

Computer Science & Information Systems

Big Data Systems – Spark Lab Sheet 6

Spark Streaming

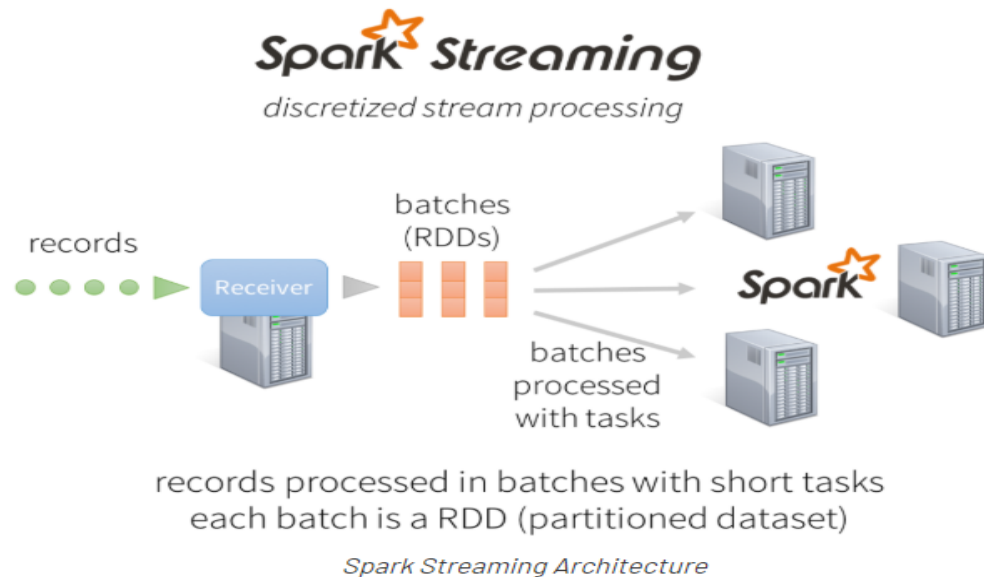
1. Objective:

Students should be able to

- A. Get familiarity with the execution of Python programmes on the Spark cluster
- B. Get hands-on experience with Spark streaming

Many applications benefit from acting on data as soon as it arrives. For example, an application might track the logs in real time. Spark streaming is Spark's module for such applications. It lets users write streaming applications using a very similar API to batch jobs and thus reuse a lot of skills and even code they built for those.

Spark streaming provides an abstraction called Dstreams or discretized streams. It is sequence of data arriving over time. Internally, each DStream is represented as sequence of RDDs arriving at each time step. DStreams can be created from various input sources, such Flume, Kafka etc. Once built they offer two types of operations: transformations, which yield a new DStream, and output operations, which write data to external system. DStreams provide many of the same operations available on RDDs, plus a new operations related to time, such as sliding window.



This lab sheet provides a quick introduction of using Spark streaming for data processing with Python. This exercise will introduce the API through pySpark package. The use case that will be considered here is for log analysis. We will assume that continuously server logs are placed into a particular location and the Spark streaming application will read on those logs in order to do some summarization based on these logs. Let's go ahead and see how we can write such type of application.

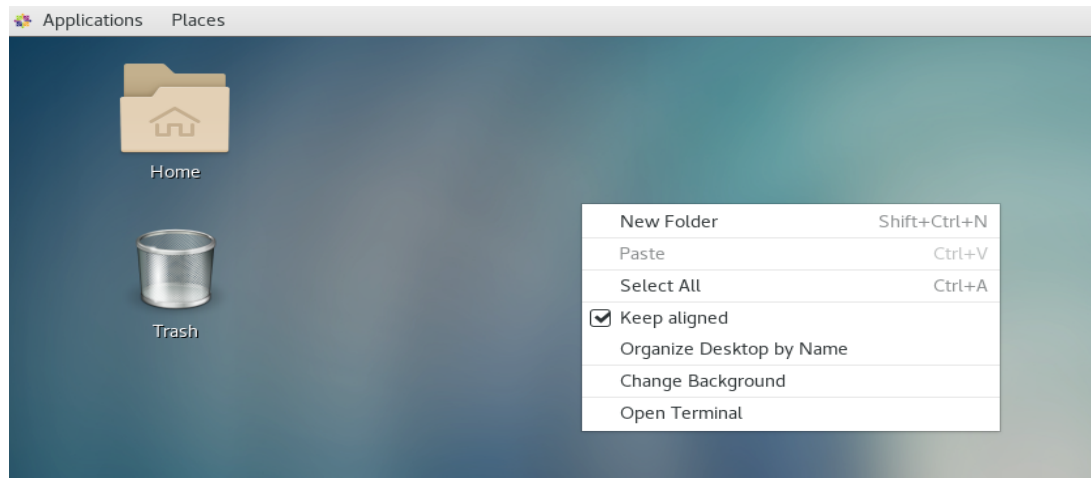
2. Steps to be performed:

Note - It's assumed that student has made a slot reservation using the slot booking interface where Apache Spark framework was selected. The details of the Apache Spark systems to be used is received through an email. If not, please contact the administrators for the same.

Also it's assumed that students are aware of the process of logging into these virtual machines. If not, then get access to the user manual maintained for the usage of remote lab setup.

Preparations -

- Open the terminal by right clicking on the desktop of the virtual machine.



- b) Look at the current directory and also file listings in it. It must have a spark installation directory present in it. Commands like pwd, ls can be used for it.

```

Applications Places Terminal
csishydlab@apache-spark:~

File Edit View Search Terminal Help
[csishydlab@apache-spark ~]$ pwd
/home/csishydlab
[csishydlab@apache-spark ~]$ ls
boston.csv          k1.py               people.json          spark-2.4.4-bin-hadoop2.7
consumer.py          kafka_2.12-2.4.0    Pictures              spark-2.4.4-bin-hadoop2.7.tgz
d1.py                kafka_2.12-2.4.0.tgz producer.py           Spark-DataFrame.ipynb
data                 log.txt              Public                spark-streaming-kafka-0-8-assembly_2.11-2.4.4.jar
Desktop              lr1.py               r1.py                 spark-warehouse
direct_kafka_wordcount.py Music                 sample_linear_regression_data.txt students.json
Documents             network_wordcount.py sample_svm_data.txt   Templates
Downloads              output                scala-2.10.1.tgz      Videos
input.txt             pl.py                sk1.py
[csishydlab@apache-spark ~]$

```

- c) Set the SPARK_HOME and HOME variable to point to the spark installations.

```
[csishydlab@apache-spark bin]$ pwd
```

```
/home/csishydlab/spark-2.4.4-bin-hadoop2.7/
```

```
[csishydlab@apache-spark bin]$ export SPARK_HOME=/home/csishydlab/spark-2.4.4-bin-hadoop2.7/bin
```

```
[csishydlab@apache-spark bin]$ export PATH="$SPARK_HOME/bin:$PATH"
```

```
echo $SPARK_HOME
```

```
echo $PATH
```

```
Applications Places Terminal Sun 19:30
csishydlab@apache-spark:~$ pwd
/home/csishydlab
csishydlab@apache-spark ~$ ls
boston.csv          k1.py              people.json        spark-2.4.4-bin-hadoop2.7
consumer.py         kafka_2.12-2.4.0   Pictures           spark-2.4.4-bin-hadoop2.7.tgz
d1.py               kafka_2.12-2.4.0.tgz producer.py        Spark-DataFrame.ipynb
data                log.txt            Public             spark-streaming-kafka-0-8-assembly_2.11-2.4.4.jar
Desktop             lr1.py             r1.py             spark-warehouse
direct_kafka_wordcount.py Music              sample_linear_regression_data.txt students.json
Documents           network_wordcount.py sample_svm_data.txt Templates
Downloads           output            scala-2.10.1.tgz   Videos
input.txt           pl.py             skl.py
csishydlab@apache-spark ~$ export SPARK_HOME=/home/csishydlab/spark-2.4.4-bin-hadoop2.7/bin
csishydlab@apache-spark ~$ export PATH=$SPARK_HOME/bin:$PATH
csishydlab@apache-spark ~$ echo $SPARK_HOME
/home/csishydlab/spark-2.4.4-bin-hadoop2.7/bin
csishydlab@apache-spark ~$ echo $PATH
/home/csishydlab/spark-2.4.4-bin-hadoop2.7/bin:/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:/home/csishydlab/.local/bin:/home/csishydlab/bin:/usr/lib/scala/bin:/home/csishydlab/spark-2.4.4-bin-hadoop2.7/bin
```

d) Create a directory named “logs” under home directory.

Note – you may need to use the root privileges in order to create this directory.

```
>> sudo su
```

```
>> mkdir logs
```

```
File Edit View Search Terminal Tabs Help
csishydlab@apache-... x csishydlab@apache-... x csishydlab@ap
[root@apache-spark csishydlab]# sudo su
[root@apache-spark csishydlab]# mkdir logs
[root@apache-spark csishydlab]#
```

e) Then create a file named “server_logs.txt” in “logs” directory. Then copy the content from attached “server_logs.txt” file to this newly created file. Spend some time to analyze the structure of this file.

```
>> cd logs
```

```
>> gedit server_logs.txt &
```



```
File Edit View Search Terminal Tabs Help
csishydlab@apache-... x csishydlab@apache-... x csishydlab@
[root@apache-spark csishydlab]# sudo su
[root@apache-spark csishydlab]# mkdir logs
[root@apache-spark csishydlab]# cd logs/
[root@apache-spark logs]# gedit server_logs.txt&
```

Installing pySpark

- f) For the execution of python programmes on the Spark, a package named pyspark is required. Using the sudo privileges, install the packages with pip command.

```
pip install pyspark
```

Writing Spark Streaming Application

- g) Open up the text editor and copy the code written in the attached streaming_demo.py file.

```
>> cd ..
```

```
>> gedit streaming_demo.py&
```

```
File Edit View Search Terminal Tabs Help
csishydlab@apache-... x csishydlab@apache-... x csishydlab@apache-... x csishydlab@
[root@apache-spark csishydlab]# sudo su
[root@apache-spark csishydlab]# mkdir logs
[root@apache-spark csishydlab]# cd logs/
[root@apache-spark logs]# gedit server_logs.txt&
[1] 18688
[root@apache-spark logs]# cd ..
[1]+  Done                  gedit server_logs.txt (wd: /home/csishydlab/logs)
(wd now: /home/csishydlab)
[root@apache-spark csishydlab]# gedit streaming_demo.py &
```

```
Applications  Places  Text Editor

streaming_demo.py
/home/csishydlab

streaming_demo.py  x  sdemo.py  x  hhh.txt  x

# -*- coding: utf-8 -*-

from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.sql import Row, SparkSession

from pyspark.sql.functions import regexp_extract

# Create a SparkSession
spark = SparkSession.builder.config("spark.sql.warehouse.dir", "file:///logs").appName("StructuredStreamingDemo").getOrCreate()

# Monitor the logs directory for new server log data, and read in the raw lines as log_lines
log_lines = spark.readStream.text("logs")

# Parse out the log format
host_exp = r'(^[\S+\.]{[\S+\.]+}[\S+])\s'
time_exp = r'\[(\d{2}/\w{3}/\d{4}:\d{2}:\d{2} -\d{4})]'
general_exp = r'"\s*(\S+)\s+(\S+)\s*(\S+)\s*" '
status_exp = r'\s(\d{3})\s'
content_size_exp = r'\s(\d+)$'

logs_DF = log_lines.select(regexp_extract('value', host_exp, 1).alias('Host'),
                           regexp_extract('value', time_exp, 1).alias('Timestamp'),
                           regexp_extract('value', general_exp, 1).alias('Method'),
                           regexp_extract('value', general_exp, 2).alias('Endpoint'),
                           regexp_extract('value', general_exp, 3).alias('Protocol'),
                           regexp_extract('value', status_exp, 1).cast('integer').alias('Status'),
```

- h) Execute the `streaming_demo.py` file using the `spark-submit` command. Accumulated the output in the text file.

```
>> spark-submit streaming_demo.py >>streaming_demo.log
```

- i) Look at the outcome printed inside the log file while the program is getting executed on the Spark cluster. It shows grouping of the requests by hosts.

```

-----
Batch: 0
-----
+-----+-----+
|          Host | count |
+-----+-----+
| 54.15.19.171 |    20 |
| 46.16.19.200 |    36 |
| 66.45.75.12  |     2 |
| 18.7.26.22   |     1 |
| 54.15.19.171 |    16 |
+-----+-----+

```

- j) Stop the execution of the program by 'ctrl' + 'c'.
- k) You may like to clear out the content of log file "streaming_demo.log".
- l) Open the streaming_demo.py file again in the text editor. Uncomment the following line in the file.


```
status_counts_DF = logs_DF.groupBy(logs_DF.Status).count()
```

Comment the following line in the file.

```
#status_counts_DF = logs_DF.groupBy(logs_DF.Host).count()
```

Save the changes.
- m) Execute the streaming_demo.py file using the spark-submit command. Accumulated the output in the text file.


```
>> spark-submit streaming_demo.py >>streaming_demo.log
```
- n) Look at the outcome printed inside the log file while the program is getting executed on the Spark cluster. It shows grouping of the requests by status.

```
-----
Batch: 0
-----
+-----+-----+
|Status|count|
+-----+-----+
| 500 | 10 |
| 301 | 11 |
| 404 | 11 |
| 200 | 18 |
| 304 | 18 |
| 405 | 7 |
+-----+-----+
```

- o) Stop the execution of the program by 'ctrl' + 'c'.
- p) Change directory to "logs" and make another copy of the "server_logs" file.

```
>> cd logs
```

```
>> ls
```

```
>> cp server_logs.txt server_logs_2.txt
```

```
csishydlab@
File Edit View Search Terminal Tabs Help
csishydlab@apache-... x csishydlab@apache-... x csishydlab@apache-..
[root@apache-spark csishydlab]#
[root@apache-spark csishydlab]# cd logs/
[root@apache-spark logs]# ls
server_logs.txt
[root@apache-spark logs]# cp server_logs.txt server_logs_2.txt
[root@apache-spark logs]# █
```

- q) Execute the streaming_demo.py file using the spark-submit command. Accumulated the output in the text file.

```
>> spark-submit streaming_demo.py >>streaming_demo.log
```

- r) Look at the outcome printed inside the log file while the program is getting executed on the Spark cluster. It shows grouping of the requests by hosts which are double is number as the same input is made available to the program.

```
-----
Batch: 0
-----
+-----+-----+
|Status|count|
+-----+-----+
| 500 | 20 |
| 301 | 22 |
| 404 | 22 |
| 200 | 36 |
| 304 | 36 |
| 405 | 14 |
+-----+-----+
```


- s) You can try the same by making multiple copies of the log file and see how continuously the outcome is changing.

3. Outputs/Results:

Students should be able to

- Execute the python streaming application on Spark cluster
- See the processing of the log statement based on the different group by options

4. Observations:

Students carefully needs to observe

- Way in which the input / log file is made available to the program using local file system
- Details provided while spark application was running
- Output grouped based upon the different keys

5. References:

- A. [Spark Documentation](#)
- B. [pySpark API Guide](#)
- C. [Spark Streaming Documentation](#)

Work Integrated
Learning Programmes



BITS Pilani
Pilani | Dubai | Goa | Hyderabad