CSE427S

HOMEWORK 9

M. Neumann

Due: THU 7 NOV 2019 4PM

Getting Started

Update your SVN repository.

When needed, you will find additional materials for *homework x* in the folder `hwx`. So, for the current assignment the folder is `hw9`.

SUBMISSION INSTRUCTIONS

WRITTEN:

- all written work needs to be submitted electronically in *pdf format*[1] via GRADESCOPE
- provide the following information on *every* page of your `pdf` file:
  - name
  - student ID
- start every problem on a *new page*
- **FOR GROUPS**: make a **group submission** on GRADESCOPE and provide names and student IDs for **all group members** on every page of your `pdf` file.

CODE:

- code needs to be submitted electronically in a *single .zip file* via GRADESCOPE (detailed submission instructions are provided under HOMEWORK 9 CODE below)
- make sure to always use the required *file name(s)* and *submission format(s)*
- **comment your code** to receive maximum credit

Preparation

1. Complete **Lab 8: Data Ingest with FLUME** as you will need the data.
2. If you have not yet executed this command, run it in the command line:

```
$ ~/training_materials/dev1/scripts/training_setup_dev1.sh
```

[1] Please, **type your solutions** or use **clear hand-writing**. If we cannot read your answer, we cannot give you credit nor will we be able to meet any regrade requests concerning your writing.

Problem 1: Transform Data using SPARK (15%)

In this problem you will load, filter, and display the web server logs in a formatted way. You can do this problem using the spark shell. After completing **Lab 8 (FLUME)** you will find the `weblogs` data in HDFS under:

```
/loudacre/weblogs
```

(a) Load the data into an RDD and use RDD transformations to create a dataset consisting of the IP address and corresponding user ID for each request for an HTML file. Display 10 entries of the data in the form ipaddress/userid, e.g.:

```
165.32.101.206/8
182.4.148.56/173
100.219.90.44/102
246.241.6.175/45395
175.223.172.207/4115
...
```

HINT: you can use **for-loops** in the SPARK shell.

Add the output to your written answer.

Include **all required pyspark or Scala commands** in your written answer in `hw9.pdf`.

Problem 2: Joining Datasets with SPARK (40%)

In this problem you will explore the Loudacre web server log files using key-value Pair RDDs. After adding some new data containing Loudacre user account information to HDFS you will join the two RDDs.

After completing **Lab 8 (FLUME)** you will find the `weblogs` data in HDFS under:

```
/loudacre/weblogs
```

Note on Development and Debugging

- You will be reducing and joining large datasets, which can take a lot of time. You may wish to develop and test your spark commands using a **smaller dataset**, consisting of only a few of the web log files, rather than all of them. One way of getting a sample is to use a wildcard while loading the data as for instance `textFile("/loudacre/weblogs/*6")` would include only filenames ending with the digit 6.
- Use the full dataset to produce the results to be included in your submission!
- If you cannot run the commands in the spark shell *locally*, launch the spark shell to execute in *client mode* on the pseudo-cluster.

You can do this problem using the **spark shell**. Include all required commands in your written answers.

(a) Load the `weblogs` data into an RDD. How many records are there in the weblogs data? Filter the RDD to only contain HTML requests. How many records for HTML requests are in the weblogs data?

(b) Using the filtered RDD, create a Pair RDD with the user ID as the key, and the integer 1 as the value. The user ID is the third field in each line. Then, sum the values for each user ID to get the hit count for each user. Your RDD data will be similar to:

```
(userid,5)
(userid,7)
(userid,2)
...
```

(c) Determine how many users visited the site for each frequency. That is, determine how many users visited once, twice, three times and so on. How many users visited the site once, 7 times, and 12 times? Use the full weblogs data filtered by HTML requests only to produce this result.

Preparation: We will perform another *simulation process* to ingest data into HDFS similar to the one we did in the lab. Now, we will use SQOOP to get data from an RDMS and put it into HDFS . Check the **Lab 8 slides** for a basic introduction to SQOOP .

Import the accounts data from the SQL database `loudacre` into the `/loudacre/accounts` folder in HDFS using SQOOP by executing the following command in the terminal:

```
$ sqoop import \
 --connect jdbc:mysql://localhost/loudacre \
 --username root --password cloudera \
 --table accounts \
 --target-dir /loudacre/accounts \
 --null-non-string '\\N'
```

(d) Create an RDD based on the accounts data consisting of key/value-array pairs: `(userid, [values...])`. The results will look sth like this:

```
(userid1,[userid1,2008-11-24 10:04:08,\N,Cheryl,West,
4905 Olive Street,San Francisco,CA,...])

(userid2,[ userid2,2008-11-23 14:05:07,\N,Elizabeth,Kerns,
4703 Eva Pearl Street,Richmond,CA,...])

(userid3,[userid3,2008-11-02 17:12:12,2013-07-18 16:42:36,Melissa,
Roman,3539 James Martin Circle,Oakland,CA,...])
```

(e) Join this Pair RDD with the set of user-id/hit-count pairs calculated in part (b). The result will look something like this:

```
(userid1,([userid1,2008-11-24 10:04:08,\N,Cheryl,West,
4905 Olive Street,San Francisco,CA,...],4))

(userid2,([ userid2,2008-11-23 14:05:07,\N,Elizabeth,Kerns,
4703 Eva Pearl Street,Richmond,CA,...],8))

(userid3,([userid3,2008-11-02 17:12:12,2013-07-18 16:42:36,Melissa,
Roman,3539 James Martin Circle,Oakland,CA,...],1))
```

Display the user ID, hit count, and first name (3rd value) and last name (4th value) for the first 5 elements. The result for the above input would look like this:

```
userid1 4 Cheryl West
userid2 8 Elizabeth Kerns
userid3 1 Melissa Roman
```

HINT: you can use **for-loops** in the SPARK shell.

Add the spark commands and output to your written answer.

(f) Draw the DAG (*directed acyclic graph*) of this spark workflow using your transformation name (such as `textFile`, `map`, etc.) and the problem part (such as (a), (b), etc.) as nodes. Indicate which steps/nodes need to re-partition the data.

Include **all required pyspark or Scala commands** in your written answer in `hw9.pdf`.

Problem 3: Write a SPARK Program (30%)

In this problem you will write a simple SPARK program in Python or Scala that counts the number of JPG requests in the web log data. You will find the `weblogs` data in HDFS under:

```
/loudacre/weblogs
```

Note that for this problem you will not need the interactive SPARK shell. So, before running your program, be sure to **exit** from the SPARK shell.

Text editor: You may use any editor you like. If you don't have a preference simply use gedit, which includes language-specific support (syntax highlighting) for Python and Scala.

Stub files: The stub files for this problem are provided in the `hw9/countjpg_stubs` folder in your SVN repository.

Scala *only*:

To compile your Scala program, use the following folder:

```
~/training_materials/dev1/exercises/spark-application/countjpgs
```

Copy your Scala program into the `src/main/scala` folder in this directory. Then, run the following command:

```
$ mvn package
```

This invokes Apache Maven to compile your program with all required dependencies. You should now find a folder called target and within this folder you should find your `.jar` file. Trouble-shooting: make sure you have a file called `pom.xml` in your `countjpg` folder.

IMPLEMENTATION SPECIFICATIONS:

- Your implementation goes into the `CountJPGs` program.
- The input data location (path and/or file name) should be passed into the program as an argument.
- Your implementation should load the file passed into the program, count the number of JPG requests, and display the count.

(a) Run your SPARK application on the `weblogs` data locally. Provide the command to submit the job and report the number of JPG requests in your written answer. Where is the Driver program executed? Where does the processing happen? Where is the result stored?

(b) Run your SPARK application on the `weblogs` data on the cluster using client mode. Provide the command to submit the job in your answer. Where is the Driver program executed? Where does the processing happen? Where is the result stored?

(c) How many *stages* and *tasks* were executed for your `CountJPGs` job? You can retrieve this information from the Yarn Resource Manager Web UI (`http://localhost:8088/jobbrowser`). Find your application and go to the very right to click on History. If you don't see a table that shows Stages and Tasks, then your SPARK event logging isn't configured correctly. Check the VM trouble-shooting page to find instructions on how to fix this.

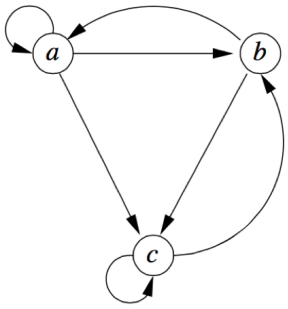
(d) We could also run the job in cluster mode. Provide the command to submit the job in your answer. Where would the Driver program be executed? Where does the processing happen? Where would the result be stored?

Problem 4: PageRank Data Representation (15%)

In this problem you will investigate the data representation of the webgraph used in the PageRank algorithm. Consider the following two data representations:

- raw input data which has a (`source-page` – `destination-page`) format or
- `links` which has a (`source-page` – list of destination-pages) format

(a) Provide both data representations for the following example graph:



(b) Which data representation is more storage efficient? State the storage requirements for both representations for a general webgraph in terms of the number of pages n and the number of links m.

continue to next page...

CSE427S

HOMEWORK 9 CODE

M. Neumann

Due: THU 7 NOV 2019 4PM

SUBMISSION INSTRUCTIONS

CODE:

- create a **single zip file** named `hw9_YourName.zip` (or `hw9_YourName_YouPartnersName.zip` for groups) including all files listed below

  Example file names:

  `hw9_MarionNeumann.zip`
  or for teams:
  `hw9_MarionNeumann_AnnaLee.zip`
  or
  `hw9_AnnaLee_MarionNeumann.zip`

- submit the *zip file* to the `hw9` CODE assignment in GRADESCOPE
- your code will be **autograded** for *correctness* and **manually inspected** for *comments*

Problem 5: Write a SPARK Program (10%)

(a) Submit your `CountJPGs.py` or `CountJPGs.scala` program. Add your *own* comments to your code to receive maximum credit.

Files to be added to your `zip` file:

```
CountJPGs.*
```

Reflection (Bonus Problem for 5% up to a max. of 100%)

Reflect on your homework experience! Write a paragraph of at least 50 words to express your experiences and feelings when working on this assignment. Answer at least 2 of the following questions:

- What did you like/dislike about the assignment and why?
- What is the most important thing you learned and why do you think so?
- What surprised you, and why?
- Assuming you could start over again (with working on the assignment), what would you do differently and why?

Do not include/copy and past the questions into your reflection!

Submission Instructions

Store your reflection in the `hw9_reflection.txt` file provided in the `hw9`folder in your SVN repository and commit it.

This file should only include the reflection, **no other personal information** such as name, wustlkey, etc. reflections are not graded based on the content, but solely for completion.

To submit your reflection `cd` into the `hw9` folder and run:

```
$ svn commit -m 'hw9 reflection submission' .
```

Take 1 minute to provide an overall **star rating** for this homework.
Submit it via this link: `https://wustl.az1.qualtrics.com/jfe/form/SV_bIRKP1xobpO8yIB`.

Grading - no group work!

You can only earn bonus points if you write a *meaningful* **reflection** of **at least 50 words** answering at least 2 of the prompted questions and provide the corresponding **star rating**. You will **not** be graded on what your reflection says and the number of stars you assign, but rather solely the completion of it.
Bonus points are given to the **owner of the repository only**. No group work!.