

Introduction to Zeppelin

Lee moon soo, NFLabs
moon@apache.org

What is Zeppelin?

Let's see demo

How Zeppelin started

Data = Understanding



War is ninety percent information
- Napoleon Bonaparte

A price of light is less than the cost of darkness
- Arthur C. Nielsen , Founder of ACNielsen

What gets measured, gets managed
- Peter Drucker

In God we trust, all others must bring data
- W. Edwards Deming

For data analysis, we need tool

But couldn't find one i like

MLLib

Cloudera-ML



MRQL

Drill

Impala



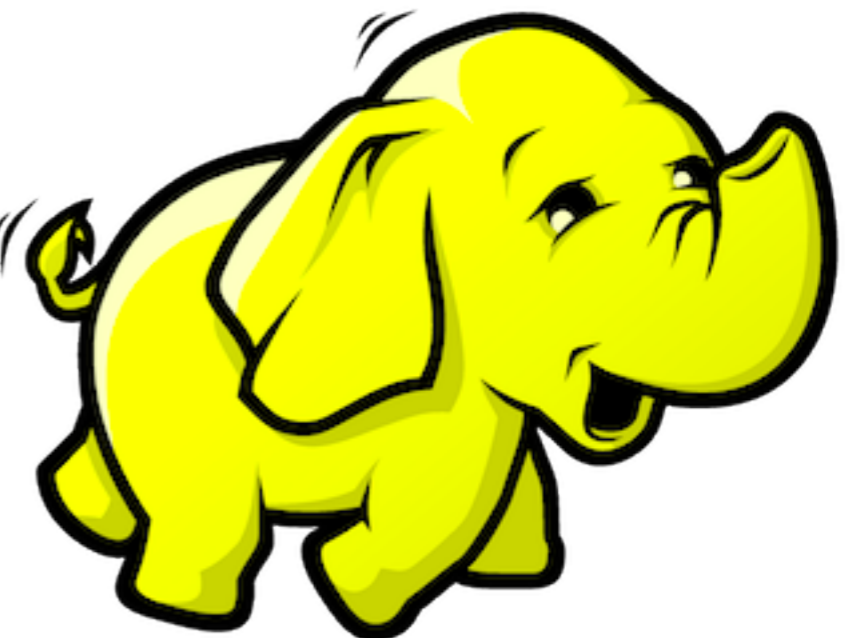
Pig

Hive

tajo



Sqoop



Decided to make one

Really good one

Good analytics environment

Analytical language

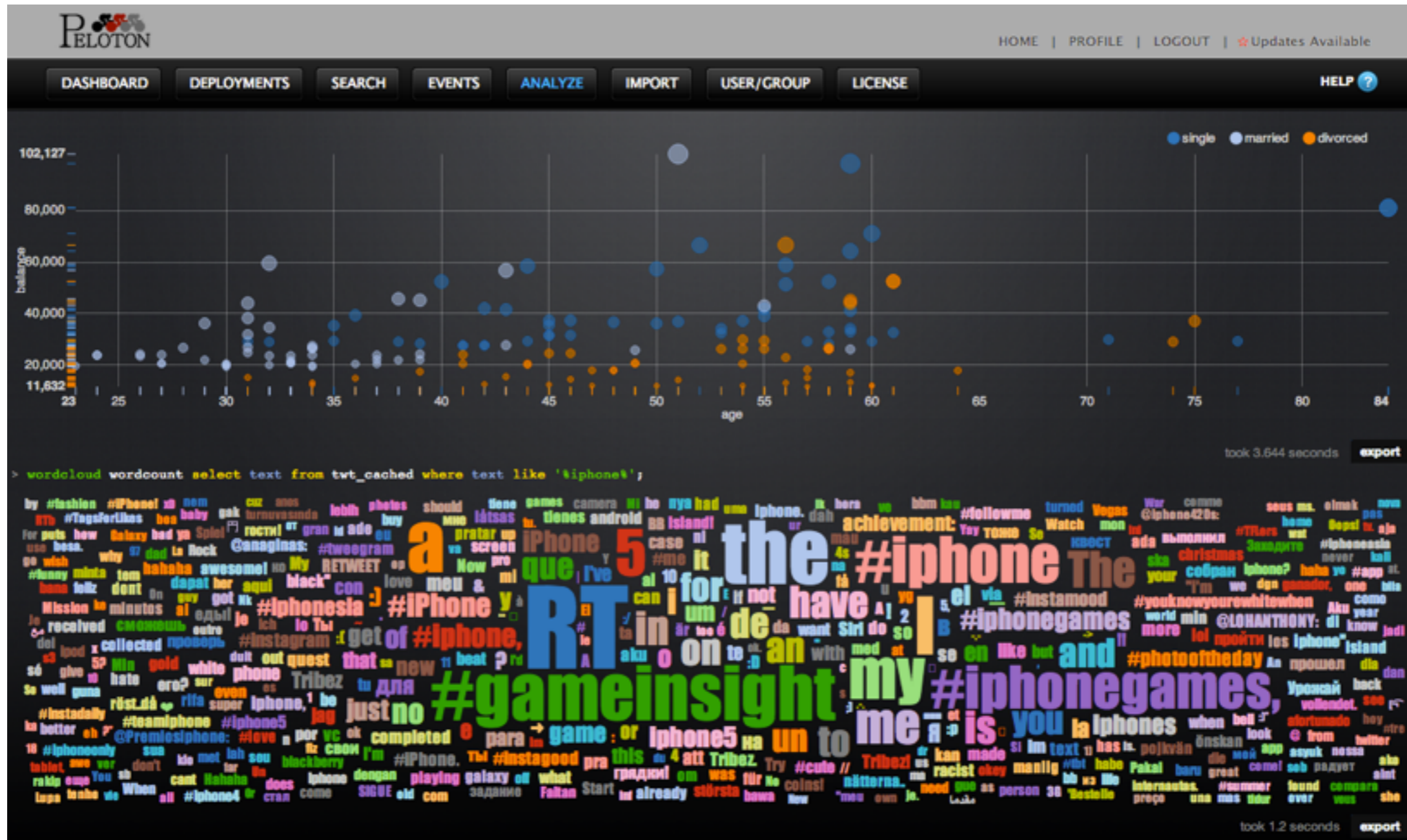
Many Libraries

Interactive

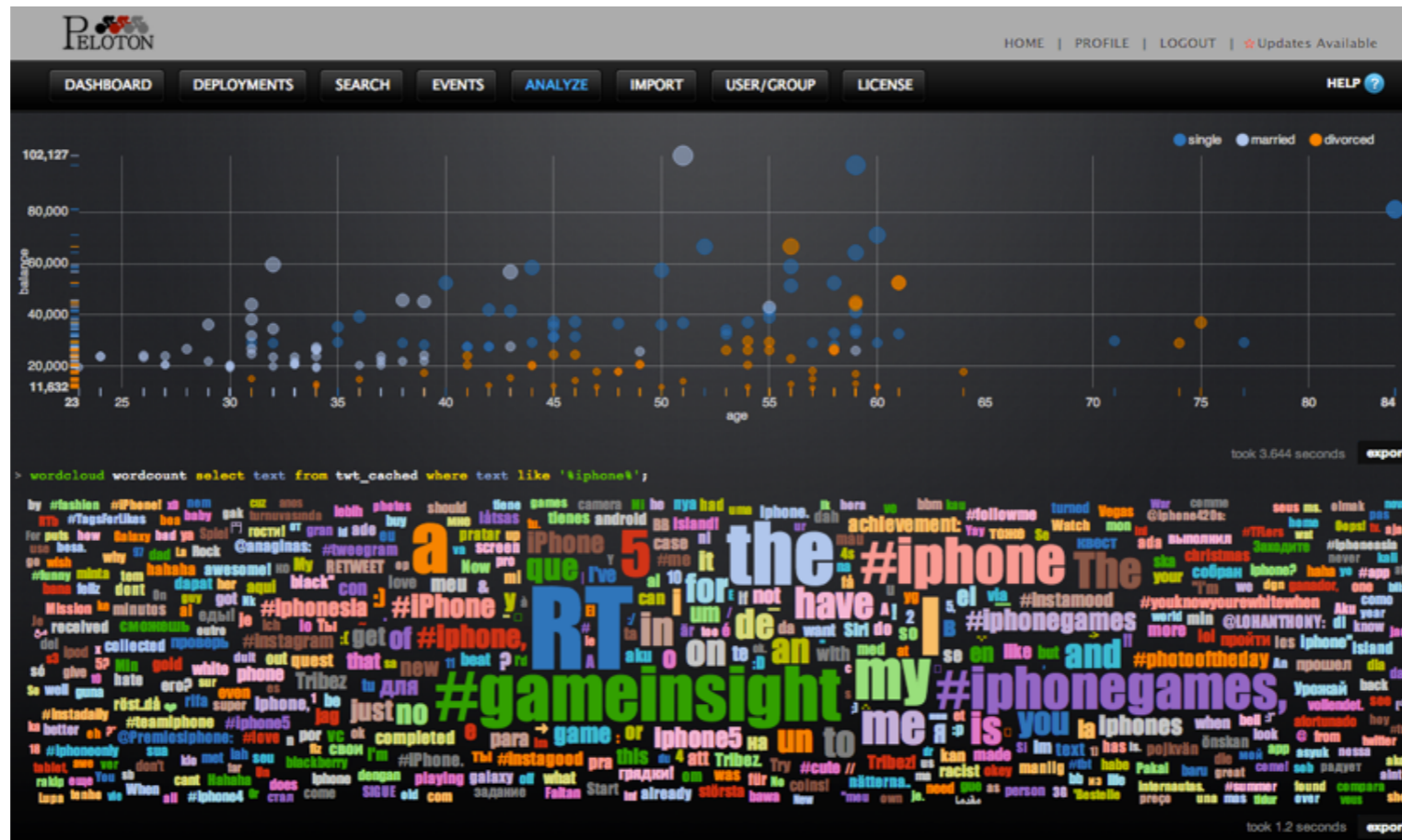
Visualization

Sharing

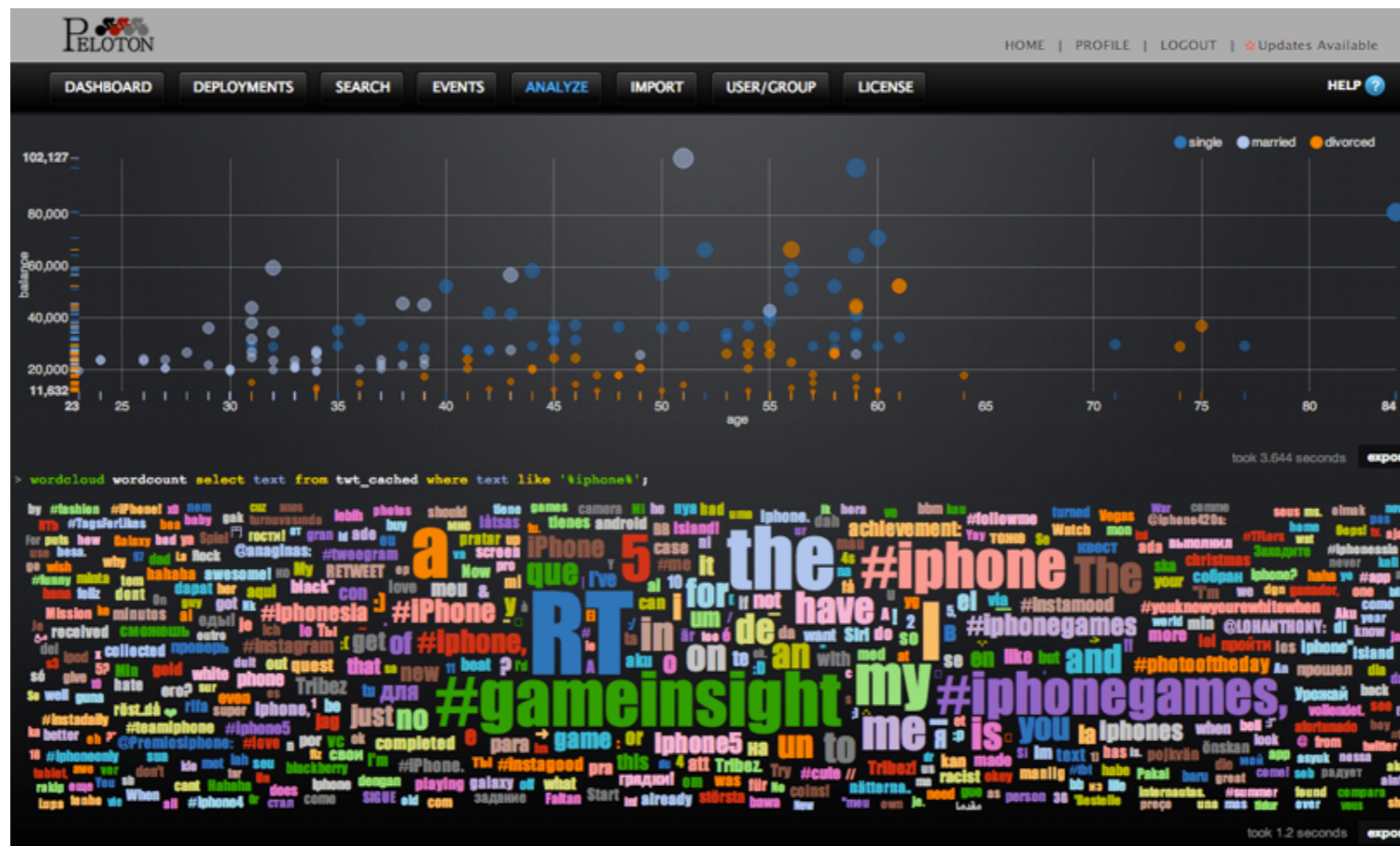
The First attempt 2012~2013



It's got graphic REPL, deployment, search, import tool

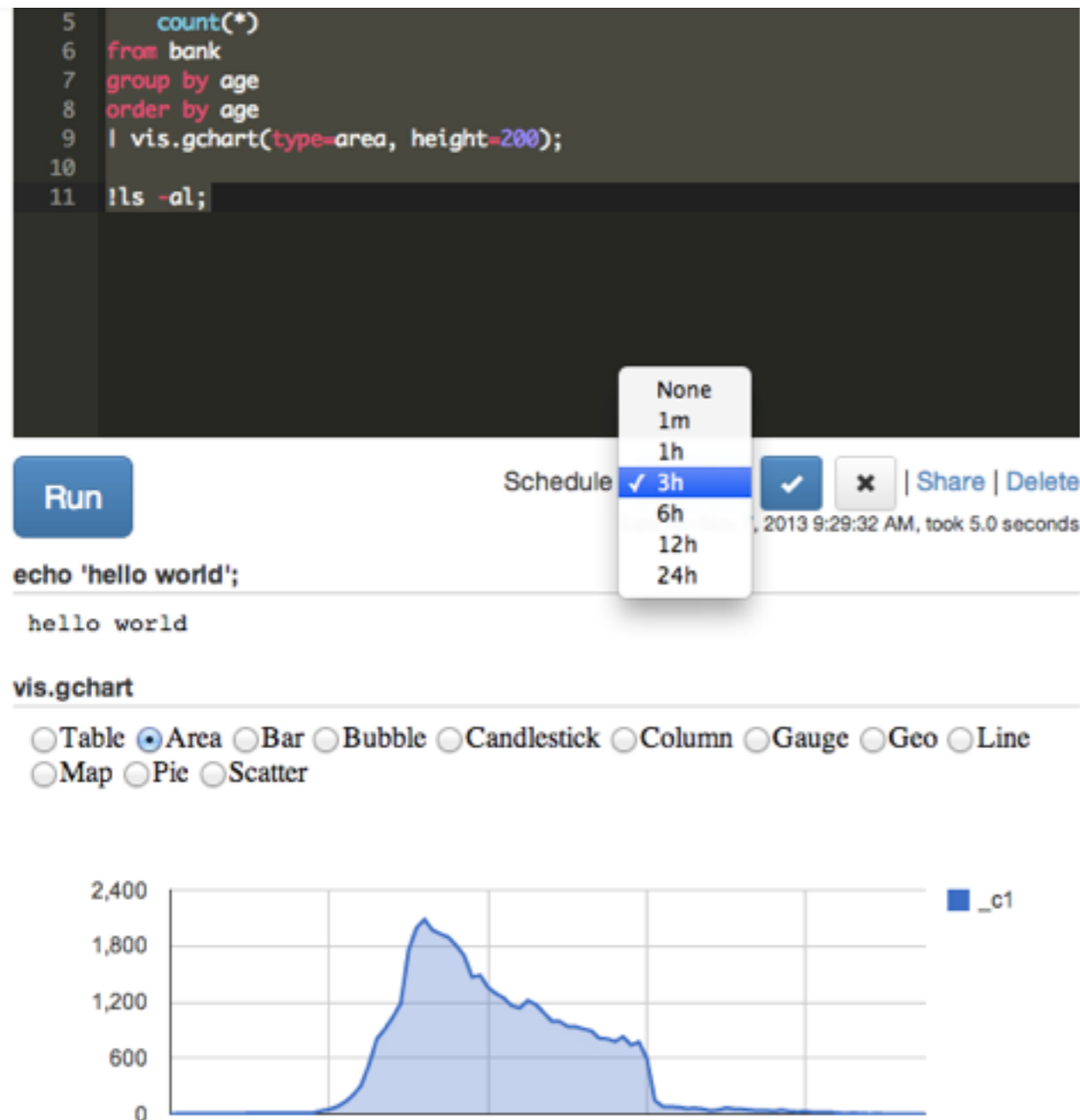


But failed, because



It wasn't widely used
It wasn't opensource

Second attempt 2013~2014



Opensourced
graphic REPL
from commercial
product

The first version
of Zeppelin

Second attempt 2013~2014



Not widely used

It was slow,
difficult to use,

...

Third attempt 2014~

After few weeks of study,
decided to rewrite Zeppelin

Graphic REPI -> Notebook
with Apache Spark
integration

The screenshot displays the Zeppelin web interface in a browser window. The address bar shows 'localhost:3333'. The interface includes a navigation bar with 'Zeppelin', 'ZQL', and 'ZAN' tabs. The main content area is divided into several sections:

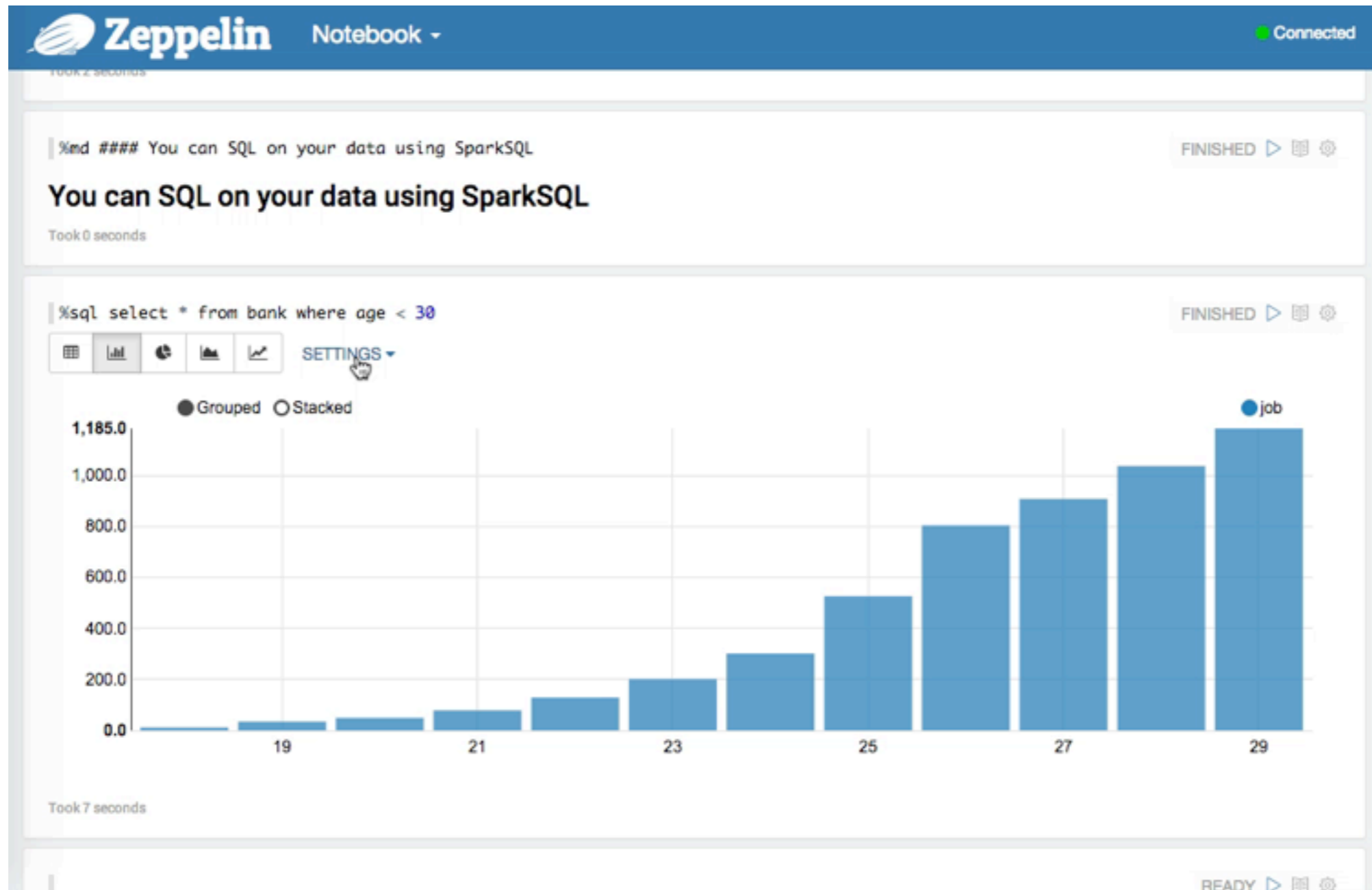
- Code Block:** Contains Scala code for loading a CSV file, parsing it into a 'Person' class, and registering it as a table named 'bank'.

```
val textData = sc.textFile("/Users/noon/Projects/zeppelin/bank-full.csv")
val bank = textData.map(s=>s.split(";")).map(
  s=>Person(s(0).toInt, s(1).replaceAll("\\\\", ""), s(2).replaceAll("\\\\", ""), s(3).replaceAll("\\\\", ""), s(5).t
bank.registerAsTable("bank")
```
- Output:** Shows the execution results, including the definition of the 'Person' class and the RDDs for 'textData' and 'bank'.

```
defined class Person
textData: org.apache.spark.rdd.RDD[String] = MappedRDD[1] at textFile at :15
bank: org.apache.spark.rdd.RDD[Person] = MappedRDD[3] at map at :19
```
- SQL Query:** A query is entered: `%sql select age, avg(balance), min(balance) from bank where age group by age order by age`.
- Chart:** A grouped bar chart visualizes the query results. The x-axis represents 'age' (ranging from 28 to 95), and the y-axis represents 'balance' (ranging from -8,019.0 to 19,511.1). The chart shows a significant peak in balance for the age group around 83.

At the bottom of the interface, there is a license notice: "Licensed under the Apache License, Version 2.0. See third-party tools/resources that Zeppelin uses and their respective authors."

Third attempt 2014~



Next week, beautifulized

Why do you like Zeppelin?

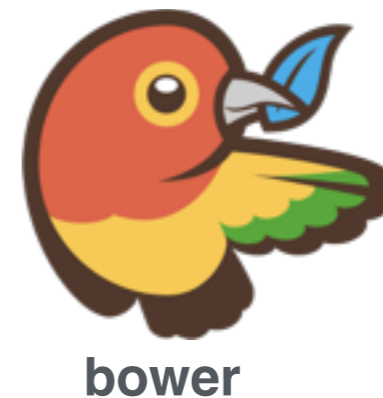
Web Based



Visualization



Package management



Build



Language



Notebook

Notebook

Code

```
val a=1
```

FINISHED

Result

```
a: Int = 1
```

Took 24 seconds

Code

```
println(s"""%table
name\tage
sun\t60
moon\t50""")
```

FINISHED



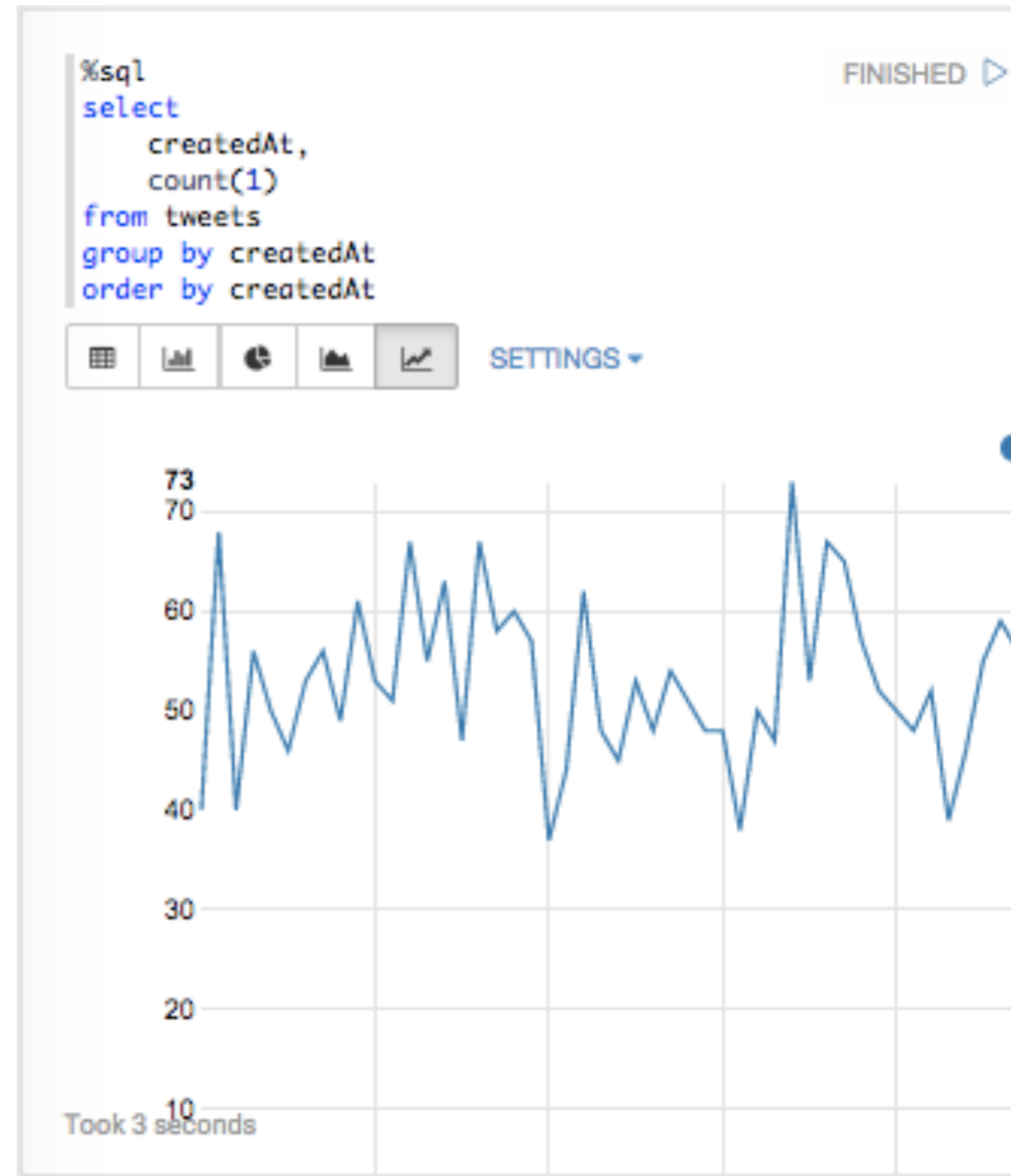
Result

name	age
sun	60
moon	50

결과가 탭과 뉴라인으로 구분된 테이블 형식이면
자동으로 테이블로 포메팅

Data Visualize

Notebook



Pivot

Pivot



Dynamic Form

Dynamic Form Creation

```
%sql  
select age, count(1) value  
from bank  
where age < ${maxAge=30}  
group by age  
order by age
```

maxAge

30



SETTINGS ▾

● Grouped ○ Stacked

1,185.0

1,000.0

800.0

600.0

400.0

200.0

0.0

19

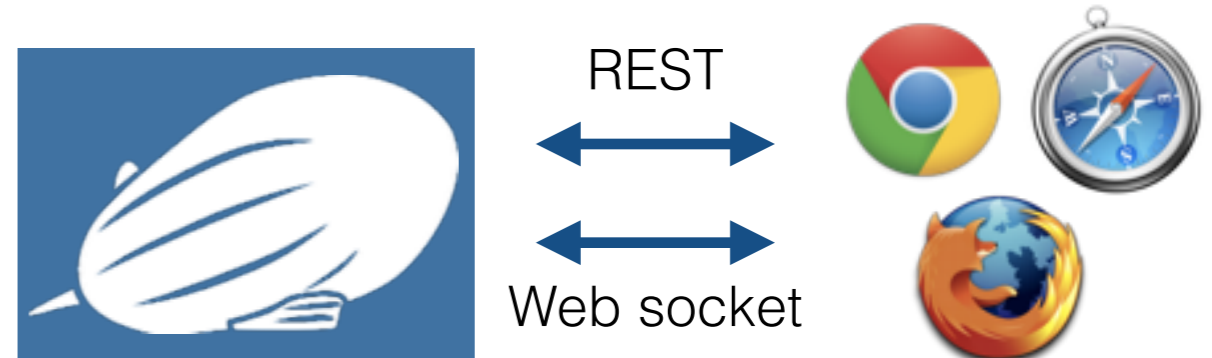
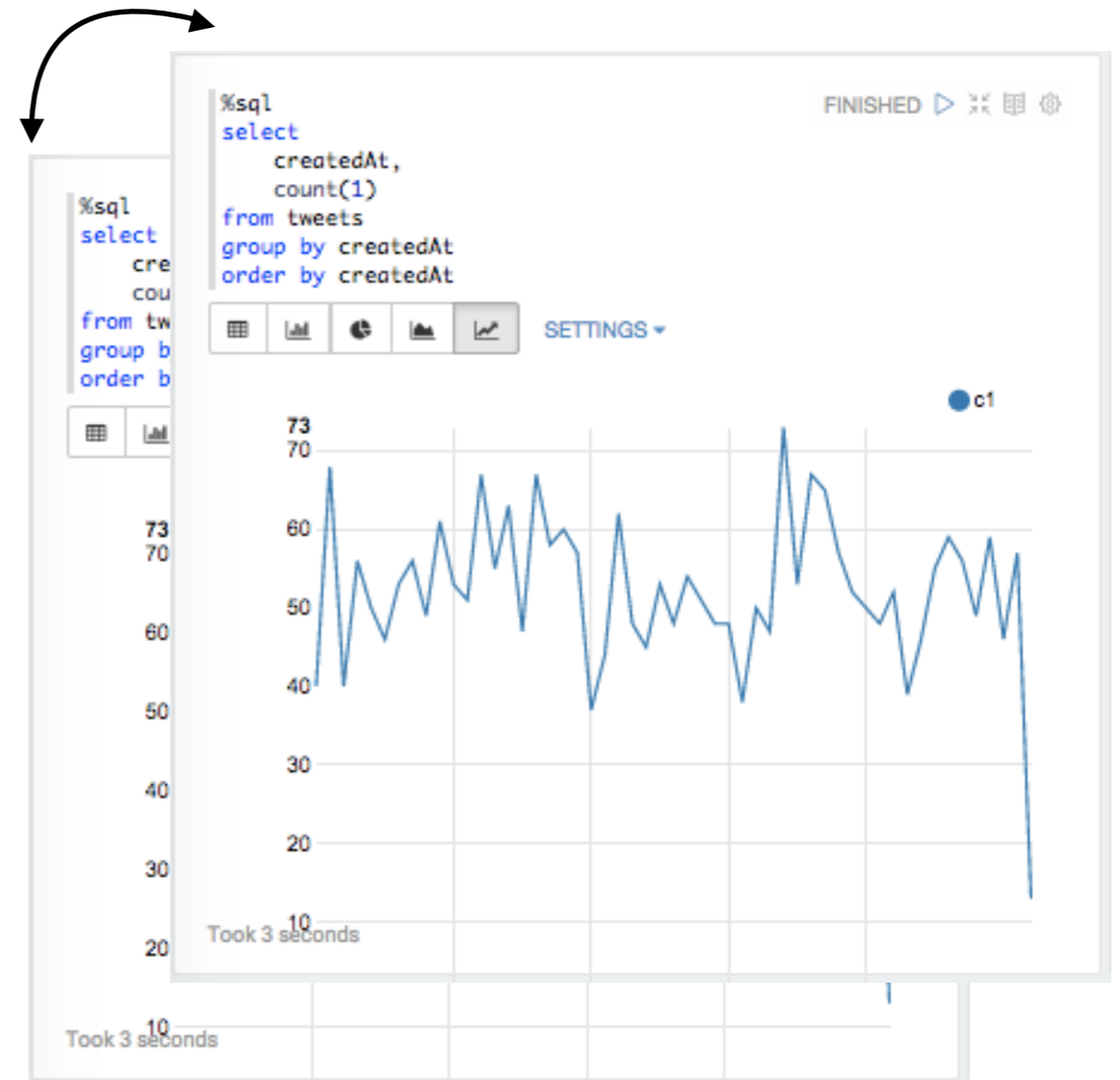
21

23

Test @ ...

Sharing

Synchronized



Zeppelin simplifies data
analysis

Why do your project likes
Zeppelin?

Easy to extend



Interpreters

Spark

PySpark

SparkSQL

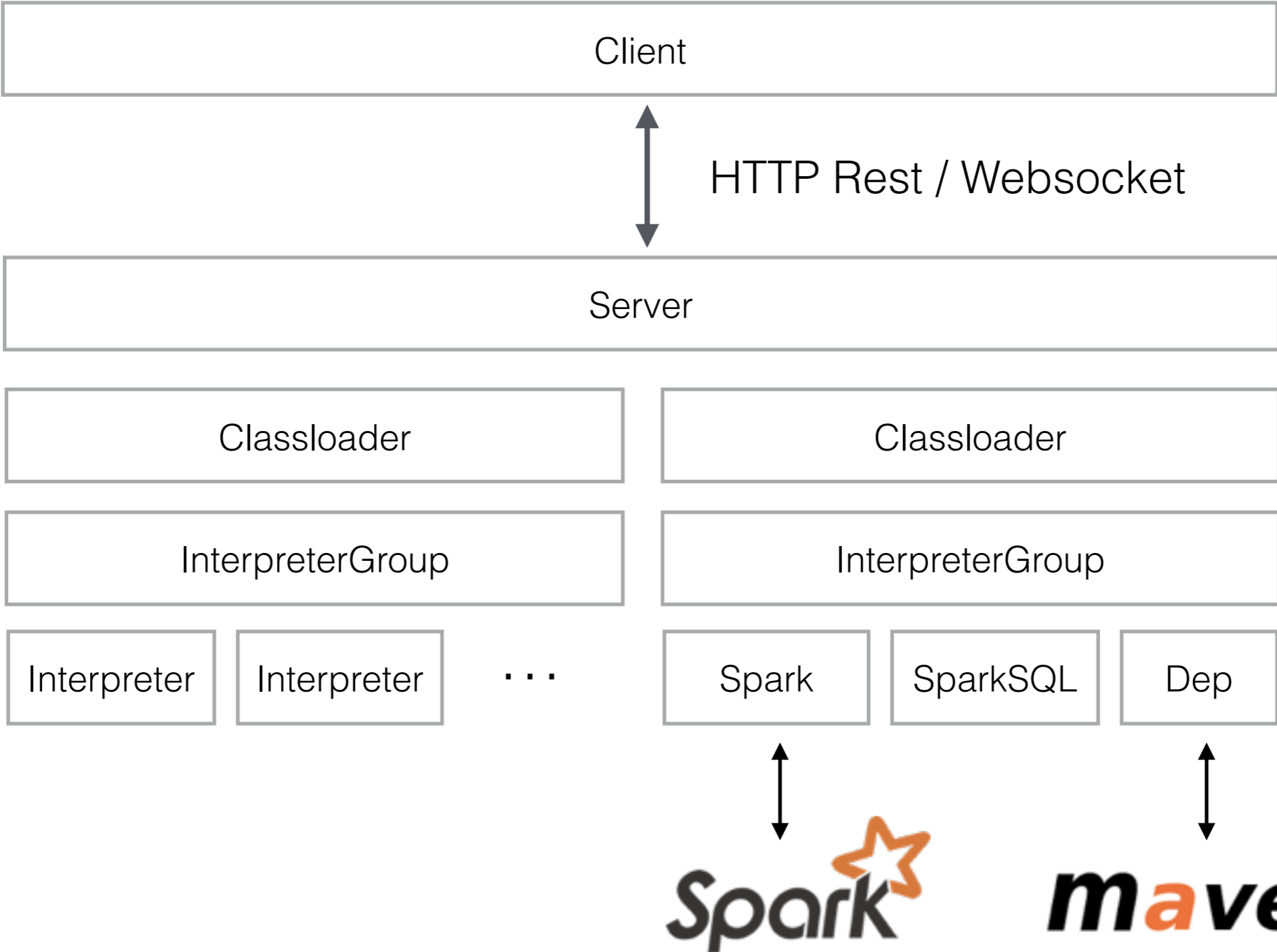
Hive

Mysql (JDBC)

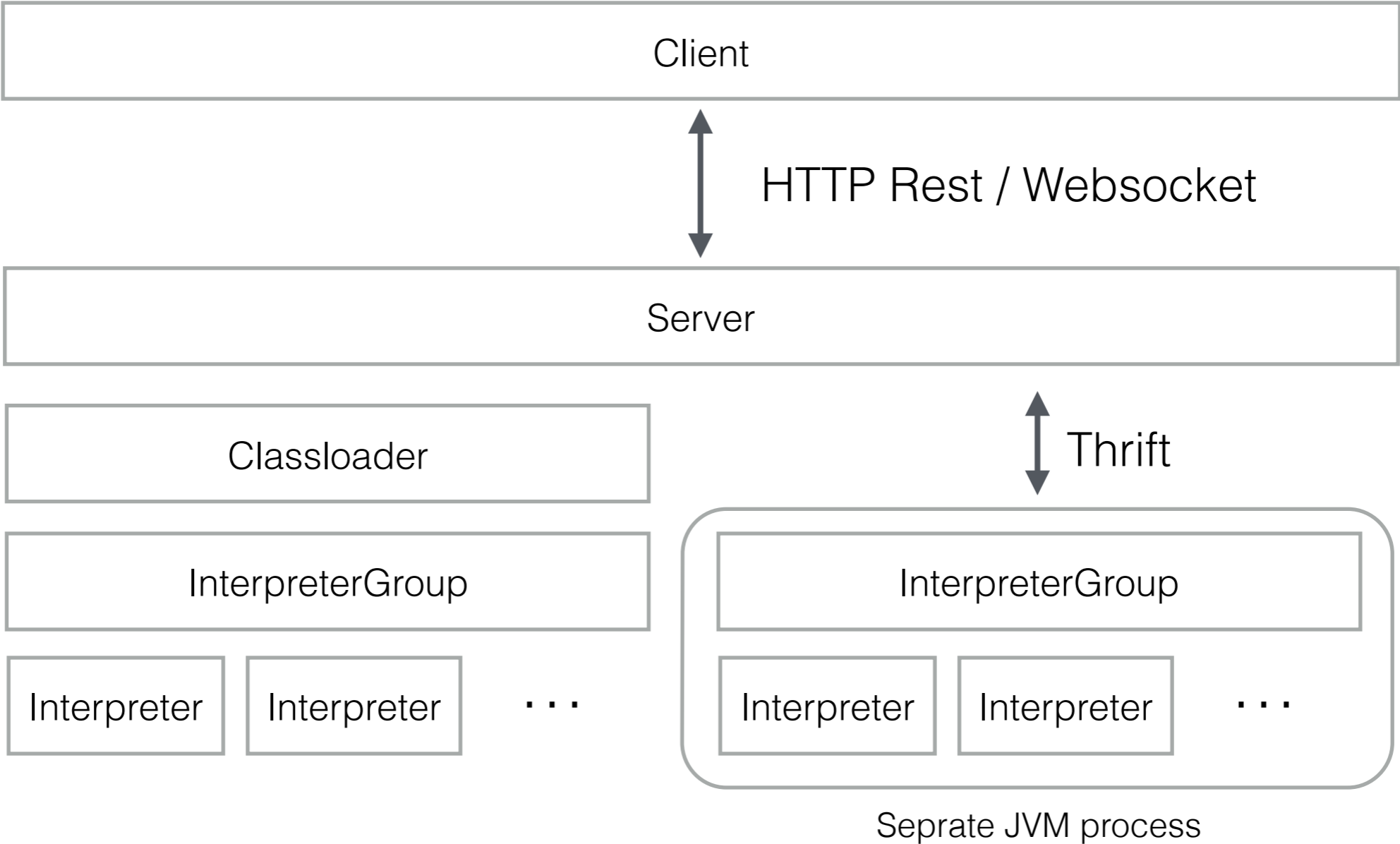
Markdown

Shell

Zeppelin Interpreter Architecture



Zeppelin Interpreter Architecture



Implementing new Interpreter

Must
have

```
public abstract void open();  
public abstract void close();  
public abstract InterpreterResult interpret(String st, InterpreterContext context);
```

Good to
have

```
public abstract void cancel(InterpreterContext context);  
public abstract int getProgress(InterpreterContext context);  
public abstract List<String> completion(String buf, int cursor);
```

More
controls

```
public abstract FormType getFormType();  
public Scheduler getScheduler();
```

Roadmap

- Integration with more distributed processing framework
 - Flink, Ignite, Tajo, etc..
- Output message streaming
- Ability to create rich GUI



MLLib

Cloudera-ML



MRQL

Drill

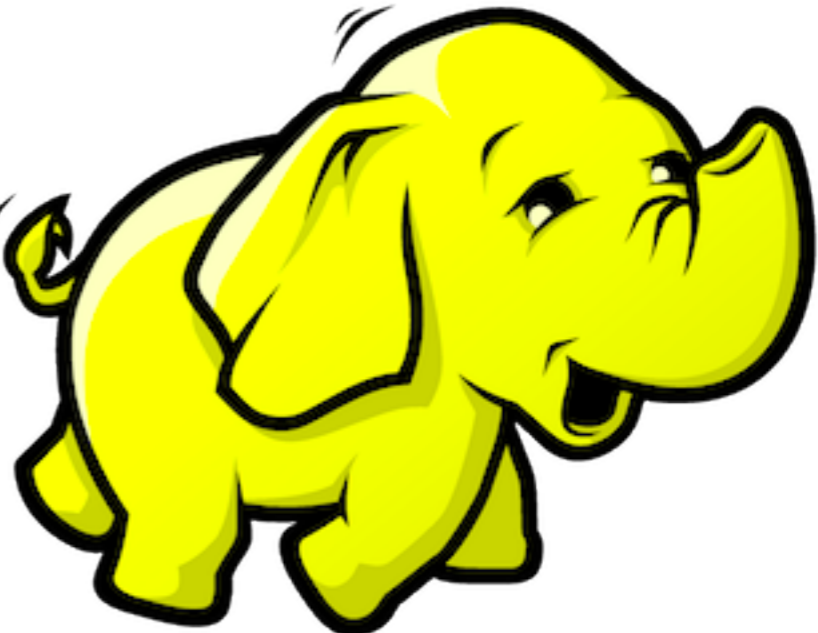
Impala



Pig

Hive

tajo



Thanks

Lee moon soo
moon@apache.org