Big Data Processing
COSC 2637/2633
Assignment 3 – HDFS Monitoring via Spark Streaming

Assessment Type	– Individual assignment. – Submit online via Canvas → Assignment 3. – Marks awarded for meeting requirements as closely as possible. – Clarifications/updates may be made via announcements or relevant discussion forums.
Due Date	At 23:59, 22 Oct, 2023
Marks	25

Overview

Write Spark programs which gives you a chance to apply the essential components you learned in lectures and labs to understand the complexity of Spark programming.

Learning Outcomes

The key course learning outcomes are:

- CLO 1 - Model and implement efficient big data solutions for various application areas using appropriately selected algorithms and data structures.
- CLO 2 - Analyse methods and algorithms, to compare and evaluate them with respect to time and space requirements and make appropriate design choices when solving real-world problems.
- CLO 4 - Explain the Big Data Fundamentals, including the evolution of Big Data, the characteristics of Big Data and the challenges introduced.
- CLO 5 - Apply non-relational databases, the techniques for storing and processing large volumes of structured and unstructured data, as well as streaming data.
- CLO 6 - Apply the novel architectures and platforms introduced for Big data, i.e., Hadoop, MapReduce and Spark.

Task – Spark Streaming

Develop a spark streaming program with Scala to monitor a folder on HDFS in real-time such that any new file in the folder will be processed (the batch interval is 3 seconds).

The following three tasks are implemented in the same Scala object:

A. For each RDD of Dstream, count the word frequency and save the output on HDFS. (5 marks)

> Each line of text is multiple words separated by a single space. For each word,
> - it is retained **if it consists of characters only**, i.e., remove the word if it includes numbers, punctuations, special characters, etc., and
> - filter out (i.e., delete) the short words (i.e., < 3 characters).
>
> For example, `"I*** like pig latin I like hive too2 I don't like hive too."` should be parsed as `"like pig latin like hive like hive"`

B. For each RDD of Dstream, process each word in the same way, and then count the co-occurrence frequency of words (refer to week 4 for the explanation of co-occurrence frequency). The words are considered co-occurred if they are in the same line. If a word appears in a line more than once, each is simply treated as an independent word (do not deduplicate). Save the output on HDFS. (10 marks)

> For example, given the input "`like pig like hive`", the co-occurrence frequency is `(like pig, 2) (like like, 2) (like hive, 2) (pig like, 2) (pig hive,1) (hive like, 2) (hive pig, 1)`.

c. For the Dstream, process each word in the same way, and then count the co-occurrence frequency of words (the words are considered co-occurred if they are in the same line); save the output on HDFS. Note you are required to use `updateStateByKey` operation to continuously update the co-occurrence frequency of words with new information. (10 marks)

Format Requirements:

Failure to follow the requirements incurs up to an 8-mark penalty.

(a) Submit scala, jar, and readme files (nothing else!) in a single .zip file. The zip file should be named sxxxxx_BDP_A3.zip (replace sxxxxx with your student ID).
(b) In README, you must specify exactly how to run the jar in AWS EMR platform.
(c) Besides the zip file, organize the source code of tasks in a separate PDF file (copy & paste them into a text editor and then save it as a PDF file). Submit the PDF file (so, there are two submissions, one is the zip file, and the other is the PDF file). The PDF file is for Turnitin plagiarism check.

Functional Requirements:

Failure to follow the requirements incurs up to a 5-mark penalty.

(a) For each task, the output on HDFS should be named with a unique sequence number as a suffix. For example, taskA-001, taskA-002, taskB-001, taskB-002, taskC-001, taskC-002 (do not use other values such as the system time when the task is running as the suffix of the output names).
(b) For each task, if an RDD is empty, do not output.
(c) For each task, the batch interval is 3 seconds.
(d) If a checkpoint directory is needed, you must set it to the current working directory.
(e) Paths of input and output should be passed as arguments of the spark-submit command.
(f) You need to create a single Scala project including all three tasks so that they work on the same stream data.

Submission

Your assignment should follow the requirements below and be submitted via Canvas > Assignment 4.

Assessment declaration: when you submit work electronically, you agree to the assessment declaration: https://www.rmit.edu.au/students/student-essentials/assessment-and-exams/assessment/assessment-declaration

Academic integrity and plagiarism (standard warning)

Academic integrity is about the honest presentation of your academic work. It means acknowledging the work of others while developing your own insights, knowledge, and ideas. You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks, and/or ideas of others you have quoted (i.e., directly copied), summarized, paraphrased, discussed, or mentioned in your assessment through the appropriate referencing methods,
- Provide a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from Internet sites.

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.

RMIT University treats plagiarism as a very serious offense constituting misconduct. Plagiarism covers a variety of inappropriate behaviors, including:

- Failure to properly document a source.
- Copyright material from the internet or databases.
- Collusion between students.

For further information on our policies and procedures, please refer to
https://www.rmit.edu.au/students/student-essentials/rights-and-responsibilities/academic-integrity

Marking Guide

- Late submission results in a penalty of 10% marks for (up to) every 24 hours being late.
- If unexpected circumstances affect your ability to complete the assignment, you can apply for special consideration.
  - Requests for special consideration within 7*24 hours please can be via emailing the course coordinator directly with supporting evidence.
  - Request for special consideration of more than 7*24 hours must be via the University Special consideration: https://www.rmit.edu.au/students/student-essentials/assessment-and-exams/assessment/special-consideration.

Task a Implementation Correctness	0 marks cannot operate, no output, >1 major logic error in code	1 mark 1 major logic error in the code	2 marks >1 minor logic error in the code	3-4 marks output incorrect due to 1 minor logic error in the code	5 marks output correct and no code error
Task b Implementation Correctness	0 marks cannot operate, no output, >1 major logic error in code	1-3 marks 1 major logic error in the code	4-6 marks >1 minor logic error in the code	7-9 marks output incorrect due to non-logic errors or 1 minor logic error in the code	10 marks output correct and no code error
Task c Implementation Correctness	0 marks cannot operate, no output, >1 major logic error in code	1-3 marks 1 major logic error in the code	4-6 marks >1 minor logic error in the code	7-9 marks output incorrect due to non-logic errors or 1 minor logic error in the code	10 marks output correct and no code error
Functional requirement	Penalty applied for not following the functional requirements (detailed in the specification above)				
Format requirement	Penalty applied for not following the format requirements (detailed in the specification above)				