Spark Uber Development Kit

Kelvin Chu, Hadoop Platform, Uber
Gang Wu, Hadoop Platform, Uber

Spark Summit 2016
June 07, 2016

UBER

Uber Mission

“Transportation as reliable as running water, everywhere, for everyone”

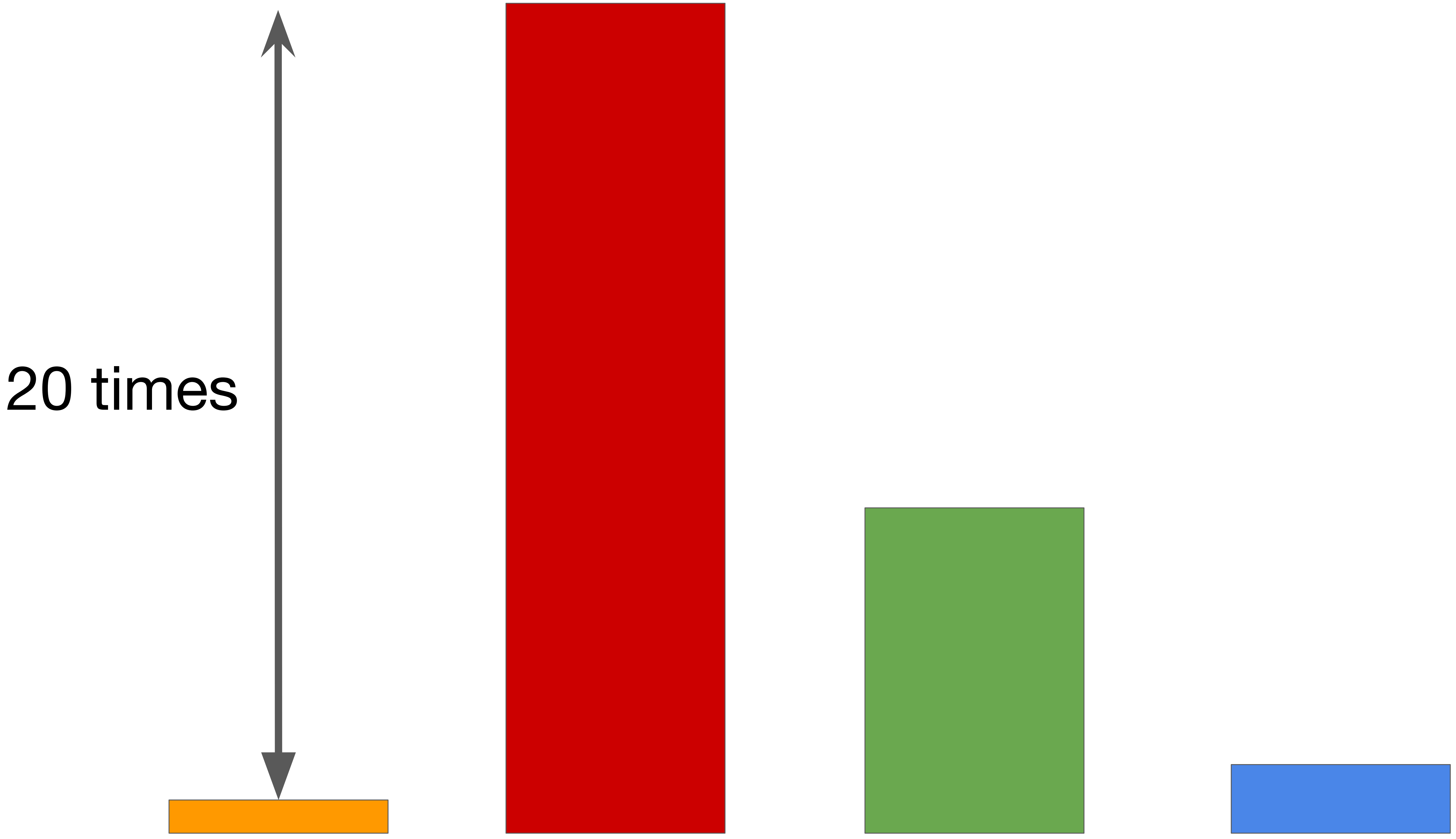
About Us

- Hadoop team of Data Infrastructure at Uber
- Schema systems
- HDFS data lake
- Analytics engines on Hadoop
- Spark computing framework and toolings

Execution Environment

Complexity

Cluster Sizes



YARN

Mesos

Docker

JVM

Parquet

ORC

Sequence

Text

Home Built Services

Hive

Kafka

ELK

Consequence:

Pretty hard for beginners, sometimes hard for experienced users too.

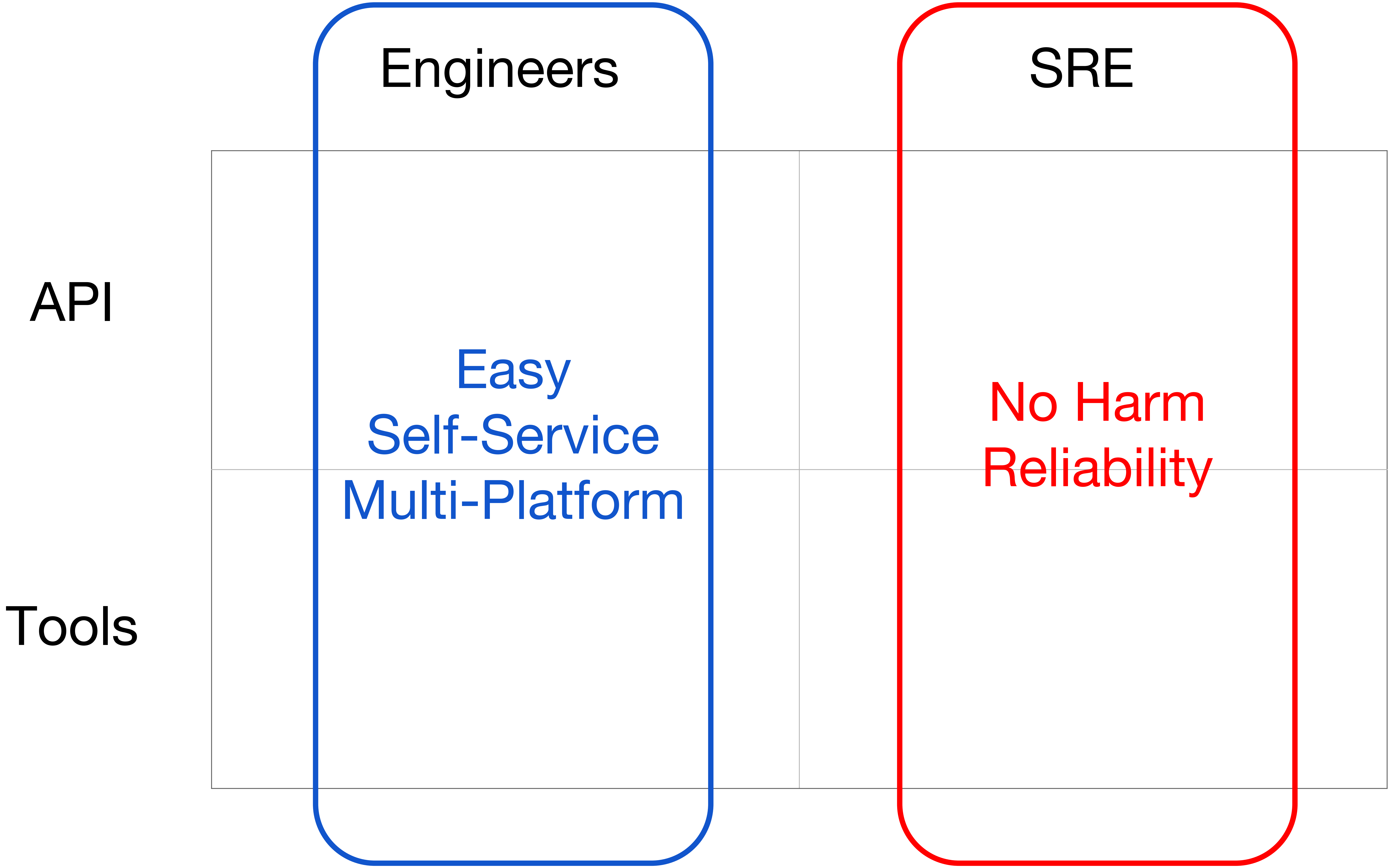
Goals:

Multi-Platform: Abstract out environment

Self-Service: Create and run Spark jobs super easily

Reliability: Prevent harm to infrastructure systems

	Engineers	SRE
API		
Tools		



Engineers

SRE

API

Tools

Easy
Self-Service
Multi-Platform

No Harm
Reliability

	Engineers	SRE
API	• SCBuilder • Kafka dispersal	
Tools	• SparkPlug	

SCBuilder

Encapsulate cluster environment details

- Builder Pattern for SparkContext
- Incentive for users:
  - performance optimized (default can’t pass 100GB)
  - debug optimized (history server, event logs)
  - don’t need to ask around YARN, history servers, HDFS configs
- Best practices enforcement:
  - SRE approved CPU and memory settings
  - resource efficient serialization

Kafka Dispersal

Kafka as data sink of RDD result

```
publish(data: RDD, topic: String, schemaId: Int, appId: String)
```

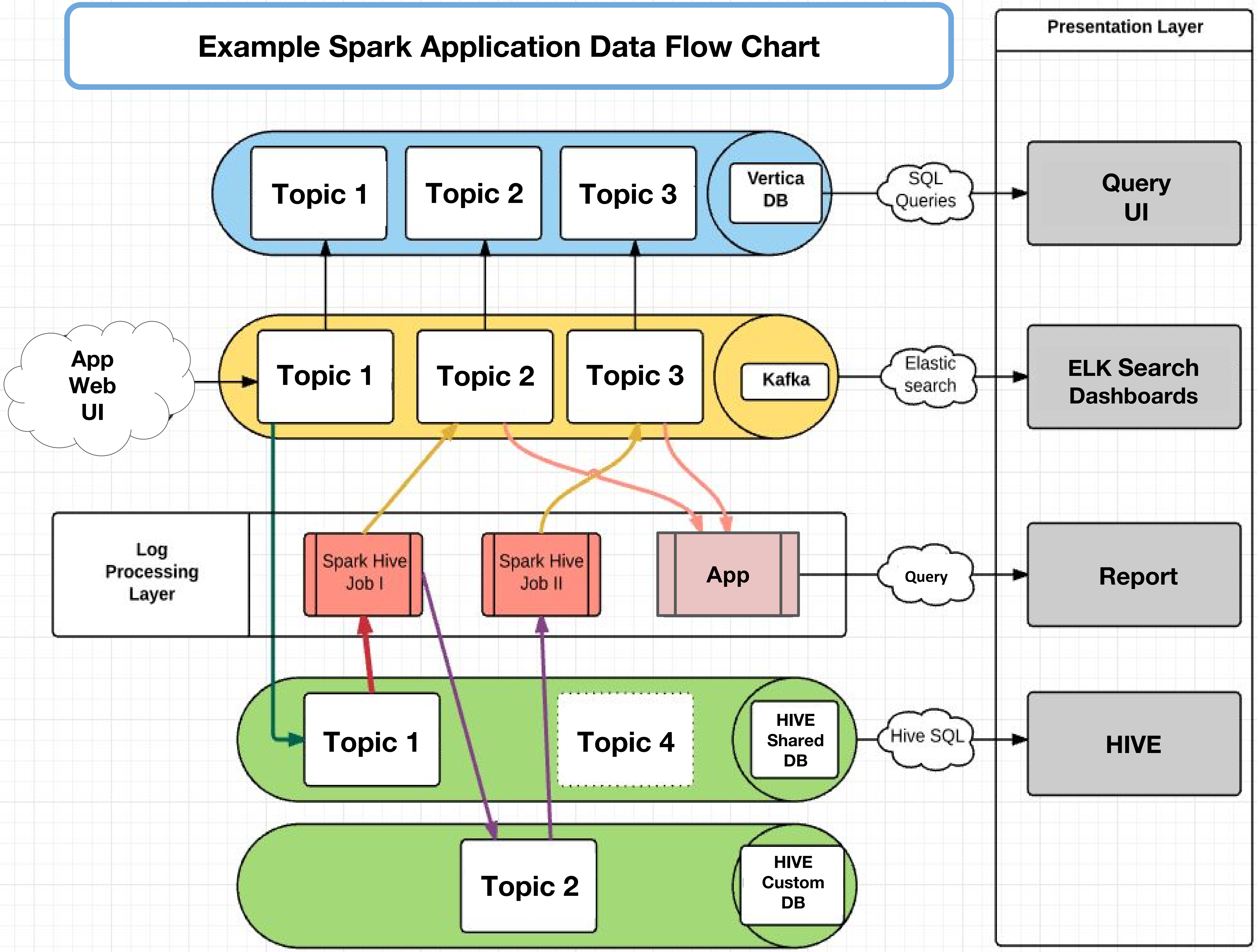
- Incentive for users:
  - RDD as first class citizen => parallelization
  - built-in HA
- Best practices enforcement:
  - rate limiting
  - message integrity by schema
  - bad messages tracking

SparkPlug

Kickstart job development

- A collection of popular job templates
  - Two commands to run the first job in Dev
- One use case per template
  - e.g. Ozzie + SparkSQL + Incremental processing
  - e.g. Incremental processing + Kafka dispersal
- Best Practices
  - built-in unit tests, test coverage, Jenkins
  - built-in Kafka, HDFS mocks

Example Spark Application Data Flow Chart



NUMBER OF EVENTS
View ▸ | 🔍 Zoom Out | ● message
● msg.event_name : SkillRecord (0) count per 1h | (789 hits)
100
80
60
40
20
0
00:00
03-03
00:00
03-05
00:00
03-07
EVENT NAME
1000
800
600
400
200
0
● OperatorSession (789)
● Missing field (0)
● Other values (0)

Title: Job

Term	Count	Action
TERM_NAME_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	404	🔍 ⊘
TERM_NAME_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	58	🔍 ⊘
TERM_NAME_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	57	🔍 ⊘
TERM_NAME_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	53	🔍 ⊘
TERM_NAME_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	40	🔍 ⊘
TERM_NAME_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	33	🔍 ⊘
TERM_NAME_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	25	🔍 ⊘
TERM_NAME_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	22	🔍 ⊘
TERM_NAME_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	14	🔍 ⊘
TERM_NAME_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	6	🔍 ⊘
Missing field	0	🔍 ⊘
Other values	77	

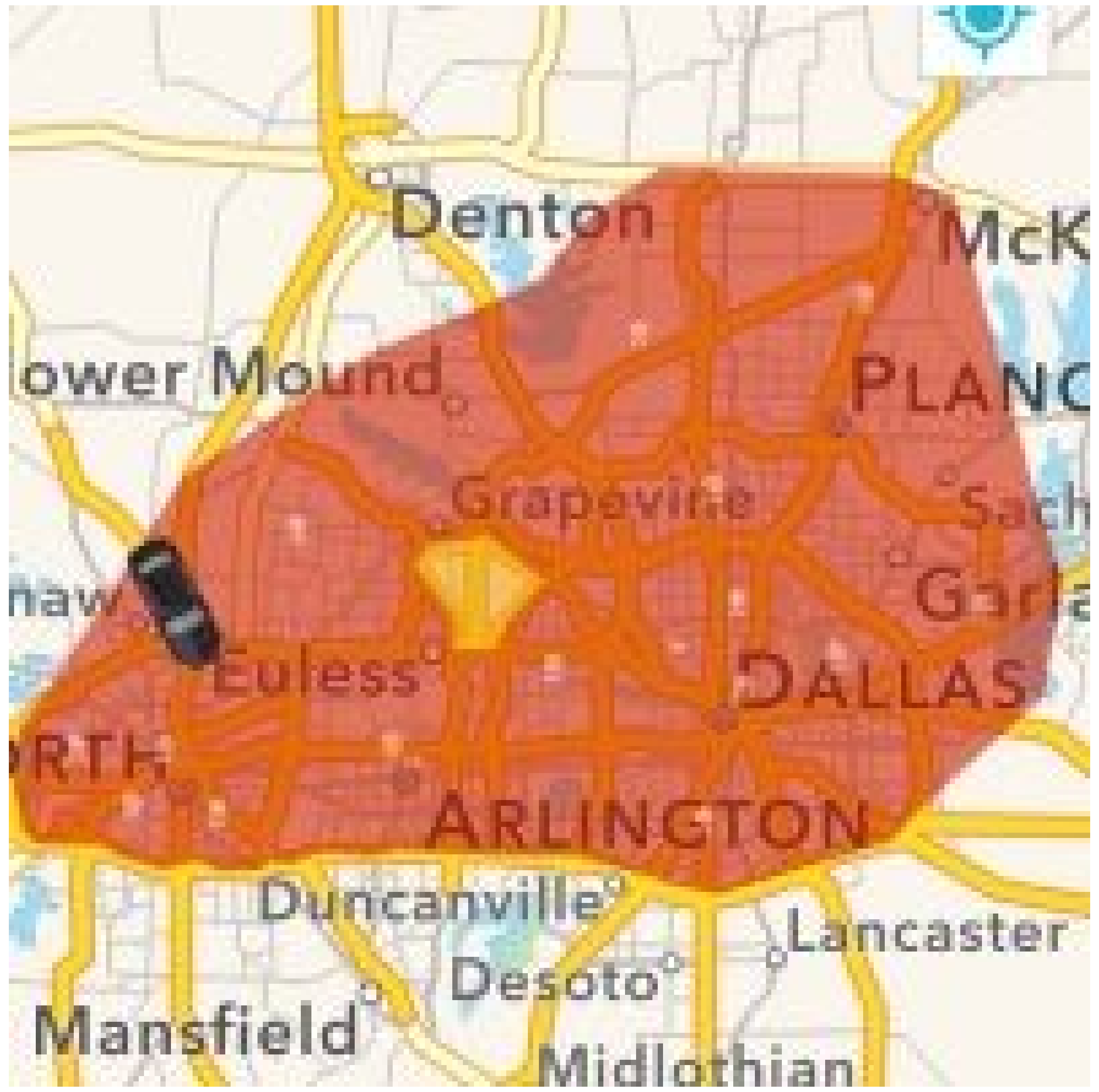
ENV

Term	Count	Action
production	412	🔍 ⊘
staging	377	🔍 ⊘
Missing field	0	🔍 ⊘
Other values	0	

	Engineers	SRE
API	• Geo-spatial processing • SCBuilder • Kafka dispersal	
Tools	• SparkPlug	

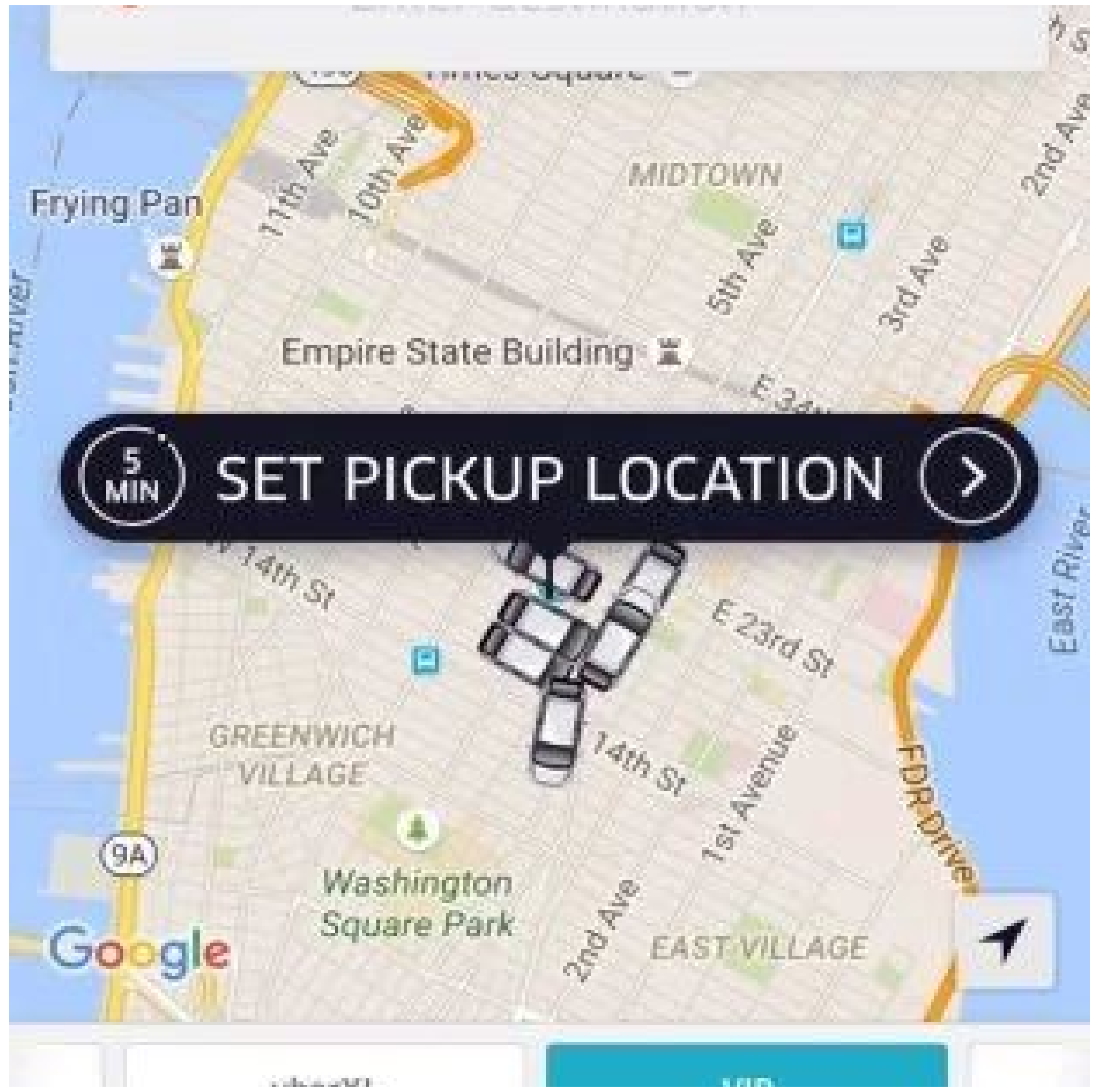
GeoSpatial UDF

Commonly used UDFs



within(trip_location, city_shape)

Find if a car is inside a city

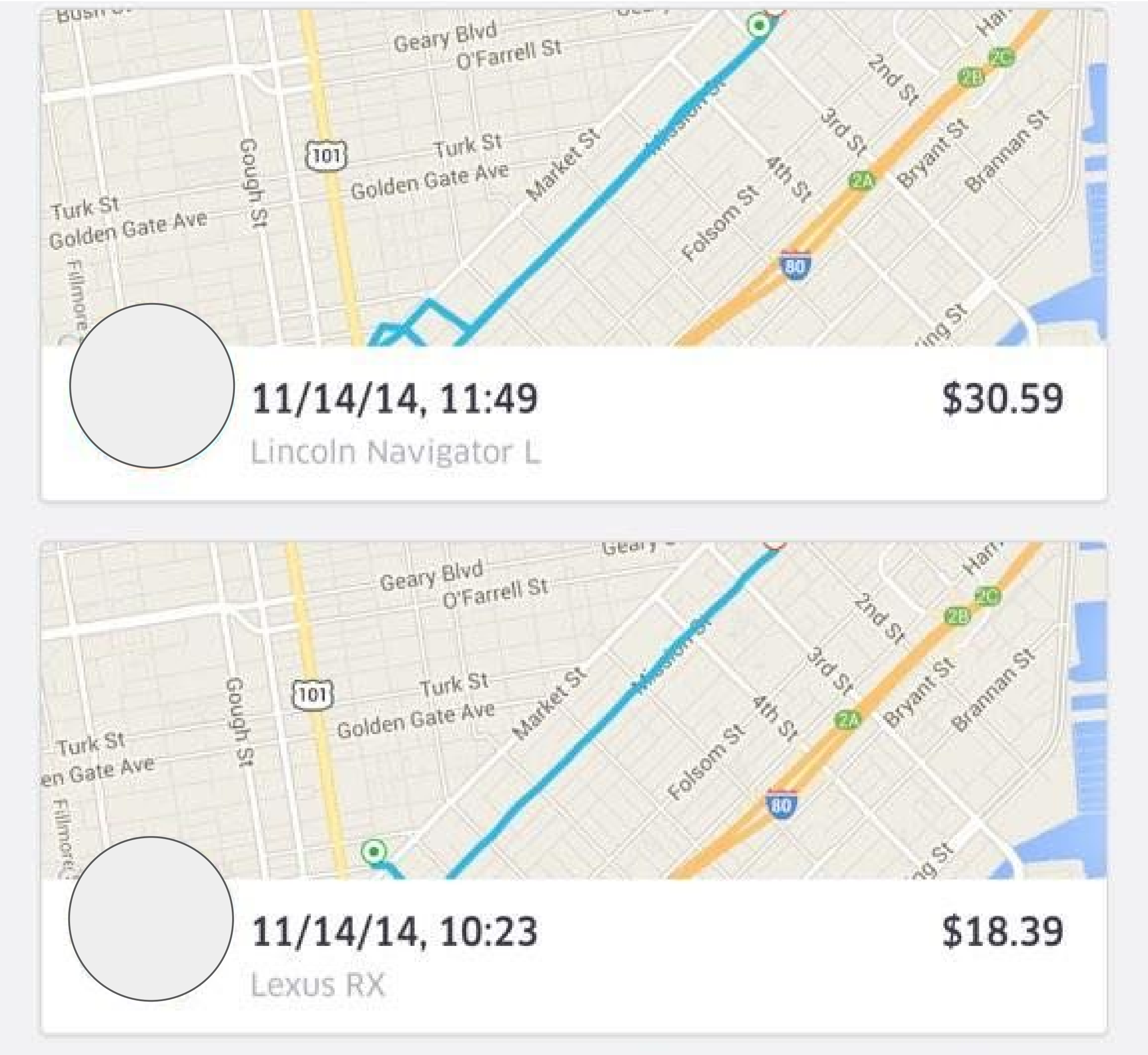


contains(geofence, auto_location)

Find all autos in one area

GeoSpatial UDF

Commonly used UDFs

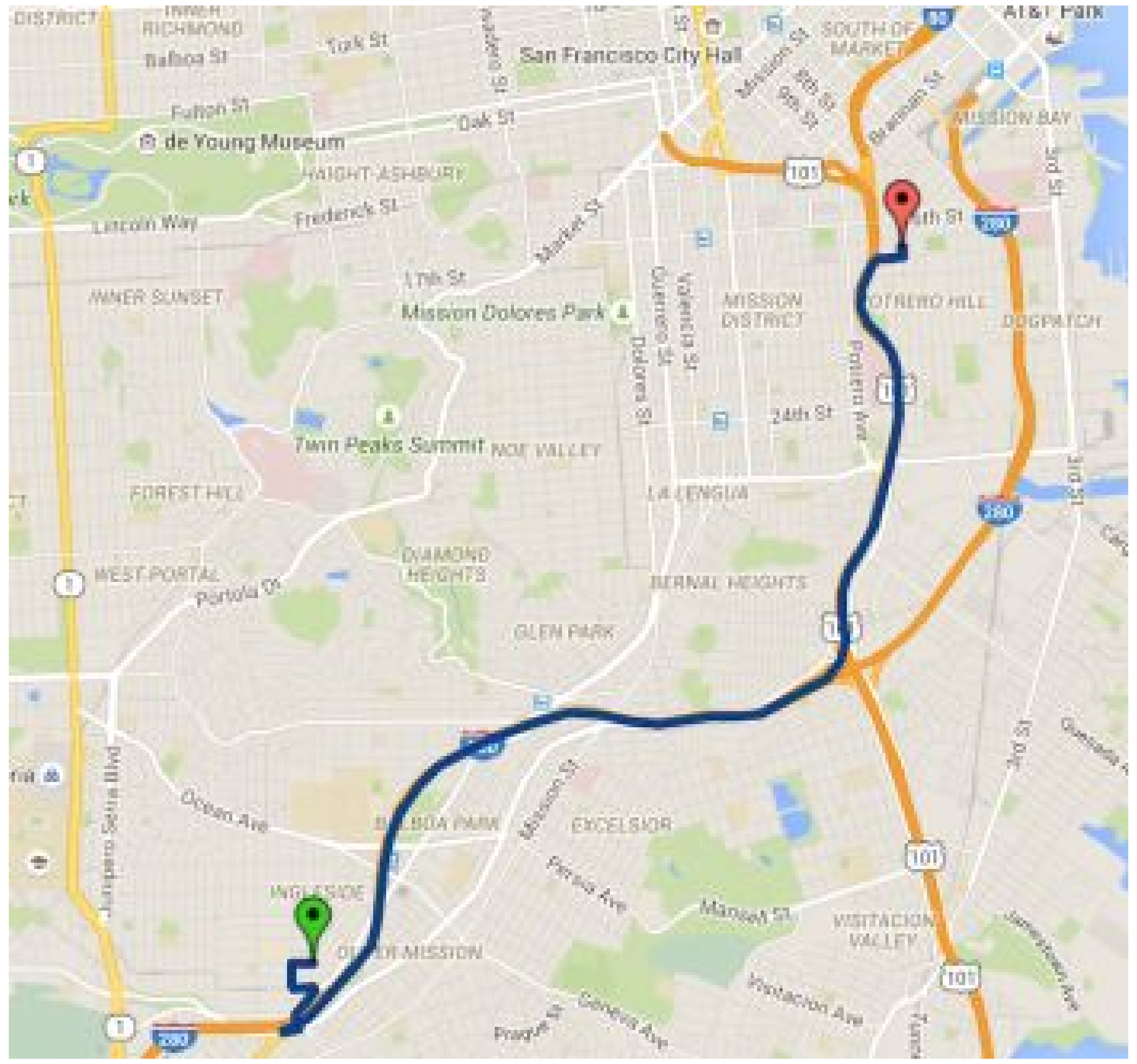


11/14/14, 11:49 $30.59

Lincoln Navigator L

11/14/14, 10:23 $18.39

Lexus RX



overlaps(trip1, trip2)

Find trips that have similar routes

intersects(trip_location, gas_location)

Find all gas stations a trip route has passed by

Spatial Join

Common query at Uber

Objective: associate all trips with city_id for a single day.

```
SELECT trip.trip_id, city.city_id
FROM trip JOIN city
WHERE contains(city.city_shape, trip.start_location)
AND trip.datestr = ‘2016-06-07’
```

Spatial Join

Problem

It takes nearly ONE WEEK to run at Uber’s data scale.

1. Spark does not have broadcast join optimization for non-equation join.

2. Not scalable, only one executor is used for cartesian join.

Spatial Join

Build a UDF to broadcast geo-spatial index

Spatial Join

Runtime Index Generation

1. Build Index

Index data is small but change often (city table)

Get fields from geo tables (city_id and city_shape)

Build QuadTree or RTree index at Spark Driver

Spatial Join

Executor Execution

2. Broadcast Index

UDF code is part of the Spark UDK jar.

⇒ get_city_id(location), returns city_id of a location

Use the broadcasted spatial index for fast spatial retrieval

Spatial Join

Runtime UDF Generation

3. Rewrite Query (2 mins only! compared to 1 week before)

```
SELECT
    trip_id,  get_city_id(start_location)
FROM
    trip
WHERE
    datestr = ‘2016-06-07’
```

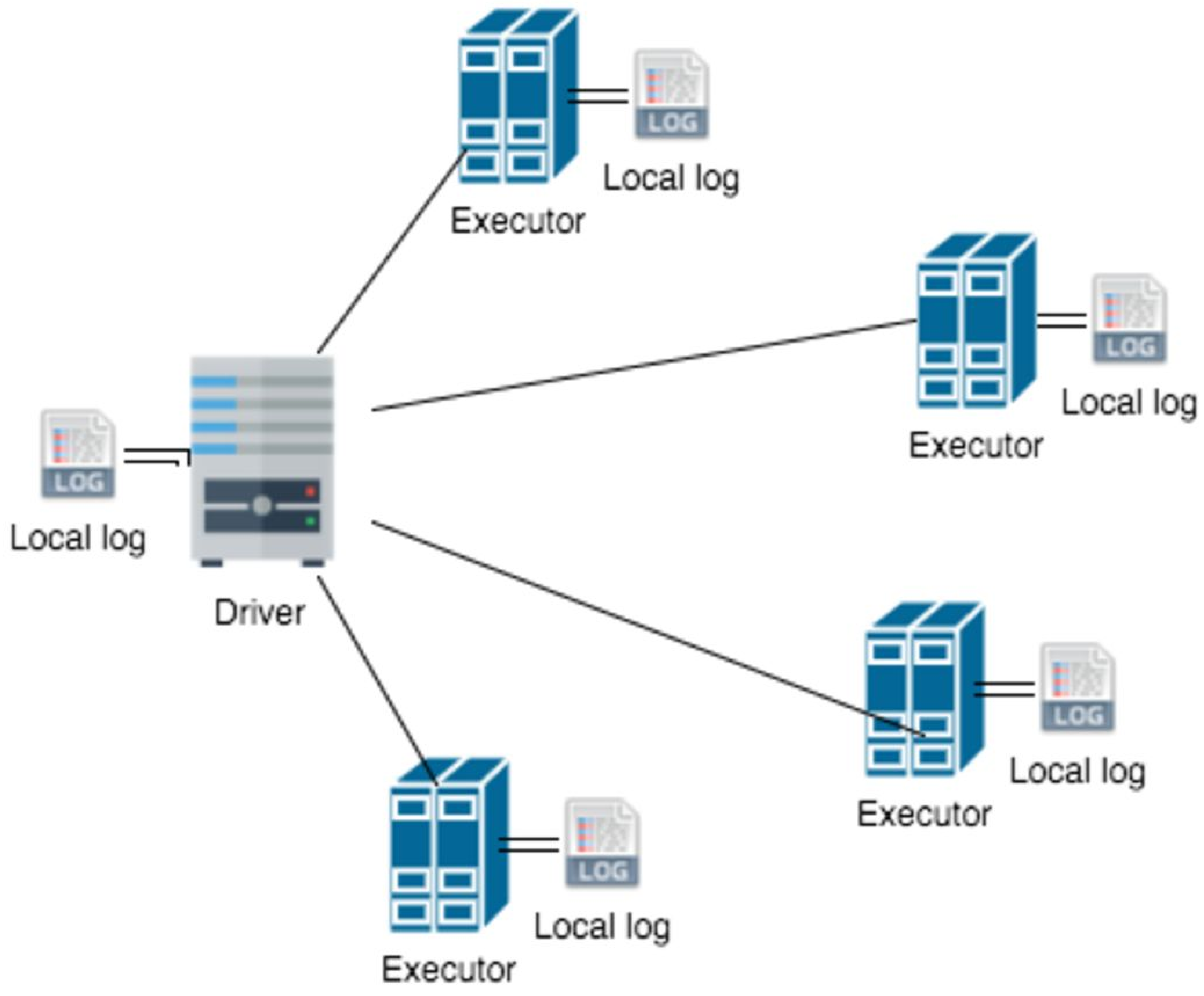
	Engineers	SRE
API	• Geo-spatial processing • SCBuilder • Kafka dispersal	
Tools	• SparkChamber • SparkPlug	

Spark Debugging

1. **Tons of local log files across many machines.**

2. **Overall file size is huge and difficult to be handled by a single machine.**

3. **Painful for debugging, which log is useful?**



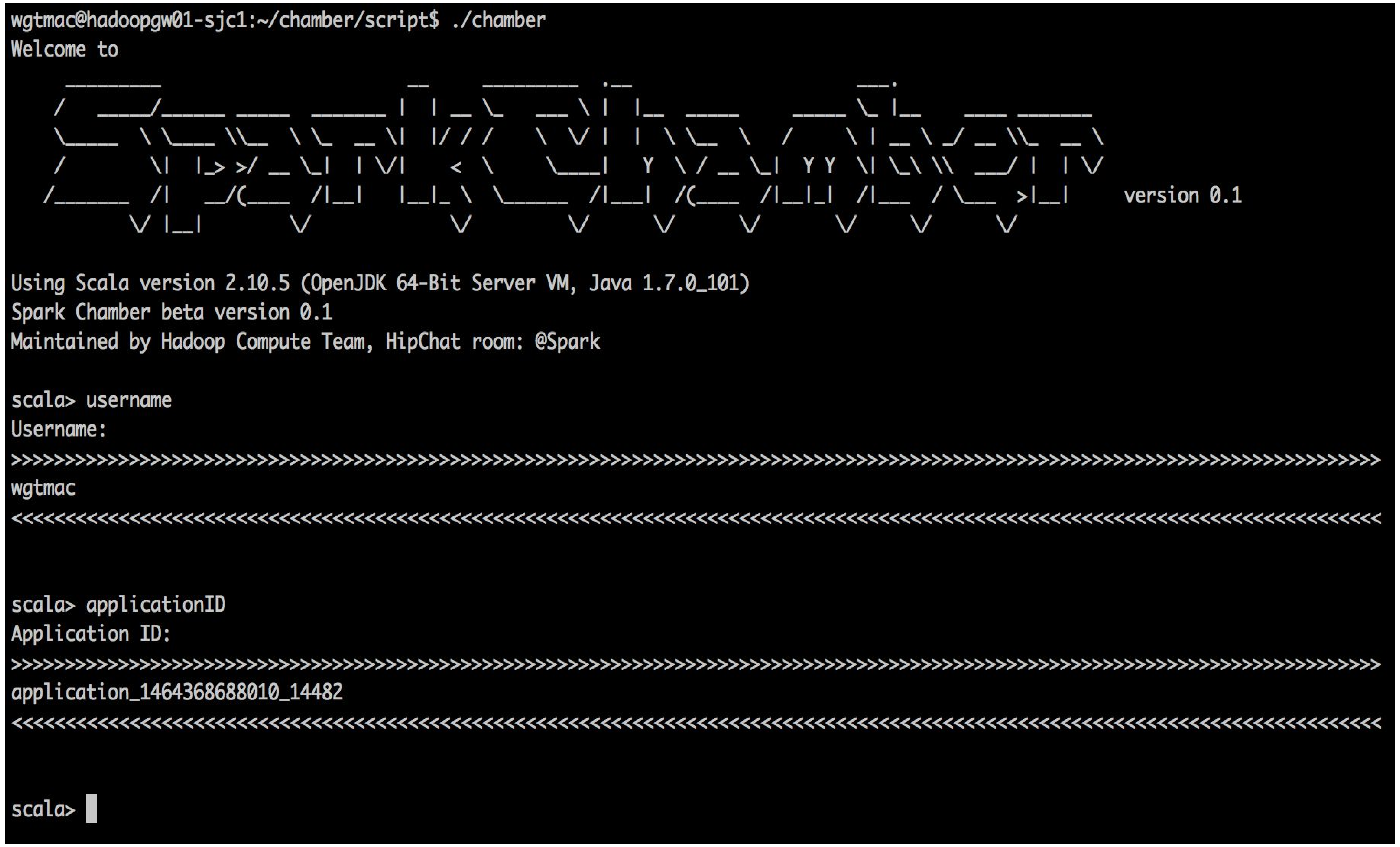
Spark Chamber

Distributed Log Debugger for Spark

Interactive

Extend Spark Shell by Hooks.

Easy to adopt for Spark developers.



```
wgtmac@hadoopgw01-sjc1:~/chamber/script$ ./chamber
Welcome to
    _________                   __    _________ .__                  ___.
   /   _____/_____ _____  _______|  | __\_   ___ \|  |__ _____    _____\_ |__   ____ _______
   \_____  \\____ \\__  \ \_  __ \|  |/ / /    \  \/|  |  \\__  \  /     \| __ \_/ __ \\_  __ \
   /        \|  |_> >/ __ \_|  | \/|    <  \     \___|   Y  \/ __ \_|  Y Y  \| \_\ \  ___/ |  | \/
  /_______  /|   __/(____  /|__|   |__|_ \ \______  /|___|  /(____  /|__|_|  /|___  / \___  >|__|     version 0.1
          \/ |__|        \/             \/        \/      \/      \/       \/     \/      \/

Using Scala version 2.10.5 (OpenJDK 64-Bit Server VM, Java 1.7.0_101)
Spark Chamber beta version 0.1
Maintained by Hadoop Compute Team, HipChat room: @Spark

scala> username
Username:
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
wgtmac
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


scala> applicationID
Application ID:
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
application_1464368688010_14482
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


scala> 
```

Spark Chamber Session

```
Welcome to
     _________                       __     _________  .__                     ___.
    /   _____/______ _____  _______ |  | __ \_   ___ \ |  |__ _____    _____ \_ |__   ____ _______
    \_____  \ \____ \\__  \ \_  __ \|  |/ / /   \  \/ |  |  \ \__  \  /     \ | __ \_/ __ \\_  __ \
    /        \|  |_> >/ __ \_|  | \/|    <  \     \___|   Y  \/ __ \_|  Y Y  \| \_\ \\  ___/ |  | \/
   /_______  /|   __/(____  /|__|  |__|_ \ \______  /|___|  /(____  /|__|_|  /|___  / \___  >|__|      version 0.1
           \/ |__|       \/            \/          \/      \/      \/       \/      \/       \/

Using Scala version 2.10.5 (OpenJDK 64-Bit Server VM, Java 1.7.0_101)
Spark Chamber beta version 0.1
Maintained by Hadoop Compute Team, HipChat room: @Spark

scala> username
Username:
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
 my_user_name
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


scala> applicationID
Application ID:
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
application_1464368688010_14482
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


scala> 
```

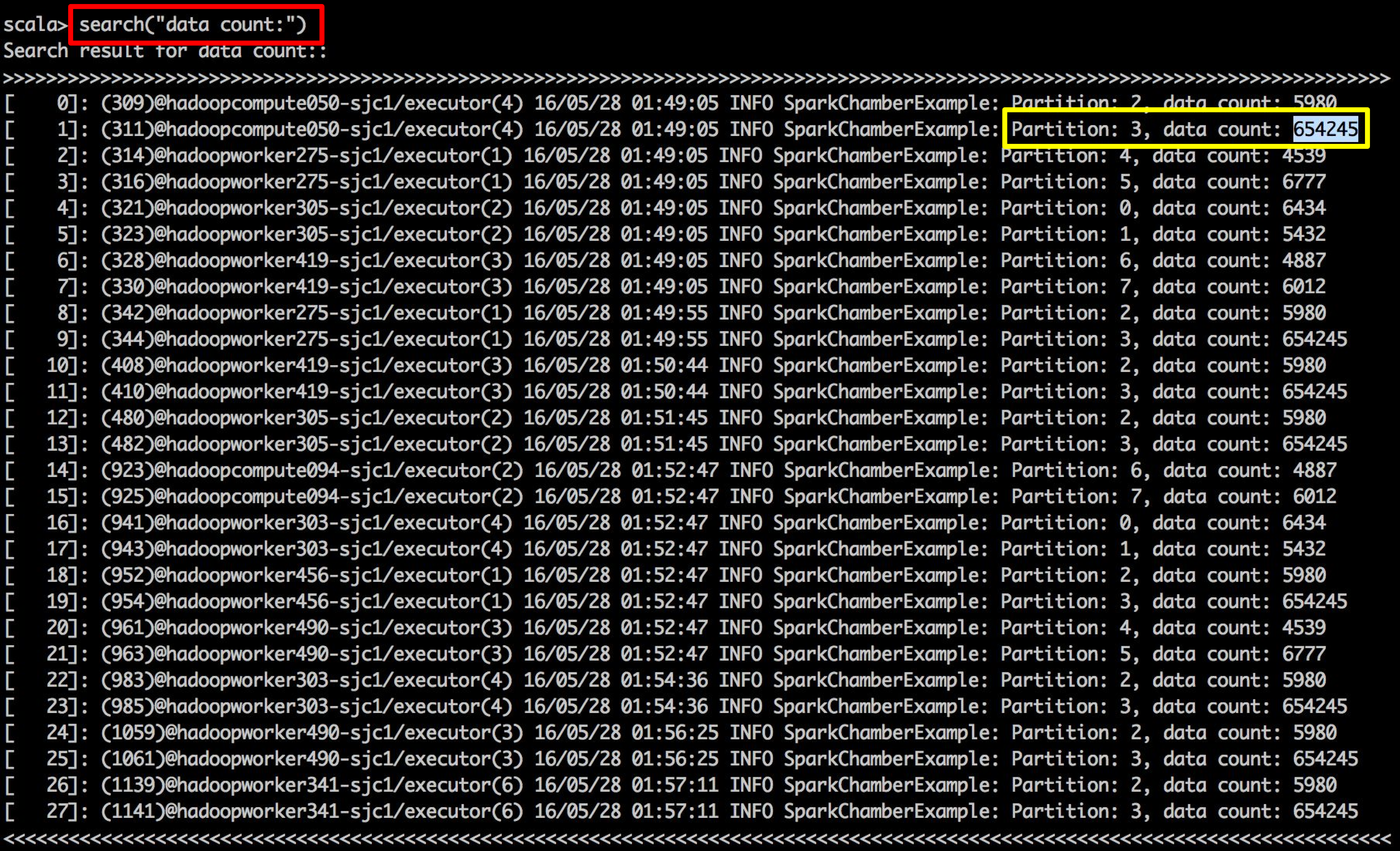
```
scala> allApplicationIds
Recent Spark Applications:
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
[    0]: application_1463943621508_147799    InspectorGadget                2016-05-24T17:42:56.237GMT
[    1]: application_1463943621508_148157    InspectorGadget                2016-05-24T17:42:56.773GMT
[    2]: application_1463943621508_147498    InspectorGadget                2016-05-24T17:28:01.696GMT
[    3]: application_1463943621508_148089    InspectorGadget                2016-05-24T17:42:57.023GMT
[    4]: application_1463943621508_147842    InspectorGadget                2016-05-24T17:42:59.347GMT
[    5]: application_1463943621508_147798    InspectorGadget                2016-05-24T17:43:01.423GMT
[    6]: application_1463943621508_147589    InspectorGadget                2016-05-24T17:28:36.430GMT
[    7]: application_1463943621508_147805    InspectorGadget                2016-05-24T17:42:57.561GMT
[    8]: application_1463943621508_147845    InspectorGadget                2016-05-24T17:42:56.266GMT
[    9]: application_1463943621508_147937    InspectorGadget                2016-05-24T17:42:59.013GMT
[   10]: application_1463943621508_148051    InspectorGadget                2016-05-24T17:42:55.401GMT
[   11]: application_1463943621508_148117    InspectorGadget                2016-05-24T17:42:55.583GMT
[   12]: application_1463943621508_148160    InspectorGadget                2016-05-24T17:42:59.121GMT
[   13]: application_1463943621508_147984    InspectorGadget                2016-05-24T17:43:00.954GMT
[   14]: application_1463943621508_147962    InspectorGadget                2016-05-24T17:43:00.844GMT
[   15]: application_1463943621508_148114    InspectorGadget                2016-05-24T17:42:56.078GMT
[   16]: application_1463943621508_148142    InspectorGadget                2016-05-24T17:43:11.641GMT
[   17]: application_1463943621508_148097    InspectorGadget                2016-05-24T17:43:00.286GMT
[   18]: application_1463943621508_147777    InspectorGadget                2016-05-24T17:28:38.652GMT
[   19]: application_1463943621508_148106    InspectorGadget                2016-05-24T17:42:55.861GMT
[   20]: application_1463943621508_148045    InspectorGadget                2016-05-24T17:42:59.113GMT
[   21]: application_1463943621508_147989    InspectorGadget                2016-05-24T17:42:56.853GMT
[   22]: application_1463943621508_148002    InspectorGadget                2016-05-24T17:43:02.013GMT
```

```
scala> setApplicationId("application_1463943621508_147842")

scala> hosts
Hosts (ExecutorId, HostName, LogWebUI):
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
[    0]: driver               hadoopworker348-sjc1      http://hadoopworker348-sjc1.pro
[    1]: executor(1)          hadoopworker347-sjc1      http://hadoopworker347-sjc1.pro
[    2]: executor(10)         hadoopworker313-sjc1      http://hadoopworker313-sjc1.pro
[    3]: executor(11)         hadoopcompute093-sjc1     http://hadoopcompute093-sjc1.pr
[    4]: executor(12)         hadoopworker490-sjc1      http://hadoopworker490-sjc1.pro
[    5]: executor(13)         hadoopworker373-sjc1      http://hadoopworker373-sjc1.pro
[    6]: executor(14)         hadoopworker255-sjc1      http://hadoopworker255-sjc1.pro
[    7]: executor(15)         hadoopworker542-sjc1      http://hadoopworker542-sjc1.pro
[    8]: executor(16)         hadoopworker549-sjc1      http://hadoopworker549-sjc1.pro
[    9]: executor(17)         hadoopworker311-sjc1      http://hadoopworker311-sjc1.pro
[   10]: executor(18)         hadoopworker437-sjc1      http://hadoopworker437-sjc1.pro
[   11]: executor(19)         hadoopworker288-sjc1      http://hadoopworker288-sjc1.pro
[   12]: executor(2)          hadoopcompute034-sjc1     http://hadoopcompute034-sjc1.pr
[   13]: executor(20)         hadoopcompute032-sjc1     http://hadoopcompute032-sjc1.pr
[   14]: executor(21)         hadoopcompute018-sjc1     http://hadoopcompute018-sjc1.pr
```

```
scala> firstException()
0th exception on ALL_HOSTS  :
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
(338)@hadoopcompute050-sjc1/executor(4) 16/05/28 01:49:54 ERROR executor.Executor: Exception in task 1.0 in stage 0.0 (TID 1)
java.lang.OutOfMemoryError: GC overhead limit exceeded
        at java.lang.Character.valueOf(Character.java:4389)
        at scala.runtime.BoxesRunTime.boxToCharacter(BoxesRunTime.java:58)
        at scala.util.Random$$anonfun$nextString$1.apply(Random.scala:88)
        at scala.collection.generic.GenTraversableFactory.fill(GenTraversableFactory.scala:91)
        at scala.util.Random.nextString(Random.scala:88)
        at com.uber.sparkplug.SparkSQLExample$$anonfun$main$1.apply$mcVI$sp(SparkSQLExample.scala:51)
        at com.uber.sparkplug.SparkSQLExample$$anonfun$main$1.apply(SparkSQLExample.scala:42)
        at com.uber.sparkplug.SparkSQLExample$$anonfun$main$1.apply(SparkSQLExample.scala:42)
        at scala.collection.Iterator$class.foreach(Iterator.scala:727)
        at org.apache.spark.InterruptibleIterator.foreach(InterruptibleIterator.scala:28)
        at org.apache.spark.rdd.RDD$$anonfun$foreach$1$$anonfun$apply$28.apply(RDD.scala:890)
        at org.apache.spark.rdd.RDD$$anonfun$foreach$1$$anonfun$apply$28.apply(RDD.scala:890)
        at org.apache.spark.SparkContext$$anonfun$runJob$5.apply(SparkContext.scala:1848)
        at org.apache.spark.SparkContext$$anonfun$runJob$5.apply(SparkContext.scala:1848)
        at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:66)
        at org.apache.spark.scheduler.Task.run(Task.scala:88)
        at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:214)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
        at java.lang.Thread.run(Thread.java:745)
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


scala> 
```



```
scala> search("data count:")
Search result for data count::
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
[    0]: (309)@hadoopcompute050-sjc1/executor(4) 16/05/28 01:49:05 INFO SparkChamberExample: Partition: 2, data count: 5980
[    1]: (311)@hadoopcompute050-sjc1/executor(4) 16/05/28 01:49:05 INFO SparkChamberExample: Partition: 3, data count: 654245
[    2]: (314)@hadoopworker275-sjc1/executor(1) 16/05/28 01:49:05 INFO SparkChamberExample: Partition: 4, data count: 4539
[    3]: (316)@hadoopworker275-sjc1/executor(1) 16/05/28 01:49:05 INFO SparkChamberExample: Partition: 5, data count: 6777
[    4]: (321)@hadoopworker305-sjc1/executor(2) 16/05/28 01:49:05 INFO SparkChamberExample: Partition: 0, data count: 6434
[    5]: (323)@hadoopworker305-sjc1/executor(2) 16/05/28 01:49:05 INFO SparkChamberExample: Partition: 1, data count: 5432
[    6]: (328)@hadoopworker419-sjc1/executor(3) 16/05/28 01:49:05 INFO SparkChamberExample: Partition: 6, data count: 4887
[    7]: (330)@hadoopworker419-sjc1/executor(3) 16/05/28 01:49:05 INFO SparkChamberExample: Partition: 7, data count: 6012
[    8]: (342)@hadoopworker275-sjc1/executor(1) 16/05/28 01:49:55 INFO SparkChamberExample: Partition: 2, data count: 5980
[    9]: (344)@hadoopworker275-sjc1/executor(1) 16/05/28 01:49:55 INFO SparkChamberExample: Partition: 3, data count: 654245
[   10]: (408)@hadoopworker419-sjc1/executor(3) 16/05/28 01:50:44 INFO SparkChamberExample: Partition: 2, data count: 5980
[   11]: (410)@hadoopworker419-sjc1/executor(3) 16/05/28 01:50:44 INFO SparkChamberExample: Partition: 3, data count: 654245
[   12]: (480)@hadoopworker305-sjc1/executor(2) 16/05/28 01:51:45 INFO SparkChamberExample: Partition: 2, data count: 5980
[   13]: (482)@hadoopworker305-sjc1/executor(2) 16/05/28 01:51:45 INFO SparkChamberExample: Partition: 3, data count: 654245
[   14]: (923)@hadoopcompute094-sjc1/executor(2) 16/05/28 01:52:47 INFO SparkChamberExample: Partition: 6, data count: 4887
[   15]: (925)@hadoopcompute094-sjc1/executor(2) 16/05/28 01:52:47 INFO SparkChamberExample: Partition: 7, data count: 6012
[   16]: (941)@hadoopworker303-sjc1/executor(4) 16/05/28 01:52:47 INFO SparkChamberExample: Partition: 0, data count: 6434
[   17]: (943)@hadoopworker303-sjc1/executor(4) 16/05/28 01:52:47 INFO SparkChamberExample: Partition: 1, data count: 5432
[   18]: (952)@hadoopworker456-sjc1/executor(1) 16/05/28 01:52:47 INFO SparkChamberExample: Partition: 2, data count: 5980
[   19]: (954)@hadoopworker456-sjc1/executor(1) 16/05/28 01:52:47 INFO SparkChamberExample: Partition: 3, data count: 654245
[   20]: (961)@hadoopworker490-sjc1/executor(3) 16/05/28 01:52:47 INFO SparkChamberExample: Partition: 4, data count: 4539
[   21]: (963)@hadoopworker490-sjc1/executor(3) 16/05/28 01:52:47 INFO SparkChamberExample: Partition: 5, data count: 6777
[   22]: (983)@hadoopworker303-sjc1/executor(4) 16/05/28 01:54:36 INFO SparkChamberExample: Partition: 2, data count: 5980
[   23]: (985)@hadoopworker303-sjc1/executor(4) 16/05/28 01:54:36 INFO SparkChamberExample: Partition: 3, data count: 654245
[   24]: (1059)@hadoopworker490-sjc1/executor(3) 16/05/28 01:56:25 INFO SparkChamberExample: Partition: 2, data count: 5980
[   25]: (1061)@hadoopworker490-sjc1/executor(3) 16/05/28 01:56:25 INFO SparkChamberExample: Partition: 3, data count: 654245
[   26]: (1139)@hadoopworker341-sjc1/executor(6) 16/05/28 01:57:11 INFO SparkChamberExample: Partition: 2, data count: 5980
[   27]: (1141)@hadoopworker341-sjc1/executor(6) 16/05/28 01:57:11 INFO SparkChamberExample: Partition: 3, data count: 654245
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

Spark Chamber

Distributed Log Debugger for Spark

Features

1. Get all recent Spark Application IDs.

2. Get first exception, all exceptions grouped by types sorted by time, etc.

3. Display CPU, memory, I/O metrics.

4. Dive into a specific driver/executor/machine

5. Search

Spark Chamber

Distributed Log Debugger for Spark

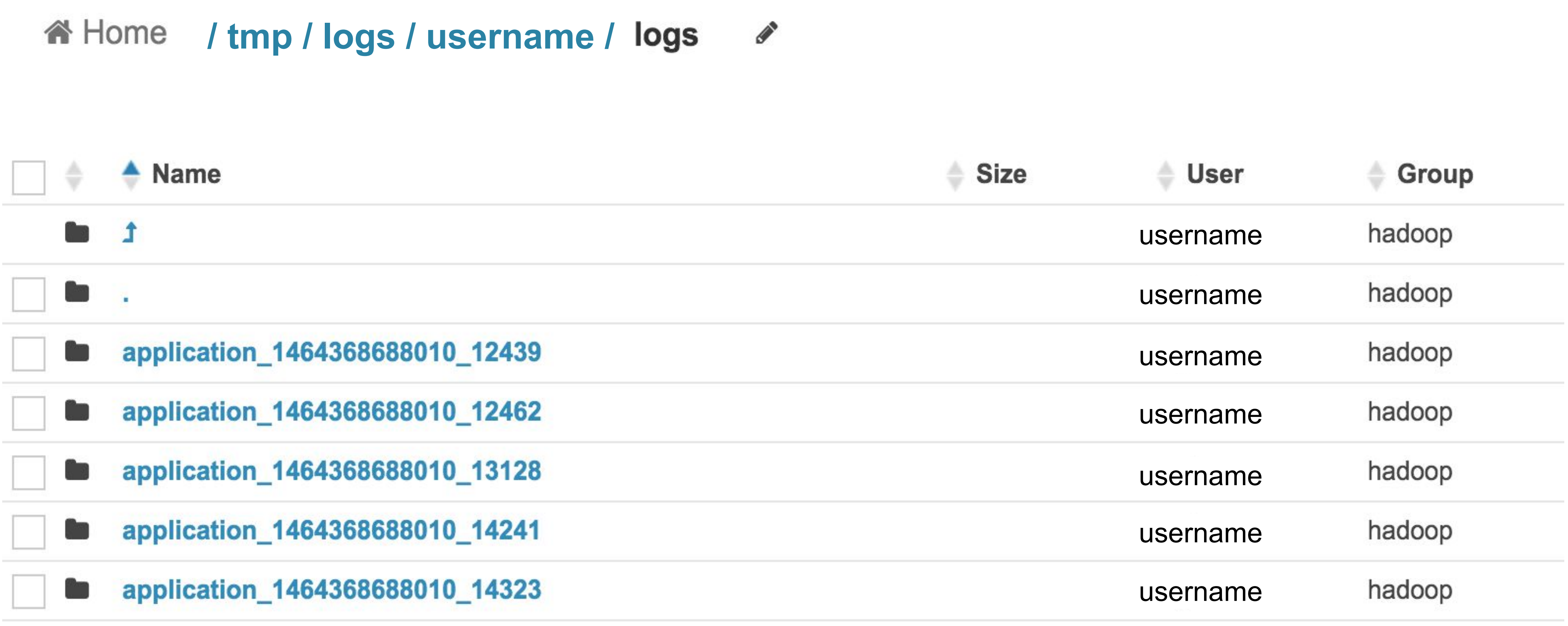
Security

Developer mode: debug developer’s own Spark job.

SRE mode: view and check all users’ Spark job information.

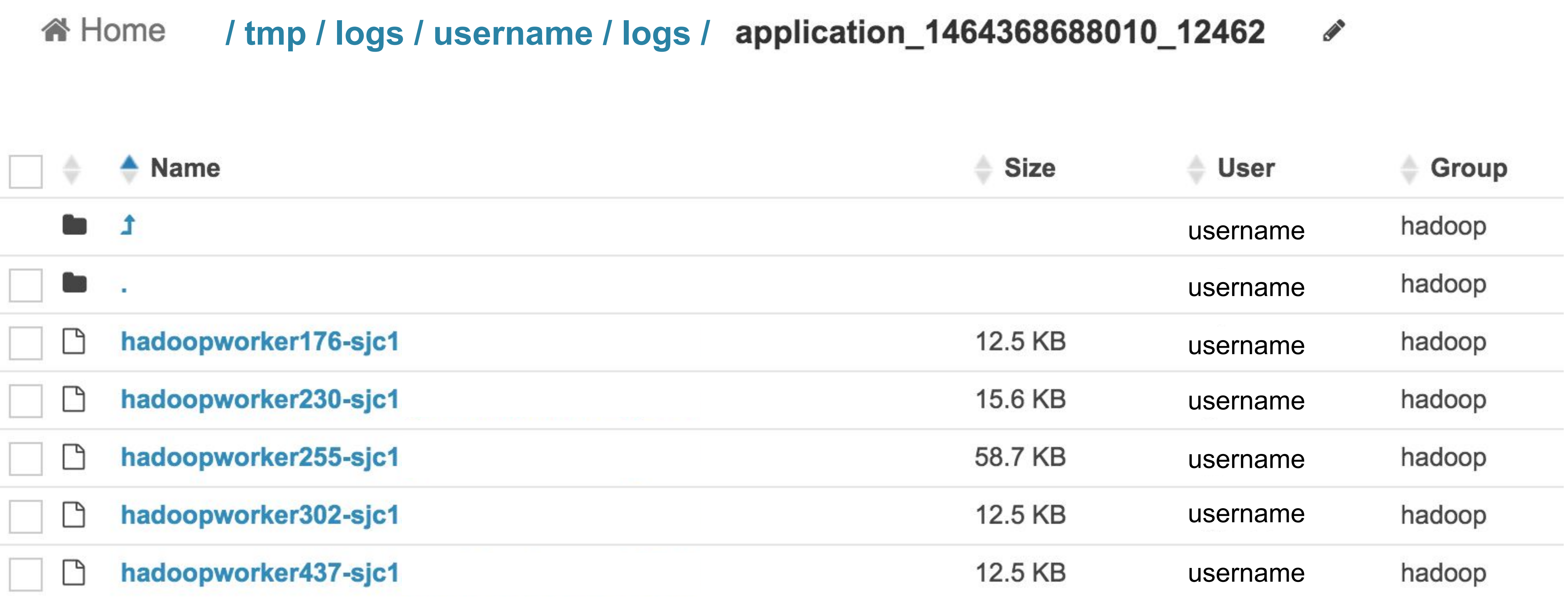
Spark Chamber

Enable Yarn Log Aggregation



Home / tmp / logs / username / logs

□	Name	Size	User	Group
	↥		username	hadoop
□	.		username	hadoop
□	application_1464368688010_12439		username	hadoop
□	application_1464368688010_12462		username	hadoop
□	application_1464368688010_13128		username	hadoop
□	application_1464368688010_14241		username	hadoop
□	application_1464368688010_14323		username	hadoop



Home / tmp / logs / username / logs / application_1464368688010_12462

□	Name	Size	User	Group
	↥		username	hadoop
□	.		username	hadoop
□	hadoopworker176-sjc1	12.5 KB	username	hadoop
□	hadoopworker230-sjc1	15.6 KB	username	hadoop
□	hadoopworker255-sjc1	58.7 KB	username	hadoop
□	hadoopworker302-sjc1	12.5 KB	username	hadoop
□	hadoopworker437-sjc1	12.5 KB	username	hadoop

YARN aggregates log files on HDFS

All application IDs of the same user are under same place.

Files are named after host names

One machine has one log file, regardless of # executors on that machine.

Spark Chamber

Use Spark to debug Spark

Extend the Spark Shell by Hooks:

1. For ONE application Id, distribute log files to different executors.

2. Extract each lines and save into DataFrame.

3. Sort log dataframe by time and hostname.

4. Retrieve target log via SparkSQL DataFrame APIs.

	Engineers	SRE
API	• Geo-spatial processing • SCBuilder • Kafka dispersal	
Tools	• SparkChamber • SparkPlug	• SparkChamber

Future Work

Spark Chamber

SRE version - Cluster wide insights

- Dimensions - Jobs
  - All
  - Single team
  - Single engineer
- Dimensions - Time
  - Last month, week, day
- Dimensions - Hardware
  - Specific rack, pod

Spark Chamber

SRE version - Analytics and Machine Learning

- Analytics
  - Resource auditing
  - Data access auditing
- Machine Learning
  - Failures diagnostics
  - Malicious jobs detection
  - Performance optimization

Future Work

	Engineers	SRE
API	• Geo-spatial processing • SCBuilder • Kafka dispersal • Hive table registration (Didn’t cover today) • Incremental processing (Didn’t cover today) • Debug logging • Metrics • Configurations • Data Freshness	• Resource usage
Tools	• SparkChamber • SparkPlug • Unit testing (Didn’t cover today) • Oozie integration (Didn’t cover today)	• SparkChamber • Resource usage auditing • Data access auditing • Machine learning on jobs

SPARK: INTERACTIVE TO PRODUCTION

Today, Tuesday, June 7
4:50 PM – 5:20 PM
Room: Ballroom B

Dara Adib, Uber

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

Thank you

UBER