Hadoop setup / Spark with HDFS

Lab 2
October 19th, 2017

Jonghyun Bae(jonghbae@snu.ac.kr)
Computer Science and Engineering
Seoul National University

Index

- Hadoop distributed file system (HDFS)
- Installation
- Configuration
- Starting / Stopping cluster
- Hadoop Web Interface
- Running Spark with HDFS
- Exercise

Before we start…

- Please connect your VM using SSH
- https://docs.google.com/spreadsheets/d/1X9Uavr2PACqgfLC3rOcQ7Gqo4-NQNKoBhvcaL_86g9E/edit?usp=sharing

```
# Please your public IP address in xxx.xxx.xxx.xxx
student@computer:~$ ssh -X -i bde3.pem ubuntu@xxx.xxx.xxx.xxx
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 3.13.0-125-generic x86_64)
[...snipp...]
ubuntu@ip-x-x-x:~$
```

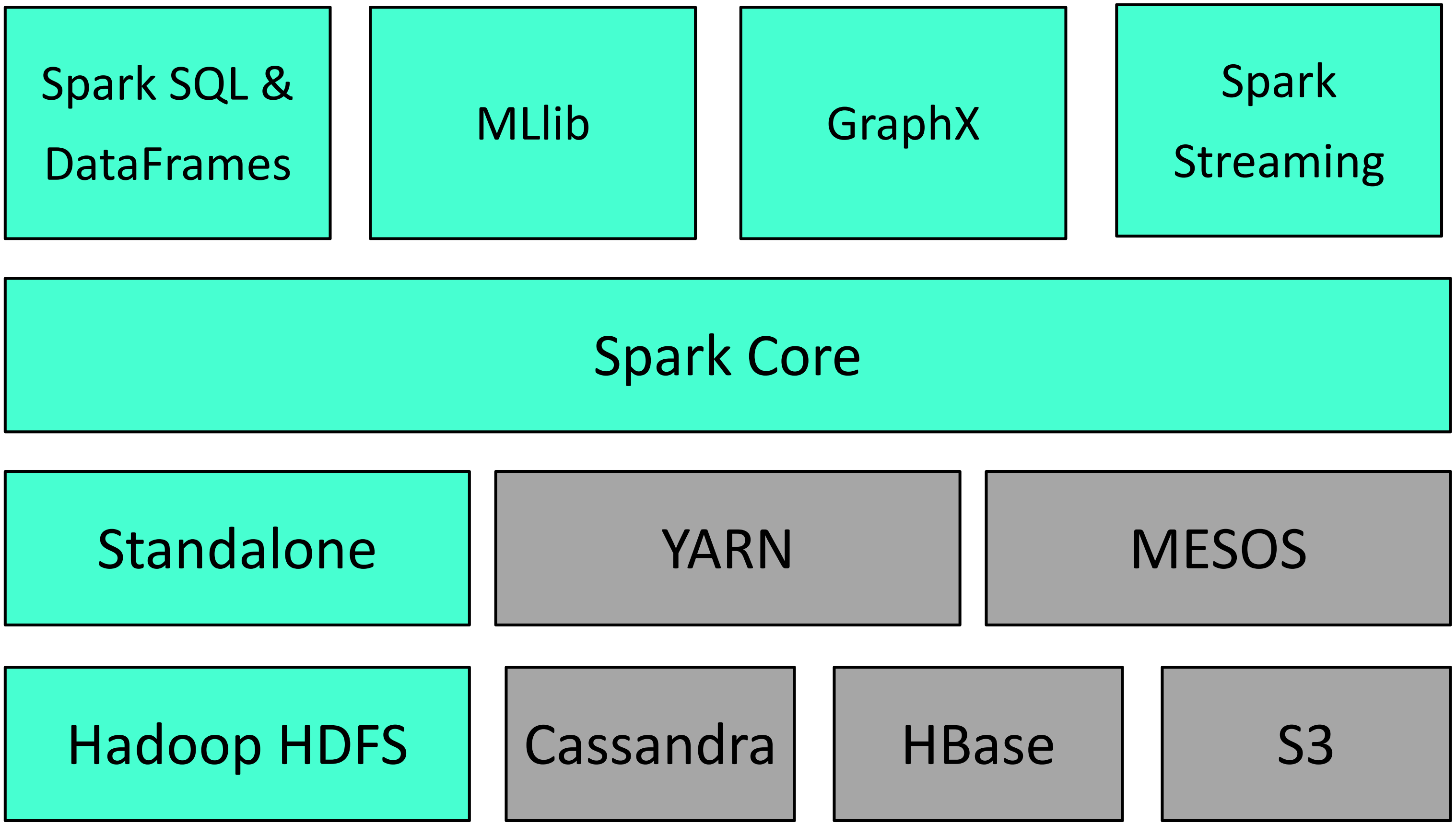
Hadoop distributed file system (1)

- **What is Hadoop?**

  - Developed by Doug Cutting and Mike Cafarella in 2006
  - Open-source software for reliable, scalable, distributed computing

Hadoop distributed file system (2)

- **Why do we need Hadoop?**
  - Spark needs file system for saving / loading data



* Image from https://www.safaribooksonline.com/library/view/data-analytics-with/9781491913734/ch04.html

Installation (1)

- Download Hadoop from Apache Download Mirrors

```
ubuntu@ip-x-x-x:~$ wget http://mirror.navercorp.com/apache/hadoop/common/hadoop-2.7.4/hadoop-2.7.4.tar.gz
```

- Unzip hadoop-2.7.4.tar.gz

```
ubuntu@ip-x-x-x:~$ tar xzf hadoop-2.7.4.tar.gz
ubuntu@ip-x-x-x:~$ cd hadoop-2.7.4
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ cd
ubuntu@ip-x-x-x:~$
```

Installation (2)

- **Update $HOME/.bashrc**

```
ubuntu@ip-x-x-x:~$ vi ~/.bashrc
```

$HOME/.bashrc

```
# ~/.bashrc: executed by bash(1) for non-login shells.
                                   (...)
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export SPARK_HOME=/home/ubuntu/spark-2.1.0
# Add new environment variable for Hadoop!
export HADOOP_HOME=/home/ubuntu/hadoop-2.7.4
```

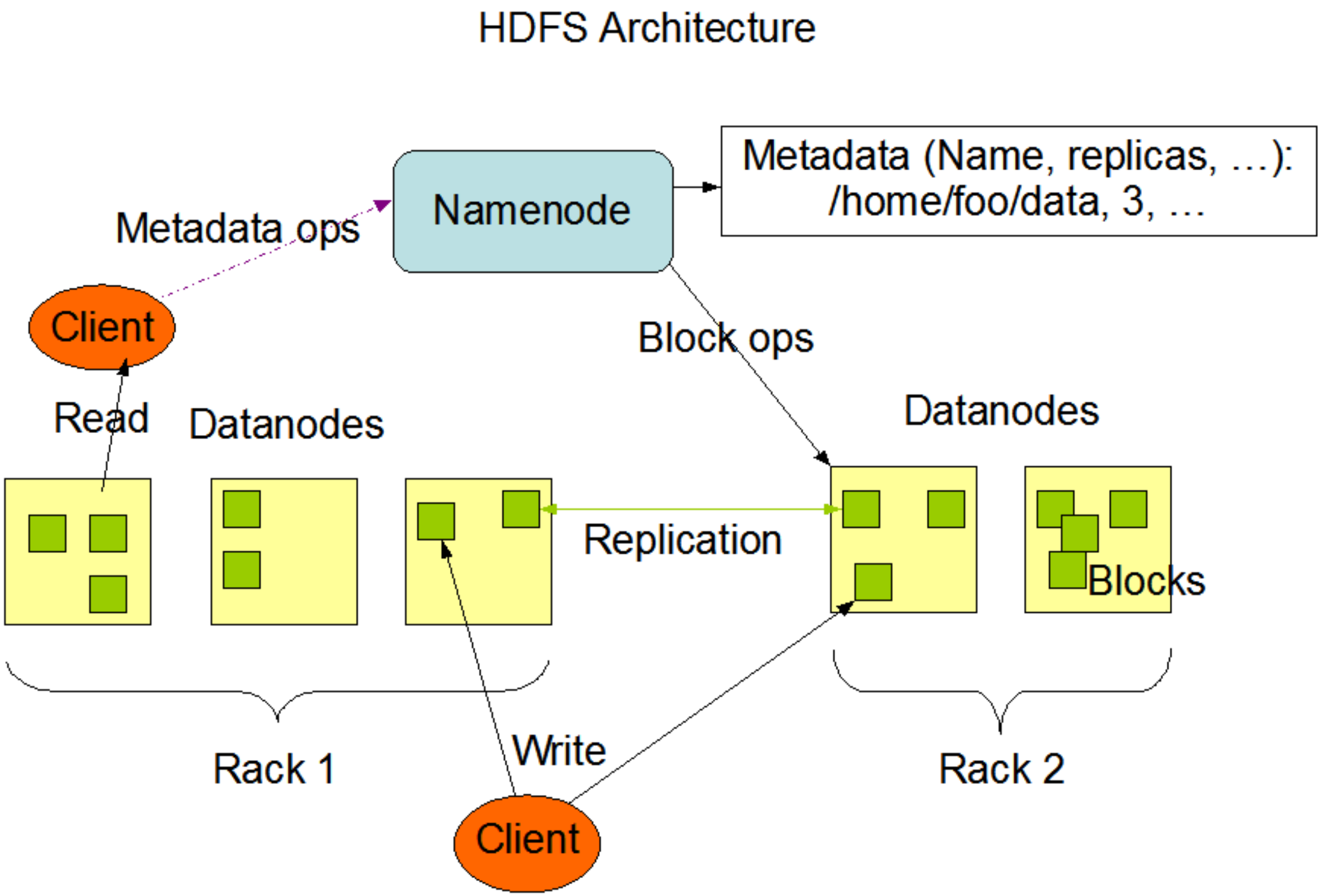
Installation (3)

- **Apply changed setup in $HOME/.bashrc**

```
ubuntu@ip-x-x-x:~$ source ~/.bashrc
ubuntu@ip-x-x-x:~$ cd $HADOOP_HOME
ubuntu@ip-x-x-x:~/hadoop-2.7.4$
```

Configuration (1)

- **Architecture of namenode and datanode**



* Image from http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

Configuration (2)

- **hadoop-env.sh**
  - Set your default configuration for Hadoop

```
ubuntu@ip-x-x-x:~$ vi $HADOOP_HOME/etc/hadoop/hadoop-env.sh
```

$HADOOP_HOME/etc/hadoop/hadoop-env.sh

```
# The java implementation to use.
export JAVA_HOME=/usr/lib/jvm/java-8-oracle

# Set your Hadoop configuration directory
export HADOOP_CONF_DIR=/home/ubuntu/hadoop-2.7.4/etc/hadoop
```

Configuration (3)

- **core-site.xml**
  - You can set cluster information for master and slave model
  - Write properties between `<configuration>` and `</configuration>`

```
ubuntu@ip-x-x-x:~$ vi $HADOOP_HOME/etc/hadoop/core-site.xml
```

$HADOOP_HOME/etc/hadoop/core-site.xml

```
<property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
</property>
```

Configuration (4)

- **hdfs-site.xml**
  - You can set internal HDFS information for namenode and datanode
  - Write properties between `<configuration>` and `</configuration>`

$HADOOP_HOME/etc/hadoop/hdfs-site.xml

```
<property>
    <name>dfs.replication</name>
    <value>1</value>
</property>
<property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/home/ubuntu/hadoop-2.7.4/hdfs/namenode</value>
</property>
(continued to next page)
```

Configuration (5)

- **hdfs-site.xml**
  - You can set internal HDFS information for namenode and datanode
  - Write properties between <configuration> and </configuration>

$HADOOP_HOME/etc/hadoop/hdfs-site.xml (continued)

```
<property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/home/ubuntu/hadoop-2.7.4/hdfs/datanode</value>
</property>
```

Configuration (6)

- **Formatting the HDFS file system**

```
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ bin/hdfs namenode -format
```

- **The output will look like this:**

```
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ bin/hdfs namenode -format
INFO namenode.NameNode: STARTUP_MSG:
                                   (...)
INFO Storage directory (...)/hdfs/namenode has been successfully formatted.
                                   (...)
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at ‘username’/127.0.0.1
************************************************************/
```

Starting / Stopping cluster (1)

- **Run the command:**

```
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ sbin/start-dfs.sh
```

Starting / Stopping cluster (2)

- **The output will look like this:**

```
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ sbin/start-dfs.sh
Starting namenodes on [localhost]
localhost: starting namenode, logging to ‘Namenode logging directory’
localhost: starting datanode, logging to ‘Datanode logging directory’
Starting secondary namenodes [0.0.0.0]
The authenticity of host '0.0.0.0 (0.0.0.0)' can't be established.
ECDSA key fingerprint is 16:20:01:83:ef:85:41:fb:ad:90:19:20:59:e1:7e:65.
Are you sure you want to continue connecting (yes/no)? yes
0.0.0.0: starting secondarynamenode, logging to ‘logging directory’
ubuntu@ip-x-x-x:~/hadoop-2.7.4$
```

Typing!!!

Starting / Stopping cluster (3)

- **Checking whether the Hadoop processes are running**

```
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ jps
1001 NameNode
1002 DataNode
1003 SecondaryNameNode
1004 Jps
ubuntu@ip-x-x-x:~/hadoop-2.7.4$
```

Starting / Stopping cluster (4)

- **Run the command:**

```
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ sbin/stop-dfs.sh
```

- **The output will look like this:**

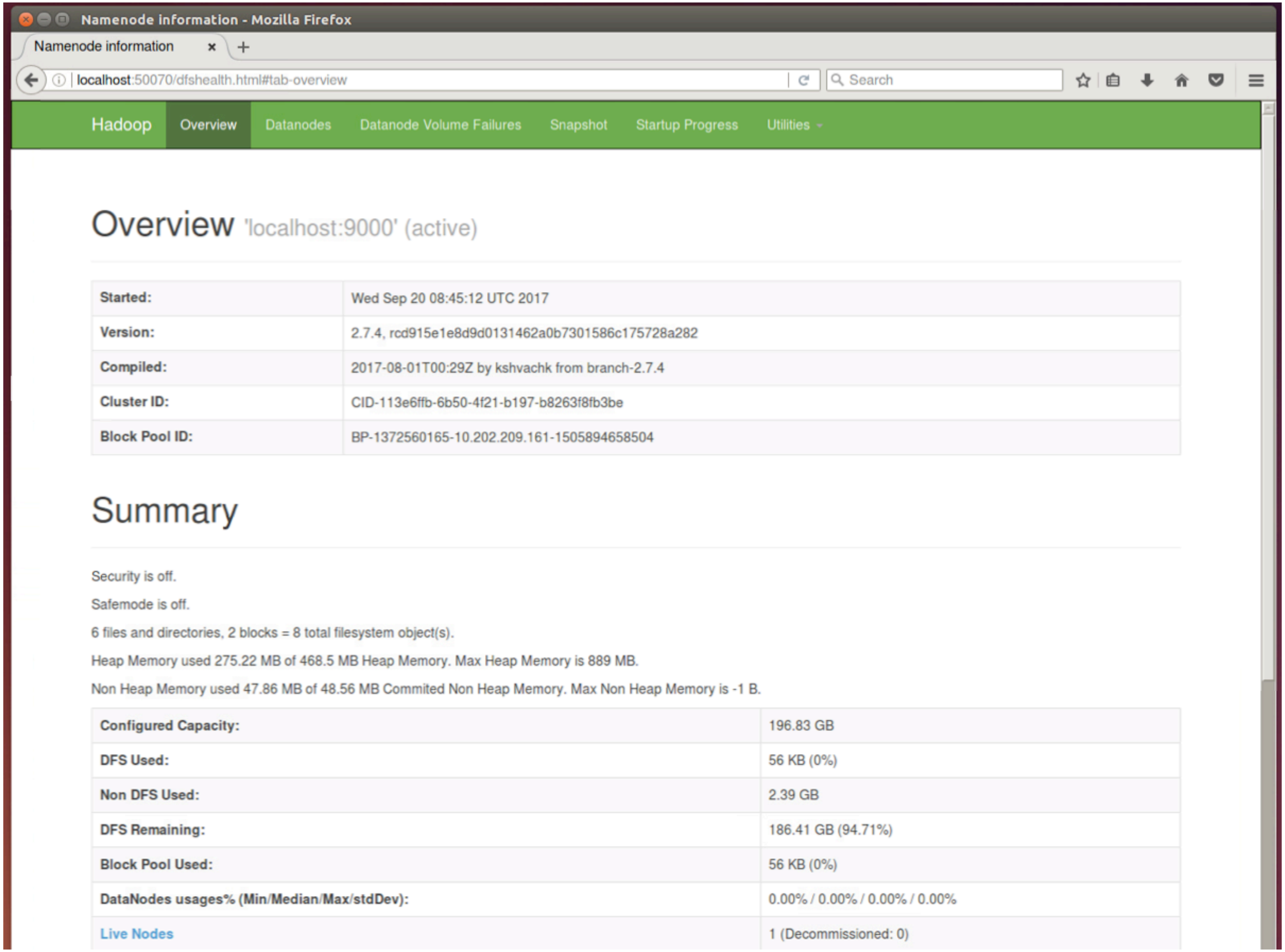
```
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ sbin/stop-dfs.sh
Stopping namenodes on [localhost]
localhost: stopping namenode
localhost: stopping datanode
Stopping secondary namenodes on [0.0.0.0]
0.0.0.0: stopping secondarynamenode
ubuntu@ip-x-x-x:~/hadoop-2.7.4$
```

Hadoop Web Interface (1)

- **Web UI of the NameNode daemon**
  - http://localhost:50070

```
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ firefox
```

Hadoop Web Interface (2)



Overview 'localhost:9000' (active)

Started:	Wed Sep 20 08:45:12 UTC 2017
Version:	2.7.4, rcd915e1e8d9d0131462a0b7301586c175728a282
Compiled:	2017-08-01T00:29Z by kshvachk from branch-2.7.4
Cluster ID:	CID-113e6ffb-6b50-4f21-b197-b8263f8fb3be
Block Pool ID:	BP-1372560165-10.202.209.161-1505894658504

Summary

Security is off.

Safemode is off.

6 files and directories, 2 blocks = 8 total filesystem object(s).

Heap Memory used 275.22 MB of 468.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 47.86 MB of 48.56 MB Commited Non Heap Memory. Max Non Heap Memory is -1 B.

Configured Capacity:	196.83 GB
DFS Used:	56 KB (0%)
Non DFS Used:	2.39 GB
DFS Remaining:	186.41 GB (94.71%)
Block Pool Used:	56 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	1 (Decommissioned: 0)

Running Spark with HDFS (1)

- **WordCount example**
  - Save three ebooks from Project Gutenberg

```
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ wget http://www.gutenberg.org/cache/epub/20417/pg20417.txt
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ wget http://www.gutenberg.org/files/5000/5000-8.txt
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ wget http://www.gutenberg.org/files/4300/4300-0.txt
```

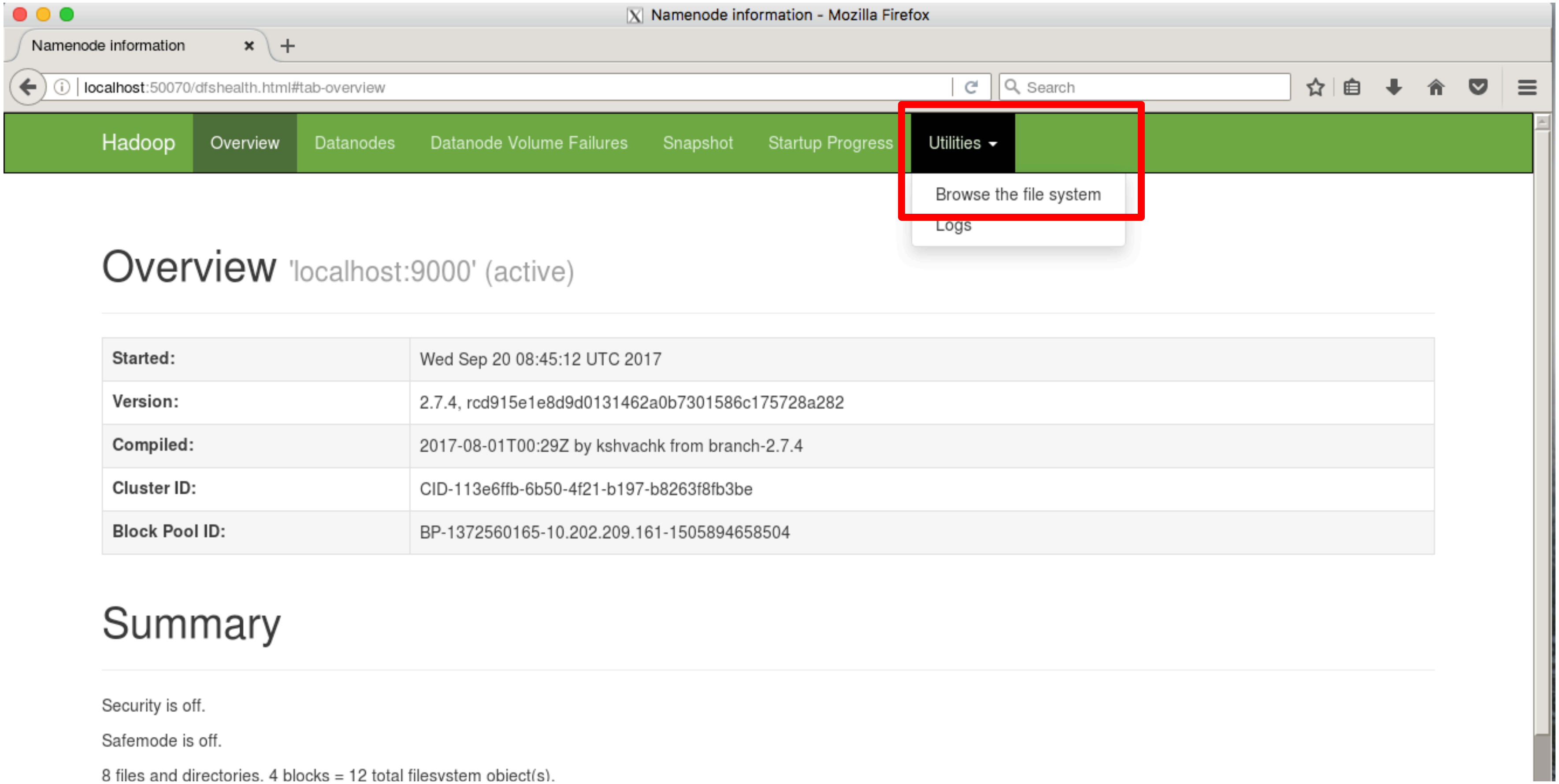
Running Spark with HDFS (2)

- **WordCount example**
  - Put .txt files into HDFS

```
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ bin/hdfs dfs –mkdir /input
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ bin/hdfs dfs –put pg20417.txt /input/sample1.txt
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ bin/hdfs dfs –put 5000-8.txt /input/sample2.txt
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ bin/hdfs dfs –put 4300-0.txt /input/sample3.txt
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ bin/hdfs dfs –ls /input
drwxr-xr-x - ubuntu supergroup   674570  2017-10-19 14:00 /input/sample1.txt
drwxr-xr-x - ubuntu supergroup  1428841  2017-10-19 14:00 /input/sample2.txt
drwxr-xr-x - ubuntu supergroup  1580890  2017-10-19 14:00 /input/sample3.txt
ubuntu@ip-x-x-x:~/hadoop-2.7.4$
```

Running Spark with HDFS (3)

- **WordCount example**
  - Check the text files in HDFS



Overview 'localhost:9000' (active)

Started:	Wed Sep 20 08:45:12 UTC 2017
Version:	2.7.4, rcd915e1e8d9d0131462a0b7301586c175728a282
Compiled:	2017-08-01T00:29Z by kshvachk from branch-2.7.4
Cluster ID:	CID-113e6ffb-6b50-4f21-b197-b8263f8fb3be
Block Pool ID:	BP-1372560165-10.202.209.161-1505894658504

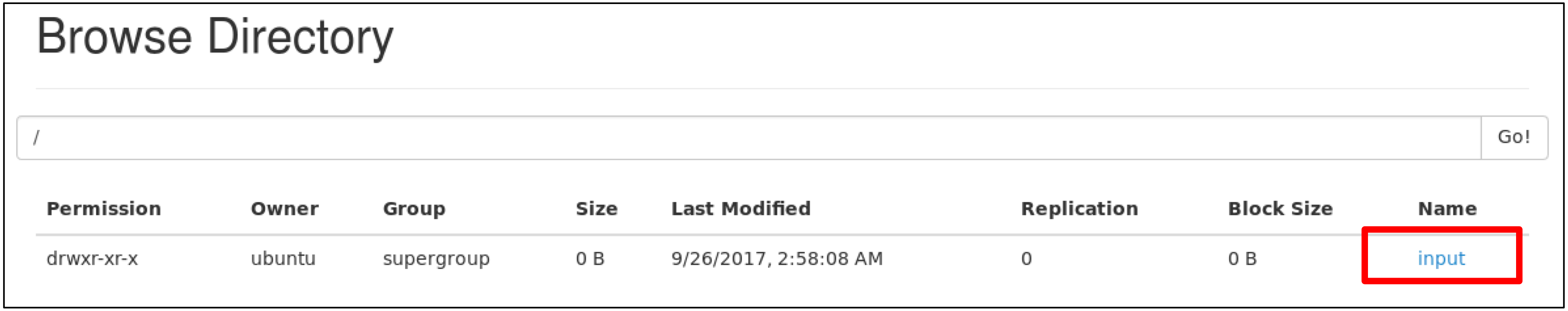
Summary

Security is off.

Safemode is off.

8 files and directories. 4 blocks = 12 total filesystem object(s).

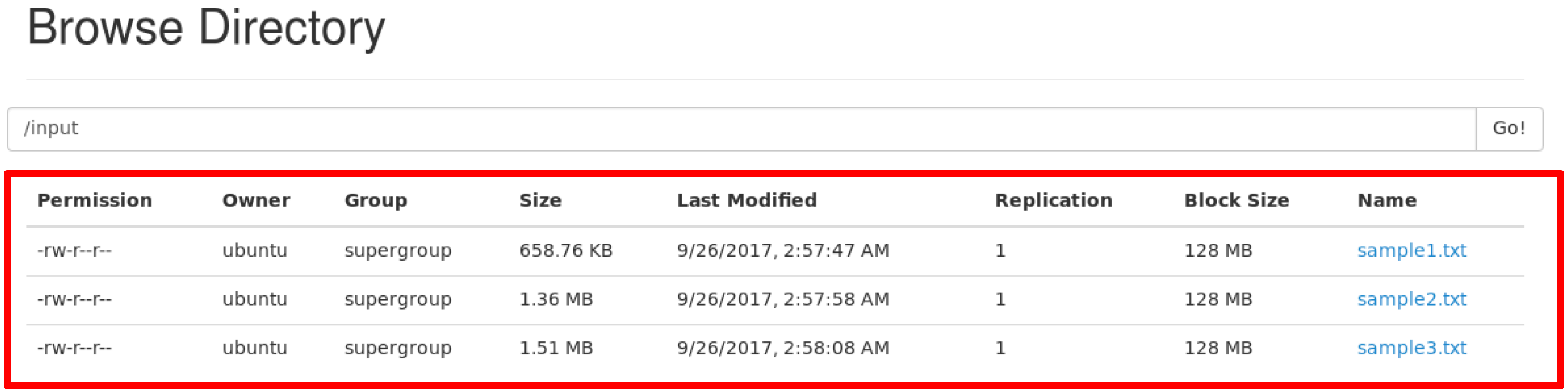
Running Spark with HDFS (4)



Browse Directory

/ Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	ubuntu	supergroup	0 B	9/26/2017, 2:58:08 AM	0	0 B	input



Browse Directory

/input Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	ubuntu	supergroup	658.76 KB	9/26/2017, 2:57:47 AM	1	128 MB	sample1.txt
-rw-r--r--	ubuntu	supergroup	1.36 MB	9/26/2017, 2:57:58 AM	1	128 MB	sample2.txt
-rw-r--r--	ubuntu	supergroup	1.51 MB	9/26/2017, 2:58:08 AM	1	128 MB	sample3.txt

Running Spark with HDFS (5)

- **Start and open your Spark shell**

```
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ cd $SPARK_HOME
ubuntu@ip-x-x-x:~/spark-2.1.0$ sbin/start-all.sh
ubuntu@ip-x-x-x:~/spark-2.1.0$ bin/pyspark
Python 2.7.6 (default, Oct 26 2016, 20:30:19)
[GCC 4.8.4] on linux2
Type "help", "copyright", "credits" or "license" for more information.
[...snipp...]
Using Python version 2.7.6 (default, Oct 26 2016 20:30:19)
SparkSession available as 'spark'.
>>>
```

Running Spark with HDFS (6)

- **Example 1: Count the total words in text files**

```
>>> text_file = sc.textFile(“hdfs://localhost:9000/input”)
>>> counts = text_file.flatMap(lambda line: line.split(“ “)) \
...              .count()
[Stage 0:>>>>>>>>>>>>>>                                    (1 + 1) / 3]
>>>
```

Running Spark with HDFS (7)

- **Example 1: Check the output**

```
>>> print counts
664559
>>>
```

Running Spark with HDFS (8)

- **Example 2: Count the occurrence of each word in text files**

```
>>> text_file = sc.textFile(“hdfs://localhost:9000/input”)
>>> wordcounts = text_file.flatMap(lambda line: line.split(“ “)) \
...             .map(lambda word: (word, 1)) \
...             .reduceByKey(lambda a, b: a + b)
>>> wordcounts.saveAsTextFile(“hdfs://localhost:9000/output”)
[Stage 0:>>>>>>>>>>>>>>                                   (1 + 1) / 3]
[Stage 1:>>>>>>>>>>>>>>                                   (1 + 1) / 3]
>>> exit()
```

Running Spark with HDFS (9)

- **Example 2: Check the output**

```
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ bin/hdfs dfs -cat /output /part-00000
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ bin/hdfs dfs -getmerge /output result.txt
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ vi result.txt
```

Running Spark with HDFS (10)

- **Example 3: Count the occurrence of each word in text files and sort words by frequency**

```
>>> text_file = sc.textFile(“hdfs://localhost:9000/input”)
>>> sortcounts = text_file.flatMap(lambda line: line.split(“ ”)) \
...             .map(lambda word: (word, 1)) \
...             .reduceByKey(lambda a, b: a + b) \
...             .sortBy(lambda x: -x[1])
>>> sortcounts.saveAsTextFile(“hdfs://localhost:9000/output2”)
[Stage 0:>>>>>>>>>>>>>>                                   (1 + 1) / 3]
[Stage 1:>>>>>>>>>>>>>>                                   (1 + 1) / 3]
>>> exit()
```

Running Spark with HDFS (11)

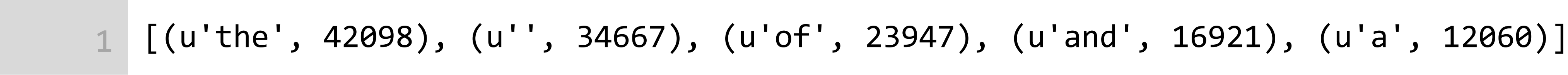
- **Example 3: Check the output**

```
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ bin/hdfs dfs -cat /output2 /part-00000
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ bin/hdfs dfs -getmerge /output2 result2.txt
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ vi result2.txt
```

Exercise 1

- **Find the top 5 most used words only**
- **Hints**
  - `takeOrdered(`*`N`*`, `*`function`*`)`
    - Description: get the *N* elements from an RDD ordered in ascending order or specified by the optional *function*

- **Please show me your result like this!**



```
[(u'the', 42098), (u'', 34667), (u'of', 23947), (u'and', 16921), (u'a', 12060)]
```

Exercise 2

- **Make a bigram count program using pyspark**
  - Example

apple banana banana apple banana banana

↓

((apple, banana), 1) , ((banana, banana), 1) , ((banana, apple), 1) ,
((apple, banana), 1) , ((banana, banana), 1)

↓

((apple, banana), 2) , ((banana, banana), 2) , ((banana, apple), 1)

Exercise 2

- **Fill in the blank (???)**

```
>>> text_file = sc.textFile(“hdfs://localhost:9000/input”)
>>> bicounts = text_file.map(lambda line: line.split(“ ”)) \
...             .flatMap(lambda x: ???) \
...             .reduceByKey(lambda a, b: a + b)
>>> bicounts.saveAsTextFile(“hdfs://localhost:9000/output3”)
[Stage 0:>>>>>>>>>>>>>>                                   (1 + 1) / 3]
[Stage 1:>>>>>>>>>>>>>>                                   (1 + 1) / 3]
>>> exit()
```

Exercise 2

- **Check the output**

```
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ bin/hdfs dfs -cat /output3
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ bin/hdfs dfs -getmerge /output3 result3.txt
ubuntu@ip-x-x-x:~/hadoop-2.7.4$ vi result3.txt
```

Appendix