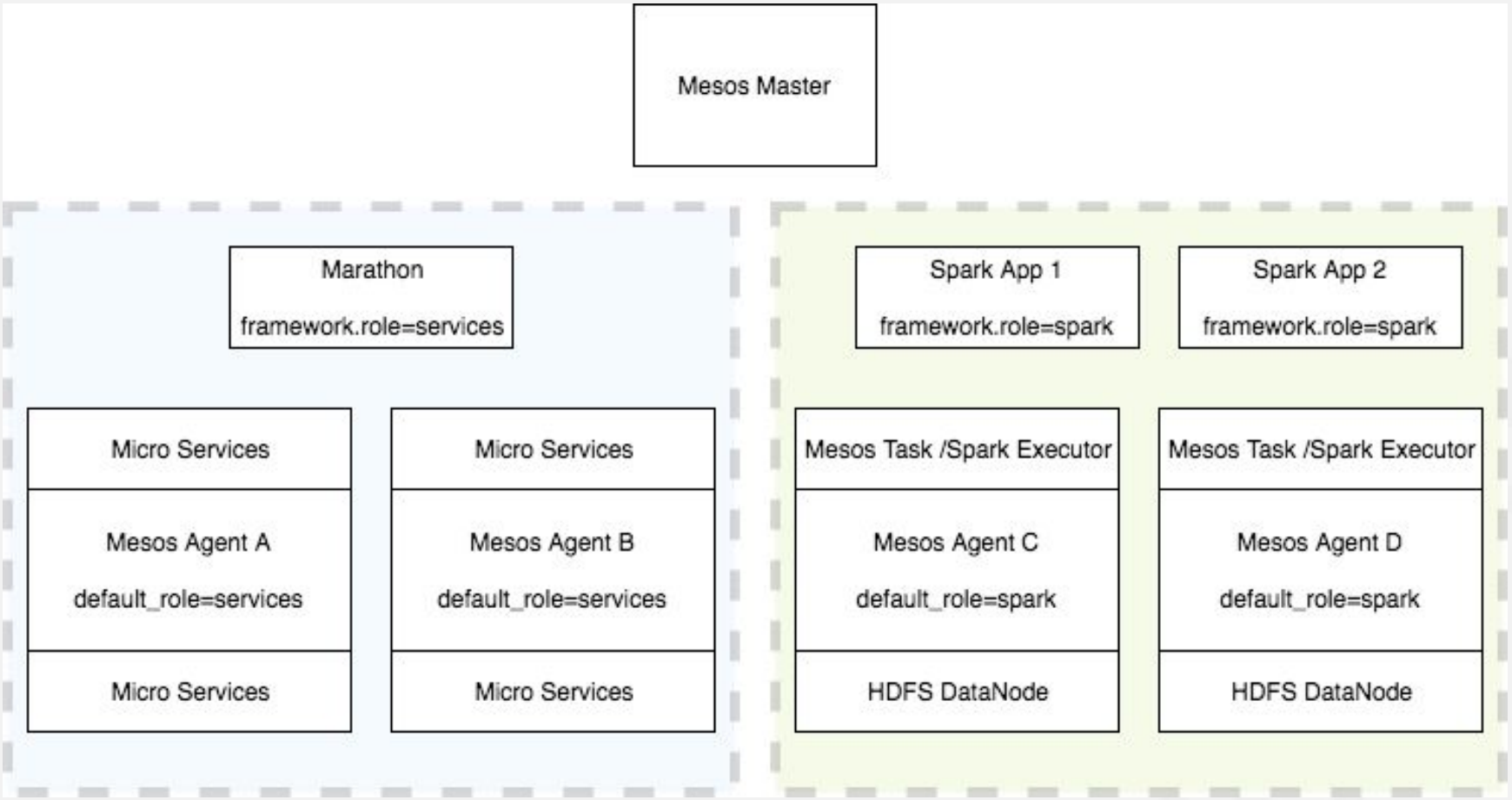
Mesos & Spark

- A Spark app/driver = a Mesos framework
- Spark executors = Mesos tasks

Mesos & Spark: Experience

- Single Cluster
- Marathon for long running services
- Constraints to pin spark tasks on certain nodes

Experience - Single Cluster



Experience - Single Cluster

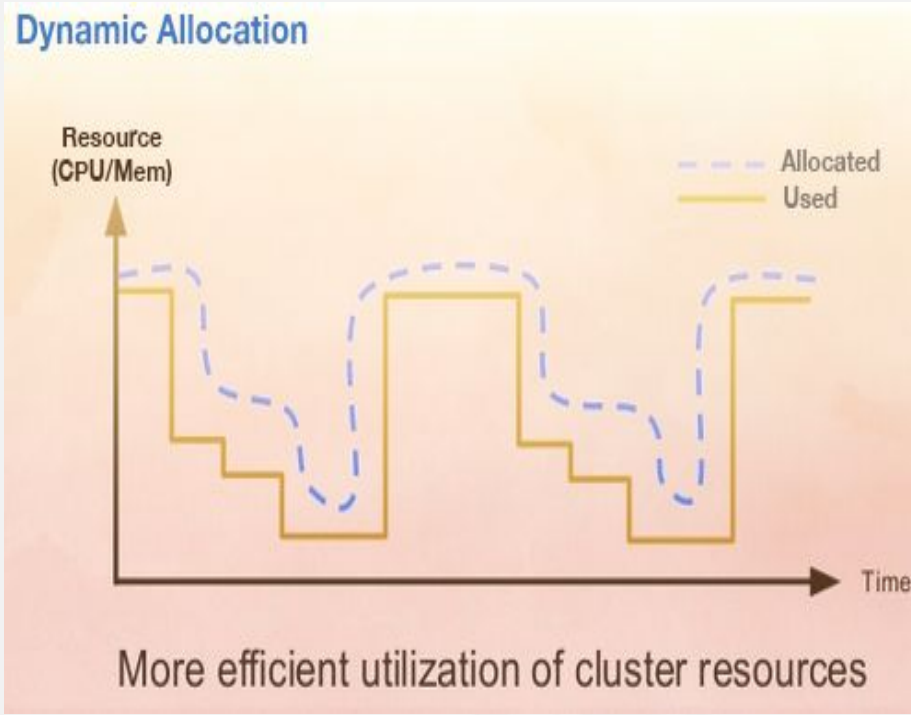
- Pros & Cons

Experience: Dynamic Allocation is a must

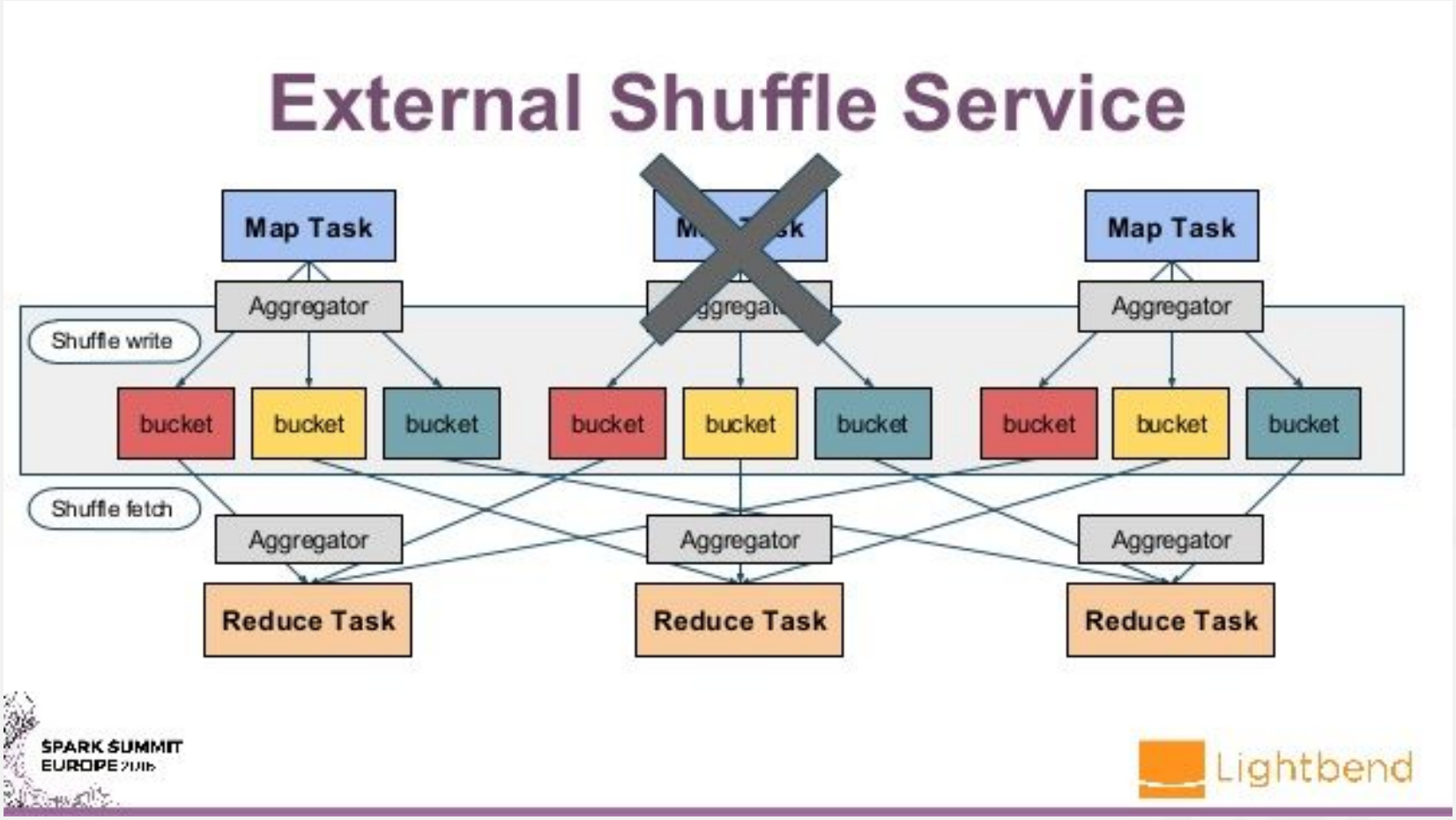
- People tend to leave their spark executors running, even if they end their day of work
- No resource available for new launched spark apps, even if the cluster is doing no work
- Enable dynamic allocation: idle spark executors are terminated after a while

Spark Dynamic Allocation

- Spark executors are:
  - Killed after being idle for a while
  - Launched later when there are tasks waiting in the queue
- Requires long-running “spark external shuffle service” on each mesos node



Spark Dynamic Allocation - External Shuffle Service



Experience: Battery-included docker base image

- Basics:
  - libmesos
  - java 8
- Libs:
  - Python 2 & Python 3 & libaries
  - Hadoop jars for AWS, kafka jars
- Configuration:
  - Resource spec (cpu/ram) for spark driver/executors
  - Dynamic allocation
  - Constraints: pin spark executors to colocate with HDFS DataNodes

Experience: Battery-included docker base image

```
# Default system properties included when running spark-submit.
# This is useful for setting default environmental settings.

spark.files                 file:///etc/hadoop/conf/hdfs-site.xml,file:///etc/hadoop/conf/core-site.xml
spark.mesos.coarse          true
spark.cores.max             2
spark.executor.cores        1
spark.executor.memory       1g
spark.mesos.role            spark
spark.mesos.constraints     role:spark-node
spark.mesos.uris            file:///root/.dockercfg

##########################################################################################
# Settings for dynamic resource allocation, DO NOT EDIT IT unless you know what you're doing.
##########################################################################################
spark.dynamicAllocation.enabled                         true
spark.shuffle.service.enabled                           true
# default 0
spark.dynamicAllocation.initialExecutors                2
# default 1min
spark.dynamicAllocation.executorIdleTimeout             10m
# default infinity
spark.dynamicAllocation.cachedExecutorIdleTimeout       1h
# default infinity
spark.dynamicAllocation.maxExecutors                    10
# The executor must share the same data dir with the external shuffle service
# See https://issues.apache.org/jira/browse/SPARK-17555
spark.mesos.executor.docker.volumes                    /tmp/spark:/tmp/spark:rw
spark.local.dir                                        /tmp/spark
```

Experience: Save Jupyter notebooks in database

- Jupyter does not support saving notebooks in databases
- but it provides a pluggable storage backend API
- pgcontents: Postgres backend, open sourced by Quantopian
- we ported it to support MySQL (straightforward thx to SQLAlchemy)

https://github.com/quantopian/pgcontents
https://github.com/scrapinghub/pgcontents/tree/mysql

Recap

Prev:

- How Spark and Mesos work together
- Experience & Caveats
  - Role & Constraints
  - Dynamic Allocation is a must

Next:

- Looking into the future
- Q & A

Looking into the Future

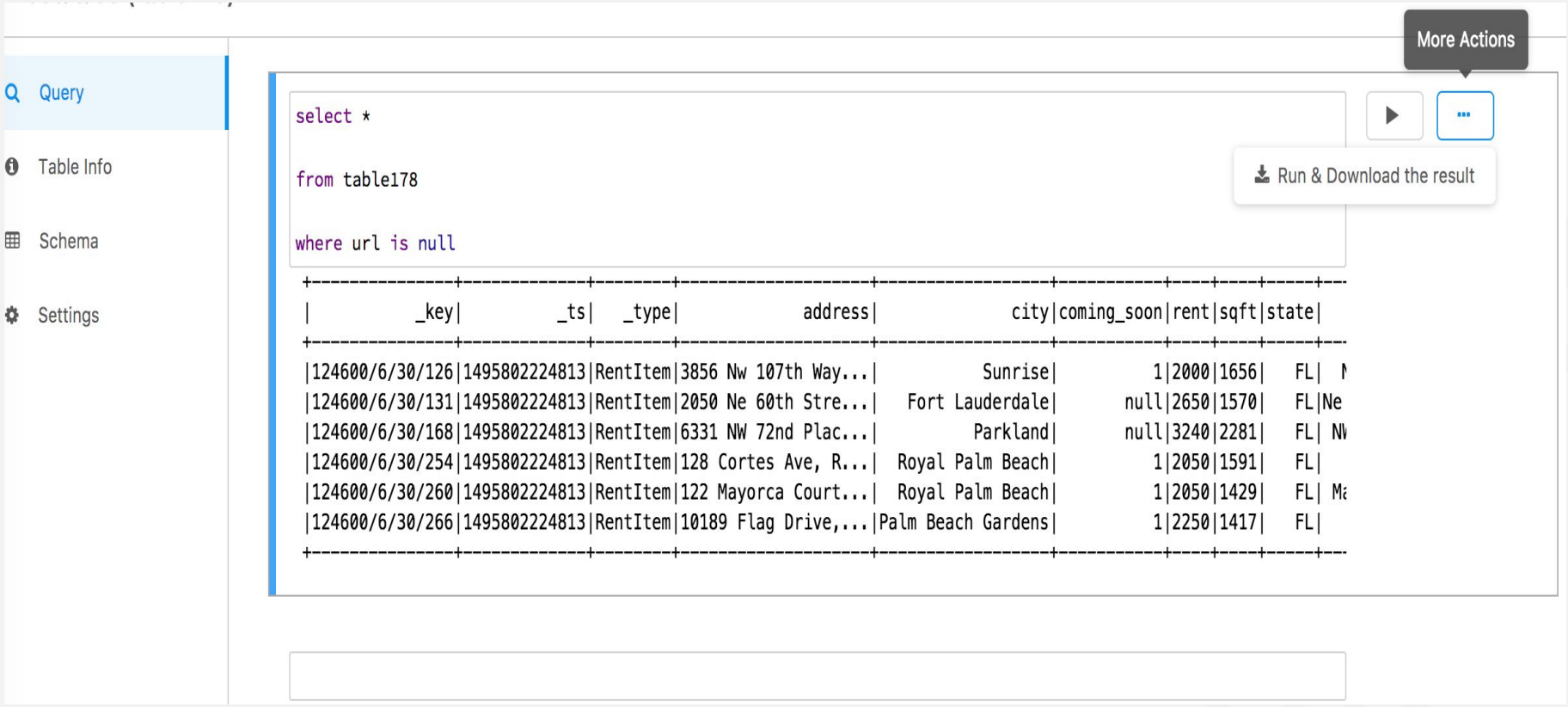
- Resource isolation between notebooks
- Python environment isolation between notebooks

Spark JobServer

- Learning spark+python is a bit too much for people like sales & QA
- But almost everyone knows about SQL
- So why not we just provide a web ui to execute spark sql?

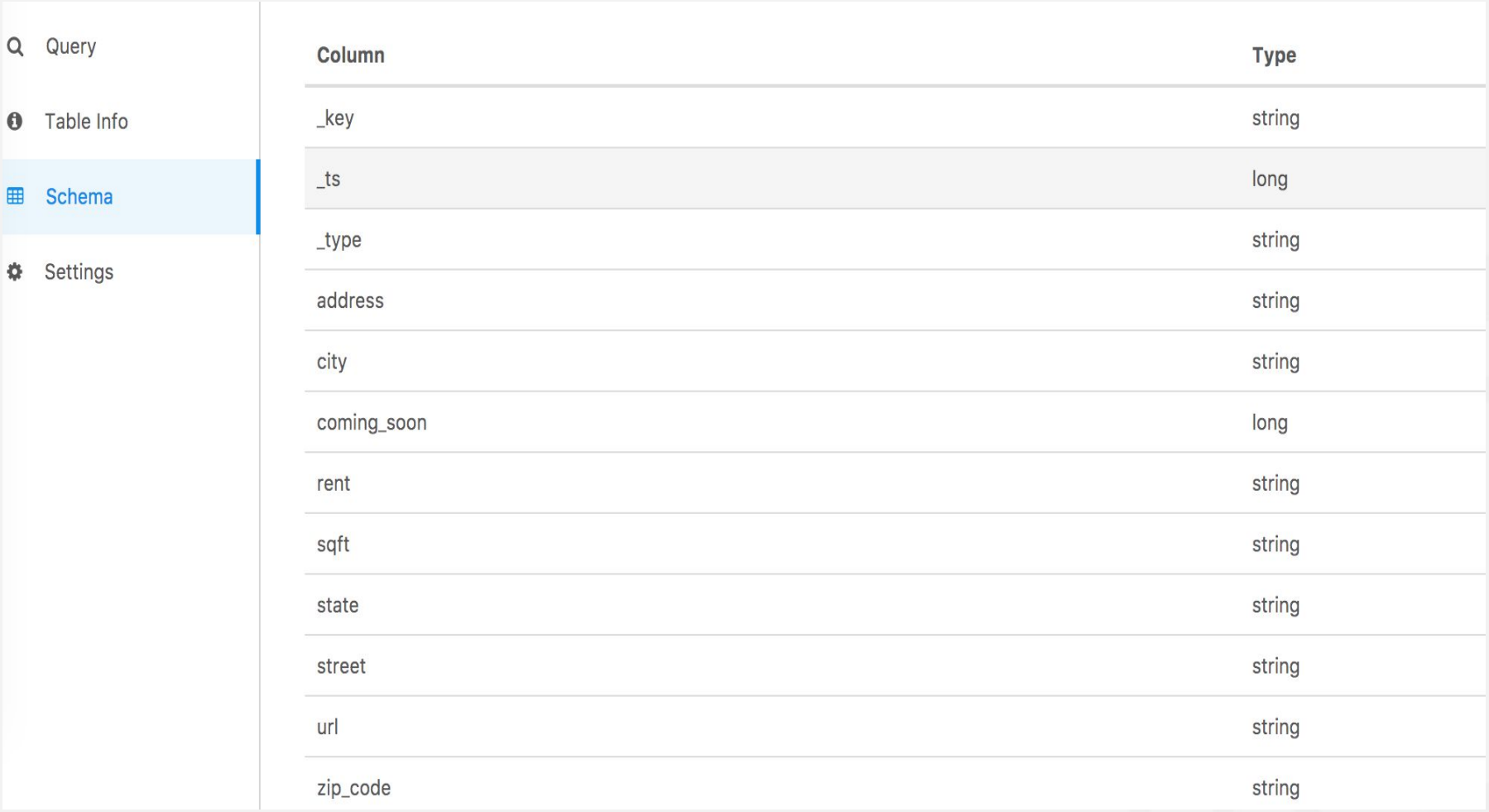
Spark JobServer

- Much like AWS Athena, but tailored to our own use cases



Spark JobServer

- Much like AWS Athena, but tailored to our own use cases



Q & A

scrapinghub