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

Cassandra
HBase

accumulo

Spark  MapReduce

HDFS

Scoop

oozie
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 REPl -> Notebook with Apache Spark integration
Third attempt 2014~

Next week, beautifulized
Why do you like Zeppelin?
Web Based

**Web framework**
- AngularJS by Google
- d3.js

**Language**
- TypeScript

**Package management**
- bower

**Build**
- Grunt
결과가 텍스트로 표시된 테이블 형식이 멀티줄로 구분된 경우, 각 줄이 자동으로 테이블로 포맷됩니다.
Data Visualize

Notebook

```sql
select createdAt, count(1) from tweets group by createdAt order by createdAt
```
Pivot
Dynamic Form
Sharing
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

Client

HTTP Rest / Websocket

Server

Classloader

InterpreterGroup

Interpreter Interpreter Interpreter

InterpreterGroup

Interpreter Interpreter Interpreter

Seperate JVM process
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
Thanks

Lee moon soo
moon@apache.org