

Hadoop Distributed File System (HDFS) + Spark Integration Guidance

Written by Jia Yu (jiayu2@asu.edu)

Please follow the steps below exactly and read the explanations carefully

Note: All the commands should be entered and run in Ubuntu Terminal (In Fedora commands may vary) if no special reminders. Assume we have three machines which are in the same local network: worker1 (IP: 192.168.0.1), worker2 (IP: 192.168.0.2) and worker3 (IP: 192.168.0.3). Among them, worker1 is the master, worker2 and worker3 are workers/slaves.

Contents

1. Prerequisites	2
2. Password-less SSH login (Required by both of Hadoop and Spark).....	2
3. Hadoop Configuration.....	3
4. Spark Configuration	5
5. First Java Application of Spark (Optional. Do this step if you want to call Spark Java API)	6
6. References and helpful links	9

1. Prerequisites

(For all machines in your cluster)

I assume you already have the basic knowledge to finish the “Prerequisites” part.

- 1) Operating system: Ubuntu 14.04.1 LTS 32-bit
Download Link:
<http://www.ubuntu.com/download/desktop>
- 2) OpenJDK 1.7.0
In Ubuntu Terminal (In Fedora commands may vary), enter the following lines:
`sudo apt-get update`
`sudo apt-get install default-jdk`
After this step, the default Java folder will be `“/usr/lib/jvm/YOURJAVAFOLDER”`
- 3) Hadoop 2.6.0
Download Link:
<http://www.apache.org/dyn/closer.cgi/hadoop/common/>
Assume Hadoop local folder is `“/home/USERNAME/Downloads/hadoop”`
- 4) Spark 1.2.0
Download Link:
<https://spark.apache.org/downloads.html>
Package Type: Pre-built for Hadoop 2.4 or later
Assume Spark local folder is `“/home/USERNAME/Downloads/spark”`
- 5) SSH (Recommended even your operating system already has SSH.)
In Ubuntu Terminal (In Fedora commands may vary), enter the following lines:
`sudo apt-get install ssh`
- 6) Maven2 (Optional. Do this step if your want to call Spark JavaAPI.)
In Ubuntu Terminal (In Fedora commands may vary), enter the following lines:
`sudo apt-get install maven2`
- 7) Eclipse Luna for Linux 32-bit (Optional. Do this step if your want to call Spark JavaAPI.)
Download Link:
<https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/luna/SR1a/eclipse-java-luna-SR1a-linux-gtk.tar.gz>

2. Password-less SSH login (Required by both of Hadoop and Spark)

- 1) Generate the public key and private key (On worker1 and worker2)
`ssh-keygen -t rsa -P "" -f ~/.ssh/id_rsa_worker1`
`ssh-keygen -t rsa -P "" -f ~/.ssh/id_rsa_worker2`
- 2) Copy the public key of worker2 to master (On worker2)
`scp ~/.ssh/id_rsa_worker2.pub USERNAME@192.168.0.1:~/.ssh/id_rsa_worker2.pub`

- 3) Shell into master (worker1) with passwords (On worker2)
`ssh USERNAME@192.168.0.1`
- 4) Authorize the public key by adding it to the list of authorized keys (On worker2)
`cat ~/.ssh/id_rsa_worker2.pub >> ~/.ssh/authorized_keys`
- 5) Log out of the current shell (On worker2)
`exit`
- 6) Test that we can log in master with no password (On worker2)
`ssh 192.168.0.1`
- 7) After the steps above, we have a one-direction password-less SSH login from worker2 to master (worker1). Please implement the password-less SSH among your cluster based on the following topology.

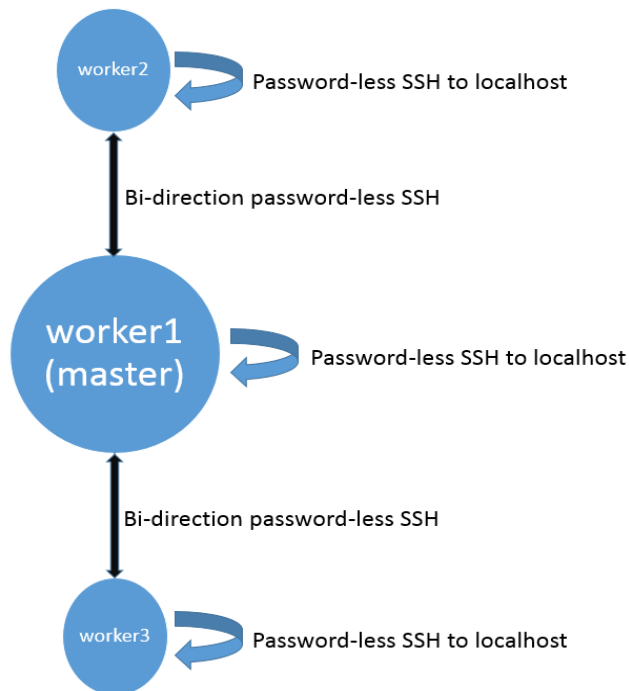


Figure 1 Password-less SSH Topology

3. Hadoop Configuration

- 1) Append the following lines in /etc/hosts. (Master only)
`192.168.0.1 master`
`192.168.0.2 worker2`
`192.168.0.3 worker3`
- 2) Append the following line in /etc/hosts. (Workers only)
`192.168.0.1 master`

- 3) Add the following line in HadoopFolder/etc/hadoop/hadoop-env.sh to point out Java folder. (All machines)
`export JAVA_HOME=/usr/lib/jvm/YOURJAVAFOLDER`
- 4) Clear content and add the following lines in HadoopFolder/etc/Hadoop/slaves. This step gives the list of all the machines who can be workers to Hadoop master. (Master only)
`master
worker2
worker3`
- 5) Append the following lines in HadoopFolder/etc/Hadoop/core-site.xml. (All machines)
`<property>
 <name>fs.default.name</name>
 <value>hdfs://master:54310</value>
</property>`
- 6) Append the following lines in HadoopFolder/etc/Hadoop/mapred-site.xml. (All machines)
`<property>
 <name>mapred.job.tracker</name>
 <value>master:54311</value>
</property>`
- 7) Append the following lines in HadoopFolder/etc/Hadoop/hdfs-site.xml. The value should be consistent with the number of machines in the step 3(4). (All machines)
`<property>
 <name>dfs.replication</name>
 <value>3</value>
</property>`
- 8) Delete “hadoop-USERNAME” folder in /tmp folder in all machines (if it exists) before you format HDFS in Step 3(9). (All machines)
- 9) Run the following line in HadoopFolder/bin/ folder to format the HDFS system if this is the first time to run this Hadoop or after you change any machine attributes such as IP address. (Master only).
`./hadoop namenode -format`
- 10) Run the following line in HadoopFolder/sbin/ folder to start your cluster. It will take a while. (Master only)
`./start-all.sh`

- 11) Check the status of Hadoop cluster by entering “localhost:50070” in your linux browser. If you reach a page like the following, that means you have started Hadoop successfully. (Master only)

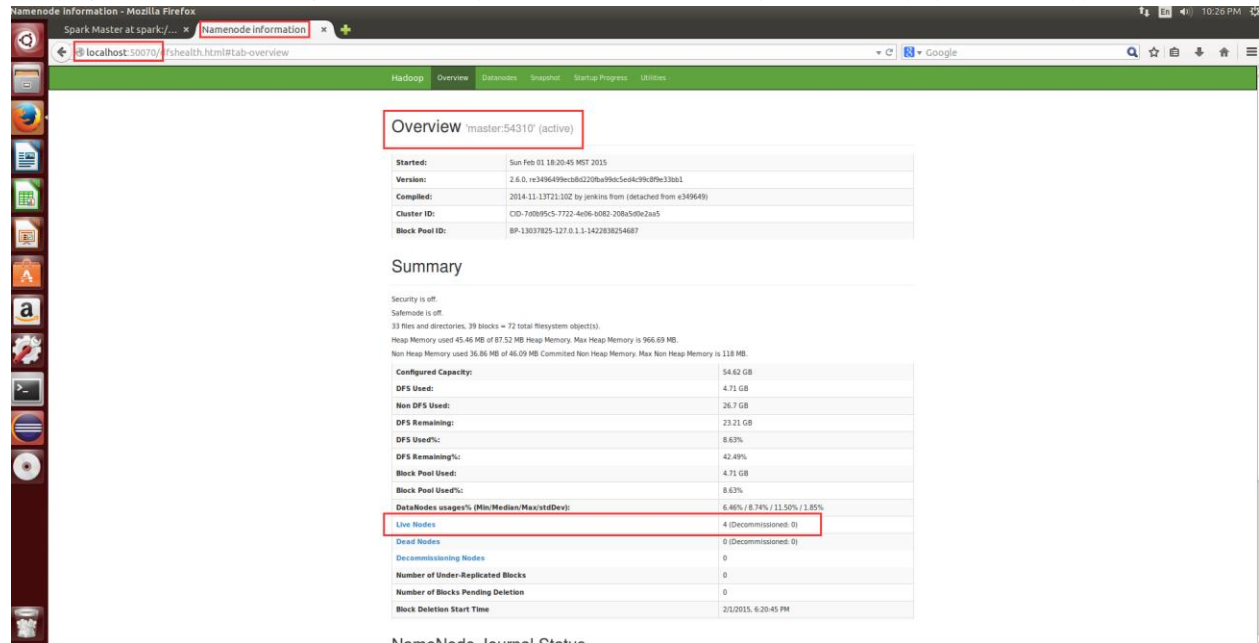


Figure 2 Hadoop Namenode Web UI

4. Spark Configuration

- 1) Add the following line in SparkFolder/conf/spark-env.sh. In our case, master (worker 1) has IP address 192.168.0.1. (Master only)
`SPARK_MASTER_IP=192.168.0.1`
- 2) Run the following line in SparkFolder/sbin/ folder to start Spark master. (Master only)
`./start-master.sh`
- 3) Check the status of Spark master by entering “localhost:8080” in your linux browser. If you reach a page like the following, that means your master has been started. (Master only)

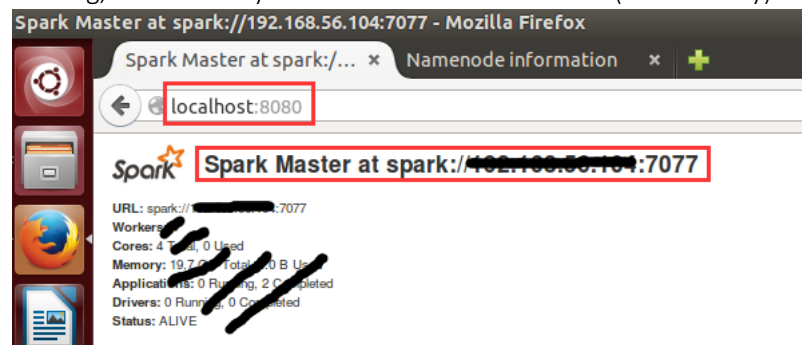
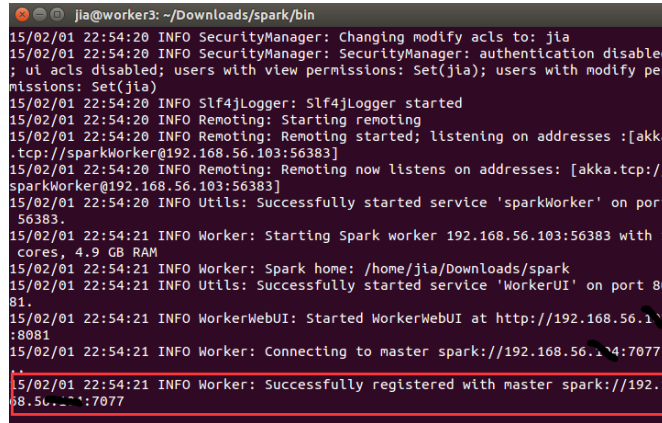


Figure 3 Spark Master Web UI

- 4) Run the following line in SparkFolder/bin/ folder to register and run the worker in Spark master. 192.168.0.101:7077 is on the master. (worker1). (Workers only)

```
./spark-class org.apache.spark.deploy.worker.Worker spark://192.168.0.101:7077
```

- 5) If you see prompts like the following, that means this worker has been registered to the master. (Workers only)



```
jia@worker3: ~/Downloads/spark/bin
15/02/01 22:54:20 INFO SecurityManager: Changing modify acls to: jia
15/02/01 22:54:20 INFO SecurityManager: SecurityManager: authentication disabled
; ul acls disabled; users with view permissions: Set(jia); users with modify per
missions: Set(jia)
15/02/01 22:54:20 INFO Slf4jLogger: Slf4jLogger started
15/02/01 22:54:20 INFO Remoting: Starting remoting
15/02/01 22:54:20 INFO Remoting: Remoting started; listening on addresses :[akka
.tcp://sparkWorker@192.168.56.103:56383]
15/02/01 22:54:20 INFO Remoting: Remoting now listens on addresses: [akka.tcp://
sparkWorker@192.168.56.103:56383]
15/02/01 22:54:20 INFO Utils: Successfully started service 'sparkWorker' on port
56383.
15/02/01 22:54:21 INFO Worker: Starting Spark worker 192.168.56.103:56383 with 1
cores, 4.9 GB RAM
15/02/01 22:54:21 INFO Worker: Spark home: /home/jia/Downloads/spark
15/02/01 22:54:21 INFO Utils: Successfully started service 'WorkerUI' on port 80
81.
15/02/01 22:54:21 INFO WorkerWebUI: Started WorkerWebUI at http://192.168.56.103
:8081
15/02/01 22:54:21 INFO Worker: Connecting to master spark://192.168.56.101:7077.
15/02/01 22:54:21 INFO Worker: Successfully registered with master spark://192.1
68.56.101:7077
```

Figure 4 Registered a worker to the master successfully

- 6) Do not close the terminal in workers after step 4(5).

This terminal will listen the port after connect and prompt useful information. Otherwise the connection between this worker and the master will be shut down. Then the status of this worker will be “DEAD” in the Web UI of the master.

- 7) Check the status of Spark master again. If you see a page like the following, that means you are all set. (Master only)

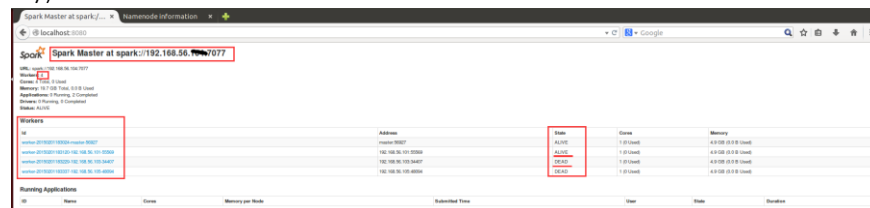


Figure 5 Spark Master Web UI after configuration

5. First Java Application of Spark (Optional. Do this step if you want to call Spark Java API.)

I assume you already have the basic knowledge of Maven project in Eclipse, such as how to create and run program in Maven project.

1) Create Maven project in Eclipse. See Figure 6 – 11.

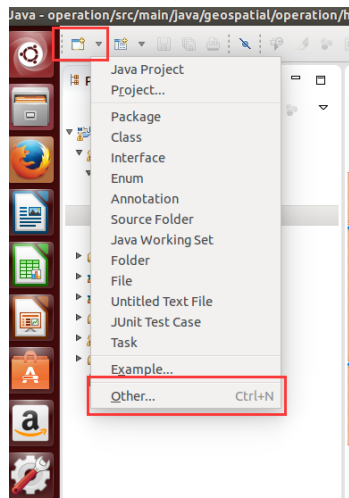


Figure 6 Create Maven project Step 1

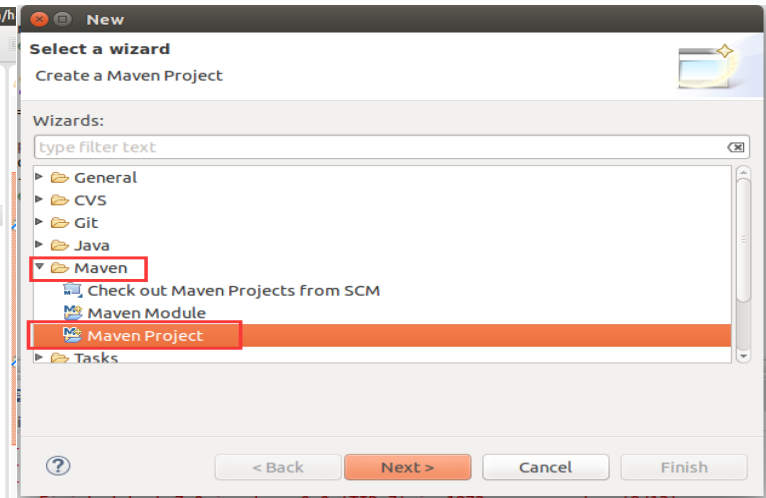


Figure 7 Create Maven project Step 2

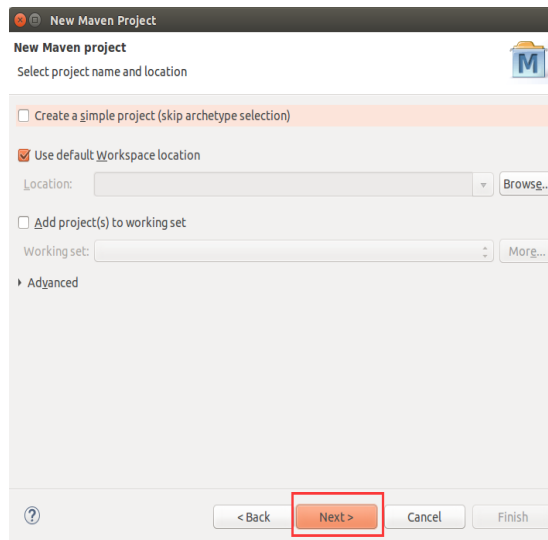


Figure 8 Create Maven project Step 3

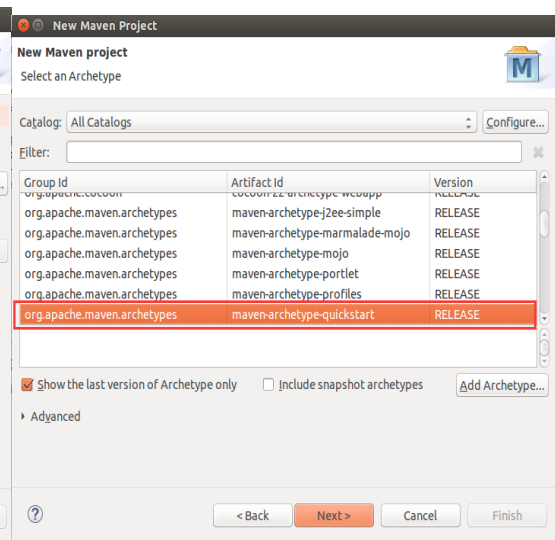


Figure 9 Create Maven project Step 4

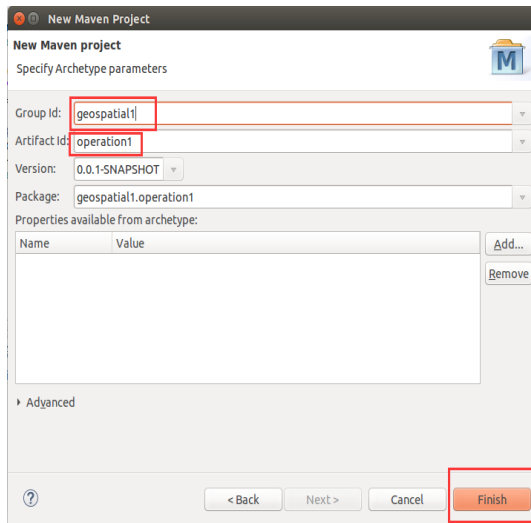


Figure 10 Create Maven project Step 5

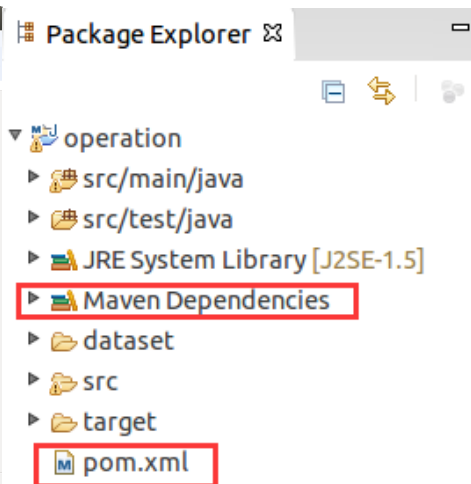


Figure 11 Maven project structure

2) Add Spark Dependency. See Figure 12 and 13.

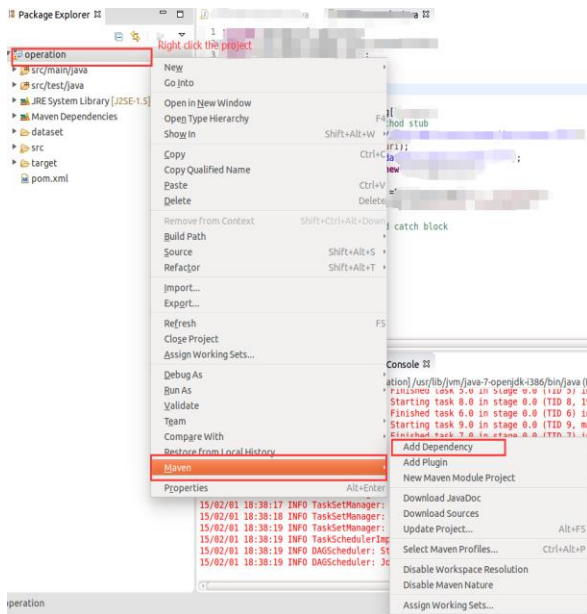


Figure 12 Add Spark Dependency for Maven

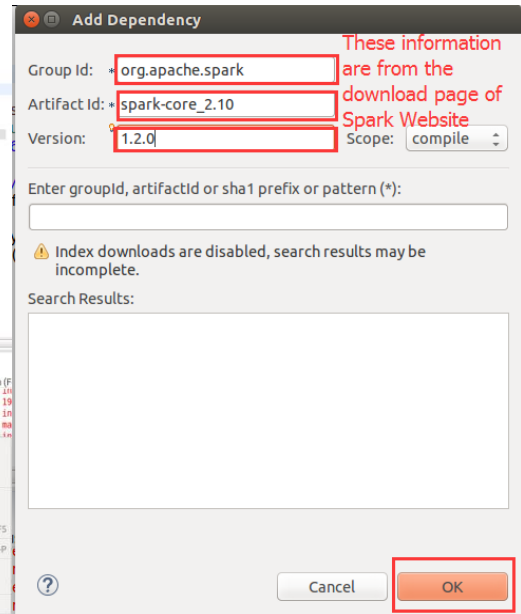


Figure 13 Enter Dependency Package Information

- 3) Let Maven download the dependency automatically by clicking this button. See Figure 14.

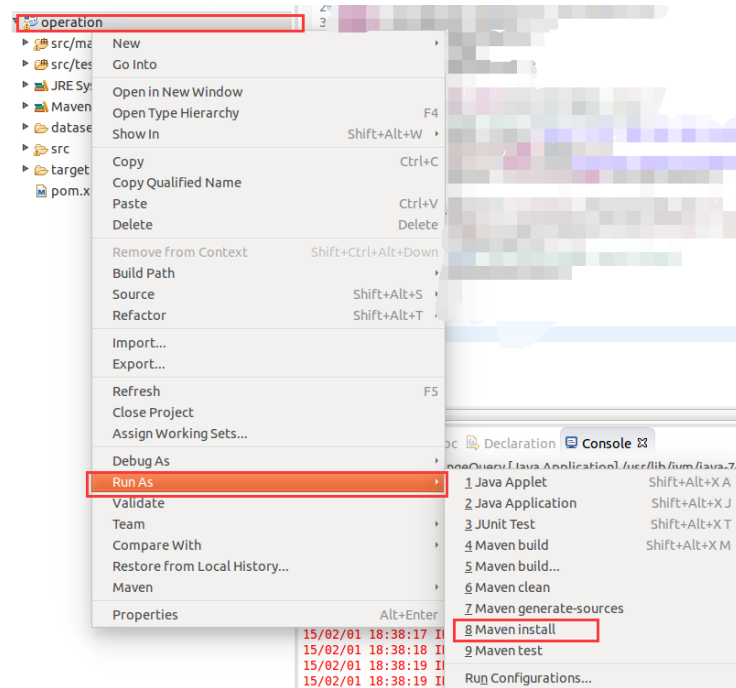


Figure 14 Download Spark dependency package by Maven

- 4) Write Java Code with Spark Java API. Note that you must read input from HadoopDFS (HDFS) and save output at HadoopDFS (HDFS). Regarding how to use Spark API, please check out the following two links.

<https://spark.apache.org/docs/latest/quick-start.html>

<https://spark.apache.org/examples.html>

Hint: Read txt files into HDFS by one program firstly. Then read input from HDFS into Spark and save output from Spark into HDFS by another program.

6. References and helpful links:

- 1) SSH:

<http://hortonworks.com/kb/generating-ssh-keys-for-passwordless-login/>

- 2) Hadoop:

<http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/>

<http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/>

<http://tutorials.techmytalk.com/2014/08/16/hadoop-hdfs-java-api/>

<https://hadoop.apache.org/docs/current/api/org/apache/hadoop/fs/FileSystem.html>

- 3) Spark:

<https://spark.apache.org/docs/latest/api/java/index.html>

<https://spark.apache.org/docs/latest/spark-standalone.html>