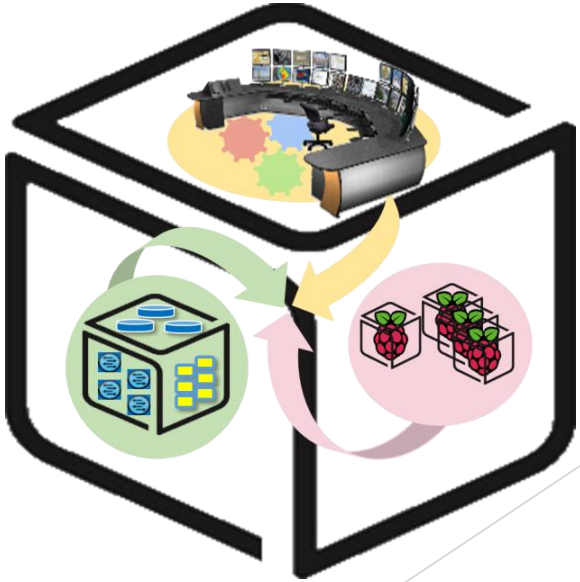
Computer Systems Lab.

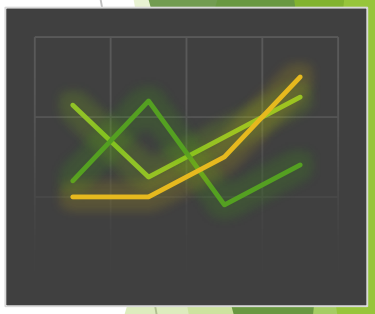
Computer Systems Lab @ Spring 2016



CSLab: Analytics LAB

Data Processing & Visualization

Data



Analytics Lab
- Final Goal

- Install Mesos, Spark, Zeppelin on NUC
- Data Processing with Spark & Zeppelin

Week 1

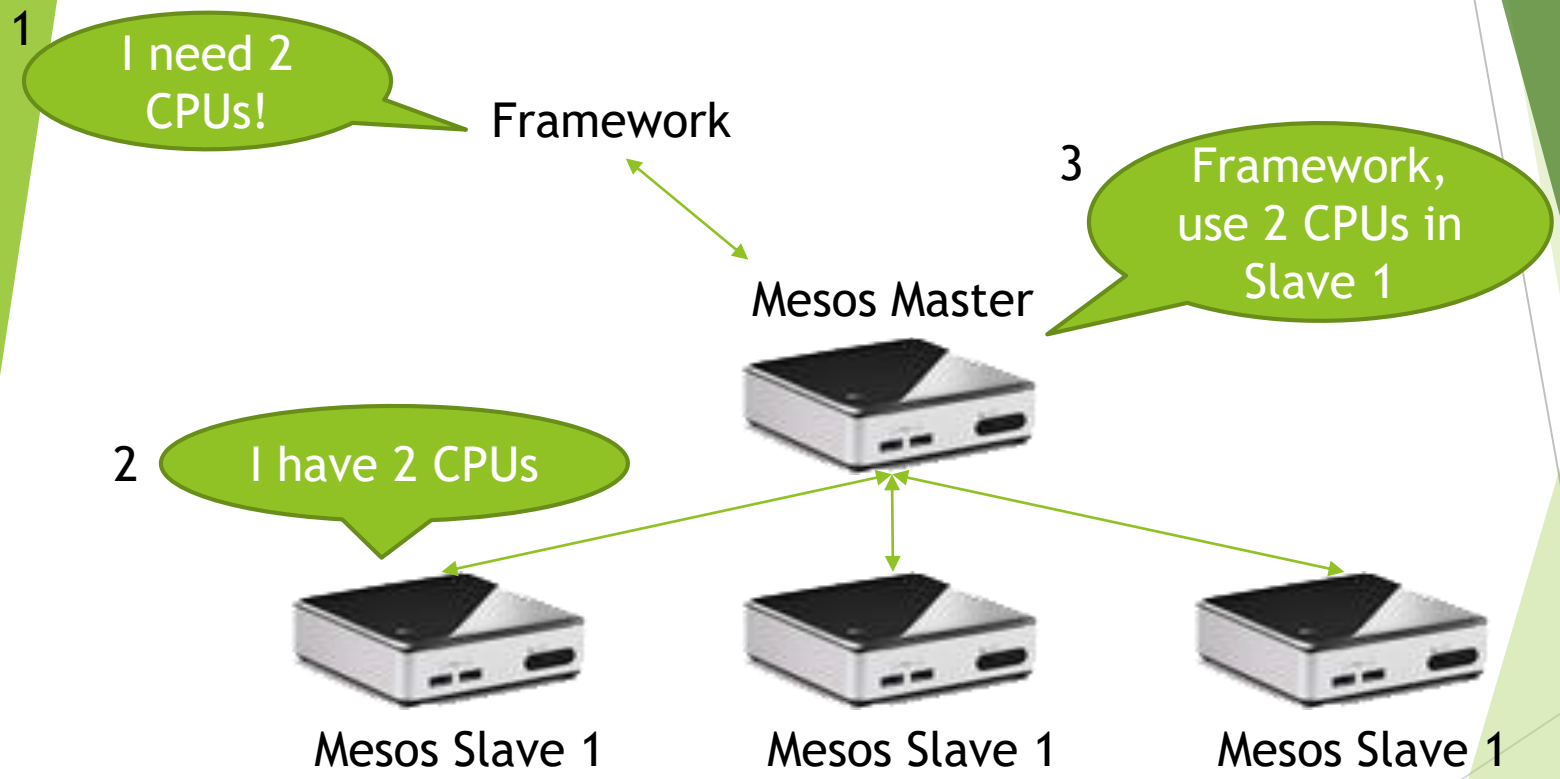
- Mesos, Spark and Zeppelin: Introduction and Configuration

Apache Mesos
- Concept

What is Mesos?
A distributed systems kernel
Mesos is built using the same principles as the Linux kernel, only at a different level of abstraction. The Mesos kernel runs on every machine and provides applications (e.g., Hadoop, Spark, Kafka, Elastic Search) with API's for resource management and scheduling across entire datacenter and cloud environments.

- Cloud as a single computer
- Share resources across the machines

Apache Mesos
- Architecture

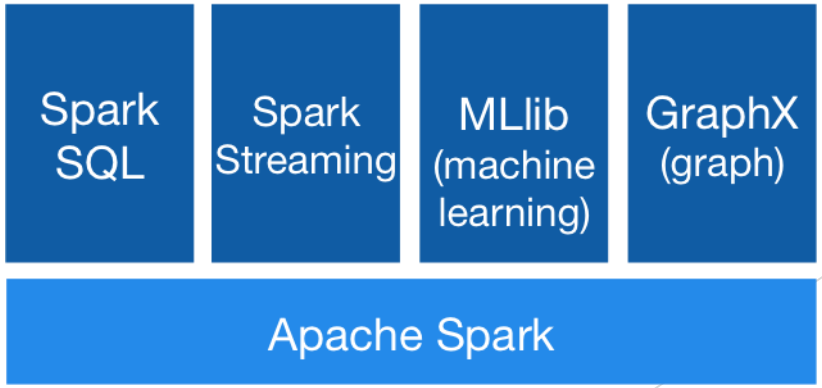


Apache Spark
- Concept

Apache Spark™ is a fast and general engine for large-scale data processing.

- In-memory data processing framework: Fast!
- Easy to use, community fastly growing
- Libraries: SQL and DataFrame, Streaming, MLlib, GraphX
- Run on standalone or Mesos, Yarn, etc

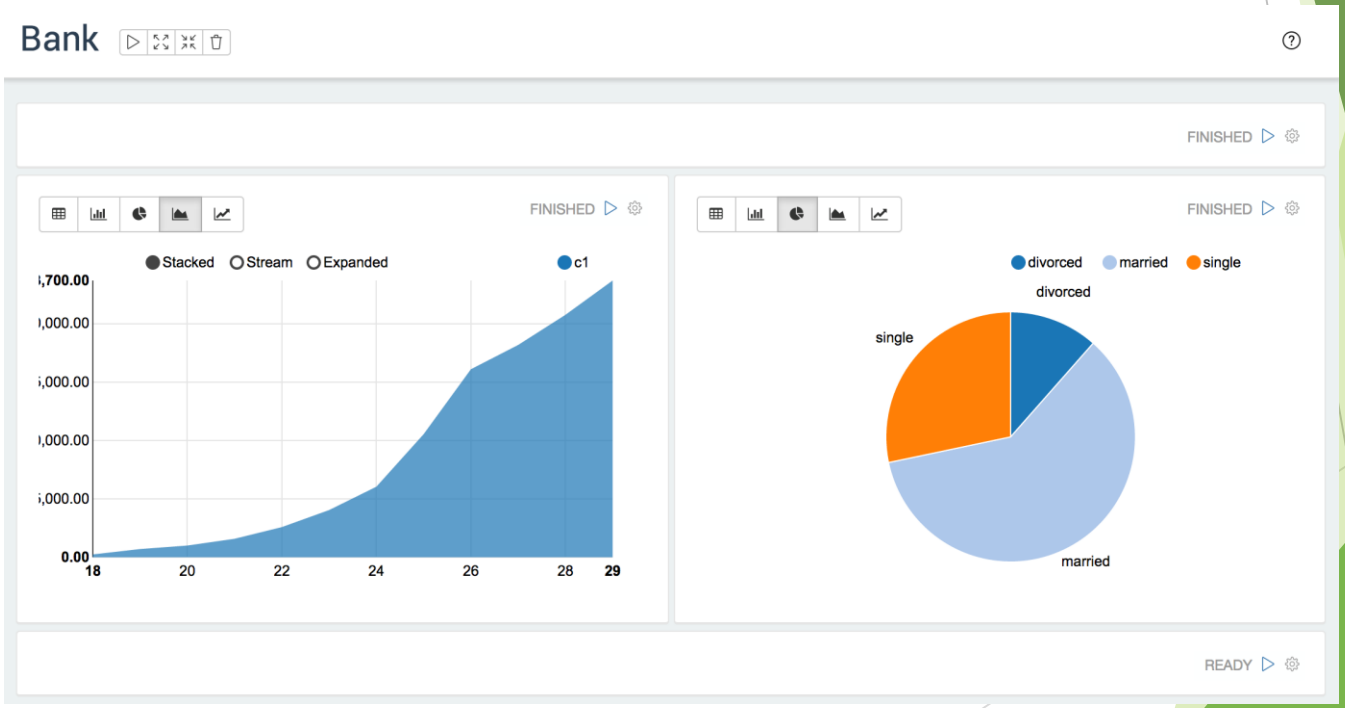
- Scala, Java, Python



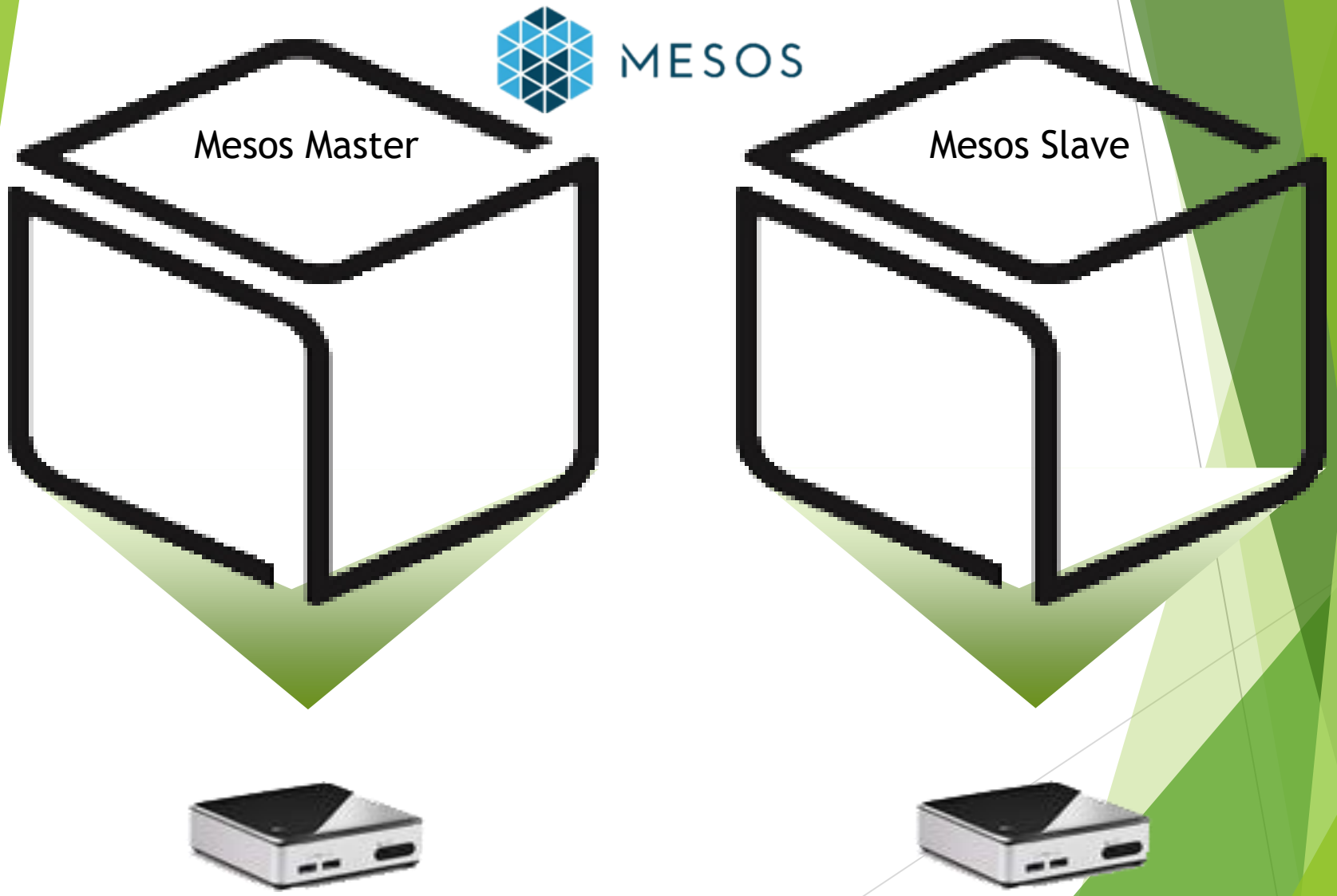
Apache Zeppelin
-Concept

A web-based notebook that enables interactive data analytics.

Support Spark



1. Apache Mesos
- Install & Configuration



1. Apache Mesos
- Install

Prerequest: Ubuntu must be 64bit

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv E56151BF

DISTRO=$(lsb_release -is | tr '[:upper:]' '[:lower:]')

CODENAME=$(lsb_release -cs)

echo "deb http://repos.mesosphere.io/${DISTRO} ${CODENAME} main" | sudo
tee /etc/apt/sources.list.d/mesosphere.list

sudo apt-get -y update
```

1. Apache Mesos
- Install: Mesos Master

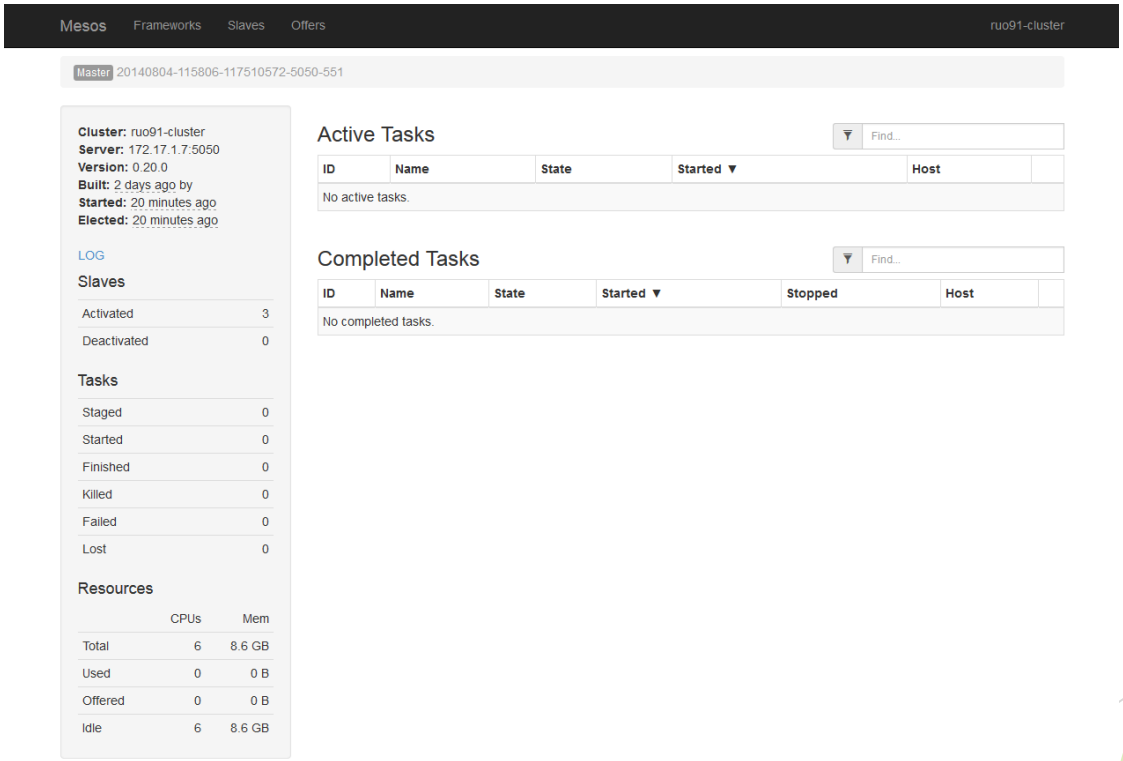
```
sudo apt-get -y install mesos marathon
sudo reboot

sudo service mesos-slave stop
echo manual | sudo tee /etc/init/mesos-slave.override
echo <IP_ADDR> | sudo tee /etc/mesos-master/ip
echo <IP_ADDR> | sudo tee /etc/mesos-master/hostname
echo zk://<IP_ADDR>:2181/mesos | sudo tee /etc/mesos/zk
echo <YOUR_NAME> | sudo tee /etc/mesos-master/cluster
sudo service zookeeper restart
sudo service mesos-master restart
sudo service marathon restart

echo 1 | sudo tee /etc/zookeeper/conf/myid
```

1. Apache Mesos
- Install: Mesos Slave

```
sudo apt-get -y install mesos
sudo reboot

sudo service mesos-master stop
echo manual | sudo tee /etc/init/mesos-master.override
sudo service zookeeper stop
echo manual | sudo tee /etc/init/zookeeper.override
sudo apt-get -y remove --purge zookeeper

echo <SLAVE_IP_ADDR> | sudo tee /etc/mesos-slave/ip

echo <SLAVE_IP_ADDR> | sudo tee /etc/mesos-slave/hostname

echo zk://<MASTER_IP_ADDR>:2181/mesos | sudo tee /etc/mesos/zk
sudo reboot
```

1. Apache Mesos
- Web UI

`http://<MASTER-IP-ADDR>:5050`



2. Apache Spark
- Install

※ Install on every NUC

```
wget http://mirror.apache-kr.org/spark/spark-1.5.2/spark-1.5.2-bin-hadoop2.6.tgz
tar xzf spark-1.5.2-bin-hadoop2.6.tgz

cd spark-1.5.2-bin-hadoop2.6

cd spark-1.5.2-bin-hadoop2.6/conf/
cp spark-env.sh.template spark-env.sh
vi spark-env.sh
```

Edit:

```
export MESOS_NATIVE_JAVA_LIBRARY=/usr/local/lib/libmesos.so
export MASTER=mesos://<MESOS MASTER IP ADDR>:5050
```

```
# Test Spark
cd ..
bin/pyspark

data = range(1, 10001)
distData = sc.parallelize(data)
distData.filter(lambda x: x < 10).collect()
```

Go to Mesos web UI and see Spark framework running.

3. Apache Zeppelin
- Install (on Mesos)

wget http://mirror.apache-kr.org/incubator/zeppelin/0.5.5-incubating/zeppelin-0.5.5-incubating-bin-all.tgz

tar xzf zeppelin-0.5.5-incubating-bin-all.tgz

cd zeppelin-0.5.5-incubating-bin-all/conf
cp zeppelin-env.sh.template zeppelin-env.sh
vi zeppelin-env.sh

Edit:

export MESOS_NATIVE_JAVA_LIBRARY=/usr/local/lib/libmesos.so
export MASTER=mesos://<MESOS MASTER IP ADDR>:5050

cd ..
bin/zeppelin-daemon.sh start

http://<IP-ADDR>:8080

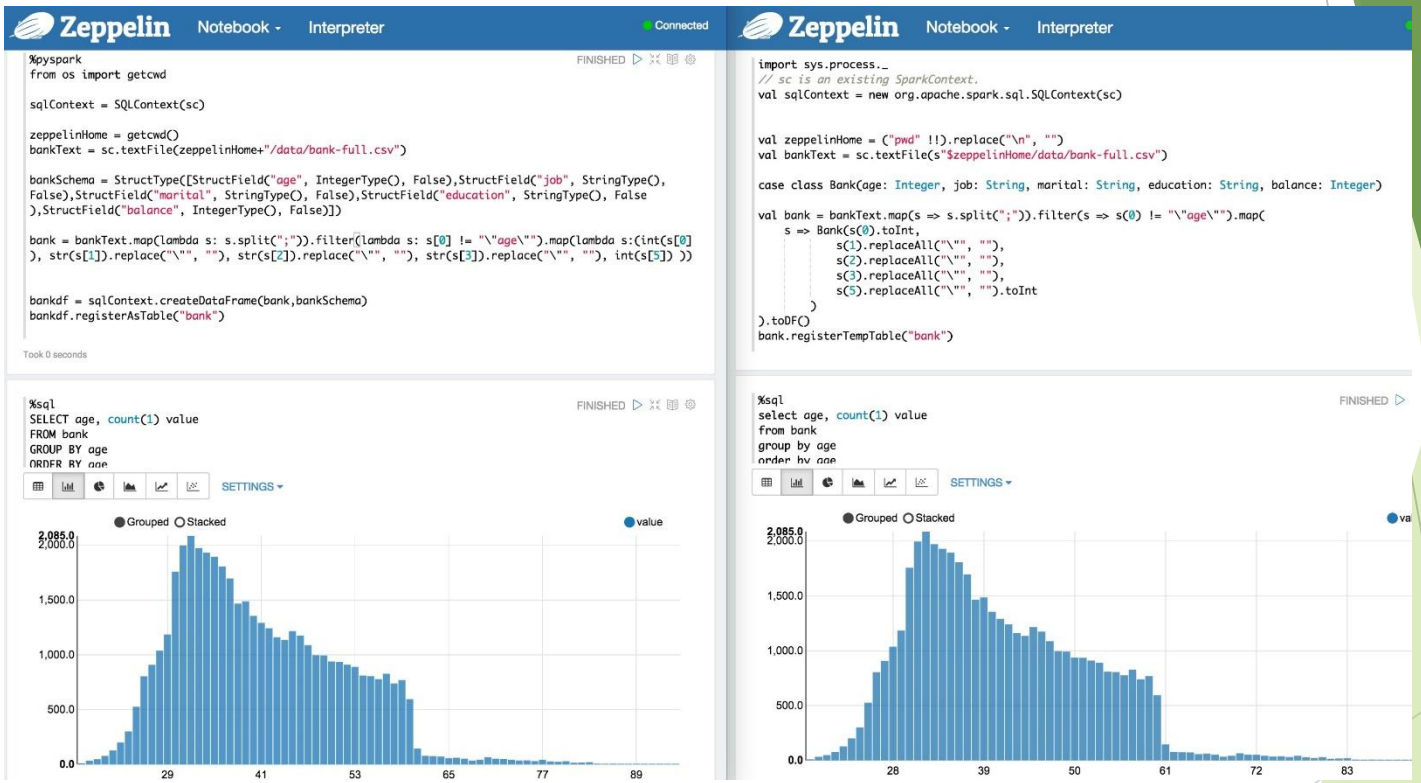
3. Apache Zeppelin
- Install (standalone mode)

If you have trouble running Zeppelin on Mesos, you can run Zeppelin in standalone mode.

```
rm conf/zeppelin-env.sh
bin/zeppelin-daemon.sh start      #(or if daemon is already running, use 'restart' instead of 'start.')
```

http://<IP-ADDR>:8080

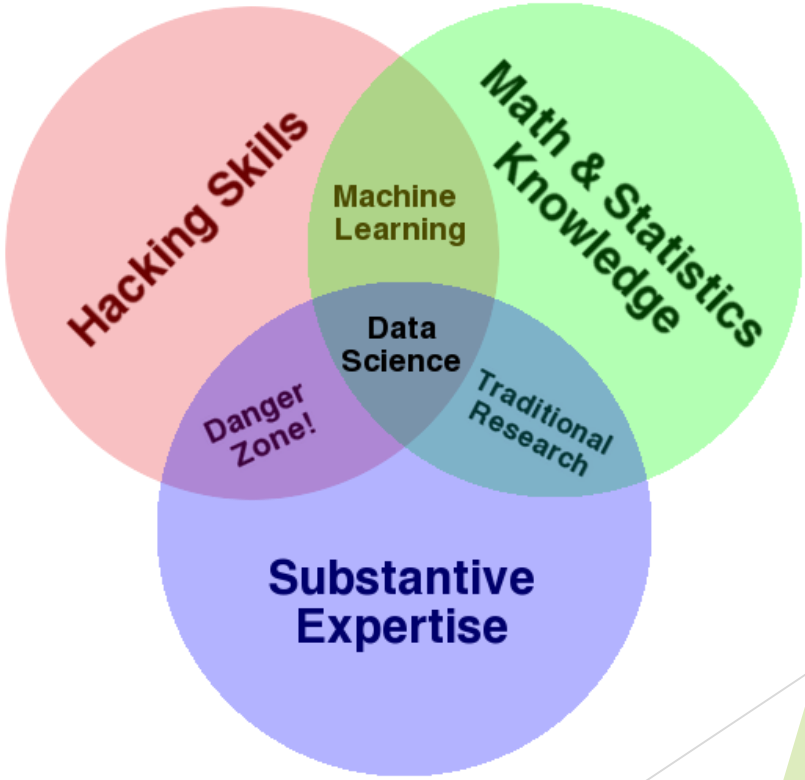
3. Apache Zeppelin
- Run Big Data Job



Press ‘Run’ button to test the sample codes.

Week 2

- More about Spark and Zeppelin
- Big data processing



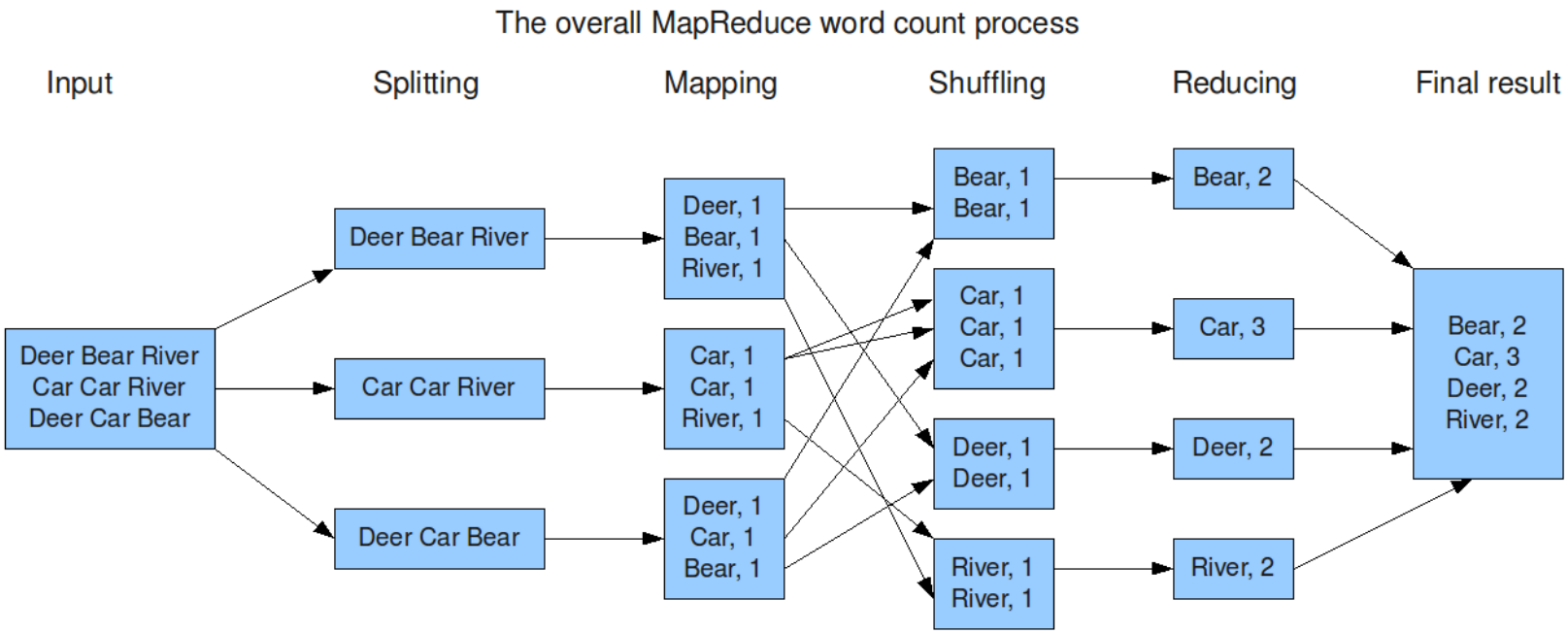
Tip. Pyspark: Spark in Python - Sample

Zeppelin tutorial converted to pyspark

```
%pyspark

from pyspark.sql.types import *

zeppelinHome = os.getcwd()
bankText = sc.textFile(zeppelinHome + "/data/bank.csv")

bankSchema = StructType([StructField("age", IntegerType(), False),
            StructField("job", StringType(), False),
            StructField("marital", StringType(), False),
            StructField("education", StringType(), False),
            StructField("balance", IntegerType(), False)])

bank = bankText.map(lambda s: s.split(";")).filter(lambda s: s[0] != "\"age\"").map(lambda s: (
            int(s[0]),
            str(s[1]).replace("\"", ""),
            str(s[2]).replace("\"", ""),
            str(s[3]).replace("\"", ""),
            int(s[5])))

bankdf = sqlContext.createDataFrame(bank, bankSchema)
bankdf.registerTempTable("bank")
```

```
# In zeppelin directory, make data directory and download sample data file.
$ cd zeppelin-0.5.5-incubating-bin-all
$ mkdir data
$ cd data
$ wget https://s3.amazonaws.com/apache-zeppelin/tutorial/bank/bank.csv
```

2. Processing Big Data
- Map and Reduce



2. Processing Big Data
- Map and Reduce in Spark

RDD (Resilient Distributed Datasets): a distributed memory abstraction that allows programmers to perform in-memory computations on large clusters while retaining the fault tolerance of data flow models like MapReduce.

class pyspark.RDD

map()
groupBy(), groupByKey()
reduce(), reduceByKey()
join()
sort(), sortByKey()
union()
…

http://spark.apache.org/docs/latest/api/python/pyspark.html

3. Apache Zeppelin
- Wordcount

```
# Prepare data
cd zeppelin-0.5.5-incubating-bin-all
mkdir data
cd data
wget https://www.dropbox.com/s/dvtrxdr8am49yvv/wordcount.txt
```

wordcount.txt: **Remarks by President Obama at Hankuk University** (https://www.whitehouse.gov/the-press-office/2012/03/26/remarks-president-obama-hankuk-university)

3. Apache Zeppelin
- Wordcount

```
%pyspark

from pyspark.sql.types import *
import os

zeppelinHome = os.getcwd()
lines = sc.textFile(zeppelinHome + "/data/wordcount.txt")
counts = lines.flatMap(
                    ?

countSchema = StructType([
    StructField("word", StringType(), True),
    StructField("counts", IntegerType(), True)])
countdf = sqlContext.createDataFrame(counts, countSchema)
countdf.registerTempTable("wordcount")
```

Hint: filter(), map(), reduceByKey() and page 20

(If you have trouble using Zeppelin,
restart Zeppelin daemon.)
$ bin/zeppelin-daemon.sh restart

3. Apache Zeppelin
- Wordcount (Result)

```
%sql
select word, counts value
from wordcount
where counts > 20
order by counts
```

FINISHED

word	value
for	21
with	22
have	24
this	26
will	28
And	38
I	40
is	42
our	44
we	47

Took 0 seconds.

3. Apache Zeppelin
- Data Cleaning

Actually, there are 26 ‘have’, 37 ‘this’ in wordcount.txt

We need to remove punctuation marks like ‘.’, ‘,’, etc.

It can be done by re.sub() and map() functions.

Add some codes to make dirty data clean and get the right answer.

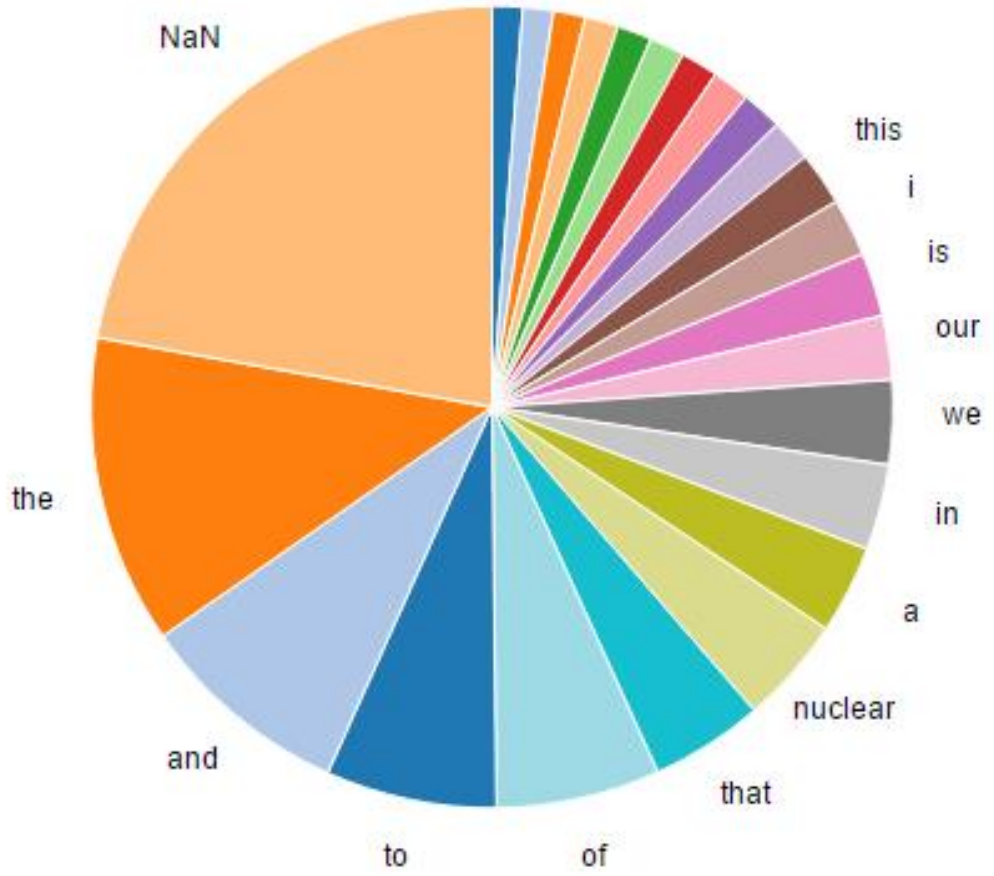
This is to make complex model simple, or get better answer.

3. Apache Zeppelin
- Wordcount (fixed)

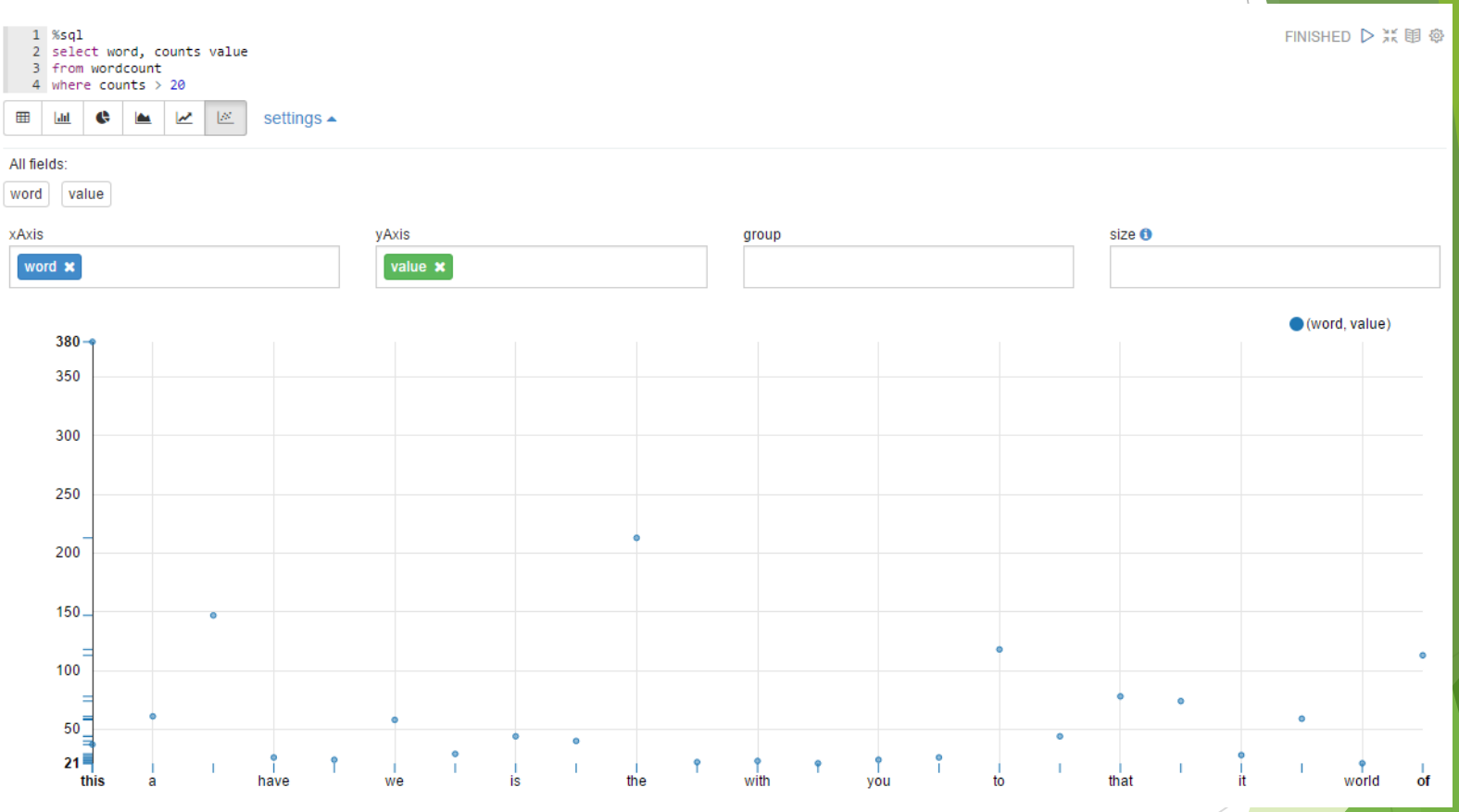
word	value
world	21
as	21
but	22
with	23
you	24
weapons	24
have	26
for	26
it	28
will	29

Took 0 seconds.

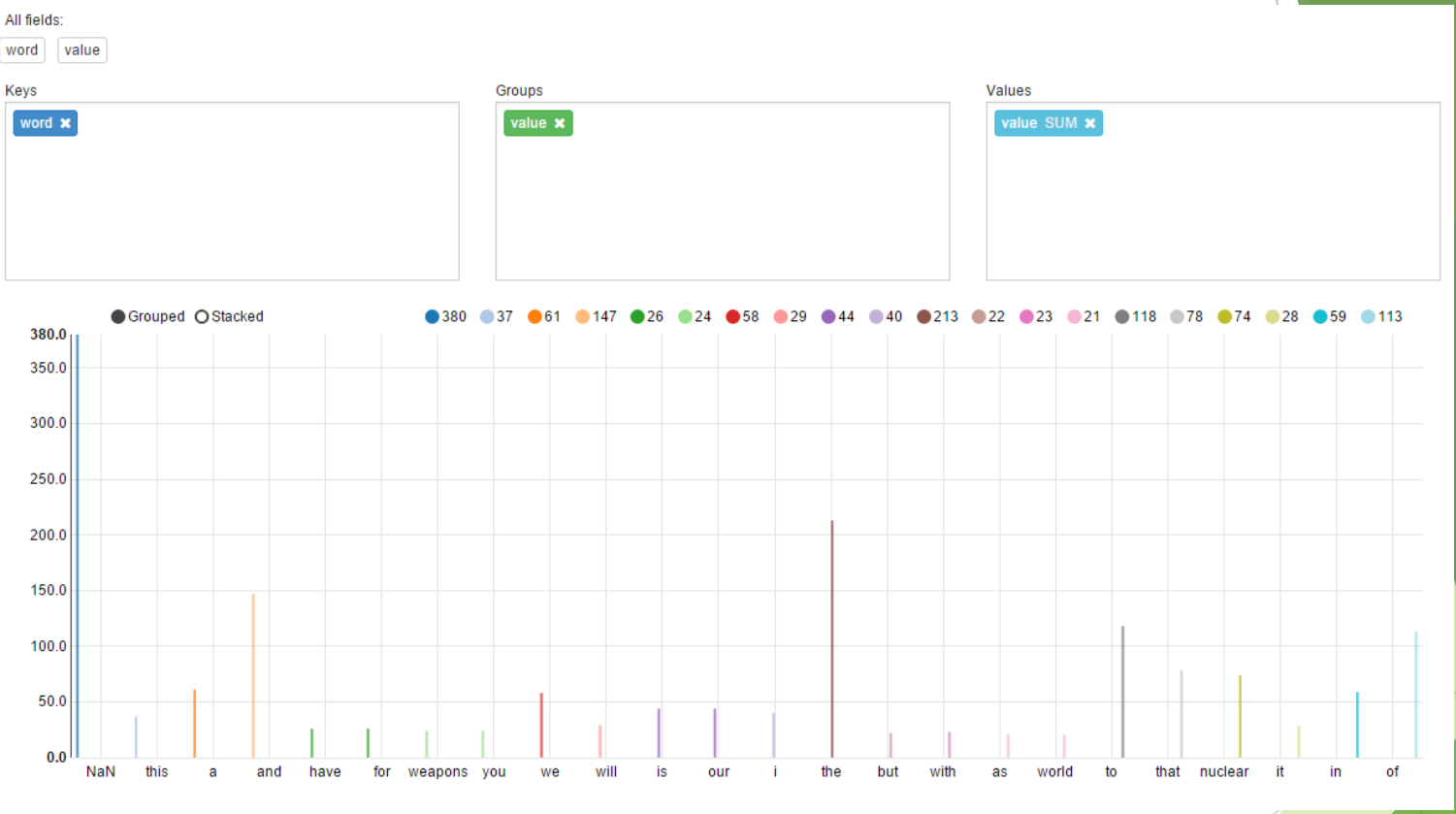
3. Apache Zeppelin
- Wordcount - graph



3. Apache Zeppelin
- Wordcount - graph



3. Apache Zeppelin
- Wordcount - graph



Thank You for Your Attention Any Questions?