Big Data Engineer Bootcamp

Code 2

Agenda

Dev Environment

Work with Kafka

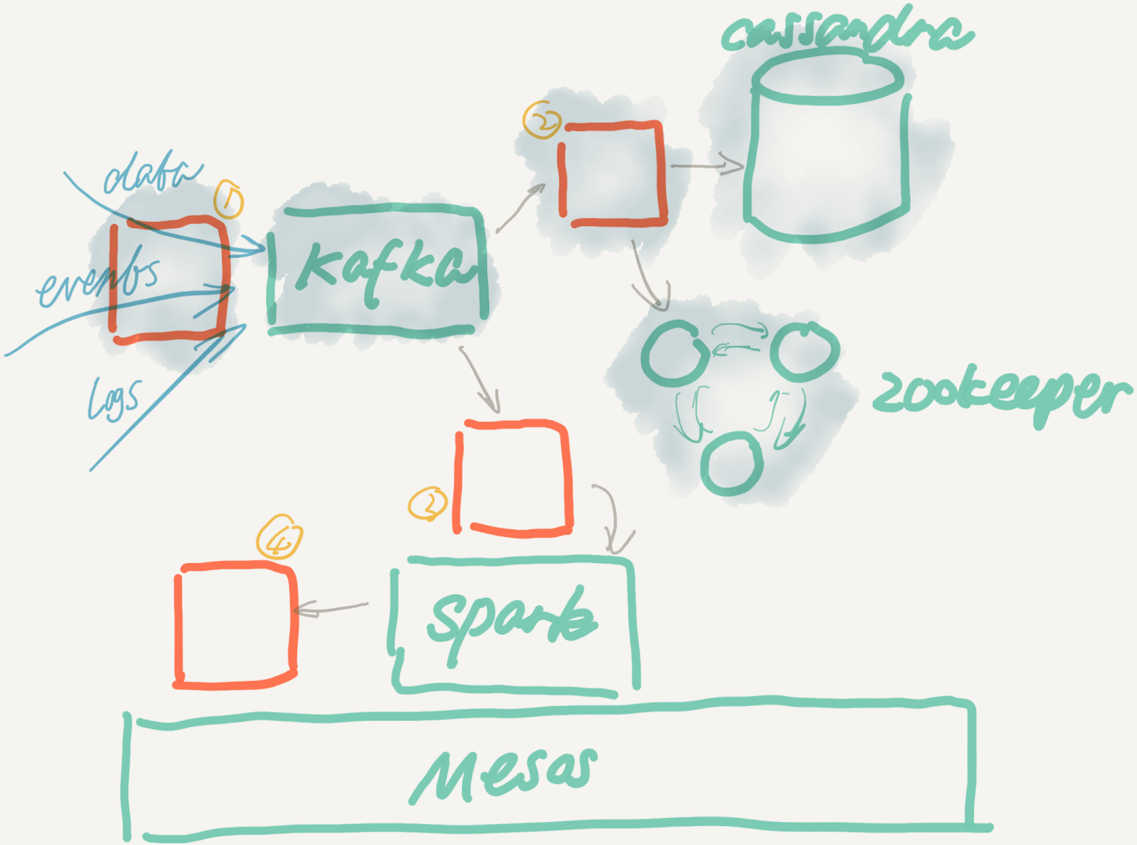
Work with Cassandra

Github Link

- https://github.com/UncleBarney/big-data-bootcamp

Project Structure

- Apache Kafka
- Apache Zookeeper
- Apache Cassandra



Start Docker Environment (MacOS, *nix)

- Have a docker-machine vm called bigdata
- Start a Zookeeper Container

  -
    ```
    docker run -d -p 2181:2181 -p 2888:2888 -p 3888:3888 --name zookeeper confluent/zookeeper
    ```
- Start a Kafka Container

  -
    ```
    docker run -d -p 9092:9092 -e KAFKA_ADVERTISED_HOST_NAME=`docker-machine ip bigdata` -e KAFKA_ADVERTISED_PORT=9092 --name kafka --link zookeeper:zookeeper confluent/kafka
    ```
  - If backtick is not working for you, use your virtual machine ip directly
- Start a Cassandra Container

  -
    ```
    docker run -d -p 7199:7199 -p 9042:9042 -p 9160:9160 -p 7001:7001 --name cassandra cassandra:3.7
    ```

Start Docker Environment (Windows)

- Have a docker-machine vm called bigdata
- Start a Zookeeper Container

  ```
  docker run -d -p 2181:2181 -p 2888:2888 -p 3888:3888 --name zookeeper confluent/zookeeper
  ```

- Start a Kafka Container

  ```
  docker run -d -p 9092:9092 -e KAFKA_ADVERTISED_HOST_NAME=`docker-machine ip bigdata` -e KAFKA_ADVERTISED_PORT=9092 --name kafka --link
  zookeeper:zookeeper confluent/kafka
  ```

  - If backtick is not working for you, use your virtual machine ip directly

- Start a Cassandra Container

  ```
  docker run -d -p 7199:7199 -p 9042:9042 -p 9160:9160 -p 7001:7001 --name cassandra cassandra:3.7
  ```

Why Use Docker For This

- Fast iteration
  - Develop once, deploy everywhere
  - Continuous integration, Continuous delivery
- Isolated environment
  - Experiment with unsafe stuff

Agenda

Dev Environment

Work with Kafka

Work with Cassandra

Functionality

- Write data to Kafka
  - Should be able to write to any kafka cluster
  - Should be able to write to any kafka topic
- Fetch data from stock exchange
  - Should be able to specify which stock

Work with Kafka Using Python

- `pip install schedule`
- `pip install kafka-python`
- `pip install googlefinance`
- `pip freeze > requirements.txt`

- 可以使用virtualenv来进行开发环境隔离
  - `pip install virtualenv`
  - `virtualenv env`
  - `source env/bin/active` (MacOS *nix)
  - 直接去env/Scripts/目录下运行active脚本 (Windows)

Code LifeCycle

- Help you release resources properly
  - ThreadPool
  - Database Connections
  - Network Connections
- Otherwise you might create leak on server side

Agenda

Dev Environment

Work with Kafka

Work with Cassandra

Functionality

- Read data from Kafka
    - Should be able to read from any kafka cluster
    - Should be able to read from any kafka topic
- Write data to Cassandra
    - Should be able to write to any Cassandra cluster
    - Should be able to write to any Cassandra table, etc

Work with Cassandra Using Python

- `pip install virtualenv`
- `virtualenv env`
- `pip install cassandra-driver`
- `pip freeze > requirements.txt`

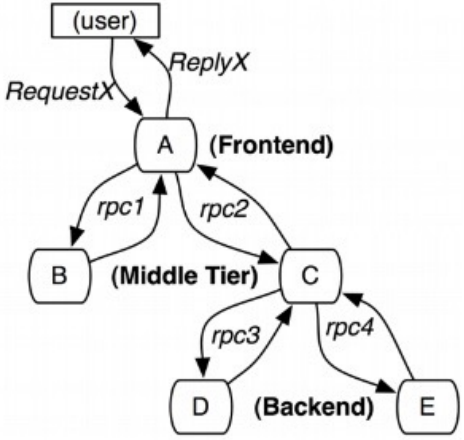
Work with Cassandra Using Python

- How to model data?
  - Our data is time series
  - Leveraging this will give us better performance

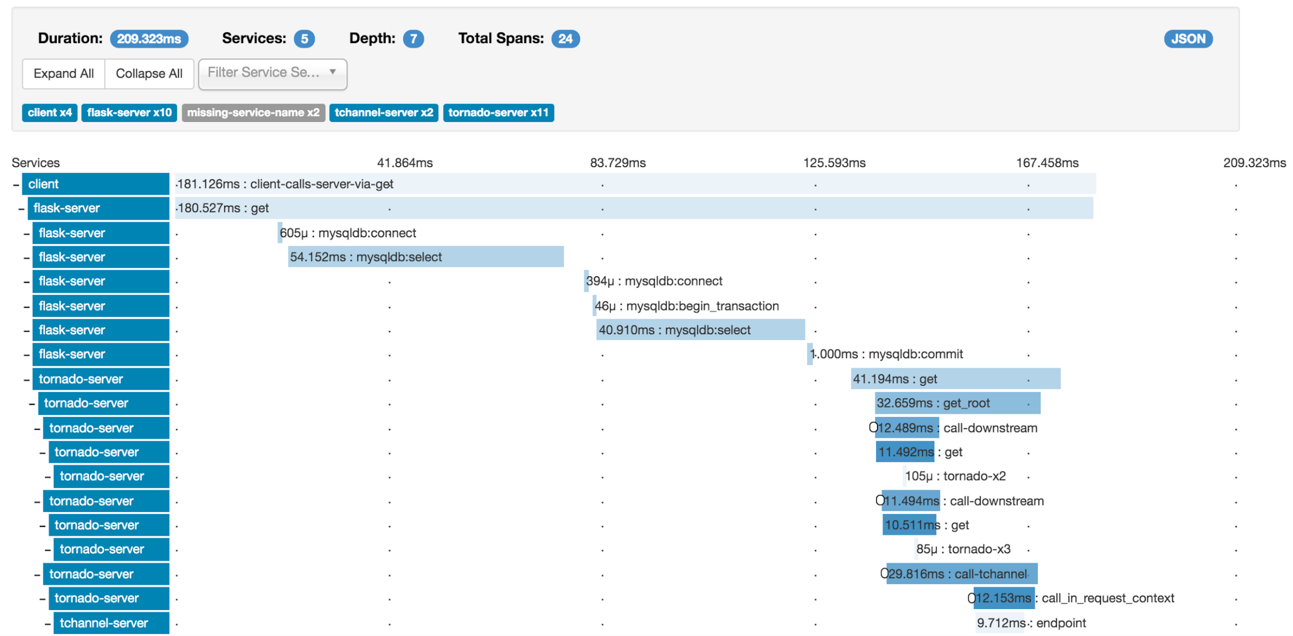
- CREATE KEYSPACE "stock" WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1} AND durable_writes = 'true';
- USE stock;
- CREATE TABLE stock (stock_symbol text, trade_time timestamp, trade_price float, PRIMARY KEY (stock_symbol,trade_time));

Logging

- Log is your god when things go south
- Sometimes, logging is not enough in distributed system



Tracing

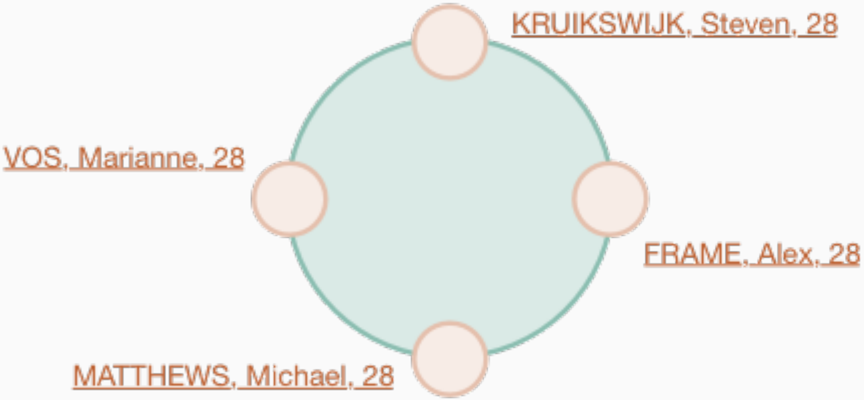


Duration: 209.323ms **Services:** 5 **Depth:** 7 **Total Spans:** 24

client x4 flask-server x10 missing-service-name x2 tchannel-server x2 tornado-server x11

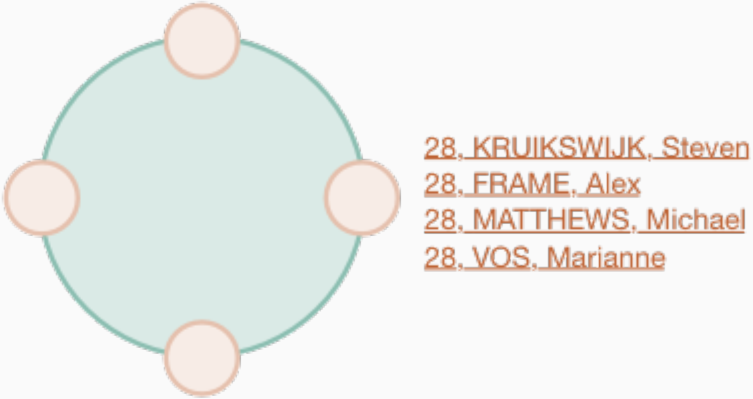
```
Services                           41.864ms     83.729ms     125.593ms     167.458ms     209.323ms
– client            181.126ms : client-calls-server-via-get
 – flask-server     180.527ms : get
  – flask-server    605µ : mysqldb:connect
  – flask-server    54.152ms : mysqldb:select
  – flask-server    394µ : mysqldb:connect
  – flask-server    46µ : mysqldb:begin_transaction
  – flask-server    40.910ms : mysqldb:select
  – flask-server    1.000ms : mysqldb:commit
  – tornado-server  41.194ms : get
   – tornado-server  32.659ms : get_root
    – tornado-server  12.489ms : call-downstream
     – tornado-server  11.492ms : get
      – tornado-server  105µ : tornado-x2
    – tornado-server  11.494ms : call-downstream
     – tornado-server  10.511ms : get
      – tornado-server  85µ : tornado-x3
    – tornado-server  29.816ms : call-tchannel
     – tornado-server  12.153ms : call_in_request_context
      – tchannel-server  9.712ms : endpoint
```

Cassandra Data Modeling



Stored by last name

Multiple partitions,
non-sequential rows

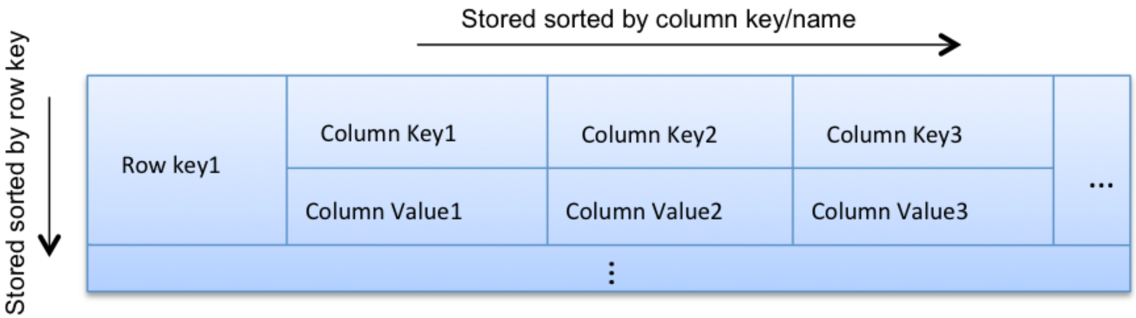


Stored by age

Single partition,
sequential rows

Cassandra Data Modeling

- `SortedMap<String, SortedMap<ColumnKey, ColumnValue>>`



Stored sorted by column key/name

Stored sorted by row key

Row key1

Column Key1

Column Value1

Column Key2

Column Value2

Column Key3

Column Value3

...

⋮

Further Reading

- Google Dapper Paper: http://research.google.com/pubs/pub36356.html
- Zipkin: http://zipkin.io
- Cassandra Data Modeling
    - http://www.planetcassandra.org/blog/the-most-important-thing-to-know-in-cassandra-data-modeling-the-primary-key/
- HBase Internal
    - https://www.mapr.com/blog/in-depth-look-hbase-architecture

Before Next Class

- `docker pull mesosphere/mesos-master:0.28.0-2.0.16.ubuntu1404`
- `docker pull mesosphere/mesos-slave:0.28.0-2.0.16.ubuntu1404`
- `docker pull mesosphere/marathon:v1.1.1`
- `docker pull redis:alpine`