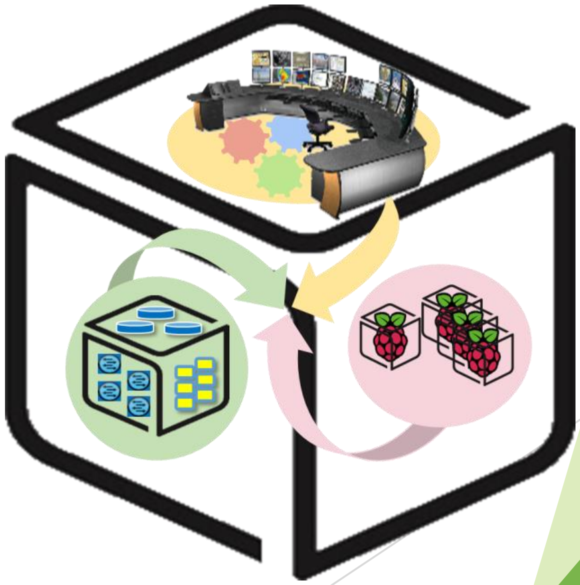
SmartX Labs for Computer Systems

Cluster Lab

(2016, Spring)

NetCS Lab



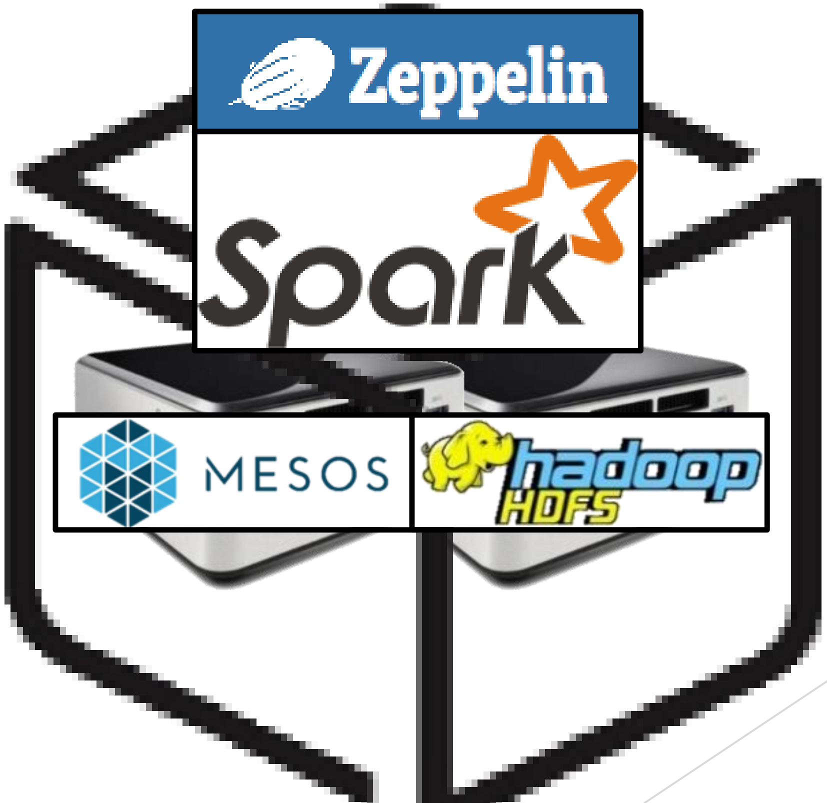
History and Contributor of Cluster Lab (2016. 05. 02.)

Version	Updated Date	Updated Contents	Contributor
-	2015/10	(구) Analytics Lab 작성	송지원
v1	2016/04	Cluster Lab 초안 작성	김승룡
v2	2016/05	Cluster Lab 수정	송지원

CSLab: Cluster LAB - Goal

SETUP to run data processing and visualization

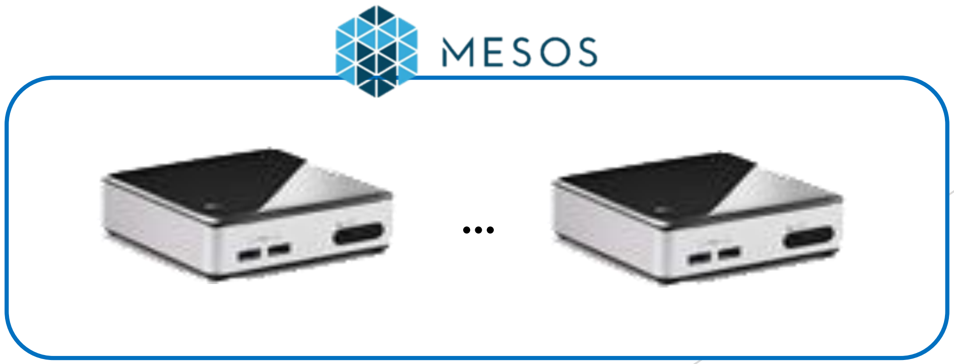
- Install Mesos, HDFS, Spark, Zeppelin on NUC



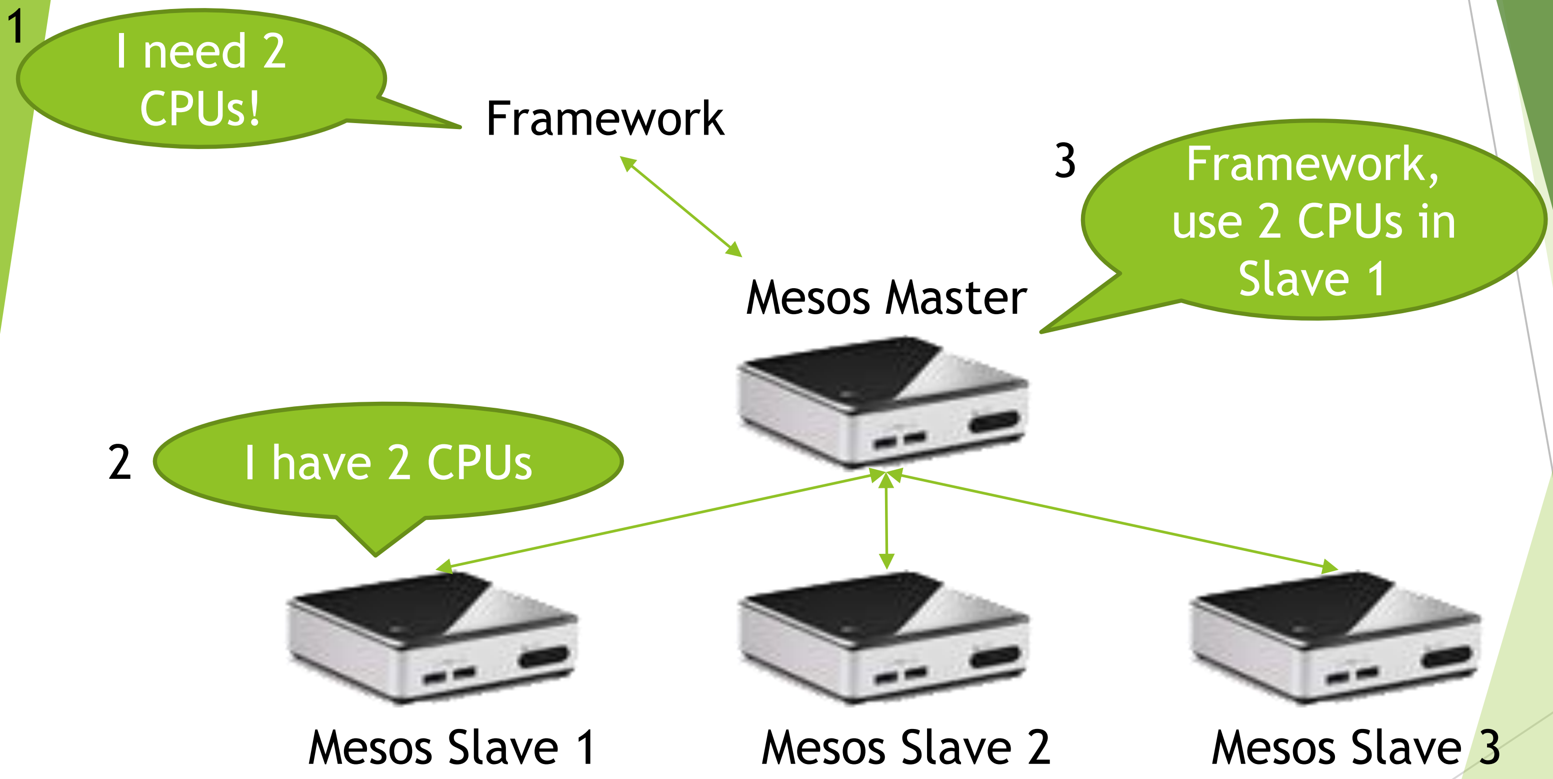
Apache Mesos
- Concept

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



HDFS
- Concept

Hadoop Distributed FileSystem

- A distributed file system that provides high-throughput access to application data.

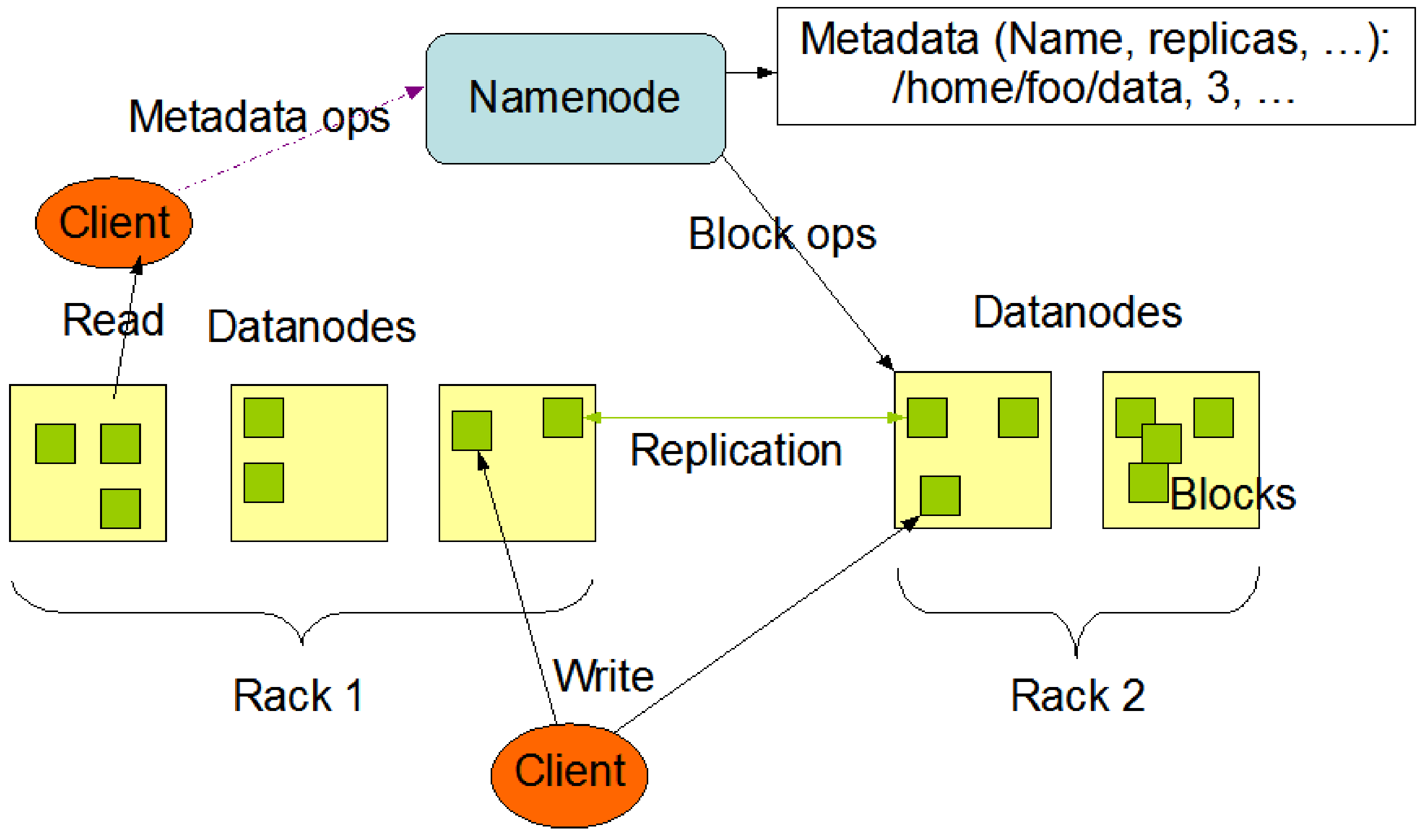
Features

- Fault tolerance by detecting faults and applying quick, automatic recovery
- Portability across heterogeneous commodity hardware and operating systems
- Scalability to reliably store and process large amounts of data
- Economy by distributing data and processing across clusters of commodity personal computers
- Efficiency by distributing data and logic to process it in parallel on nodes where data is located
- Reliability by automatically maintaining multiple copies of data and automatically redeploying processing logic in the event of failures

HDFS
- Architecture

<Master/Slave architecture>

- NameNode: A single node which manages the file system namespace and regulates access to files by clients.
- DataNode: DataNodes manage storage attached to the nodes that they run on.

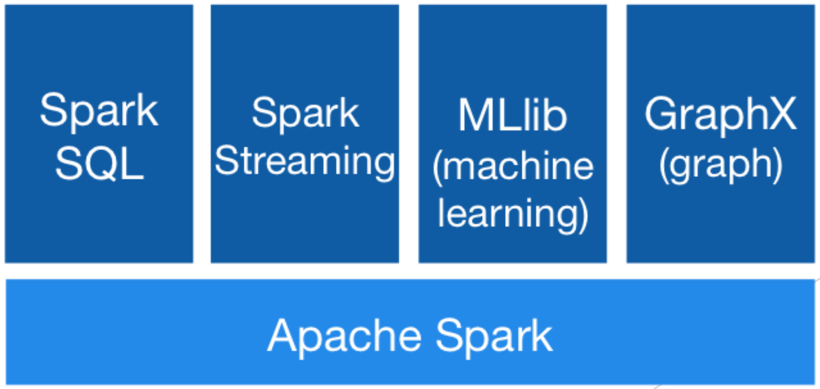


Apache Spark
- Concept

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

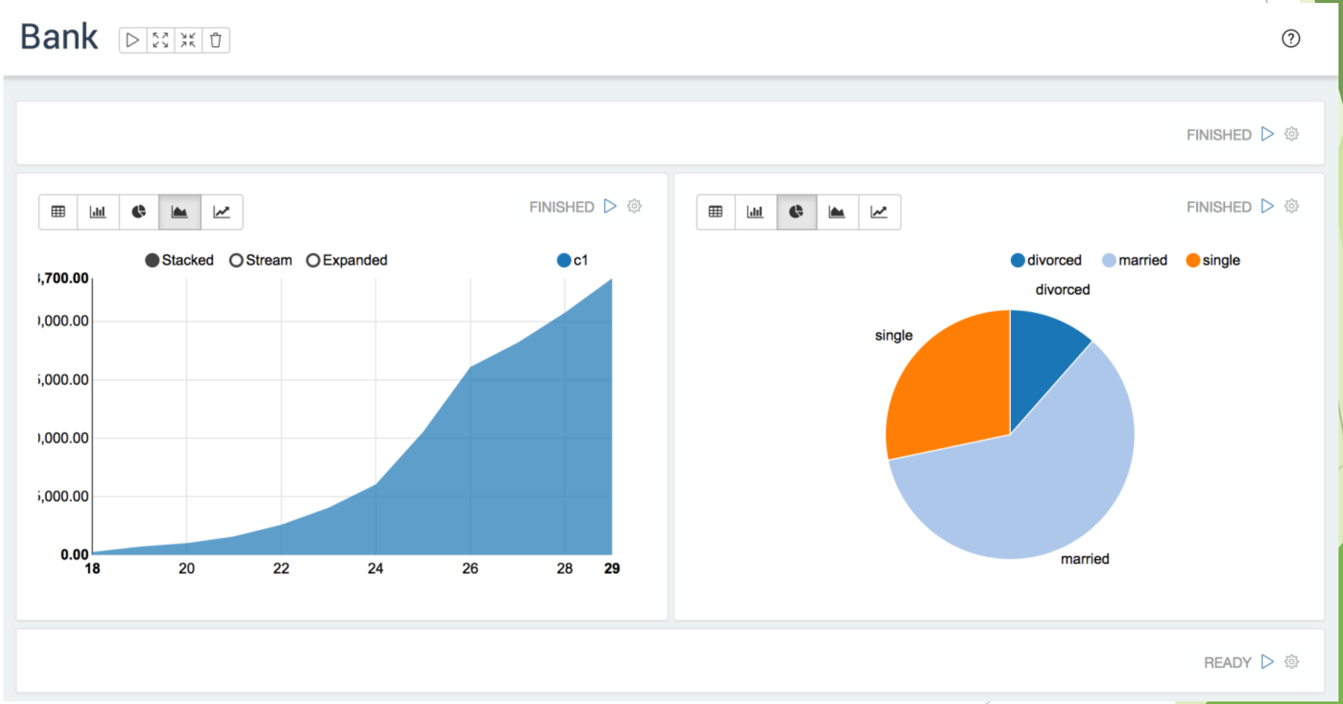
- Scala, Java, Python



Apache Zeppelin
-Concept

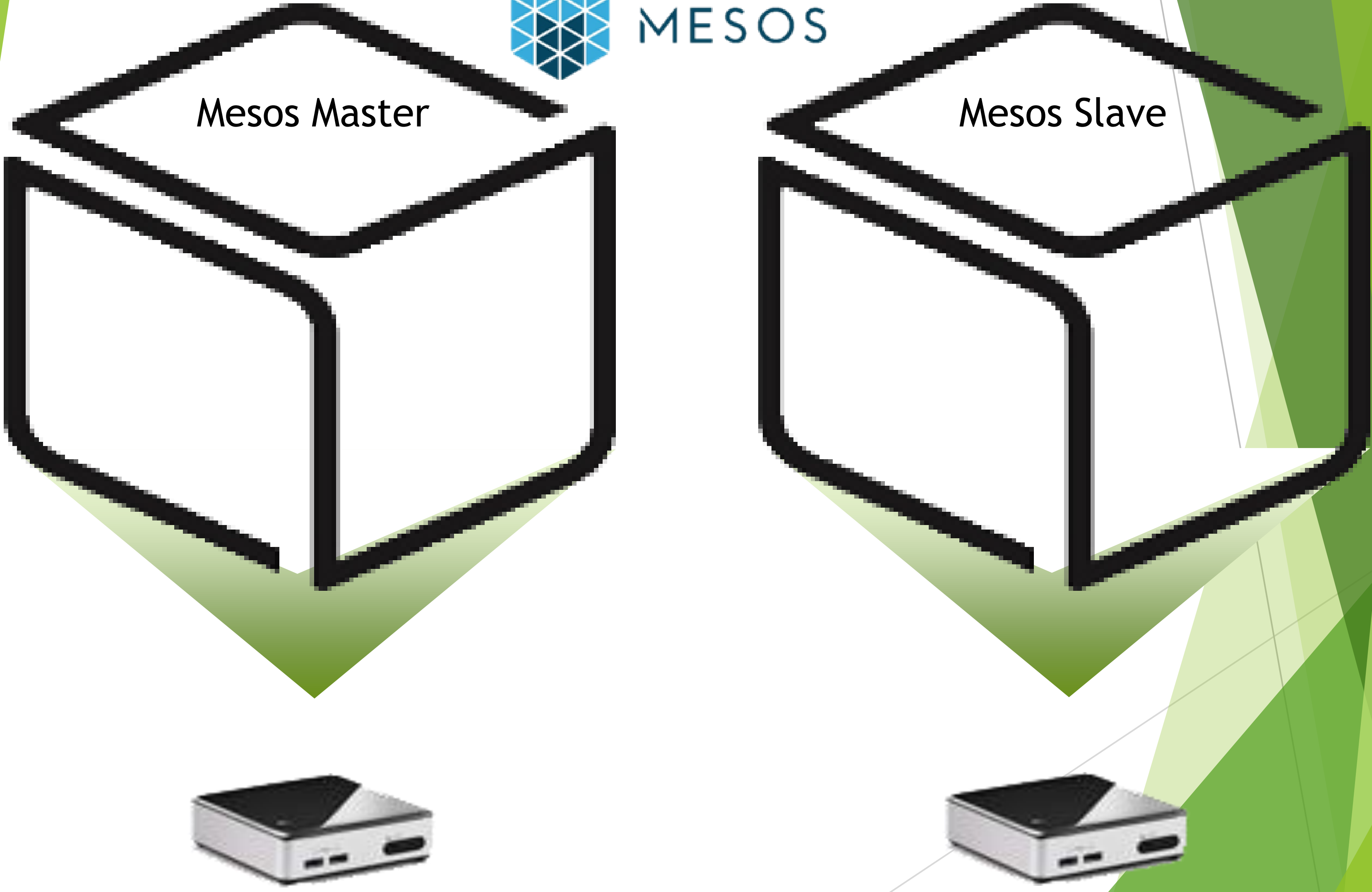
A web-based notebook that enables interactive data analytics.

Support Spark



1. Apache Mesos
- Install & Configuration

MESOS



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

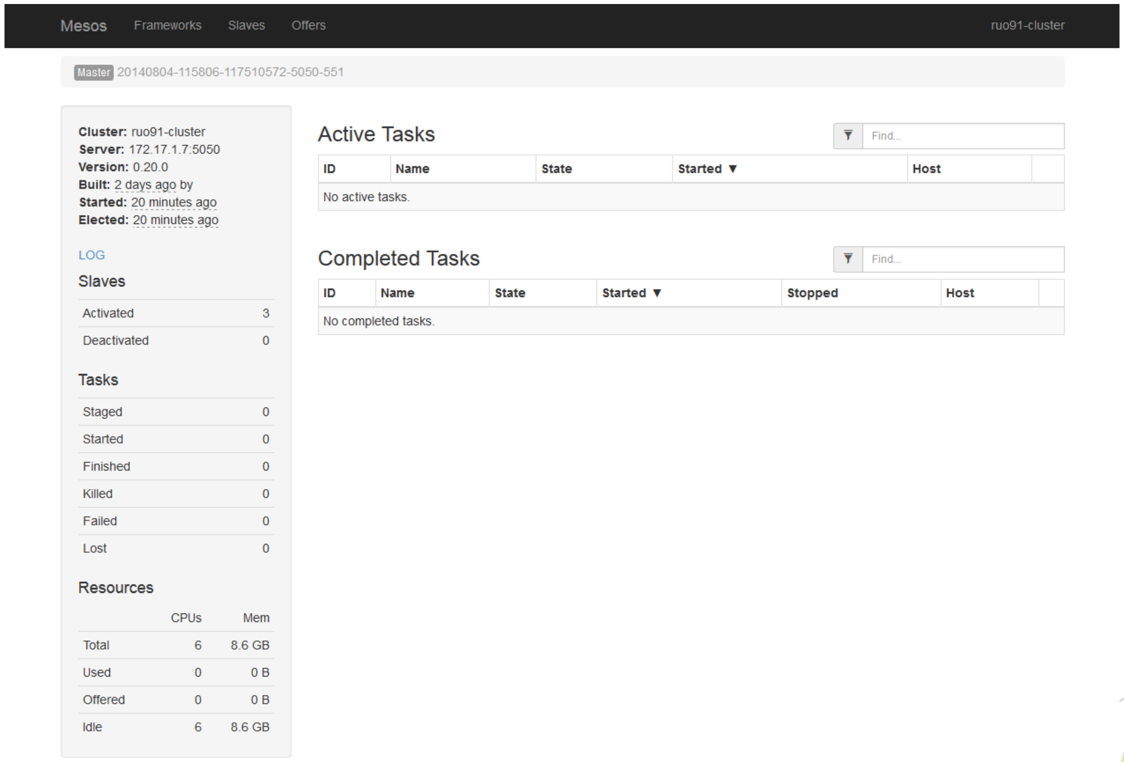
```
sudo apt-get -y install mesos marathon
sudo reboot

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
```



20140804-115806-117510572-5050-551

Cluster: ruo91-cluster
Server: 172.17.1.7:5050
Version: 0.20.0
Built: 2 days ago by
Started: 20 minutes ago
Elected: 20 minutes ago

LOG

Slaves

Activated	3
Deactivated	0

Tasks

Staged	0
Started	0
Finished	0
Killed	0
Failed	0
Lost	0

Resources

	CPUs	Mem
Total	6	8.6 GB
Used	0	0 B
Offered	0	0 B
Idle	6	8.6 GB

Active Tasks

ID	Name	State	Started ▼	Host
No active tasks.				

Completed Tasks

ID	Name	State	Started ▼	Stopped	Host
No completed tasks.					

2. HDFS
- SSH Key Registration

1. Generate key

```
• ssh-keygen -t rsa
```

2. Modify key permission

```
• cd .ssh
• chmod 700 ./
• chmod 755 ../
• chmod 600 id_rsa
• chmod 644 id_rsa.pub
• chmod 644 authorized_keys
• chmod 644 known_hosts
```

3. Copy key from Master to Slaves

```
• scp id_rsa.pub username@hostname:id_rsa.pub
```

4. Registration (for each slave)

```
• cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
```

2. HDFS
- Install Hadoop

1. Download and Unzip

```
• wget http://apache.mirror.cdnetworks.com/hadoop/common/hadoop-
  2.6.0/hadoop-2.6.0.tar.gz
• tar -xvzf hadoop-2.6.0.tar.gz
• mv hadoop-2.6.0 hadoop
```

2. Modify environment values

```
• vi .bashr
Edit: export HADOOP_HOME=$HOME/hadoop
      export PATH=$PATH:$HADOOP_HOME/bin
      export PATH=$PATH:$HADOOP_HOME/sbin
      export HADOOP_MAPRED_HOME=$HADOOP_HOME
      export HADOOP_COMMON_HOME=$HADOOP_HOME
      export HADOOP_HDFS_HOME=$HADOOP_HOME
      export YARN_HOME=$HADOOP_HOME
      export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
      export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
      export HADOOP_CLASSPATH=$JAVA_HOME/lib/tools.jar
      export CLASSPATH=$HADOOP_HOME/share/hadoop/common/hadoop-common-
      2.6.0.jar:$HADOOP_HOME/share/hadoop/mapreduce/:$HADOOP_HOME/share/hado
      op/common/lib/commons-cli-1.2.jar
```

2. HDFS
- Install Hadoop

3. Make directory to use HDFS

```
• source .bashrc (for every node)
• mkdir -p hadoop_tmp/hdfs/namenode
• mkdir -p hadoop_tmp/hdfs/datanode
```

4. Modify configuration files

```
• cd ~/hadoop/etc/Hadoop
```

- hadoop-env.sh, core-site.xml, hdfs-site.xml, mapred-site.xml, yarn-site.xml

2. HDFS
- Configuration

1. Registration host names

```
      vi /etc/hosts
Edit: Type all nodes' IP address and hostname
```

2. <Hadoop-env.sh>

```
Edit: export JAVA_HOME=~/java
      export HADOOP_PREFIX=~/hadoop
```

3. <core-site.xml>

```
Edit: <configuration>
        <property>
          <name>fs.defaultFS</name>
          <value>hdfs://hostname:9000/</value>
        </property>
      </configuration>
```

2. HDFS
- Configuration

4. <hdfs-site.xml>

Edit:
```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>3</value>
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file:~/hadoop_tmp/hdfs/namenode</value>
    </property>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>file:~/hadoop_tmp/hdfs/datanode</value>
    </property>
</configuration>
```

5. <hdfs-site.xml>

Edit:
```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
</configuration>
```

2. HDFS
- Configuration

6. <yarn-site.xml>

Edit:

```
<configuration>
<!-- Site specific YARN configuration properties -->
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
        <value>org.apache.hadoop.mapred.ShuffleHandler</value>
    </property>
    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>hostname.univ.ac.kr</value>
    </property>
    <property>
        <name>yarn.resourcemanager.resource-tracker.address</name>
        <value>hostname.univ.ac.kr:8025</value>
    </property>
    <property>
        <name>yarn.resourcemanager.scheduler.address</name>
        <value>hostname.univ.ac.kr:8030</value>
    </property>
    <property>
        <name>yarn.resourcemanager.address</name>
        <value>hostname.univ.ac.kr:8050</value>
    </property>
    <property>
        <name>yarn.resourcemanager.webapp.address</name>
        <value>hostname.univ.ac.kr:8055</value>
    </property>
</configuration>
```

2. HDFS
- Configuration

7. Deploying configuration files

```
scp -r hadoop username@hostname:
...

scp .bashrc username@hostname:
...
```

3. Apache Spark
- Install

```
wget http://mirror.apache-kr.org/spark/spark-1.6.1/spark-1.6.1-bin-hadoop2.6.tgz
hadoop fs –put /user/spark-1.6.1/spark-1.6.1-bin-hadoop2.6.tgz

tar xzf spark-1.6.1-bin-hadoop2.6.tgz

cd spark-1.6.1-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
export SPARK_EXECUTOR_URI=hdfs://<HDFS_IP_ADDR>/user/spark-1.6.1-bin-hadoop2.6.tgz
```

# 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.

4. Apache Zeppelin
- Install (on Mesos)

```
wget http://mirror.apache-kr.org/incubator/zeppelin/0.6.0-incubating/zeppelin-0.6.0-incubating-bin-all.tgz

tar xzf zeppelin-0.6.0-incubating-bin-all.tgz

cd zeppelin-0.6.0-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
export SPARK_EXECUTOR_URI=hdfs://<HDFS_IP_ADDR>/user/spark-1.6.1-bin-hadoop2.6.tgz
```

```
cd ..
bin/zeppelin-daemon.sh start
```

http://<IP-ADDR>:8080

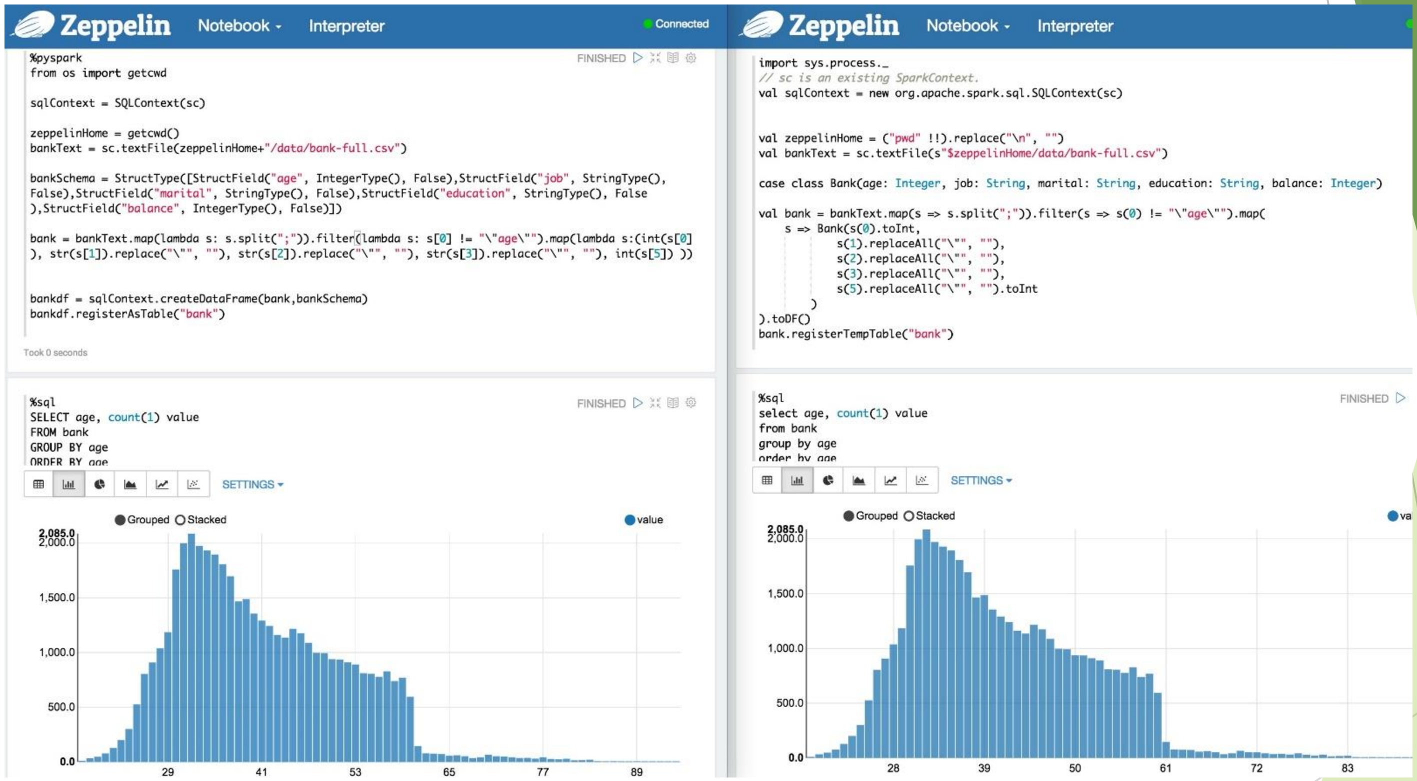
4. Apache Zeppelin
- Install (standalone mode)

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

4. Apache Zeppelin
- Run Big Data Job



```
%pyspark
from os import getcwd

sqlContext = SQLContext(sc)

zeppelinHome = getcwd()
bankText = sc.textFile(zeppelinHome+"/data/bank-full.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.registerAsTable("bank")
```

```
%sql
SELECT age, count(1) value
FROM bank
GROUP BY age
ORDER BY age
```

```
import sys.process._
// sc is an existing SparkContext.
val sqlContext = new org.apache.spark.sql.SQLContext(sc)


val zeppelinHome = ("pwd" !!).replace("\n", "")
val bankText = sc.textFile(s"$zeppelinHome/data/bank-full.csv")

case class Bank(age: Integer, job: String, marital: String, education: String, balance: Integer)

val bank = bankText.map(s => s.split(";")).filter(s => s(0) != "\"age\"").map(
    s => Bank(s(0).toInt,
            s(1).replaceAll("\"", ""),
            s(2).replaceAll("\"", ""),
            s(3).replaceAll("\"", ""),
            s(5).replaceAll("\"", "").toInt
        )
).toDF()
bank.registerTempTable("bank")
```

```
%sql
select age, count(1) value
from bank
group by age
order by age
```

Press ‘Run’ button to test the sample codes.

Thank You for
Your Attention
Any Questions?