CSE427S

HOMEWORK 7

M. Neumann

Due: THU 25 OCT 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 `hw7`.

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 7 CODE below)
- make sure to always use the required *file name(s)* and *submission format(s)*
- **comment your code** to receive maximum credit

Problem 1: Top-10-List of Most Popular Movies (10%)

In this problem we will analyze movie rating data and compute a list of the most popular movies. This is essentially LAB 6!

[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.

IMPLEMENTATION SPECIFICATIONS:

- **Find more detailed information on how to get the data and implementation specifications in the lab instructions.** Use the filenames specified in the lab instructions.
- Write a MAPREDUCE program to compute the N most popular movies for the data provided in the `TrainingRatings.txt` file.
- Use the **sum of the ratings** as measure for popularity! N should be a parameter, that you can provide via the command line.
- Your program should output the sum of the ratings and the movie title for the top-N-list.
- Comment your code to receive maximum credit.
- Programs to implement and submit (cf. HOMEWORK `hw7CODE` below): `AggregateRatings.java`, `SumReducer.java`, `AggregateRatingsMapper.java`, `TopNReducer.java`, `TopNMapper.java`, `TopNDriver.java`
- Use the following test input and output for a top-3-list for testing and debugging:

  Input:

  ```
  8,1148143,2.0
  8,1174811,5.0
  9,63493,5.0
  9,516722,4.0
  1,1232582,2.0
  5,1631874,4.0
  5,721546,4.0
  8,2035299,3.0
  5,826193,5.0
  8,1793777,4.0
  3,125713,3.0
  ```

  Output:

  ```
  14 What the #$*! Do We Know!?
  13 The Rise and Fall of ECW
  9 Class of Nuke 'Em High 2
  ```

(a) Run your implementation for $N = 15$ on the pseudo-cluster using the `TrainingRatings.txt` provided in the Netflix data. Add the result to your written answer.

Copyrights

The data used in this assignment is taken from Pedro Domingos' class on Data Mining/Machine Learning at University of Washington, 2012.

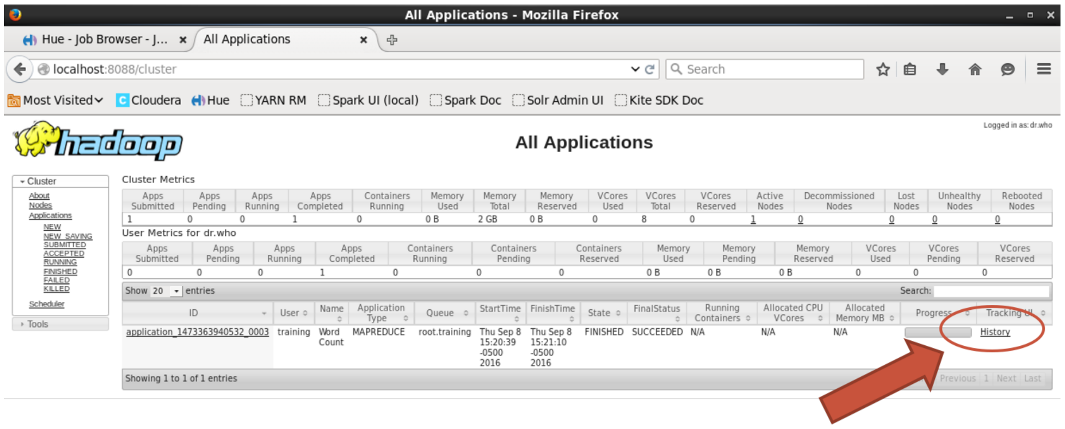
Problem 2: Combiner (20%)

For this problem you will add a Combiner to the first job of your Top-N-List MapReduce program and examine its effect on the job execution.

(a) Add a Combiner to the first job of your Top-N-List implementation from problem 1 (your Combiner should be a "*SumCombiner*"). Provide the respective line(s) of code you added in your driver to your written submission. Do **not** submit this version for Problem 4! Does the **result** of your job change compared to the original implementation without Combiner?

(b) Run this job on the `TrainingRatings.txt` data **twice**, once with Combiner (rename your job to "*Aggregation with Combiner*") and once without Combiner (name this job "*Aggregation no Combiner*") and observe the job execution statistics of both jobs.

To do so, open the **YARN Resource Manager Web UI** in a web browser (`http://localhost:8088/`). On this page you will find an overview of completed jobs. Find the ones with names "*Aggregation with Combiner*" and "*Aggregation no Combiner*" and open their **History** (on the very right). Then click on **Counters** (in the panel on the left).



All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1	0	0	1	0	0 B	2 GB	0 B	0	8	0	1	0	0	0	0

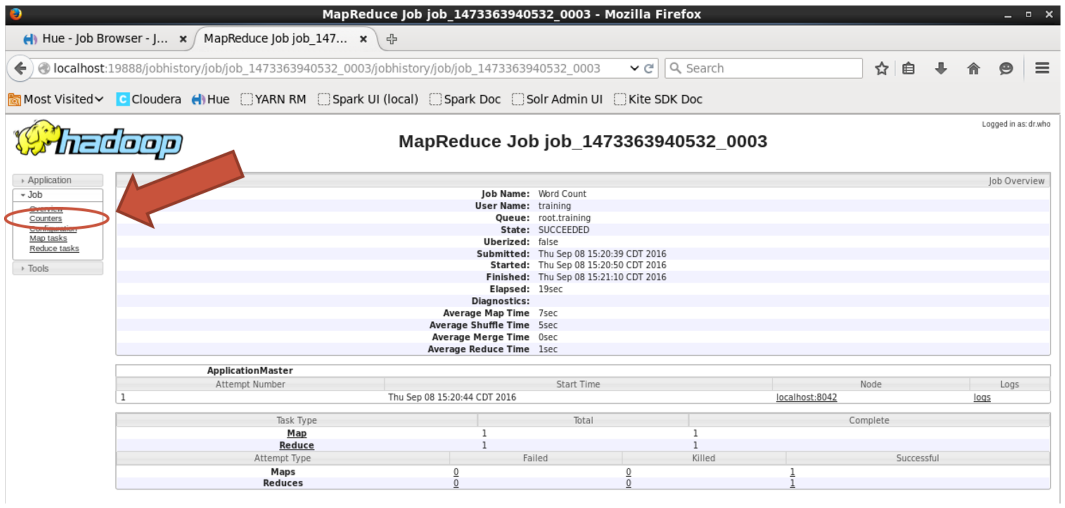
User Metrics for dr.who

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	VCores Used	VCores Pending	VCores Reserved
0	0	0	1	0	0	0	0 B	0 B	0 B	0	0	0

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Progress	Tracking UI
application_1473363940532_0003	training	Word Count	MAPREDUCE	root.training	Thu Sep 8 15:20:39 -0500 2016	Thu Sep 8 15:21:10 -0500 2016	FINISHED	SUCCEEDED	N/A	N/A	N/A		History

Showing 1 to 1 of 1 entries Previous 1 Next Last


MapReduce Job job_1473363940532_0003 - Mozilla Firefox

localhost:19888/jobhistory/job/job_1473363940532_0003/jobhistory/job/job_1473363940532_0003

Logged in as: dr.who

MapReduce Job job_1473363940532_0003

Job Overview

Job Name:	Word Count
User Name:	training
Queue:	root.training
State:	SUCCEEDED
Uberized:	false
Submitted:	Thu Sep 08 15:20:39 CDT 2016
Started:	Thu Sep 08 15:20:50 CDT 2016
Finished:	Thu Sep 08 15:21:10 CDT 2016
Elapsed:	19sec
Diagnostics:	
Average Map Time	7sec
Average Shuffle Time	5sec
Average Merge Time	0sec
Average Reduce Time	1sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Thu Sep 08 15:20:44 CDT 2016	localhost:8042	logs

Task Type	Total	Complete
Map	1	1
Reduce	1	1

Attempt Type	Failed	Killed	Successful
Maps	0	0	1
Reduces	0	0	1

Now, report and compare the total number of counters (last column). Do the following statistics (counters) change? Explain why or why not.

- FILE: `number of bytes read`
- FILE: `number of bytes written`
- HDFS: `number of bytes read`
- HDFS: `number of bytes written`

How many key-value pairs could be combined using the Combiner (HINT:look at those counters: `Combine input records` and `Combine output records`)?

(c) Why can you use the `SumReducer` as Combiner for this job? Give two resons.

(d) Come up with an example Reducer operation (which is **not** the mean/average computation discussed in the lecture) where you cannot use the Reducer as Combiner.

Problem 3: Computing Word Co-Occurrence (25%)

In the lecture we discussed co-occurrence based recommendations. A very similar Big data application is to compute the co-occurrence of words in text documents. These word co-occurrences, also called *bi-grams*, and their frequencies can be used as features in many text mining and natural language processing tasks such as document categorization or plagiarism detection.

You will implement a MAPREDUCE program that counts the words-pairs in `shakespeare` that occur right next to each other. For this application the order of the words actually matters, so you will have to adapt the pseudo code of job 2 in the RS3 slides accordingly. For instance, in the toy example below `(now,no)` is a different bi-gram than `(no,now)`.
Use the stubs provided in:

```
~/workspace/word_co-occurence/src/stubs/
```

IMPLEMENTATION SPECIFICATIONS:

- Implement the word co-occurrence MAPREDUCE program using the *pairs* approach discussed for job 2 of the co-occurrence based recommendation system.
- Note that in our use-case we do not look at data from different documents, hence, job 1 is not required.
- Before you start think about the Reducer you will need for this program. Reuse the *appropriate* existing program from a prior lab/hw assignment!
- Do **not** perform any preprocessing such as stemming or stop-word filtering.
- Comment your code to receive maximum credit.
- Programs to implement and submit (cf. HOMEWORK hw7CODE below): `WordCo.java`, `WordCoMapper.java`
- Use the following **test input** to test your implementation:

  ```
  No now is no now no time
  ```

  The output for this test input would be:

  ```
  is,no     1
  no,now    2
  no,time   1
  now,is    1
  now,no    1
  ```

(a) Once, everything works run your job on the `shakespeare` data. Check the output and provide the following statistics:

- How many different word pairs did your program produce?
- How often does the following word pairs occur:
  - the,lovers
  - loved,you
  - you,loved
  - verona,julia

HINT: the UNIX command **grep** will be useful to find those pairs in the output of your MAPREDUCE program.

(b) What are the following statistics for the job you just ran on the `shakespeare` input data:

- number of key-value pairs that are Mapper input
- number of key-value pairs that are Mapper output
- number of key-value pairs that are Reducer input

- number of key-value pairs that are Reducer output

What is the total/actual *communication cost* including the input and output of all tasks?

What is the *communication cost* using the simplified formula?

HINT: You can find the required statistics to compute the actual communication cost of your job from the HADOOP provided **job counters** in the YARN Resource Manager Web UI (cf. **Problem 2**).

(c) What is the drawback of merely counting the co-occurrence of words *directly* next to each other? Create two example sentences to illustrate the problem and describe the problem using your example.

(d) How does a *stripes* approach implementing word co-occurrence compare to your implementation?

Consider both simplified communication cost and worst-case memory requirements of the reduce() functions in your answer. Assume that the total number of lines in your input data is ℓ, the total number of words in your input is w and the number of different words is q (size of the vocabulary).

continue to next page...

CSE427S

HOMEWORK 7 CODE

M. Neumann

Due: THU 25 OCT 2019 4PM

SUBMISSION INSTRUCTIONS

CODE:

- create a **single zip file** named `hw7_YourName.zip` (or `hw7_YourName_YouPartnersName.zip` for groups) including all files listed below

  Example file names:

  `hw7_MarionNeumann.zip`
  or for teams:
  `hw7_MarionNeumann_AnnaLee.zip`
  or
  `hw7_AnnaLee_MarionNeumann.zip`

- submit the *zip file* to the `hw7` CODE assignment in GRADESCOPE
- your code will be **autograded** for *correctness* and **manually inspected** for *comments*

Problem 4: Top-10-List of Most Popular Movies (30%)

In this problem we will analyze movie rating data and compute a list of the most popular movies. This is essentially LAB 6! Find more detailed information on how to get the data and implementation specifications in the lab instructions. Use the filenames specified in the lab instructions.

(a) Submit your `AggregateRatings.java` program. If you started from a template, remove the provided comments and add your *own* comments to your code to receive maximum credit.

(b) Submit your `AggregateRatingsMapper.java` program. If you started from a template, remove the provided comments and add your *own* comments to your code to receive maximum credit.

(c) Submit your `TopNMapper.java` program. If you started from a template, remove the provided comments and add your *own* comments to your code to receive maximum credit.

(d) Submit your `TopNReducer.java` program. If you started from a template, remove the provided comments and add your *own* comments to your code to receive maximum credit.

(e) Submit your `TopNDriver.java` program. If you started from a template, remove the provided comments and add your *own* comments to your code to receive maximum credit.

Files to be added to your `zip` file:

```
AggregateRatings.java
SumReducer.java
AggregateRatingsMapper.java
TopNMapper.java
TopNReducer.java
TopNDriver.java
```

Problem 5: Computing Word Co-Occurrence (15%)

(a) Submit your `WordCo.java` program. Remove the provided comments and add your *own* comments to your code to receive maximum credit.

(b) Submit your `WordCoMapper.java` program. Remove the provided comments and add your *own* comments to your code to receive maximum credit.

Files to be added to your `zip` file:

```
WordCo.java
WordCoMapper.java
```

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 `hw7_reflection.txt` file provided in the `hw7`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 `hw7` folder and run:

```
$ svn commit -m 'hw7 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!.