Telefónica
Investigación y Desarrollo Chile

Procesamiento masivo de datos con Spark

Oscar Peredo A.
Telefónica I+D Chile
oscar.peredoa@telefonica.com

2 Junio 2016

ELEGIMOS TODO_

Contenidos

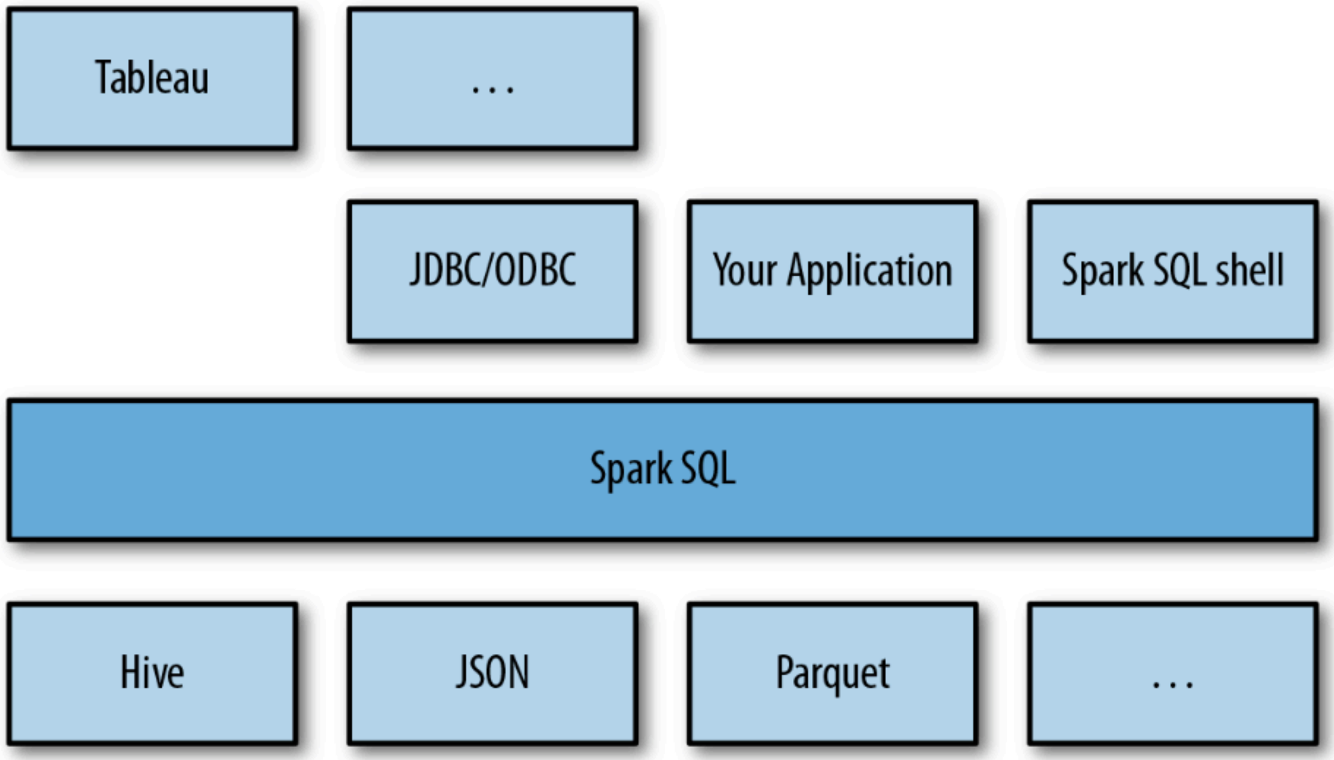
- Spark SQL

- Tez versus Spark

Spark SQL

Herramienta útil cuando:

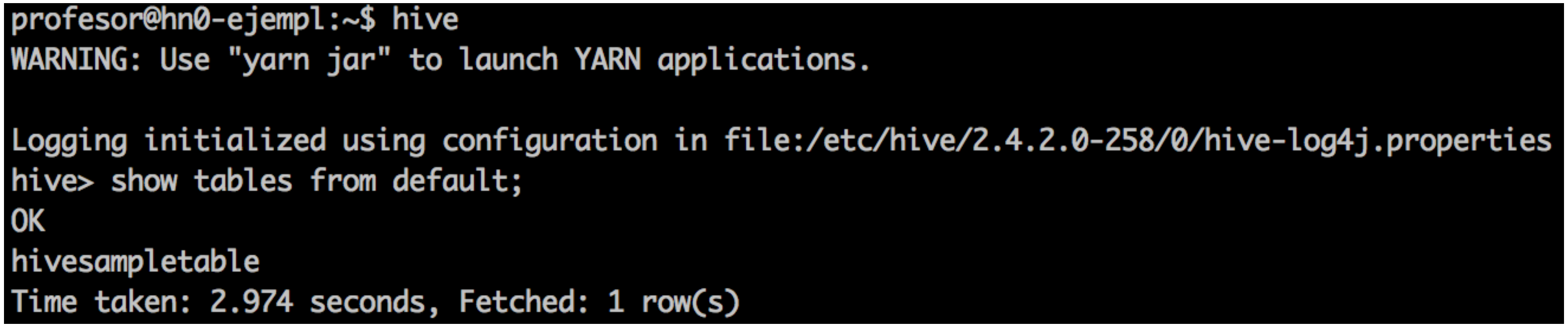
- Los datos (semi) estructurados
- La base de conocimiento en SQL es amplia
- Se necesita integración con equipo de data scientists (Python)
- Se quiere usar toda la potencia de procesamiento de Spark como backend de cómputo



Karau et al. (2015) Learning Spark: Lightning-Fast Big Data Analysis

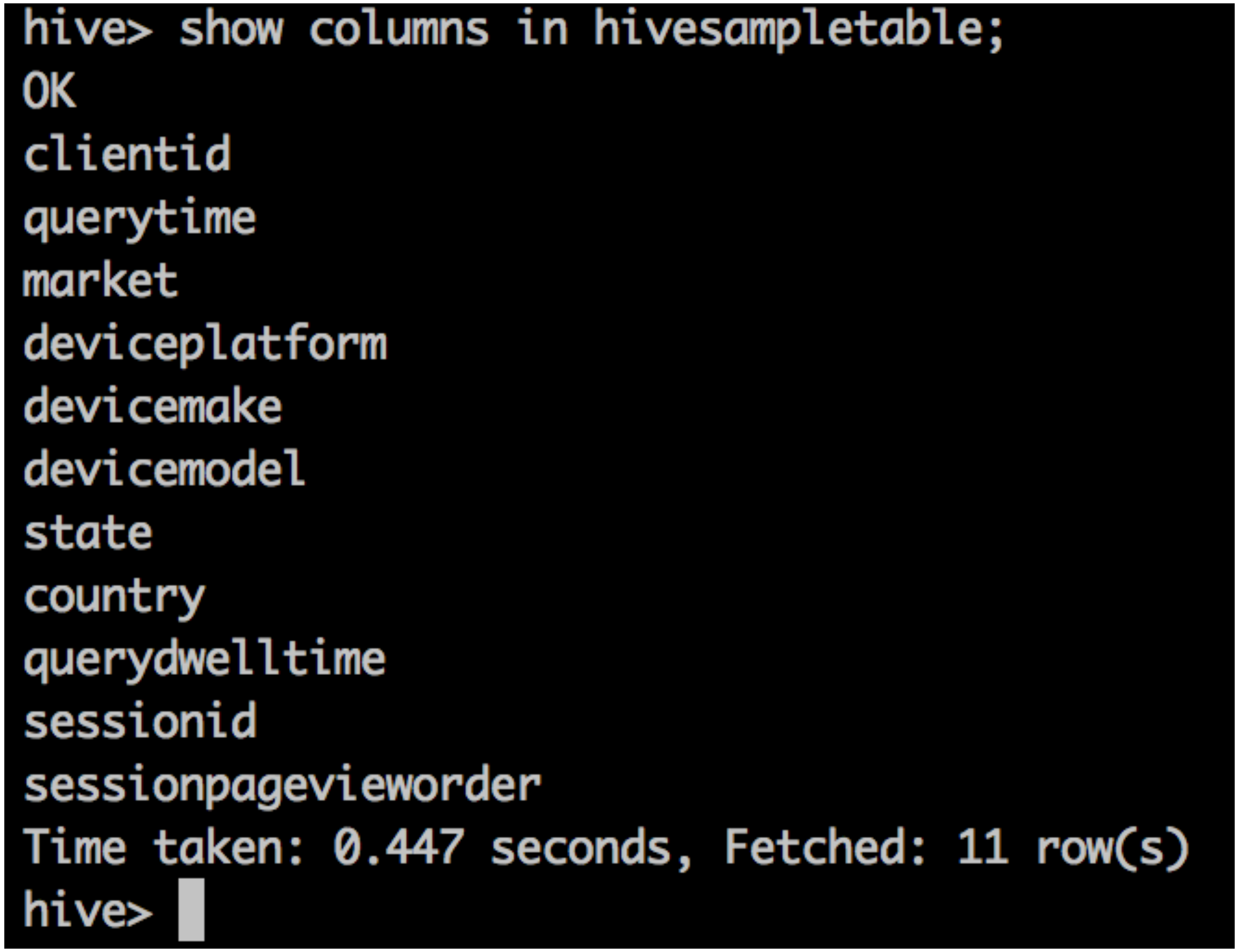
Ejemplo: SparkSQL + Hive

Por el lado de Hive (datos)…



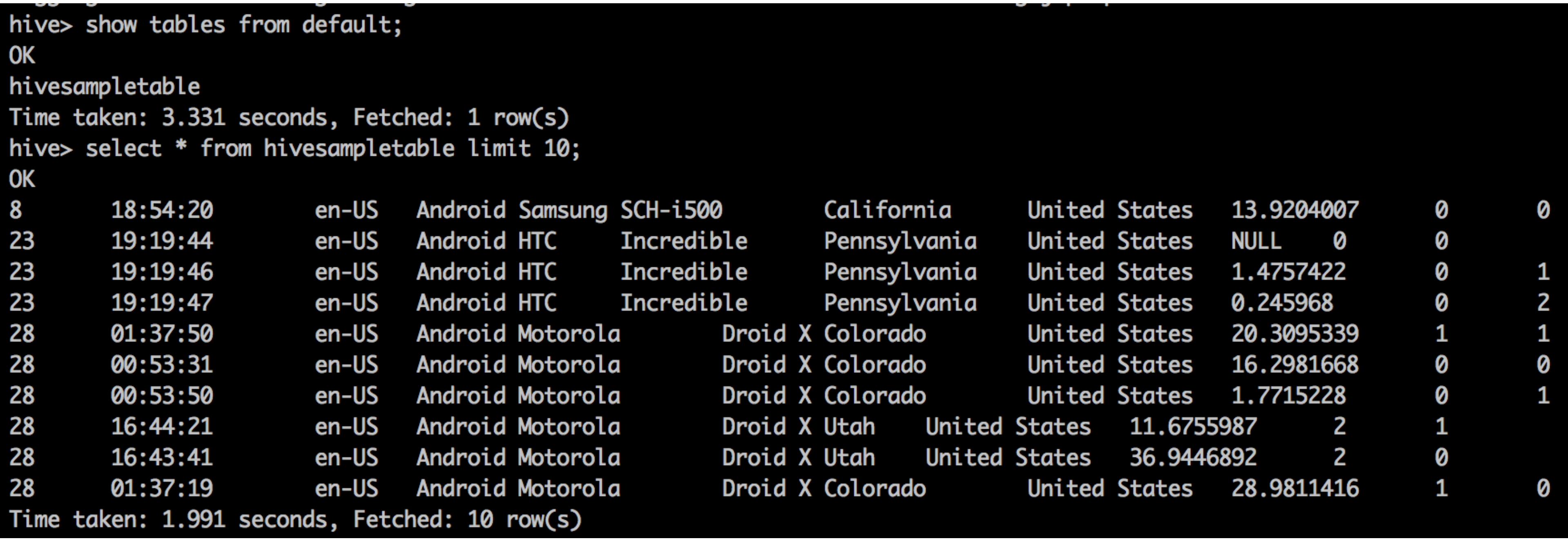
```
profesor@hn0-ejempl:~$ hive
WARNING: Use "yarn jar" to launch YARN applications.

Logging initialized using configuration in file:/etc/hive/2.4.2.0-258/0/hive-log4j.properties
hive> show tables from default;
OK
hivesampletable
Time taken: 2.974 seconds, Fetched: 1 row(s)
```

Ejemplo: SparkSQL + Hive



```
hive> show columns in hivesampletable;
OK
clientid
querytime
market
deviceplatform
devicemake
devicemodel
state
country
querydwelltime
sessionid
sessionpagevieworder
Time taken: 0.447 seconds, Fetched: 11 row(s)
hive> 
```

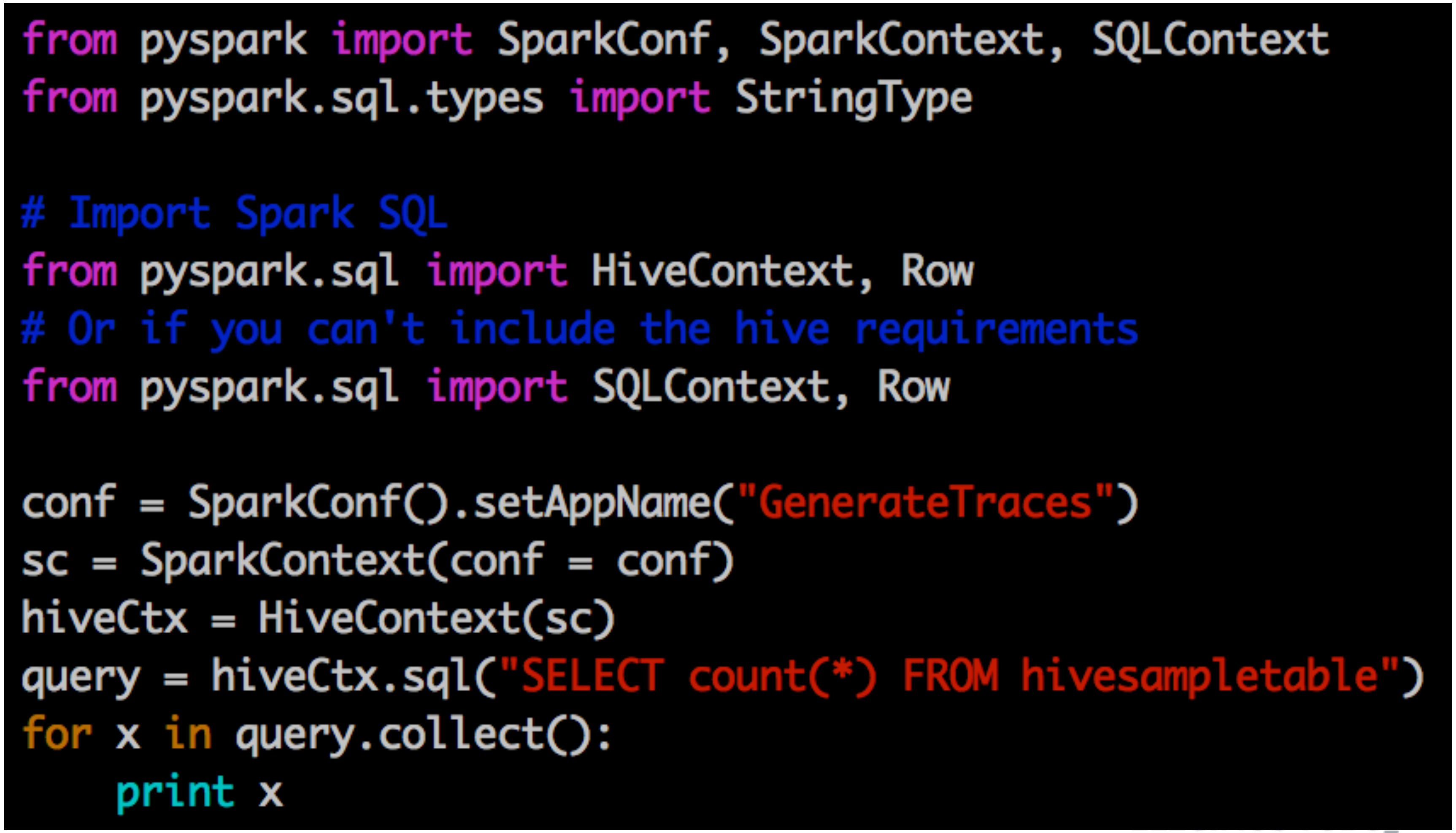
Ejemplo: SparkSQL + Hive



```
hive> show tables from default;
OK
hivesampletable
Time taken: 3.331 seconds, Fetched: 1 row(s)
hive> select * from hivesampletable limit 10;
OK
8       18:54:20        en-US   Android Samsung SCH-i500        California      United States   13.9204007      0       0
23      19:19:44        en-US   Android HTC     Incredible      Pennsylvania    United States   NULL    0       0
23      19:19:46        en-US   Android HTC     Incredible      Pennsylvania    United States   1.4757422       0       1
23      19:19:47        en-US   Android HTC     Incredible      Pennsylvania    United States   0.245968        0       2
28      01:37:50        en-US   Android Motorola        Droid X Colorado        United States   20.3095339      1       1
28      00:53:31        en-US   Android Motorola        Droid X Colorado        United States   16.2981668      0       0
28      00:53:50        en-US   Android Motorola        Droid X Colorado        United States   1.7715228       0       1
28      16:44:21        en-US   Android Motorola        Droid X Utah    United States   11.6755987      2       1
28      16:43:41        en-US   Android Motorola        Droid X Utah    United States   36.9446892      2       0
28      01:37:19        en-US   Android Motorola        Droid X Colorado        United States   28.9811416      1       0
Time taken: 1.991 seconds, Fetched: 10 row(s)
```

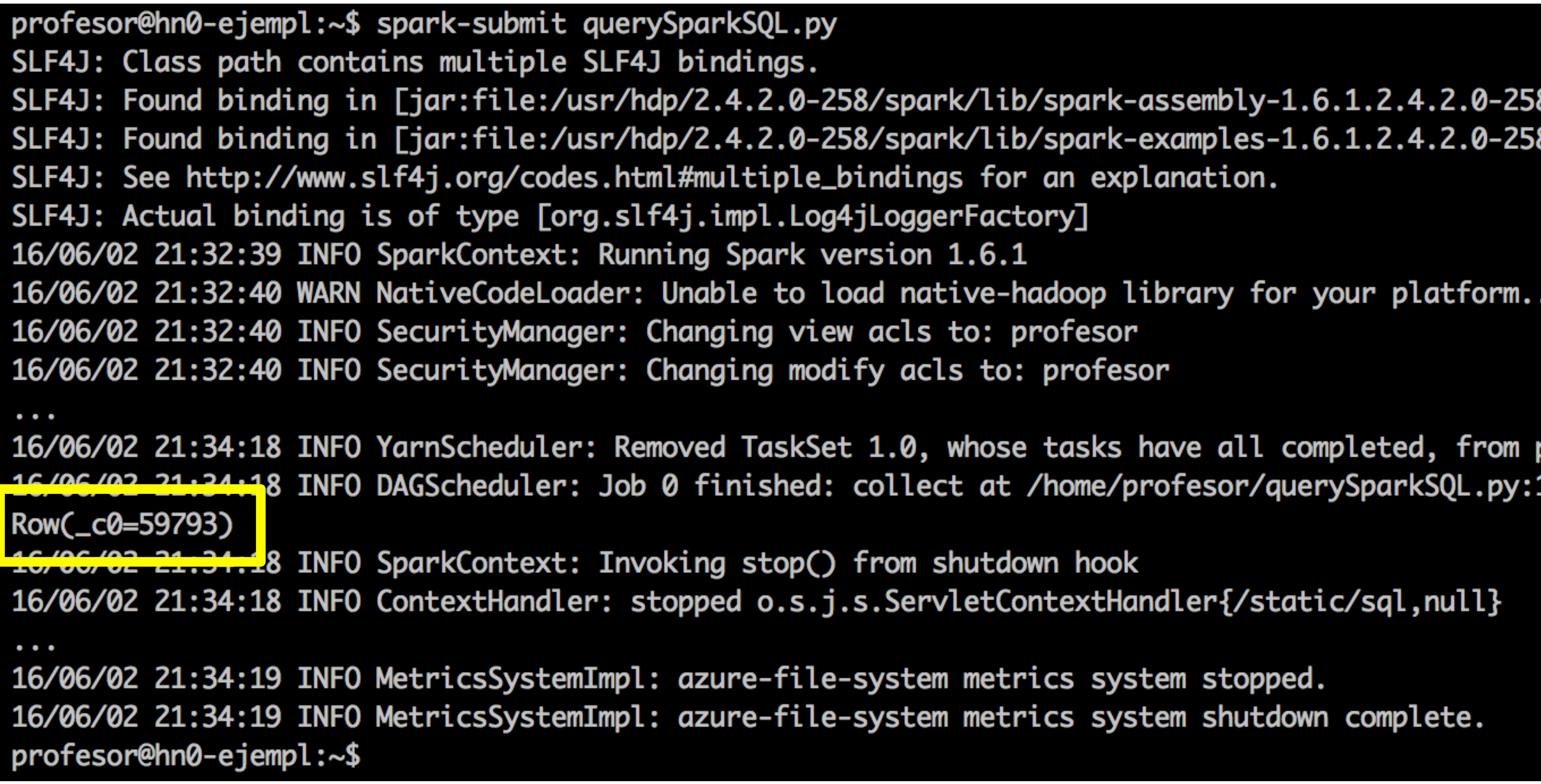
Ejemplo: SparkSQL + Hive

Por el lado de SparkSQL (query)...



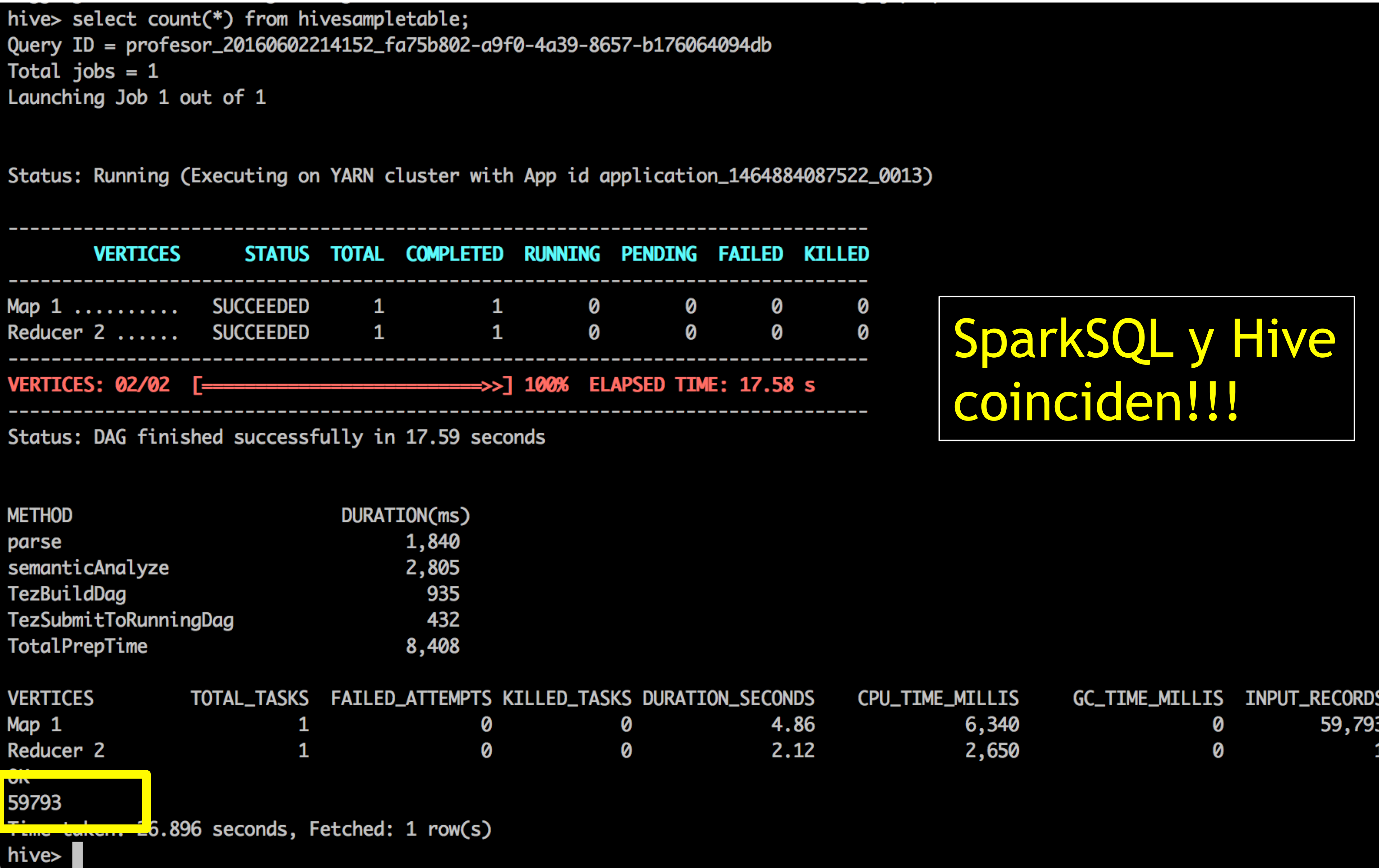
```
from pyspark import SparkConf, SparkContext, SQLContext
from pyspark.sql.types import StringType

# Import Spark SQL
from pyspark.sql import HiveContext, Row
# Or if you can't include the hive requirements
from pyspark.sql import SQLContext, Row

conf = SparkConf().setAppName("GenerateTraces")
sc = SparkContext(conf = conf)
hiveCtx = HiveContext(sc)
query = hiveCtx.sql("SELECT count(*) FROM hivesampletable")
for x in query.collect():
    print x
```

Ejemplo: SparkSQL + Hive



```
profesor@hn0-ejempl:~$ spark-submit querySparkSQL.py
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/hdp/2.4.2.0-258/spark/lib/spark-assembly-1.6.1.2.4.2.0-258
SLF4J: Found binding in [jar:file:/usr/hdp/2.4.2.0-258/spark/lib/spark-examples-1.6.1.2.4.2.0-258
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
16/06/02 21:32:39 INFO SparkContext: Running Spark version 1.6.1
16/06/02 21:32:40 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform.
16/06/02 21:32:40 INFO SecurityManager: Changing view acls to: profesor
16/06/02 21:32:40 INFO SecurityManager: Changing modify acls to: profesor
...
16/06/02 21:34:18 INFO YarnScheduler: Removed TaskSet 1.0, whose tasks have all completed, from p
16/06/02 21:34:18 INFO DAGScheduler: Job 0 finished: collect at /home/profesor/querySparkSQL.py:1
Row(_c0=59793)
16/06/02 21:34:18 INFO SparkContext: Invoking stop() from shutdown hook
16/06/02 21:34:18 INFO ContextHandler: stopped o.s.j.s.ServletContextHandler{/static/sql,null}
...
16/06/02 21:34:19 INFO MetricsSystemImpl: azure-file-system metrics system stopped.
16/06/02 21:34:19 INFO MetricsSystemImpl: azure-file-system metrics system shutdown complete.
profesor@hn0-ejempl:~$
```

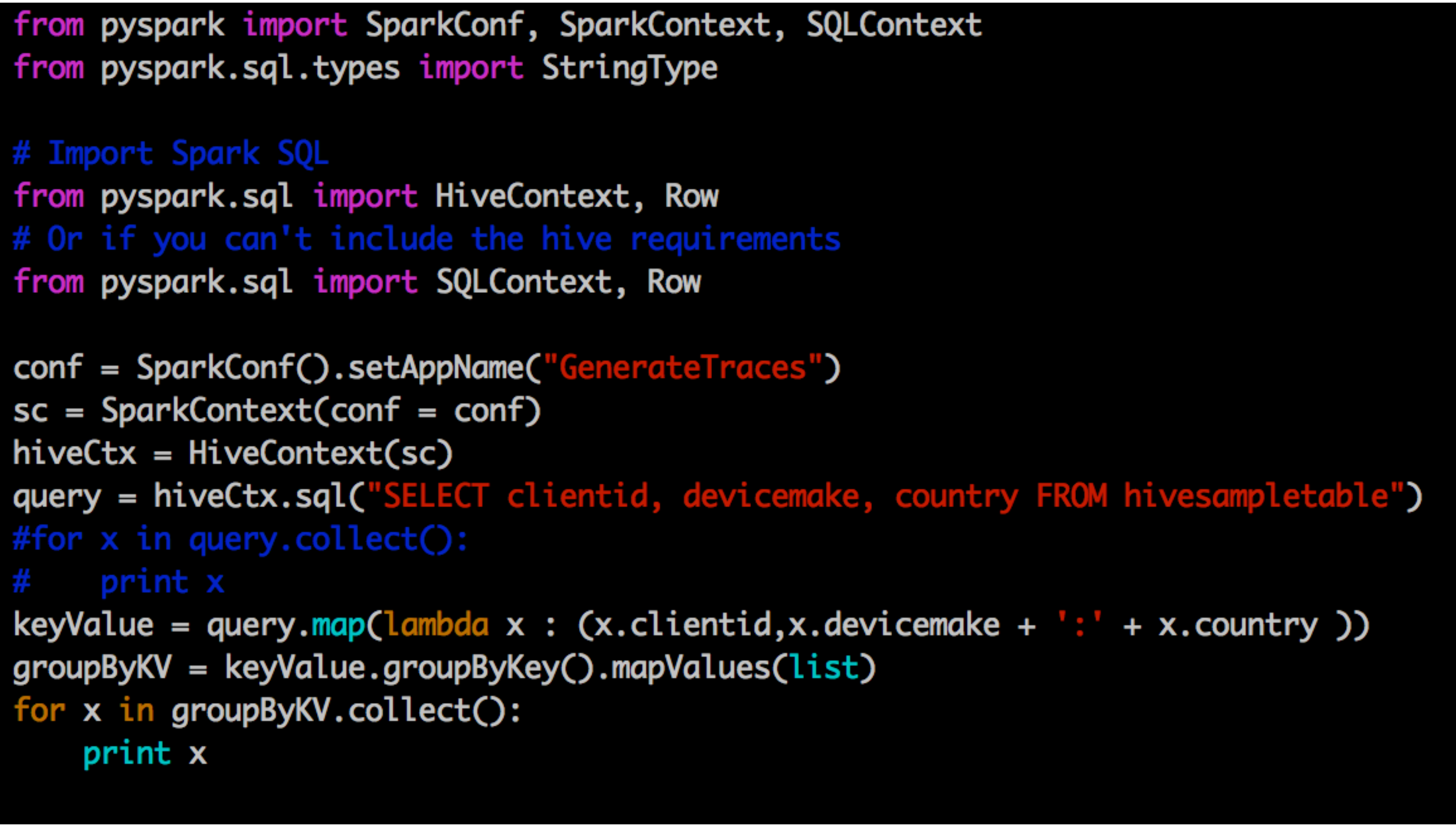
Ejemplo: SparkSQL + Hive



```
hive> select count(*) from hivesampletable;
Query ID = profesor_20160602214152_fa75b802-a9f0-4a39-8657-b176064094db
Total jobs = 1
Launching Job 1 out of 1


Status: Running (Executing on YARN cluster with App id application_1464884087522_0013)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 17.58 s
--------------------------------------------------------------------------------
Status: DAG finished successfully in 17.59 seconds


METHOD                         DURATION(ms)
parse                                 1,840
semanticAnalyze                       2,805
TezBuildDag                             935
TezSubmitToRunningDag                   432
TotalPrepTime                         8,408

VERTICES         TOTAL_TASKS  FAILED_ATTEMPTS KILLED_TASKS DURATION_SECONDS    CPU_TIME_MILLIS     GC_TIME_MILLIS  INPUT_RECORDS
Map 1                      1                0            0             4.86              6,340                  0         59,793
Reducer 2                  1                0            0             2.12              2,650                  0              1
OK
59793
Time taken: 26.896 seconds, Fetched: 1 row(s)
hive> 
```

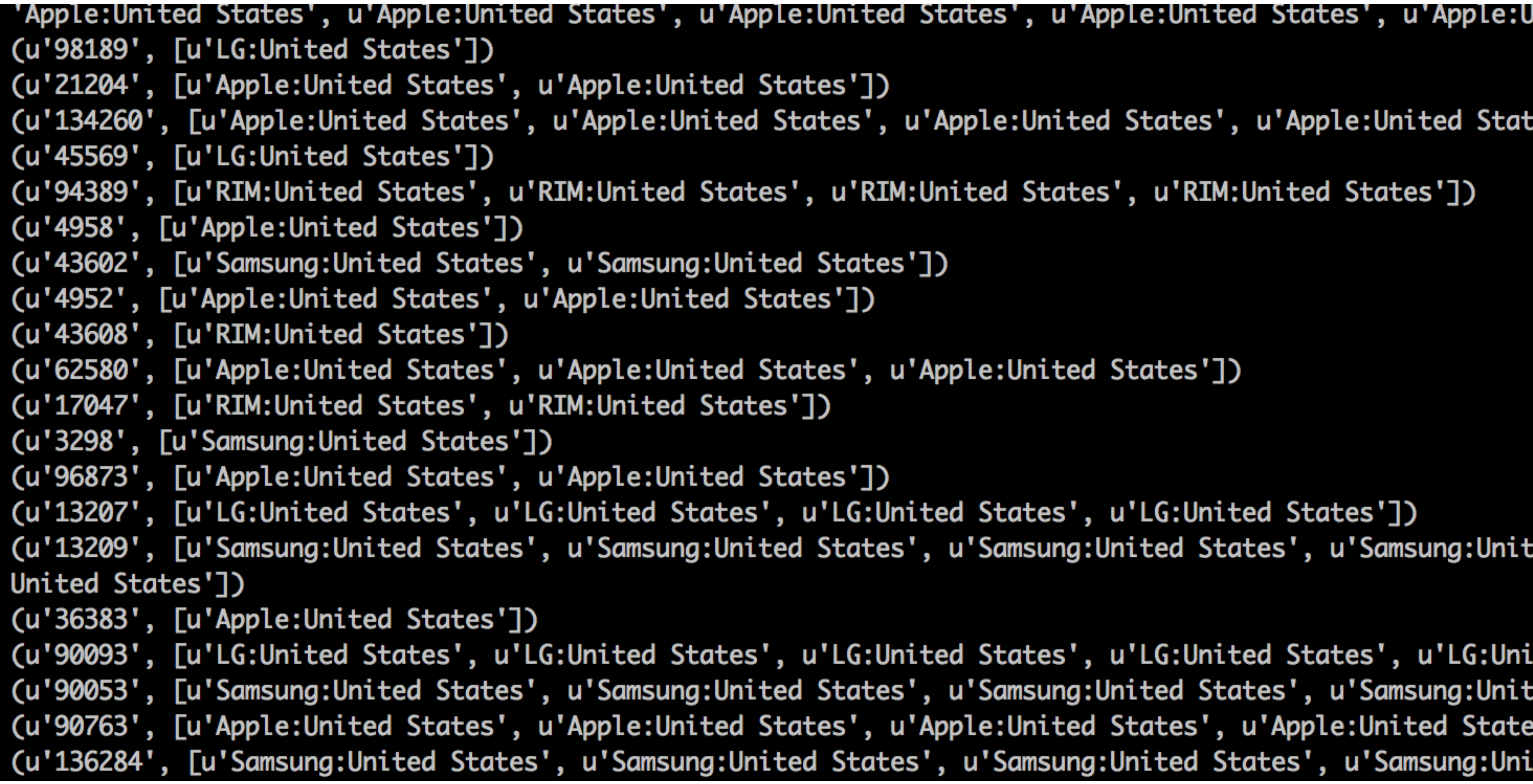
SparkSQL y Hive coinciden!!!

SQL + RDDs



```python
from pyspark import SparkConf, SparkContext, SQLContext
from pyspark.sql.types import StringType

# Import Spark SQL
from pyspark.sql import HiveContext, Row
# Or if you can't include the hive requirements
from pyspark.sql import SQLContext, Row

conf = SparkConf().setAppName("GenerateTraces")
sc = SparkContext(conf = conf)
hiveCtx = HiveContext(sc)
query = hiveCtx.sql("SELECT clientid, devicemake, country FROM hivesampletable")
#for x in query.collect():
#    print x
keyValue = query.map(lambda x : (x.clientid,x.devicemake + ':' + x.country ))
groupByKV = keyValue.groupByKey().mapValues(list)
for x in groupByKV.collect():
    print x
```

SQL + RDDs

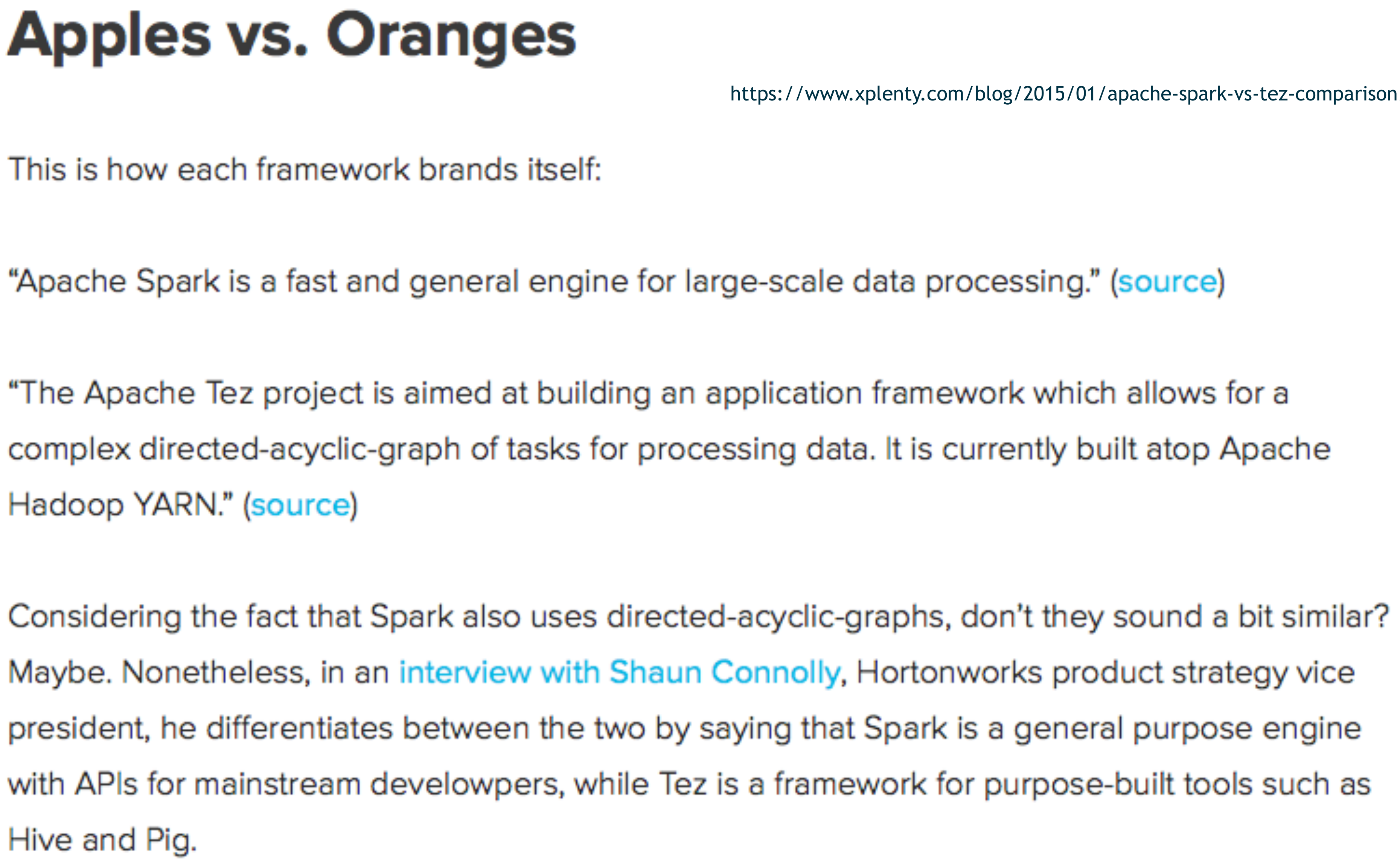


```
'Apple:United States', u'Apple:United States', u'Apple:United States', u'Apple:United States', u'Apple:U
(u'98189', [u'LG:United States'])
(u'21204', [u'Apple:United States', u'Apple:United States'])
(u'134260', [u'Apple:United States', u'Apple:United States', u'Apple:United States', u'Apple:United Stat
(u'45569', [u'LG:United States'])
(u'94389', [u'RIM:United States', u'RIM:United States', u'RIM:United States', u'RIM:United States'])
(u'4958', [u'Apple:United States'])
(u'43602', [u'Samsung:United States', u'Samsung:United States'])
(u'4952', [u'Apple:United States', u'Apple:United States'])
(u'43608', [u'RIM:United States'])
(u'62580', [u'Apple:United States', u'Apple:United States', u'Apple:United States'])
(u'17047', [u'RIM:United States', u'RIM:United States'])
(u'3298', [u'Samsung:United States'])
(u'96873', [u'Apple:United States', u'Apple:United States'])
(u'13207', [u'LG:United States', u'LG:United States', u'LG:United States', u'LG:United States'])
(u'13209', [u'Samsung:United States', u'Samsung:United States', u'Samsung:United States', u'Samsung:Unit
United States'])
(u'36383', [u'Apple:United States'])
(u'90093', [u'LG:United States', u'LG:United States', u'LG:United States', u'LG:United States', u'LG:Uni
(u'90053', [u'Samsung:United States', u'Samsung:United States', u'Samsung:United States', u'Samsung:Unit
(u'90763', [u'Apple:United States', u'Apple:United States', u'Apple:United States', u'Apple:United State
(u'136284', [u'Samsung:United States', u'Samsung:United States', u'Samsung:United States', u'Samsung:Uni
```

ELEGIMOS TODO_
]

Tez versus Spark

Tez vs Spark



https://www.xplenty.com/blog/2015/01/apache-spark-vs-tez-comparison/

Apples vs. Oranges

This is how each framework brands itself:

"Apache Spark is a fast and general engine for large-scale data processing." (source)

"The Apache Tez project is aimed at building an application framework which allows for a complex directed-acyclic-graph of tasks for processing data. It is currently built atop Apache Hadoop YARN." (source)

Considering the fact that Spark also uses directed-acyclic-graphs, don't they sound a bit similar? Maybe. Nonetheless, in an interview with Shaun Connolly, Hortonworks product strategy vice president, he differentiates between the two by saying that Spark is a general purpose engine with APIs for mainstream develowpers, while Tez is a framework for purpose-built tools such as Hive and Pig.

Nuestro benchmark

- Eventos:

```
ID_123,TIME_001,ANTENNA_A
ID_235,TIME_002,ANTENNA_A
ID_123,TIME_003,ANTENNA_B
ID_654,TIME_005,ANTENNA_D
…
ID_ABC,TIME_XYZ,ANTENNA_F
```

- Trazas de movilidad:

```
{ID_123,((TIME_001,ANTENNA_A),(TIME_003,ANTENNA_B))}
{ID_235,((TIME_002,ANTENNA_A))}
{ID_654,((TIME_005,ANTENNA_D))}
…
```

Nuestro benchmark

- Tres implementaciones
  - Pig + Tez
  - Python Spark (interpretado)
  - Scala Spark (compilado)

- ¿Qué implementación permite calcular trazas de la manera más rápida?
- Intuición:
  - Código compilado debería ser más rápido

Nuestro benchmark

Datos: 10GBytes

	Pig+Tez	PySpark	Scala
Tiempo [mm:ss]	27:22	31:56	28:33
Líneas de código	24	55	35 + 7 (makefile)
Tuning [Horas]	0	48	48

Nuestro benchmark

Datos: 10GBytes

	Pig+Tez	PySpark	Scala
Tiempo [mm:ss]	27:22	31:56	28:33
Líneas de código	24	55	35 + 7 (makefile)
Tuning [Horas]	0	48	48

¿Tez o Spark?
Depende...

ELEGIMOS TODO_

Gracias!!