

CSCI312 Big Data Management
Singapore 2022-2
Assignment 1
Published on 02 April 2022

Scope

The objectives of Assignment 1 include implementation of HDFS applications, implementation of simple MapReduce applications, and describing an implementation of complex MapReduce application.

This assignment is due on **Saturday, 23 April 2021, 9:00pm** (sharp) Singaporean Time (SGT).

This assignment is worth **10%** of the total evaluation in the subject.

The assignment consists of 4 tasks and specification of each task starts from a new page.

Only electronic submission through Moodle at:

<https://moodle.uowplatform.edu.au/login/index.php>

will be accepted. A submission procedure is explained at the end of Assignment 1 specification.

A policy regarding late submissions is included in the subject outline.

Only one submission of Assignment 1 is allowed and only one submission per student is accepted.

A submission marked by Moodle as "late" is always treated as a late submission no matter how many seconds it is late.

A submission that contains an incorrect file attached is treated as a correct submission with all consequences coming from the evaluation of the file attached.

All files left on Moodle in a state "Draft (not submitted) " will not be evaluated.

A submission of compressed files (zipped, gzipped, rared, tared, 7-zipped, lhzed, ... etc) is not allowed. The compressed files will not be evaluated.

An implementation that does not compile well due to one or more syntactical and/or run time errors scores no marks.

The first assignment is an **individual assignment** and it is expected that all its tasks will be solved **individually without any cooperation** with the other students. However, it is

allowed to declare in the submission comments that a particular component or task of this assignment has been implemented in cooperation with another student. In such a case evaluation of a task or component may be shared with another student. In all other cases plagiarism will result in a **FAIL** grade being recorded for entire assignment. If you have any doubts, questions, etc. please consult your lecturer or tutor during laboratory/tutorial classes or over e-mail.

Task 1 (2 marks)**Implementation of HDFS application**

This task is based on the source code included in a presentation 04 HDFS Interfaces.

Implement in Java HDFS application, that can be used to move a file from one location in HDFS into another location in HDFS.

The application must have the following two parameters.

- (1) A path to and a name of file to be moved from.
- (2) A path to and a new name of file to be moved to.

Perform the following steps.

- (1) Implement the application and save its source code in `.java` file. A name of file is up to you.
- (2) Compile the Java source code and create a `jar` file.
- (3) Upload to HDFS a small text file for the purpose of future testing. A name and location of the file in HDFS is up to you.
- (4) Use Hadoop to process your application that moves a file on HDFS from one location to the other.
- (5) Use Hadoop to provide an evidence that the file earlier uploaded to HDFS has been successfully moved.

Deliverables

A file `solution1.java` with a source code of the application that moves a file in HDFS. A file `solution1.pdf` that contains the contents of Terminal window with a report from compilation, creation of jar file, uploading to HDFS two small files for testing, processing of the application, and an evidence that two files uploaded into HDFS has been successful merges in one file in HDFS.

Task 2 (3 marks)

Implementation of MapReduce application

Assume, that The Bureau of Meteorology records total yearly rainfall in a number of cities located in different states. The measurements are recorded in a text file, that contains data from a period of the last year.

For example, a sample file with the recorded amounts of rainfall could be the following. The first column contains a name of a state, the second column contains a name of a city in a state and the third column contains total rainfall depth per year measured in mm.

Queensland	Gold Coast	25
Victoria	Melbourne	125
Victoria	Geelong	90
Victoria	Wodonga	10
NSW	Lismore	900
Queensland	Brisbane	50
South Australia	Adelaide	300
Western Australia	Perth	200
Western Australia	Albany	200
Western Australia	Broome	10

Your task is to implement a MapReduce application, that finds the total rainfall in each state, the largest rainfall in one location in each state and the smallest rainfall in one location in each state.

For example, your application should generate the following outputs when processing data listed above.

Queensland	75	50	25
Victoria	225	125	10
NSW	900	900	900
South Australia	300	300	300
Western Australia	410	200	10

An input file with the speed measurements must include 10 lines listed above and it must contain at least 10 other measurements. All additional measurements are up to you.

Save your solution in a file `solution2.java`.

When ready, compile, create `jar` file, and process your application. Display the results created by the application. Next, list your input file with the speed measurements. When finished, Copy and Paste the messages from a Terminal screen into a file `solution2.pdf`.

Deliverables

A file `solution2.java` with a source code of the application that implement the functionality of `SELECT` statement given above. A file `solution2.pdf` with a report

from compilation, creating `jar` file, processing, displaying the results of processing `solution2.java`, and listing of your input file with the rainfall measurements.

Task 3 (2 marks)

Implementation of MapReduce application

Consider a classical MapReduce application that counts the total number of occurrences of words in a given text. For example, look at `WordCount` application available in a file `WordCount.java` in Laboratory 2.

Assume the following classification of words depending on the length of each word.

```
very short: 1 <= length <= 3
short:      4 <= length <= 5
medium:     6 <= length <= 8
long:       9 <= length <= 12
X long:    13 <= length <= 15
XX long:   16 <= length
```

Extend Java code of the application such that it counts in a given text the total number of words in each category. For example, distribution of words in a text that consists of 90 words could be the following.

```
X short:      10 words
short:        15 words
medium:       35 words
long:         20 words
X long:       10 words
XX long:       0 words
```

Save your solution in a file `solution3.java`.

When ready, compile, create `jar` file, and process your application. To test your application, you can use a file `sales.txt` included in a folder with a specification of Exercise 2. Display the results created by the application. When finished, Copy and Paste the messages from a Terminal screen into a file `solution3.pdf`.

Deliverables

A file `solution3.java` with a source code of the application that implement the functionality of `SELECT` statement given above. A file `solution3.pdf` with a report from compilation, creating `jar` file, processing, and displaying the results of processing `solution3.java`.

Task 4 (3 marks)

Describing MapReduce implementation

Assume, that a text file `crime-stories.txt` contain the texts of large number of crime stories. Assume, that the file is formatted such that one statement is located in one line of the text file.

Assume, that a text file `patterns.txt` contains the text patterns, for example regular expressions. Assume, that the file is formatted, such that one pattern is located in one line of the text file.

To simplify the problem, assume that all text patterns in a file `patterns.txt` are different.

Finally, assume that a function `match(text-line, text-pattern)` returns true when `text-line` matches a pattern `text-pattern`. Otherwise, the function returns false.

Your task is to explain how to implement a MapReduce application, that for each text pattern in a file `patterns.txt` finds the total number of statements in a file `crime-stories.txt` that match the pattern.

You must specify the parameters (if any) of your application and the key-value data in the input and output of the Map and Reduce stages

There is no need to write Java code, however, if you like it then it is all right to do so. The precise explanations in plain English or in a pseudocode will do. Please note, that if you decide to use pseudocode then your explanations must precisely explain what happens at each stage of Map-Reduce application.

Save your explanations in a file `solution4.pdf`. This task does not require you to write any code in Java. However, the comprehensive explanations related to all stages of data processing are expected. You are allowed to support your explanations with the fragments of pseudocode. Try to be as specific as it is possible.

Deliverables

A file `solution4.pdf` with the comprehensive explanations how would you implement in Java a MapReduce application that for each text pattern in a file `patterns.txt` finds the total number of statements in a file `crime-stories.txt` that match the pattern.

Submission of Assignment 1

Note, that you have only one submission. So, make it absolutely sure that you submit the correct files with the correct contents. No other submission is possible !

Submit the files **solution1.java**, **solution1.pdf**, **solution2.java**, **solution2.pdf**, **solution3.java**, **solution3.pdf**, and **solution4.pdf** through Moodle in the following way:

- (1) Access Moodle at **<http://moodle.uowplatform.edu.au/>**
- (2) To login use a **Login** link located in the right upper corner the Web page or in the middle of the bottom of the Web page
- (3) When logged select a site **ISIT312 (SP222) Big Data Management**
- (4) Scroll down to a section **ASSESSMENT ITEMS (ASSIGNMENTS)**
- (5) Click at **In this place you can submit the outcomes of your work on the tasks included in Assignment 1** link.
- (6) Click at a button **Add Submission**
- (7) Move a file **solution1.java** into an area **You can drag and drop files here to add them**. You can also use a link **Add...**
- (8) Repeat step (7) for the remaining files **solution1.pdf**, **solution2.java**, **solution2.pdf**, **solution3.java**, **solution3.pdf**, and **solution4.pdf**
- (9) Click at a button **Save changes**
- (10) Click at a button **Submit assignment**
- (11) Click at the checkbox with a text attached: **By checking this box, I confirm that this submission is my own work, ...** in order to confirm authorship of your submission.
- (12) Click at a button **Continue**

End of specification