TDDE31/732A54-Big Data Analytics Lab Compendium

Notice: *Please make sure you have read the whole lab compendium before you start to work on the server from NSC.*

April 20, 2020

1 Description and Aim

In the lab you will work on the Sigma[1] set up which is a HPC cluster from the National Supercomputer Centre (NSC). You are supposed to work with the historical meteorological data from the Swedish Meteorological and Hydrological Institute (SMHI). Specifically, you will work with air temperature readings and precipitation readings from 812 meteorological stations in Sweden[2]. In these exercises, you will work with Spark 2.4.3[3].

After completing the first two labs you will have basic knowledge of the programing environment, techniques and APIs. You will work on exercises with Spark and Spark SQL and thus will be able to compare the differences between the two approaches. In the third lab, you are supposed to achieve a machine learning method with Spark. The overview of three labs is that you need to upload data to Hadoop Distributed File System (HDFS), read the data from HDFS in your code and then program with PySpark[4] to answer the labs' questions.

2 SMHI Data

The data includes air temperature and precipitation readings from 812 stations in Sweden. The stations include both currently active stations as well readings from historical stations that have been closed down. The latest readings available for active stations are from October 10, 2016. The air temperature and precipitation records are hourly readings, however some stations provide only one reading every three hours.

The provided files[5] are prepared csv files with removed headers. Values are separated with semicolons. Some files are too big to be read using some text editors. Therefore, please use either python to read the files or bash commands such as *tail* and *more* to get an overview of a file's content. Provided files:

[1]Sigma at: `https://www.nsc.liu.se/systems/sigma/`
[2]If interested in other readings please check: `http://opendata-catalog.smhi.se/explore/`
[3]Spark 2.4.3 at: `https://spark.apache.org/docs/2.4.3/index.html`
[4]PySpark 2.4.3 at: `https://spark.apache.org/docs/2.4.3/api/python/index.html`
[5]`https://www.ida.liu.se/~732A54/lab/data.zip`

- temperature-readings.csv - ca 2 GB
- precipitation-readings.csv - ca 660 MB
- stations.csv
- stations-Ostergotland.csv

All these files are available at */software/sse/manual/spark/BDA_demo/input_data* folder on Sigma. The headers of these files are shown in Table 1, Table 2 and Table 3. If you notice any mistakes in the dataset or have any comments please contact the lab assistants.

Station number	Date	Time	Air temperature (in °C)	Quality[a]

[a] G - controlled and confirmed values, Y - suspected or aggregated values

Table 1: Headers for temperature-readings.csv

Station number	Date	Time	Precipitation (in mm)	Quality[a]

[a] G - controlled and confirmed values, Y - suspected or aggregated values

Table 2: Headers for precipitation-readings.csv

Station number	Station name	Measurement height	Latitude	Longitude	Readings from (date and time)	Readings to (date and time)	Elevation

Table 3: Headers for stations.csv and stations-Ostergotland.csv

3 Running PySpark Program on Sigma

3.1 Working on Sigma

The Sigma server is available at *sigma.nsc.liu.se* (log in using your NSC accounts). You can use ssh forwarding connection or Thinlinc[6] connection to log in Sigma. It's also fine to use regular ssh connection without forwarding option, in this case, you have to program locally and use *scp* command to upload your code to Sigma, and then check the history log from text file.

- Thinlinc server is avaliable same as *sigma.nsc.liu.se*. In this way, you can get a graphical environment on Sigma and given that you work directly on Sigma there is no need to use *ssh* or *scp*. When you use Thinlinc, don't forget to uncheck the full screen mode so that you can share the screen with your lab partner or lab assistant when you ask questions. **Note:** Please remember to log out when done working on the labs so that Sigma does not keep open Thinlinc sessions. Also, each pair of students, please uses one Thinlinc connection during lab sessions, due to the limited number of Thinlinc licenses on Sigma.

[6] `https://www.cendio.com/thinlinc/download`

- ***ssh -X username@sigma.nsc.liu.se*** where username is your NSC username (not the LiU one), -X indicates forwarding function of ssh which is used for running graphics applications remotely.

- ***[username@sigma ~] $ exit*** is used to logout Sigma. If it is hung on, please use ctrl-c to terminate the connection.

- ***[username@sigma ~] $ emacs &*** You can use Emacs for coding by running ***emacs &*** in the terminal after you connect to Sigma or program locally on you machine, then use *scp* command to copy you files to Sigma.

- ***scp LOCAL_FILE_PATH username@sigma.nsc.liu.se:Documents*** is used for uploading files from your local machine to Sigma. (**Note:** you are supposed to run *scp* command before you log in Sigma when you want to upload files to Sigma.)

NSC reserves nodes for each lab session of the course, which means other jobs on Sigma will not use these nodes during our lab sessions. You can check and use the reservations by running following commands:

- ***[username@sigma ~] $ listreservations***

- ***[username@sigma ~] $ export SBATCH_RESERVATION=RESERVATION***

- ***[username@sigma ~] $ export SBATCH_RESERVATION=devel*** (outside lab sessions)

3.2 Submitting Jobs to Hadoop Cluster and Sigma

To submit your pyspark code to the Hadoop cluster, you will use:

- ***spark-submit --deploy-mode cluster --master yarn --num-executors 9 --driver-memory 2g --executor-memory 2g --executor-cores 4 CODE.py*** where CODE.py is the python script in the current folder. In this command, Yarn is used for resource management and the cluster-deploy mode is used.

To run your pyspark code on Hadoop cluster, you also need to first submit a job to Sigma. Some scripts in the demo (in next section) are provided to you so you can make the calling of your *spark-submit* command easier. You will use the non-interactive way to submit a job on Sigma. Each time after you submit, the job will enter the scheduling queue. You can use *sbatch* command to submit the job, *squeue* command to monitor the submitted job and may use *scancel* command to cancel a job.

- ***[username@sigma ~] $ sbatch -A liu-compute-2020-3 run.q*** Note: Don't forget to use '-A liu-compute-2020-3' if you are envolved in more than one project from NSC, it will guarantee to use the allocation reserved on Sigma for our course.

- ***[username@sigma ~] $ squeue -u username***

- ***[username@sigma ~] $ scancel JOB_ID***

Sigma uses Slurm for scheduling. Once you submit a job, the job will be assigned an ID. After the job is finished, you will see a *slurm-JOB_ID.out* file returned, which includes the output information of the job script. The script *run.q* is supposed to contain commands for constructing the Hadoop cluster for your account, commands for interacting with Hadoop Distributed File System (HDFS) and *spark-submit* command for running pyspark code. A detailed example of *run.q* will be shown in following section which introduces the demo.

In order to be able to access the logs after the execution, you will need to set the *spark.eventLog.enabled* flag when running your jobs:

- ***spark-submit --conf spark.eventLog.enabled=true --deploy-mode cluster --master yarn --num-executors 9 --driver-memory 2g --executor-memory 2g --executor-cores 4 CODE.py***

The script '*run_yarn_with_historyserver.q*' in the demo includes the configuration for running the history server. There are two ways to access the logs, one is integrated in the script which is:

- ***yarn logs -applicationId "$APPLICATION_ID"***

The other way is to run following commands. Then a firefox session that points at the history server web UI will be opened. In this way, it only lists the latest finished job on the historyserver.

- ***module load spark/.2.4.3-hadoop-2.7-nsc1***
- ***spark_browse_historyserver -A liu-compute-2020-3 --reservation RESERVATION***

3.3 Demo

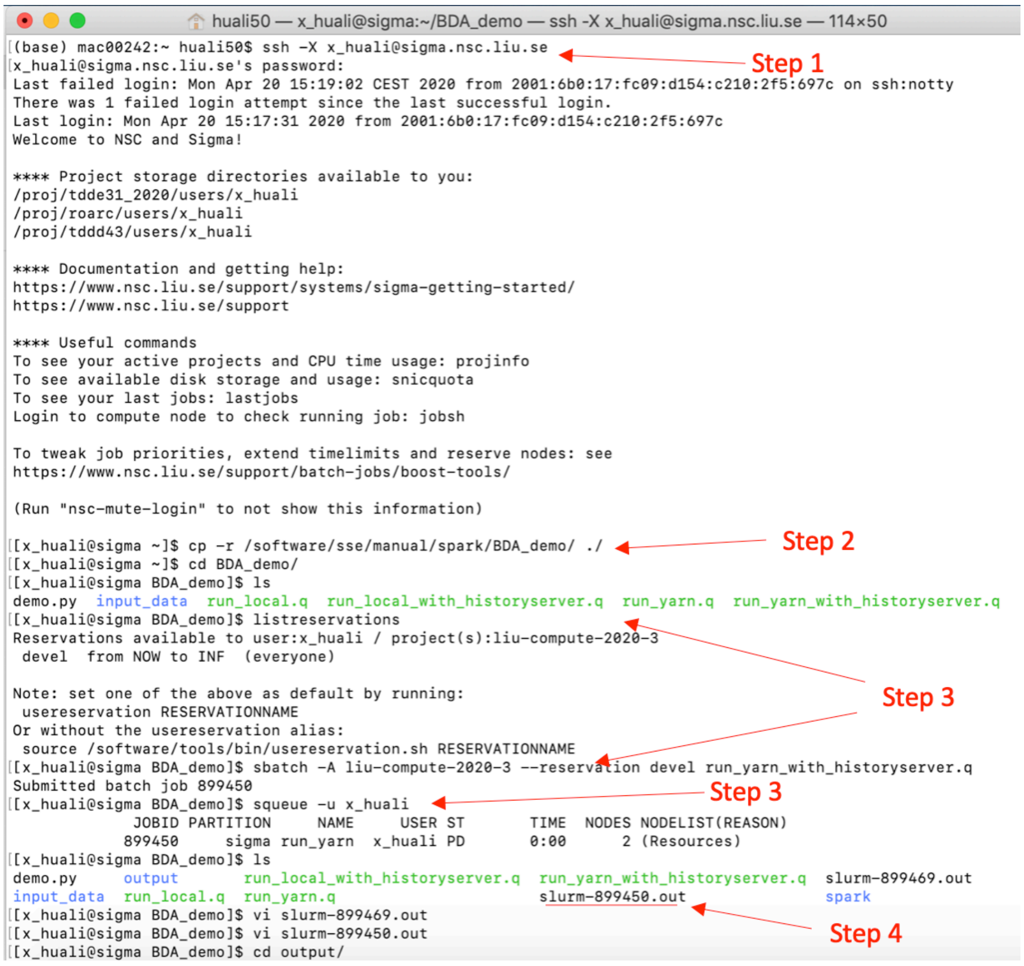
You can find the demo from following folders after you log in Sigma.

- */software/sse/manual/spark/BDA_demo/*
- */software/sse/manual/spark/examples/pyspark_on_hdfs/*

The steps to run the *BDA_demo* are shown as below, and in Figure 1 and Figure 2.

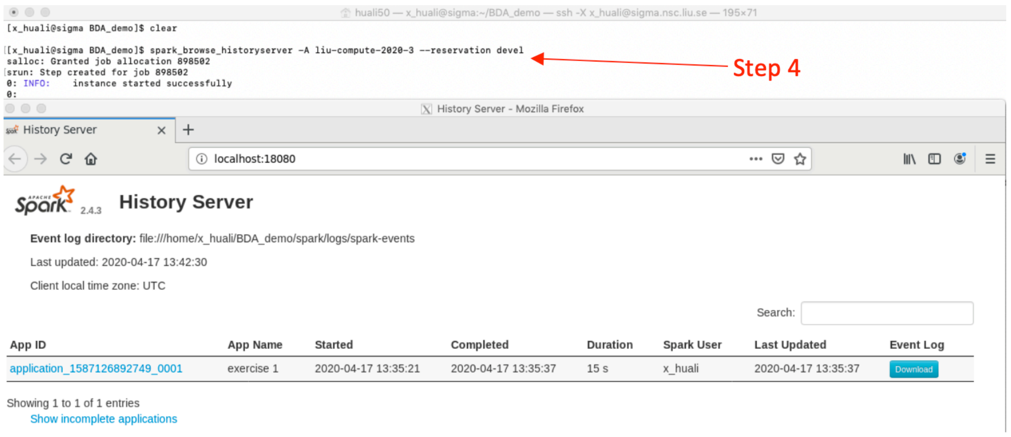
- Step 1: Login in to Sigma with '*ssh -X*' connection or Thinlinc.
- Step 2: Copy the demo to your home folder on Sigma.
- Step 3: Check and enable the reservation. Use *sbatch* to submit the job. Then use *squeue* command to monitor your job. Once the job is finished, check the returned file named *slurm-ID.out*.
- Step 4: Check the history log from *slurm-ID.out* or run *spark_browse_historyserver*.

In '*run_yarn_with_historyserver.q*' as shown in Figure 3, you can see a number of commands that are used to interact with HDFS.


huali50 — x_huali@sigma:~/BDA_demo — ssh -X x_huali@sigma.nsc.liu.se — 114×50
Step 1
Step 2
Step 3
Step 3
Step 4

```
(base) mac00242:~ huali50$ ssh -X x_huali@sigma.nsc.liu.se
x_huali@sigma.nsc.liu.se's password:
Last failed login: Mon Apr 20 15:19:02 CEST 2020 from 2001:6b0:17:fc09:d154:c210:2f5:697c on ssh:notty
There was 1 failed login attempt since the last successful login.
Last login: Mon Apr 20 15:17:31 2020 from 2001:6b0:17:fc09:d154:c210:2f5:697c
Welcome to NSC and Sigma!

**** Project storage directories available to you:
/proj/tdde31_2020/users/x_huali
/proj/roarc/users/x_huali
/proj/tddd43/users/x_huali

**** Documentation and getting help:
https://www.nsc.liu.se/support/systems/sigma-getting-started/
https://www.nsc.liu.se/support

**** Useful commands
To see your active projects and CPU time usage: projinfo
To see available disk storage and usage: snicquota
To see your last jobs: lastjobs
Login to compute node to check running job: jobsh

To tweak job priorities, extend timelimits and reserve nodes: see
https://www.nsc.liu.se/support/batch-jobs/boost-tools/

(Run "nsc-mute-login" to not show this information)

[x_huali@sigma ~]$ cp -r /software/sse/manual/spark/BDA_demo/ ./
[x_huali@sigma ~]$ cd BDA_demo/
[x_huali@sigma BDA_demo]$ ls
demo.py  input_data  run_local.q  run_local_with_historyserver.q  run_yarn.q  run_yarn_with_historyserver.q
[x_huali@sigma BDA_demo]$ listreservations
Reservations available to user:x_huali / project(s):liu-compute-2020-3
 devel  from NOW to INF  (everyone)

Note: set one of the above as default by running:
 usereservation RESERVATIONNAME
Or without the usereservation alias:
 source /software/tools/bin/usereservation.sh RESERVATIONNAME
[x_huali@sigma BDA_demo]$ sbatch -A liu-compute-2020-3 --reservation devel run_yarn_with_historyserver.q
Submitted batch job 899450
[x_huali@sigma BDA_demo]$ squeue -u x_huali
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            899450     sigma run_yarn  x_huali PD       0:00      2 (Resources)
[x_huali@sigma BDA_demo]$ ls
demo.py     output      run_local_with_historyserver.q  run_yarn_with_historyserver.q  slurm-899469.out
input_data  run_local.q  run_yarn.q                     slurm-899450.out               spark
[x_huali@sigma BDA_demo]$ vi slurm-899469.out
[x_huali@sigma BDA_demo]$ vi slurm-899450.out
[x_huali@sigma BDA_demo]$ cd output/
```

Figure 1: Steps to run the demo program


huali50 — x_huali@sigma:~/BDA_demo — ssh -X x_huali@sigma.nsc.liu.se — 195×71
Step 4
History Server - Mozilla Firefox
History Server
localhost:18080
2.4.3
Search:
Download

```
[x_huali@sigma BDA_demo]$ clear

[x_huali@sigma BDA_demo]$ spark_browse_historyserver -A liu-compute-2020-3 --reservation devel
salloc: Granted job allocation 898502
srun: Step created for job 898502
0: INFO:    instance started successfully
0:
```

History Server

Event log directory: file:///home/x_huali/BDA_demo/spark/logs/spark-events

Last updated: 2020-04-17 13:42:30

Client local time zone: UTC

App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
application_1587126892749_0001	exercise 1	2020-04-17 13:35:21	2020-04-17 13:35:37	15 s	x_huali	2020-04-17 13:35:37	Download

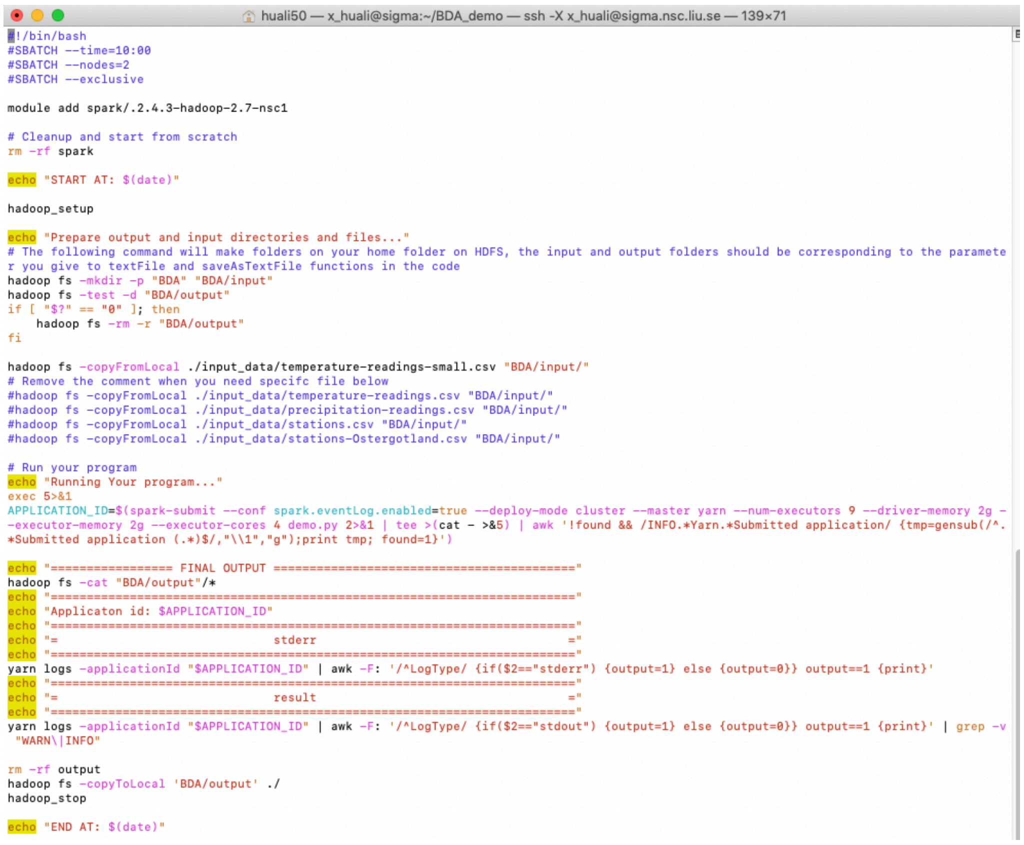
Showing 1 to 1 of 1 entries

Show incomplete applications

Figure 2: Check the log through open Spark Web UI

- ***hadoop fs -mkdir <FOLDER_NAME>*** -make a folder on HDFS
- ***hadoop fs -mkdir -p <FOLDER_NAME> <FOLDER_NAME>*** -make multiple folders

- ***hadoop fs -test -d <FOLDER_NAME>*** -if the path is a directory, return 0
- ***hadoop fs -rm -r <FOLDER_NAME>*** -deletes the directory and any content under it recursively
- ***hadoop fs -cat <FOLDER_ON_HDFS> [local]*** -copy HDFS path to stdout
- ***hadoop fs -copyFromLocal <localsrc> ... <dst>*** -copy single src, or multiple srcs from local Sigma to HDFS
- ***hadoop fs -copyToLocal <dst> ... <localsrc>*** -copy single src, or multiple srcs from HDFS to local Sigma


huali50 — x_huali@sigma:~/BDA_demo — ssh -X x_huali@sigma.nsc.liu.se — 139×71

```bash
#!/bin/bash
#SBATCH --time=10:00
#SBATCH --nodes=2
#SBATCH --exclusive

module add spark/.2.4.3-hadoop-2.7-nsc1

# Cleanup and start from scratch
rm -rf spark

echo "START AT: $(date)"

hadoop_setup

echo "Prepare output and input directories and files..."
# The following command will make folders on your home folder on HDFS, the input and output folders should be corresponding to the paramete
r you give to textFile and saveAsTextFile functions in the code
hadoop fs -mkdir -p "BDA" "BDA/input"
hadoop fs -test -d "BDA/output"
if [ "$?" == "0" ]; then
    hadoop fs -rm -r "BDA/output"
fi

hadoop fs -copyFromLocal ./input_data/temperature-readings-small.csv "BDA/input/"
# Remove the comment when you need specifc file below
#hadoop fs -copyFromLocal ./input_data/temperature-readings.csv "BDA/input/"
#hadoop fs -copyFromLocal ./input_data/precipitation-readings.csv "BDA/input/"
#hadoop fs -copyFromLocal ./input_data/stations.csv "BDA/input/"
#hadoop fs -copyFromLocal ./input_data/stations-Ostergotland.csv "BDA/input/"

# Run your program
echo "Running Your program..."
exec 5>&1
APPLICATION_ID=$(spark-submit --conf spark.eventLog.enabled=true --deploy-mode cluster --master yarn --num-executors 9 --driver-memory 2g -
-executor-memory 2g --executor-cores 4 demo.py 2>&1 | tee >(cat - >&5) | awk '!found && /INFO.*Yarn.*Submitted application/ {tmp=gensub(/^.
*Submitted application (.*)$/,"\\1","g");print tmp; found=1}')

echo "================= FINAL OUTPUT ==========================================="
hadoop fs -cat "BDA/output"/*
echo "========================================================================="
echo "Applicaton id: $APPLICATION_ID"
echo "========================================================================="
echo "=                              stderr                                   ="
echo "========================================================================="
yarn logs -applicationId "$APPLICATION_ID" | awk -F: '/^LogType/ {if($2=="stderr") {output=1} else {output=0}} output==1 {print}'
echo "========================================================================="
echo "=                              result                                   ="
echo "========================================================================="
yarn logs -applicationId "$APPLICATION_ID" | awk -F: '/^LogType/ {if($2=="stdout") {output=1} else {output=0}} output==1 {print}' | grep -v
 "WARN\|INFO"

rm -rf output
hadoop fs -copyToLocal 'BDA/output' ./
hadoop_stop

echo "END AT: $(date)"
```

Figure 3: '*run_yarn_with_historyserver.q*' for running the code

4 Hand In

You are supposed to use GitLab[7] to submit your report and code. For each lab, please submit the code and a report that contains your results (a snippet of the results is enough if the results contain many rows) and answers to the questions. In cases where a plot of your results is asked, you can include the figure directly in the report. You can use a tool of your preference to produce the plots (R, Excel, matplotlib in Python, etc.). Comment each step in your code to provide a clear picture of your reasoning when solving the problem.

[7]`https://gitlab.liu.se/olaha93/bigdata`