**Assignment2 Task 1 Report**

Jin Shuyuan A0162475B

1. **Description of my code**

* Specify input and output files location and create a SparkContext
* Read the stopwords file and cache the content
* Read the first input file, count the words not in the stopwords set, and cache the counts
* Read the second input file, count the words not in the stopwords set, and cache the counts
* Join the 2 count results, find common words count, sort by the frequency and take the top 15
* Save the final common words count result to txt file

1. **Structure of the files**

|  |  |
| --- | --- |
| Input | Task1\_data/task1-input1.txt  Task1\_data/task1-input2.txt  Task1\_data/stopwords.txt |
| Main source code | CommonWords/src/main/scala/CommonWords.scala |
| Output txt file | Task1\_data/output/ |
| Jar file built from sbt | CommonWords/target/scala-2.11/commonwords\_2.11-0.1.jar |

1. **Comparison between Hadoop and Spark**

|  |  |  |
| --- | --- | --- |
| **Comparison** | **Hadoop** | **Spark** |
| Programming Language | Java | Scala |
| Programming Paradigm | OOP | Functional programming |
| Code Length | Long | Short |
| Main Logic | Main logic of the implementations is the same. There are 4 stages: process stopwords, process input1, process input2 and find common words count. | |
| Libraries | We need to write our own map and reduce functions in the classes. Not many library functions provided. | The map and reduceByKey functions are provided. Lots of library functions to use. |
| Memory Management | After each stage, we need to specifically write to disk to save the intermediate result. | The RDDs are stored in memory which saves cost of writing intermediate results. And we can use cache() to speed up the computational process. |

1. **Comparison on runtime execution**

|  |  |  |
| --- | --- | --- |
| Comparisons | Hadoop | Spark |
| Result | Format: (count, word) | Format: (word, count) |
| Execution time |  |  |

The output from the 2 programs are the same.

As for execution time, I use System.currentTimeMillis() to record start and end time. When I run the code locally on my laptop, Spark program is roughly 50ms faster than Hadoop. If the file size gets larger and we run the program on cluster, Spark should be faster than Hadoop because Spark leverages memory to store the intermediate results to save the cost of writing to disk.

1. **Overall Pros and Cons**

|  |  |  |
| --- | --- | --- |
| **Pros and Cons** | **Hadoop** | **Spark** |
| Pros | * Resilient to failure because there are multiple copies of data * Higher security features | * Fast with in-memory operations * Simple to code |
| Cons | * I/O to disk is high * Bad with iterative process | * Limited memory space may be a problem |