Work Integrated Learning Programmes

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Computer Science & Information Systems

# Big Data Systems – Spark Lab Sheet 6

# Spark Streaming

## 1. Objective:

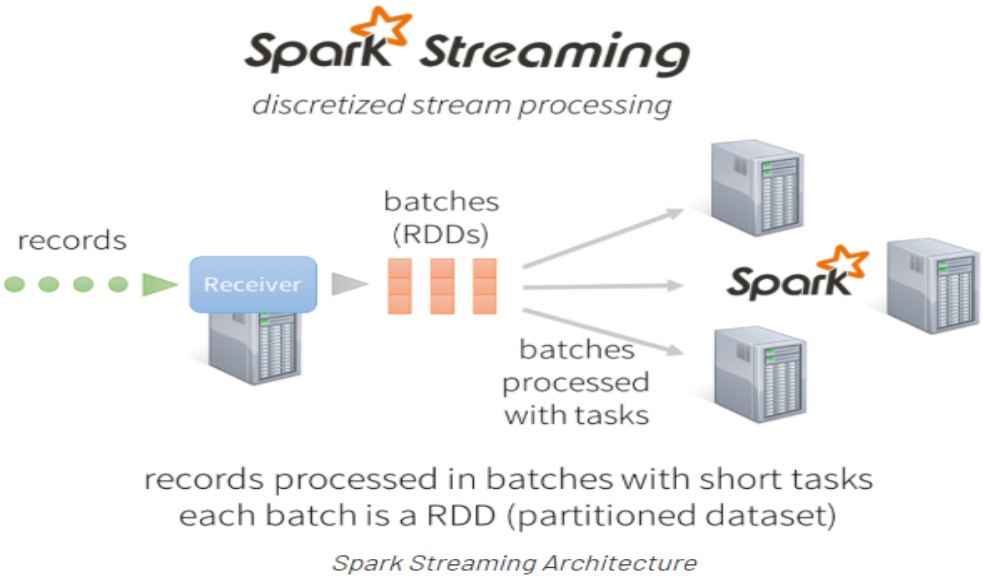
Students should be able to

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 date to external system. DStreams provide many of the same operations available on RDDs, plus a new operations related to time, such as sliding window.



Spark Streaming Architecture

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 steaming 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.

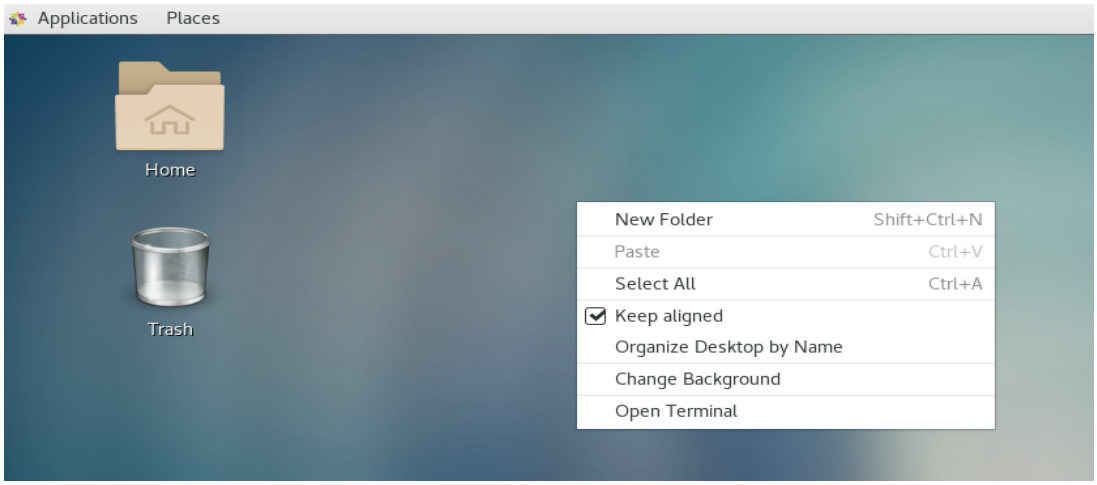
## 2. Steps to be performed:

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.

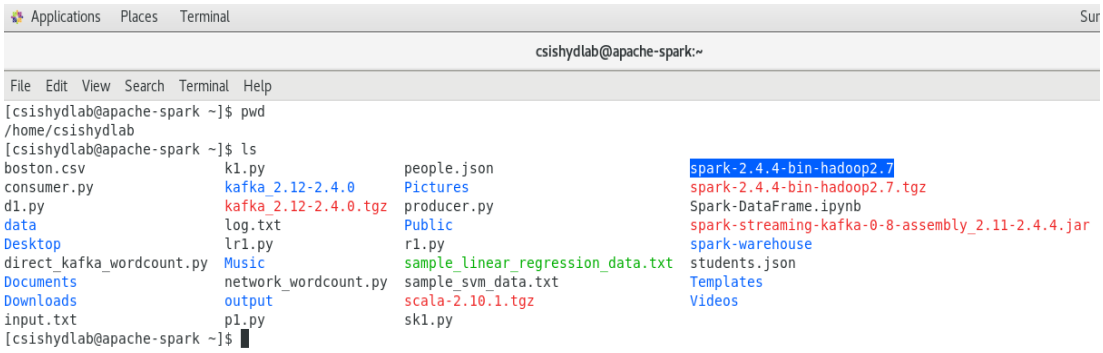
Also it's assumed that students are aware of the process of logging into these virtual machines. If not, then get access to the user manual maintained for the usage of remote lab setup.

**Preparations -**

a) 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.




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

[csishydlab@apache-spark bin]$ pwd

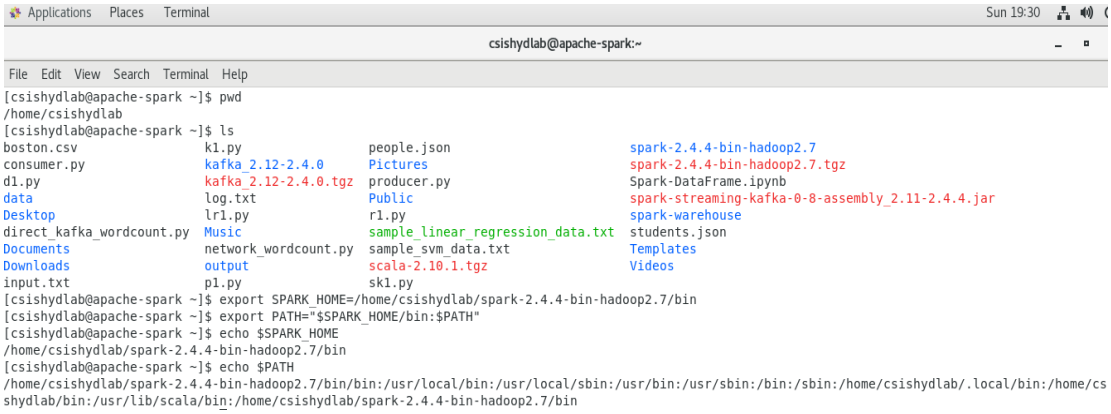
/home/csishydlab/spark-2.4.4-bin-hadoop2.7/

[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

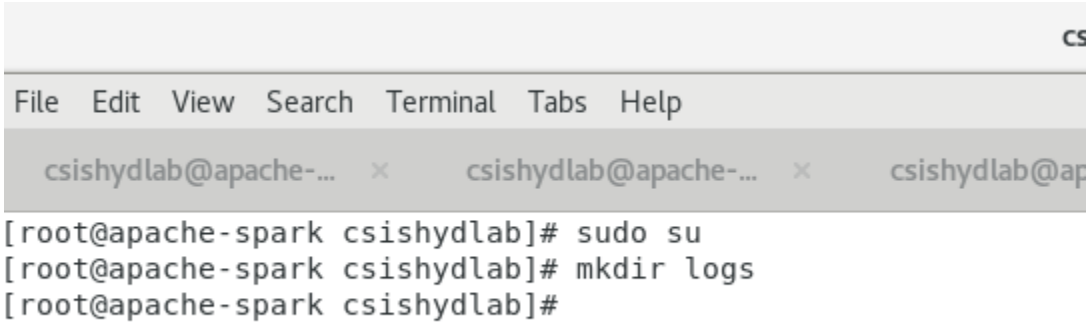



d) Create a directory named “logs” under home directory.

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

>> sudo su

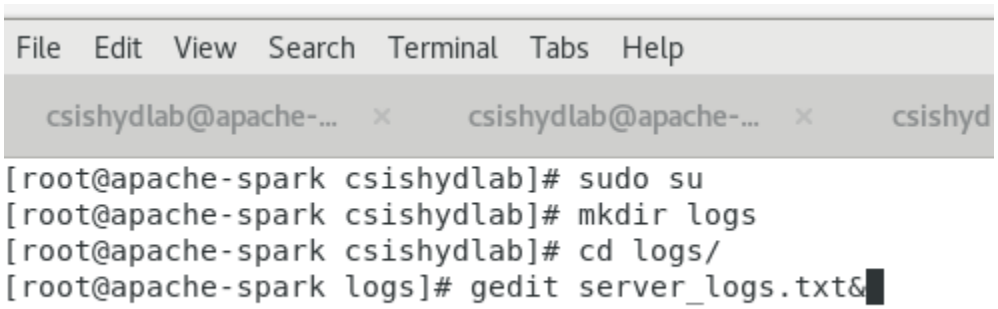
>> mkdir logs




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 &




## Installing pySpark

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

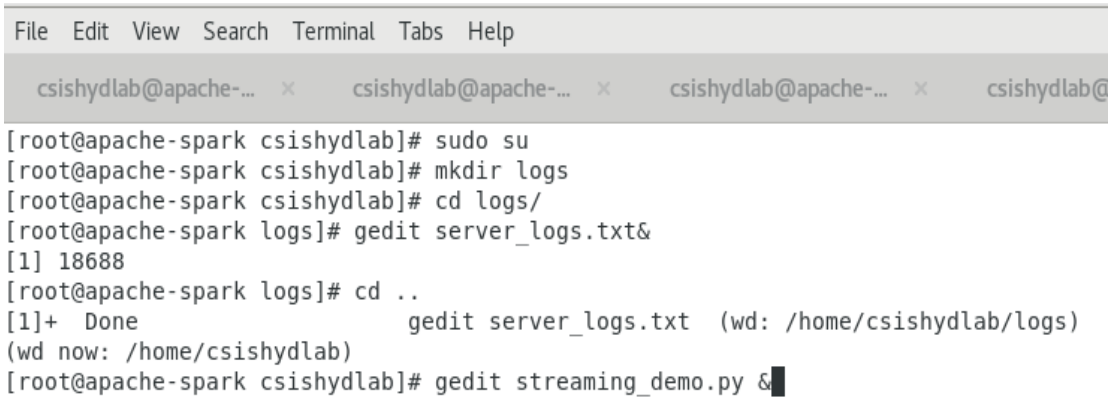
pip install pyspark

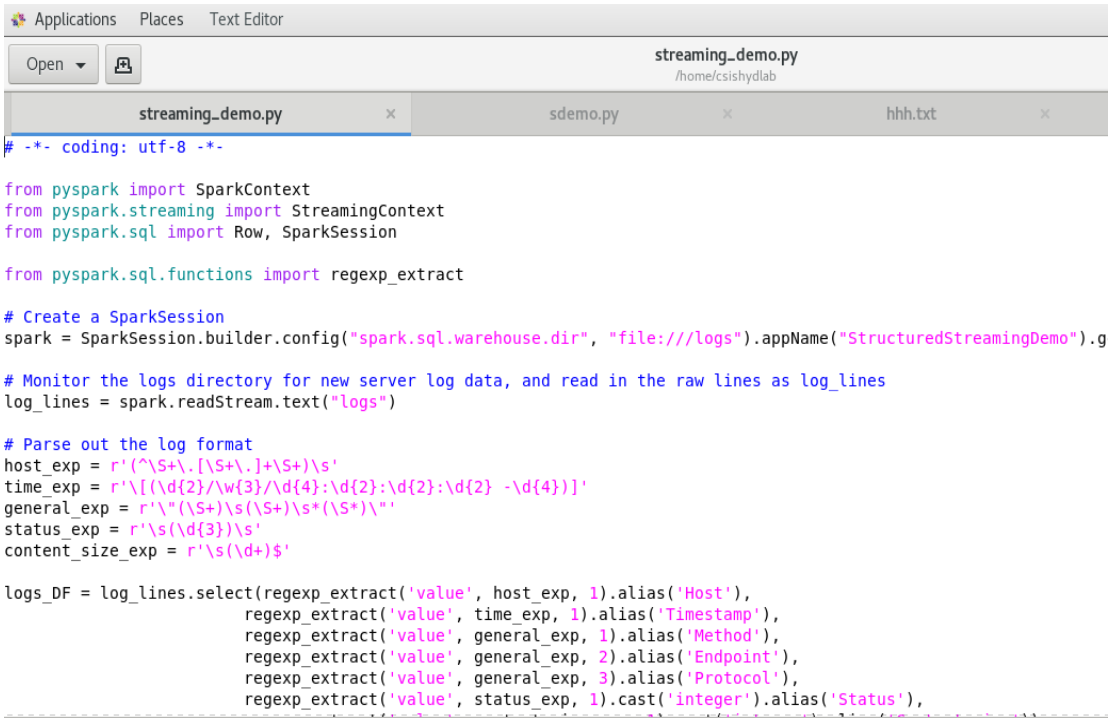
## Writing Spark Streaming Application

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

>> cd ..

>> gedit streaming_demo.py&






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|
+-------------+-----+
|54.15.19.1711|   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|
+------+-----+
```

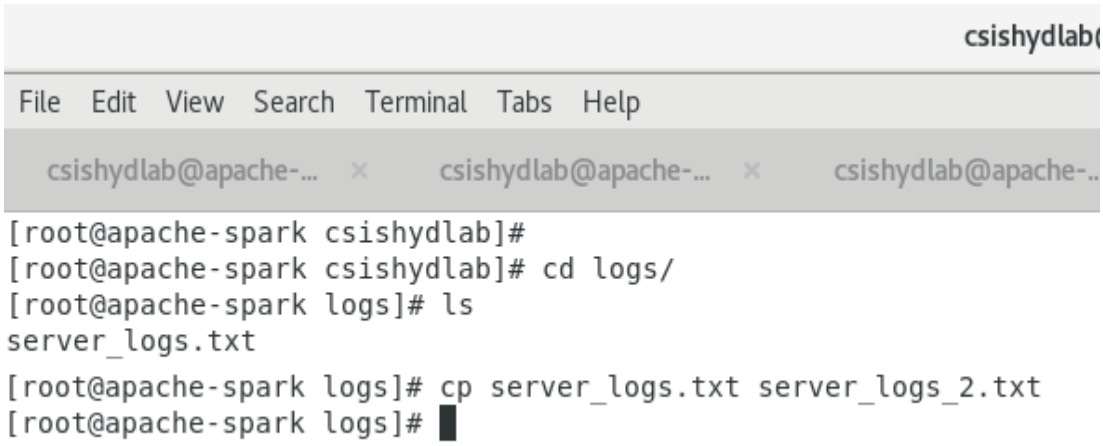
o) Stop the execution of the program by 'ctrl' + 'c'.

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




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