Universitat de Lleida

Escola Politècnica Superior

Màster en Enginyeria Informàtica

Sistemes Intensius de Processament de Dades

MapReduce

Alex Ghenghiu
X4954469Q

Contents

1 Introduction **2**

2 Classes **3**
2.1 Cleaner 3
2.1.1 Running 3
2.2 TrendingTopics 3
2.2.1 Running 3
2.3 TopNPattern 4
2.3.1 Running 4
2.4 HashtagSentiment 4
2.4.1 Running 4
2.5 JobsJoint 4
2.5.1 Running 5

3 Performance **6**
3.1 tweets.txt 6
3.2 tweets2.txt 7
3.3 tweets2_es.txt 8

4 Conclusions **9**

Introduction

This practice aims to design and implement a small **MapReduce** application to calculate the trending topics on twitter and feelings associated with such hashtags. For its realization many of the concepts presented in this course will apply.

***This implementation only works with .txt input files**

Classes

Here will be described all the classes of the project. They can be found on the **Sources/src/** folder.

2.1 Cleaner

This class uses the **chain mapper** and **hadoop-predefined FieldSelectionMapper** in order to normalize and cleanup of the input data.

- All the tweets will be changed to lower-case.
- All the tweets with lacking of the processed fields (hashtags, text) will be removed.
- Remove all the tweets in any language different from Spanish.

2.1.1 Running

yarn jar Lexicon/Sources/out/artifacts/Lexicon/Lexicon.jar Cleaner hdfs_tweets_route hdfs_Cleaner_output

2.2 TrendingTopics

This class uses the **RegexMapper** in order to generate a list of all hashtags. Also, it uses a custom **Reducer** just to count the occurrences of each hashtag to determine the **trending topics**. Also, to improve the performance, **Combiners** and **Partitioners** are used.

2.2.1 Running

yarn jar Lexicon/Sources/out/artifacts/Lexicon/Lexicon.jar TrendingTopics hdfs_Cleaner_output hdfs_TrendingTopics_output num_partitioners

2.3 TopNPattern

This class uses **TopNPattern** to get the **N** trending topics.

2.3.1 Running

yarn jar Lexicon/Sources/out/artifacts/Lexicon/Lexicon.jar TopNPattern
N hdfs_TrendingTopics_output hdfs_TopNPattern_output

2.4 HashtagSentiment

This class uses **Distributed Cached** files to define a set of positive and negative words. For each hashtag will get a ratio of the positive and negative words in order to "determine" the global sentiment about that hashtag. In this case, the **mapper** will generate, for each hashtag in each tweet of the input file an **OwnWritable** data type. This is a custom **Writable** class composed by two **IntWritables**:

- Tweet's polarity (Tweet's positive words - Tweet's negative words).
- Tweet's length.

The **reducer** will receive this structure in order to get the global sentiment about that hashtag.

2.4.1 Running

yarn jar Lexicon/Sources/out/artifacts/Lexicon/Lexicon.jar HashtagSentiment
hdfs_negativeWords hdfs_positive_words hdfs_Cleaner_output hdfs_HashtagSentiment_output

2.5 JobsJoint

This class gathers all the previous MapReduce jobs. **Uses the compressed sequence file format for the partial results.**

2.5.1 Running

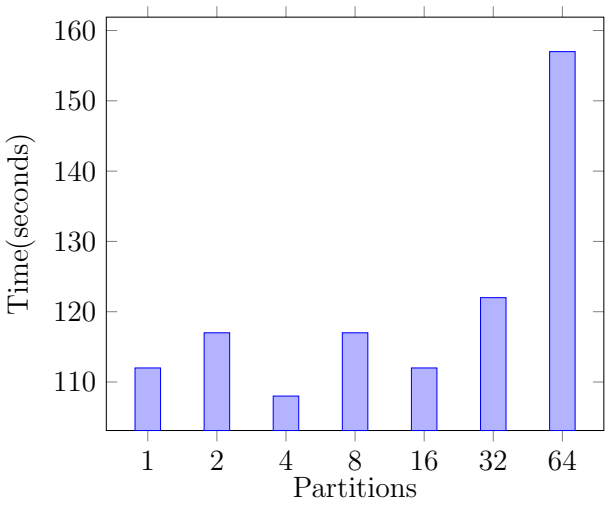
yarn jar Lexicon/Sources/out/artifacts/Lexicon/Lexicon.jar JobsJoint
hdfs_tweets_route num_partitioners N_TopNPattern hdfs_negativeWords hdfs_positive_words
hdfs_Cleaner_output hdfs_TrendingTopics_output
hdfs_TopNPattern_output hdfs_HashtagSentiment_output

Performance

Analysis of the final performance of your application, and the speed-up when increasing both the input dataset size and the number of nodes (map reduce tasks).

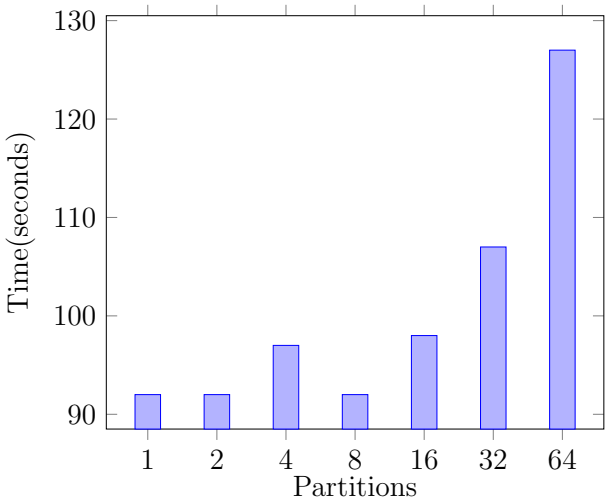
3.1 tweets.txt

Partitions	Time (seconds)	Speedup
1	112	1
2	117	0.96
4	108	1.03
8	117	0.96
16	112	1
32	122	0.91
64	157	0.71



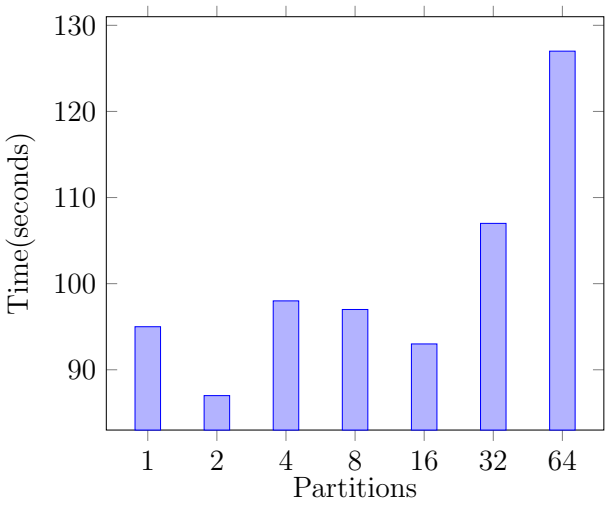
3.2 tweets2.txt

Partitions	Time (seconds)	Speedup
1	92	1
2	92	1
4	97	0.94
8	92	1
16	98	0.93
32	107	0.85
64	127	0.72



3.3 tweets2_es.txt

Partitions	Time (seconds)	Speedup
1	95	1
2	87	1.09
4	98	0.97
8	97	0.98
16	93	1.02
32	107	0.88
64	127	0.75



Conclusions

We have defined the *good tweets* as the tweets that accomplish the following rules:

- Tweets in Spanish language.
- Tweets with both hashtags and text.

So, as the input tweets file increases the number of *good tweets*, as expected, the computational time also gets increased. Then, to complete the study, several executions with different number of **partitions** were made. In all cases, between 1 and 16 partitions the execution time, more or less, remains the same. But, always with 32 and 64 the execution time raises indicating that the **I\O** processes are taking a lot of execution time.