Data Science Infrastructures – Exercise 06 (DSI E06 ST 24)

30/05/2024 (Update V02 03/06/2024) / Philipp Wieder / philipp.wieder@gwdg.de
Software needed: None
Files: mellville.txt, pi.py (downloadable from Stud.IP)

General Prerequisites

To fulfil this assignment, you need an HPC account with GWDG. You should have received it from your lecturer (if not, please contact him directly). To work with the HPC systems, you need to follow this manual to get your SSH key onto the system: https://s.gwdg.de/ANQLmW. This SSH key is needed to authenticate yourself at the HPC systems as account name and password is not accepted (the latter is considered less secure). Please test whether the login works before you start with the assignment.

NOTE: (a)The first time you log in you are asked whether you want to add continue connecting. Say yes. (b) Once logged in, you will recognise that you entered one of the so-called frontend nodes named like `gwdu101`.

HELP: In case you run into issues, please ask the colleagues via the RocketChat channel https://chat.gwdg.de/channel/hpc-users.

Assignment I – Creating a Spark Cluster

Points: 1,50

Once you have access to the HPC system, you need to create a Spark cluster. For this assignment, we will use a short and to-the-point description without too many explanations. If you interested in all the details, please consult e.g. https://spark.apache.org/docs/latest/.

Assignment I

First you need to adjust the module path and load the Spark module:

```
export MODULEPATH=/opt/sw/modules/21.12/scc/common:$MODULEPATH
module load spark/3.4.0
```

Then start an interactive shell, use the following command:

`spark-shell --master local[1]` (the `local[n]` parameter specifies the number of CPU cores to be used. Make sure to keep `n` small to not use all resources).

You can now execute Scala expressions directly within the shell. Just submit a screenshot of the terminal/command line output as the result of the assignment. If you want, also have a look at the available Spark shell commands by typing `:help`. To leave the shell, type `:quit`.

NOTE: For most commands you can get info on how to use it by invoking it with the `--help` parameter.

Assignment II – Interactive Use of Spark

Points: 1,50

For preparation, you first need to transfer the file melville.txt[1] to the HPC cluster. This is done with the `scp` (secure copy) command. You find the file in Stud.IP. The following is executed under the assumption that you use the account name `hpc-ss23-61`. Just adopt it to your own account name. Another assumption is made: normally under Linux, the SSH key files reside in the `.ssh` directory (which itself is a subdirectory of your home directory). So, in case you have your SSH key files somewhere else, make sure that you adopt the path accordingly.

To get the file on the HPC system, execute the following (from within the GÖNET; from outside, please use the command according to the manual section "Connecting to the System outside GÖNET" from here https://s.gwdg.de/ANQLmW[2]):

```
scp -i ~/.ssh/hpc-ss23-61_id_rsa melville.txt hpc-ss23-61@login-
mdc.hpc.gwdg.de:/usr/users/hpc-ss23-61/
```

Please note, that this command has to be executed on your local machine as it copies the file from there to the HPC system.

Assignment II

Execute the following Scala expressions from within the Spark shell (see Assignment I on how to start the shell):

```
var map = sc.textFile("/usr/users/hpc-ss23-61/melville.txt").flatMap(line
=> line.split(" ")).map(word => (word,1));
var counts = map.reduceByKey(_ + _);
counts.saveAsTextFile("./output");
```

This actually does a MapReduce-like word count executed on the Spark cluster. The results are stored in the `output` directory. Make a screenshot of the contents of the `output` directory and one of the last results from the first file created in the `output` folder and submit it as the result of this

---

[1] This is the book "Moby Dick" by Herman Melville freely available at Project Gutenberg (https://www.gutenberg.org/files/2701/2701-0.txt).
[2] There is more information to be found here (e.g. regarding SSH): https://docs.hpc.gwdg.de/getting_started/connecting/index.html.

assignment (instead, you can also use `scp` to download the content of the output folder to your local machine and submit it is the result[3]).

Assignment III – Run a Spark Job

Points: 3,00

In this assignment, we create a Spark cluster by submitting a batch job[4]. This allows us, compared to the interactive use as in Assignment II, to tailor the size of the cluster according to our needs.

Assignment III

First create a Spark cluster with 4 worker nodes by executing the following command:

`scc_spark_deploy.sh -N 1 -t 01:00:00 --qos=normal`[5]

This creates a cluster as a so-called batch job on the HPC cluster, which contains a Spark cluster. It runs for on hour on one node. You can experiment with the size of the cluster, but your job may end up in a queue waiting. You can always check the status of your jobs using

`squeue -u hpc-ss23-61` (remember to replace the user's name with yours)

```
gwdu102:20 12:15:16 ~ > squeue -u hpc-ws21-50
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          10852503       fat    Spark hpc-ws21  R       0:30      4 dfa[003-006]
gwdu102:20 12:15:17 ~ >
```

The first node in the `NODELIST` is the Spark master node. Please note that the node list only exists if the job is actually running (`state` is `R`). In case the state is `PD`, the job waits for resources to be allocated and you cannot continue at the moment[6]. You can execute a program directly on the master node. In our case, we run an approximation of Pi using the program `pi2.py` available from Stud.IP. It approximates Pi by using the Monte-Carlo method. You can run the program like this:

`spark-submit --master spark://dfa003:7077 pi2.py` (Please select the Spark master node of your cluster instead of `dfa003`).

The execution of the program creates a lot of output, but near to the end, you see the approximation for Pi and the time it took to execute the calculation. Please provide a screenshot of the relevant part

---

[3] See e.g. https://www.man7.org/linux/man-pages/man1/scp.1.html regarding the syntax of `scp`.
[4] More information is available here: https://docs.gwdg.de/doku.php?id=en:services:application_services:high_performance_computing:software:spark.
[5] qos is the parameter for the Queue to choose. You can either use normal or 2h.
[6] More Info on the `squeue` command can be found here https://slurm.schedmd.com/squeue.html.

as the result of the assignment. If interested, experiment with the Spark cluster setting and the number of iterations in the program (N = 10000000 is set as default).

NOTE: In case you want to kill a job, please execute:

`scancel --signal=KILL 10852503` (with `10852500` being the `JOBID`, which needs to be replaced by the one you want to stop).

Implementing the Assignments on the NHR system

Here are the things to change logging into the NHR system.

Login

Please log into `glogin.hpc.gwdg.de`. Details can be found here https://docs.hpc.gwdg.de/getting_started/connecting/configuring_ssh/index.html and here https://docs.hpc.gwdg.de/getting_started/connecting/login_nodes_and_example_commands/index.html. You should end up in your home directory on the cluster. You can check the name with the command `pwd`.

Changes Assignment I

To load the Spark module use the following command:

```
module load spark/3.5.1
```

Creating the interactive shell works as before (see original Assignment I).

Changes Assignment II

To copy the files to the NHR system, please use

```
scp -i ~/.ssh/u12345_id_rsa melville.txt
u12345@glogin.hpc.gwdg.de:/user/username/u12345/
```

Replace the file name of the ssh key (`u12345_id_rsa`) as well as the username (`u123456`) with your respective names. `/user/username/u12345/` is to be replaced with the name of your home directory (like `/user/philipp.wieder/u12345`). The same needs to be done for the path used in the Scala expression shown in the original Assignment II.

Changes Assignment III

Cluster creation (adjust N for nodes to test different settings):

```
nhr_spark_deploy.sh -N 4 -t 01:00:00 --qos=normal
```

Job submission:

```
spark-submit --master spark://dfa003:7077 pi.py
```

Adjust iterations to test performance and precision of the approximation.