CS6675/ CS4675

Homework Assignment 4

(Programming Category)

Student Name:______________________

Student Session: cs6675 or CS4675 (circle one)

You are given three types of programming problems in the HW4. You only need to choose one problem. Feel free to use any of your favorite programming language, Java, C, Perl, Python, Scala, etc..

Due Date: Midnight on Friday of Week 9 (March 7, with grace extension to 9am on Saturday of March 8)

Problem 1. Learning Mobile Trace Generation and Visualization

This problem is designed for students who are interested in mobile location based services. It contains 4 tasks.

Task1: Mobile Trace Generation
You are asked to generate a mobile trace dataset for 10 mobile objects and collect the trace for these 10 objects moving for certain period of time (say 10 minutes) using GTMobiSIM.
https://code.google.com/archive/p/gt-mobisim/wikis/QuickStart.wiki
https://code.google.com/archive/p/gt-mobisim/

Hint: You need to choose a region of the map or use one from the GTMobiSIM.

Task 2: Visualize the trace you generate for the 10 mobile objects. Report what you observe. Take a screen shot.

Task 3: Repeat Task 1 for varying N from 10, 100, 1000, 10,000 on the same size map region you choose in the task 1, and compare the trace generation time for these four traces of increasing sizes and the memory size of these four traces.

Task 4: Perform 1-2 Stress Tests on GTMobiSIM. For example, try to generate a trace of large N moving objects or running much longer time such as half hour for larger N, you may end up with out of memory error. Then try to find out you're your available memory allocated to run GTMobiSIM, what is the feasible N value? Given a fixed large N, how long can you run the trace generator, which will not cause out of memory error?
Discuss your stress test experiments and your observations.

Problem 2. Learning about Web Server Technology

You are asked to download your favorite Web server software from public domain, such as Apache TomCat (https://tomcat.apache.org/download-80.cgi) and install it on your laptop. Perform the following tasks:

1. Learn how to configure your server using a port number to create your own web service, such as hosting 5 Web pages containing a set of images or photos with 5 different designs, such as 4 photos per page, 2 photos per page, etc. You may choose your own web service of similar functionality.

2. Test your Web server from a remote client, say your cellphone, and report the result by taking a screen shot.

3. Test your Web server performance in terms of latency and throughput by sending a large number of requests and record the service time in terms of latency (per request time) and throughput (the number of requests served per time unit (minute or second).

4. Create stress tests to your Web server to see if you can saturate your Web server. Report your design and your experience.

5. Create a Web access cache and measure your request traffics in step 3 and step 4 and report the performance enhancement you observe in terms of latency and throughput.

6. Summarize your experience with this Web server and the toy workloads you created and discuss three things that you learned about the Web server through this hand on experience with a toy workload.

Problem 3. Duplicate Detection with PySpark

Discovery of multiple representations of the same real- world object.

Problems:

- Representations are not identical (Fuzzy Duplicates)
- Data sets are large: quadratic complexity if comparing every pair of records

- Similarity measures:
  – Domain-dependent vs. domain independent solutions
    - Avoid comparisons by partitioning
    - Maximize Parallel computing

Hint: Slide based on a lecture by Felix Nauman (University of Potsdam): https://hpi.de/fileadmin/user_upload/fachgebiete/naumann/folien/SS13/DPDC/DPDC_11_Duplicate_Detection.pdf

Datasets:

Download 20~50 documents from the Web or Wikipiedia that are on similar subject areas and thus contain duplicate terms. Create another 50% (10~25) documents from your download set to create duplicates.

Now you have two datasets:

(1) 20 or more documents downloaded on the same subject and thus will container duplicate words/terms (small)
(2) 40 or more documents and 50% were generated from the downloaded collection by randomly removing some paragraphs. Thus you have more duplicates.

The programming problem consists of 4 parts:

Part 1: Write a pyspark code to determine the 1,000 most popular words in the document collection provided. Please ensure that your code removes any special symbols, converts everything to lower case (and if possible: remove stop words, destemming, etc.). Stop word removal can be done either by using some nltk or creating your own list of words to be removed (e.g. and,or ,the, it,...)

Part 2:

(2.1) Write a pyspark code to create an inverted index for the 1,000 words determined in Part 1. The inverted index is supposed to be of the form

term1: doc1:weight$_{1\_1}$,doc2:weight$_{2\_1}$,doc3:weight$_{3\_1}$,...
term2: doc1:weight$_{1\_2}$,doc2:weight$_{2\_2}$,doc3:weight$_{3\_2}$,... ...

where weight$_{x\_y}$ is the no. of occurrences of term-x in document y / total number of words in document y

Hint: revisit the MapReduce lecture for the inverted index and acknowledge any open source MapReduce code if you decide to use one of them.

(2.2) Measure the execution time of the code for the large data set for three cluster sizes (e.g., 5, 10 and 15 executors)

Part 3:

(3.1) Write a pyspark code to calculate the similarity matrix Sim with each entry of Sim being

Sim(docx, docy) = $t \varepsilon V (weight_{t\_docx} \times weight_{t\_docy})$

Here V is the vocabulary (determined/obtained in part 1) and the weights having been determined and obtained in part 2

(3.2) Measure the execution time for the large data set for three cluster sizes (e.g., 5, 10 and 15 executors)

Notes: See the following paper for the full algorithm:
Tamer Elsayed, Jimmy Lin, and Douglas W. Oard. 2008. "*Pairwise document similarity in large collections with MapReduce*"
https://www.aclweb.org/anthology/P/P08/P08-2067.pdf

Part4: Provide a list of the 10 most similar (or identical) pair of documents from each of the two data sets.

Documentation.
The Documentation should contain

- (Brief) Problem description
- Solution strategy
- Description of how to run your code
- Results section
  (1) Description of resources used
  (2) Description of measurements performed
  (3) Results (graphs/tables + findings)

The document should **NOT** contain

- Replication of the entire source code – that's why you have to deliver the sources
- Screen shots of every single measurement you made
  - Actually, 1-2 screen shots only if they are illustrative to your innovative design/optimization.
- The output files