School of Computing & Information Technology
University of Wollongong

Session: **2, April 2021**
Lecturer: **Janusz R. Getta**

CSCI312 Big Data Management
Singapore 2021-2
Assignment 1

Published on 3 April 2021

Scope

The objectives of Assignment 1 include an analysis and processing of unknow HDFS application, implementation of MapReduce application without the Reduce phase, describing a MapReduce application, and implementation of MapReduce applications.

This assignment is due on **Saturday, 24 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)
Discovering the functionality and processing of unknow HDFS application.

Consider the available source code of Java application in a file `Unknown.java`.

Perform the following steps. Each step listed below is worth 1 mark.

(1) Read an analyse the contents of a file `Unknown.java` and discover what functionality is implemented by the unknown Java application. Insert your explanations as a comment located in the first few lines of the applications. "Few lines" means, that we expect the comprehensive explanations.

Next, insert the comments into a file `Unknown.java`, that explain step by step functionality of each line(s) such that while reading the comments it would it possible to easily understand how the application is implemented. Please note, that a comment like `"An expression on the right-hand side of assignment statement is computed and the results becomes a value of a variable on the left-hand side of assignment statement"` is an absolutely meaningless comment. The comments must explain the semantics of the Java statements in a context of the functionality of the application.

Assume, that solution will be evaluated by reading the comments one-by one and trying to understand how the application is implemented. Assume, that the comments will be read by someone who does not know how to write the computer programs in Java.

(2) At this point you should know what the functionality of the Java application is. Change a name of file `Unknown.java` and a name of application to a name `solution1.java`.

Few text files are zipped in a file `FewFiles.zip`. Compile a Java application `Unknown.java` and create a `jar` file, and use to it to process the files in `FewFiles.zip`.

Report all your Terminal commands and the output to demonstrate, that you successfully run the application. A simple way to create a report is to use Copy from Terminal Window and then Paste it into a text file and later on print it into a file `solution1.pdf`.

Deliverables
A file `solution1.java` with the explanations of the functionality of unknown application and with the comments explaining implementation of the application. A file `solution1.pdf` with the command use to compile and to process the application.

Task 2 (2 marks)
Implementation of MapReduce application without the Reduce phase

The application described in a document `Filter.java` has the functionality equivalent to the functionality of the following SQL statement.

```
SELECT key, value
FROM Sequence-of-key-value-pairs
WHERE value > given-value;
```

The application is a MapReduce application without the Reduce phase.

An objective of this task is to use the Java code included in a file `Filter.java` to implement a MapReduce application, that has the functionality the following `SELECT` statement.

```
SELECT key, value
FROM Sequence-of-key-value-pairs
WHERE value IN (value-1, value-2, value-3);
```

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. 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, and displaying the results of processing `solution2.java`.

Task 3 (3 marks)
Describing MapReduce implementation

Assume, that a file `customers.txt` has the following contents.

```
00001 James
00002 Harry
00003 Peter
00004 Jane
...   ...
```

The numbers in the first column represent a customer number and the names in the second column represent customer name.

Assume, that a file `orders.txt` has the following contents.

```
0000001 00001 34.5
0000002 00001 23.0
0000003 00002 123.0
0000004 00003 12.3
...     ...   ...
```

The numbers in the first column represent order number, the numbers in the second column represent customer number, and the number in the third column represent a total order value.

An objective of this task is to describe implementation of an application that finds all customers who have not submit any order yet.

Assume that both files have been loaded to HDFS. Explain would you implement Map phase and Reduce phase of MapReduce application, that lists all customers who have not submitted any orders yet.

Save you explanations in a file `solution3.pdf`. This task does not require you to write any code in Java. However, the comprehensive explanations on how to join the rows are expected. You are allowed to support your explanations with the fragments of pseudocode.

Deliverables
A file `solution3.pdf` with the comprehensive explanations on how to implement an application that finds all customers who have not submit any order yet.

Task 4 (3 marks)
Implementation of MapReduce application

Assume, that a bank records in a text file the withdrawals and deposits of certain amounts of money from the bank accounts. A single row in a file with the withdrawal/deposit records consists of an account number, a date when a withdrawal/deposit occurred, and an amount of money involved. Assume, that the withdrawals are represented by the negative numbers and the deposits are represent by the positive numbers and that each withdrawal/deposit modulo 50 = 0. All values in a single record are always separated with a single blank.

An objective of this task is to implement MapReduce application that finds the total amount of money deposited by each customer per year. For example, if a sample file with the withdrawals and deposits contains the following lines

```
1234567 12-DEC-2019 200
1234567 15-DEC-2019 50
9876543 25-JUL-2018 150
9876543 12-FEB-2018 -50
9876543 01-JAN-2019 150
1234567 21-OCT-2020 -250
9876543 22-OCT-2019 300
```

then your application supposed to produce the following outputs.

```
1234567 2019 250
9876543 2018 150
9876543 2019 450
```

The order of the lines listed above is up to you.

Perform the following steps.

Implement the application and save its source code in a file `solution4.java` file. A name of the file with the source code in a local file system is up to you.

Compile the Java source code and create a `jar` file.

Upload to a local file system a small file for the purpose of future testing. The file must contain the withdrawals and deposits and it must have an internal structure the same as it is explained and visualized above. A name of file and location of file in a local file system is up to you.

Use Hadoop to process your application that finds the total amount of money deposited by each customer per year.

Use Hadoop to list an input file with the withdrawals and deposits and the results produced by your application.

Deliverables

A file `solution4.java` with a source code of the application, that implements an application described above. A file `solution4.pdf` with a report from compilation, creating `jar` file, processing, and displaying the results of processing `solution4.java`.

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.pdf**, **solution4.java**, 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 (SP221) Big Data Management**
(4) Scroll down to a section **SUBMISSIONS**
(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.pdf**, **solution4.java**, 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