A Case Study in

Computing Word Co-occurrence Matrices with MapReduce

Minh Phan
Student Number: S3335814
Big Data Processing
RMIT University

***Abstract*—This case study attempted to investigate the scalability efficiency of the two popular words co-occurrence algorithms: "pairs" and "stripes". The experiment was designed to compare the performance of the two algorithms in the context the following changing variables: data size, number of mappers, number of reducers and number of slave nodes employed. The metric of measuring efficiency is time. The case study concludes that "stripes" algorithm is more efficient in implementing words co-occurrence.**

Keywords—scalability, pairs, stripes, co-occurrence.

I. Introduction

The words co-occurrence matrices has multiple applications, especially in the age of growing web contents. The most popular design pattern approaches for constructing co-occurrence matrices are the "pairs" and the "stripes" approaches [1]. This case study was designed to investigate the question: which approach is better in terms of time efficiency. In addition, the experiment investigated how cost effective the two approaches are by comparing their efficiency with different numbers of slaves nodes. The experiment was conducted using the Common Crawl dataset from Amazon Web Service (AWS).

II. mapreduce

A. *Input*

Both of the approaches reads in the same input with the key as Text and the value as ArchiveReader, which help iterate through the files from the Common Crawl data.

B. *"Pairs" approach*

The "pairs" approach mappers output object MyPair as key and LongWritable as value. The reducers also have MyPair and LongWritable as "key-value" pairs of the outputs.

C. *"Stripes" approach*

The "Stripes" approach mappers output Text as key and MapWritable as value. The reducers al so have TextPair and MapWritable as "key-value" pairs of the outputs.

III. Methodology

The experiment was set up to compare the time efficiency of the two design patterns using the Common Crawl dataset. The algorithms were tested against variables including: data size, number of mappers, number of reducers and number of slave nodes. The purpose of the experiment was to determine which approach is better for scaling. The context of co-occurrence is the closest 2 words in the same sentence.

A. *Metric of efficiency measuring*

The total running time of each algorithm was recorded in seconds in order to compare the efficiency.

In addition, the cost effectiveness of each approaches is inferred from the performance with the increasing number of slave nodes.

B. *Testing environments*

- Data size: a range from 2 files (201.53 mb) to 10 files (1014 mb) of the Common Crawl data set was used. Due to the nature of the Common Crawl data file, each file will use its own mapper.
  - Case 1: 1 reducer and 1 master node.
  - Case 2: 3 reducers, 1master node and 8 slave nodes.
- Numbers of mappers: the number of mappers tested were 1, 2, 4, 6, 8 and 10; with the file size of 101.52mb and 1 reducer.
- Number of reducers: the number of reducers tested were 1, 2, 3, 6, 18 and 54; with the file size of 101.52mb and 1 mapper.
- Numbers of slave nodes: the numbers of slave nodes tested was 2, 4, 6, 8 and 10 nodes; with 2 cases
  - Case 1: 101.52 mb - 1 input file.
  - Case 2: 406.89 mb – 4 input files.

IV. Results and discussion

A. *Results*

The results of each testing categories were recorded in graphs forms; for details see Appendix 1.

- Data size: Figure 1.1 and 1.2

- Numbers of mappers: Figure 1.3
- Numbers of reducers: Figure 1.4
- Numbers of slave nodes: Figure 1.5, 1.6

B. *Discussion*

Overall, we can observe that the “stripe” approach performs better than the “pairs” approach in all the testing environments.

As the size of the inputs increases, the “stripes” approach has lower running time than the “pairs” one; this suggests that the “stripes” approach scales better in terms of data size (Figure 1.1). However, when they were running on just the master node (Figure 1.2), the “stripes” approach has a much smaller rate of increase in executing time than the “pairs” approach; this aligns with the conclusion drawn from the experiment proposed by J. Lin, and C, Dyer[1].

As the number of mappers was forced to increases, the input file was split into more parts and transmitted to different mappers; this was done with a small input file of 101.52 mb. In addition, the intermediate outputs were transferred back to 1 single reducer; this results in an increase in data transmitting across the network, hence decreasing the time efficiency in both approaches (Figure 1.2).

As the number of reducers was forced to increases, at first, the running times in both approaches decrease .However, as intermediate outputs were transferred from a single mapper to many reducers; this may result in an increase in data transmitting across the network, hence decreasing the time efficiency, in the later part of the plot(Figure 1.3).

We can see that both of the approaches do worse in term in time efficiency as the number of mappers and reducers increases; this is the result of insufficient use of resources. The data was forced to split into smaller blocks and transferred across the network, while a single mapper or reducer could process the data block; this violates the big data processing principle of moving the codes to the data and minimising transfer of the data[1].

As the number of slave nodes increase, we can observe a clear increase in efficiency in both approaches. The “stripes” approach appears to take less time in all the cases. Furthermore, the rate in increase in time efficiency, decrease in time, appears to plateau after a certain amount of nodes reached (Figure 1.4, 1.5).

In both cases, the “stripes” approach performs better. In the larger dataset, the “stripes” approach appears to achieve a plateau of time decrease after 4 slave nodes; while the “pairs” approach achieves a plateau after 8 slave nodes .

V. Conclusion

We have learnt that excessive data transfer will significantly reduce the time efficiency of any algorithms. We can also observe that, the employment of more resources have a positive impact in time efficiency. However, the impact of more resources becomes less significant after a certain amount of nodes.

The results of the experiments suggested that the ”stripes” approach has a better scalability efficiency in terms of time and cost.

The “stripes” approach takes less time to complete tasks in all cases. Besides, the “stripes” approach seems to reach this point relatively “maximum” of efficiency with less nodes than the “pairs” counterpart. In terms of costing, this showed that the “stripes” approach has better value for money, since less resources were required.

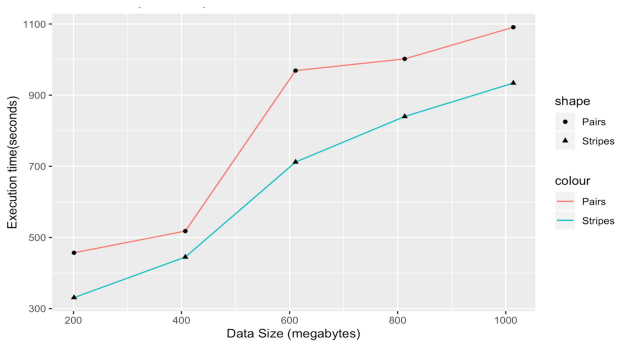
In conclusion, in order to maximise time efficiency and minimise running cost, we are required to understand not only which algorithm is better, but also how each algorithm thrives in practical context.

Reference

[1] J. Lin, and C, Dyer. *Data-intensive Text Processing with MapReduce*. San Rafael, Calif: Morgan & Claypool, 2010.

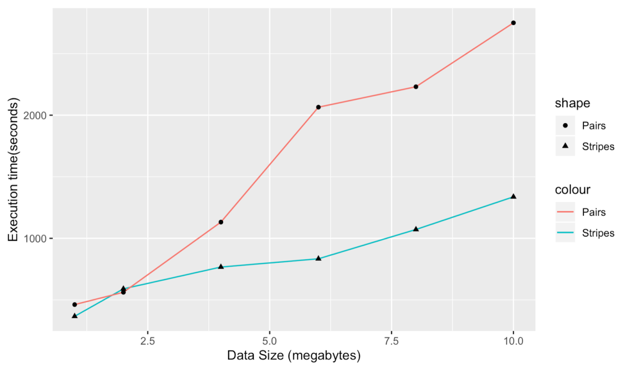
Appendix 1

Figure 1.1



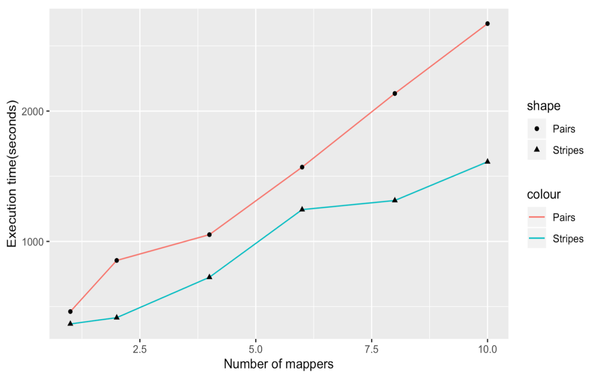
The effect of data size to time efficiency- 8 nodes

Figure 1.2



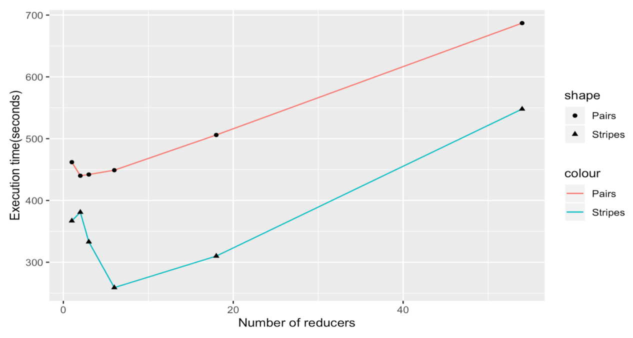
The effect of data size to time efficiency- 1 node

Figure 1.3



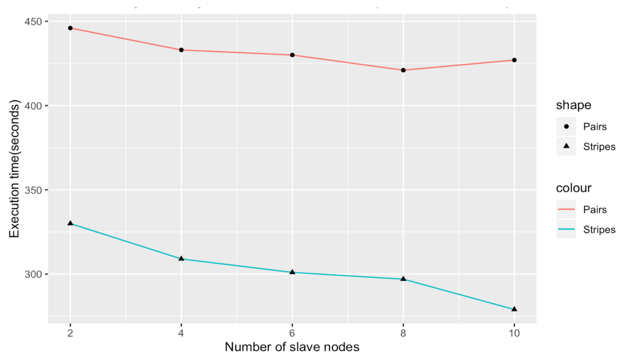
The effect of number of mappers to time efficiency

Figure 1.4



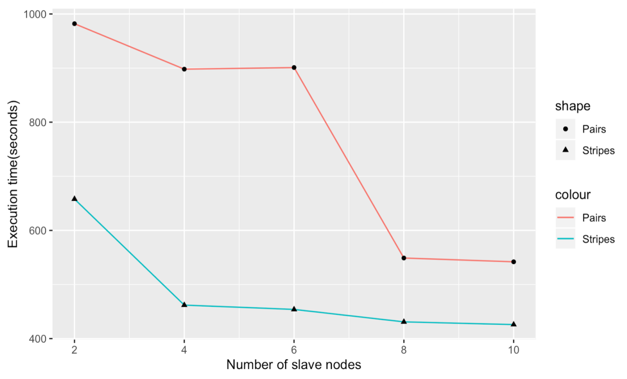
The effect of number of reducers to time efficiency

Figure 1.5



The effect of number of slave nodes to time efficiency-101mb

Figure 1.6



The effect of number of slave nodes to time efficiency- 408mb

Time log

Table 2.1

Data size(MB)	Stripes(secs)	Pairs(secs)
201	331	457
407	445	518
611	712	969
813	840	1002
1014	934	1091

The effect of data size to time efficiency- 8 nodes.

Table 2.2

Data size(MB)	Stripes(secs)	Pairs(secs)
1	367	462
2	590	562
4	767	1132
6	834	2065
8	1072	2231
10	1337	2750

The effect of data size to time efficiency-1node

Table 2.3

Mappers	Stripes(secs)	Pairs(secs)
1	367	462
2	415	854
4	725	1052
6	1244	1570
8	1314	2135
10	1611	2671

The effect of number of mappers to time efficiency-1 reducer.

Table 2.3

Reducers	Stripes (secs)	Pairs(secs)
1	367	462
2	381	440
3	333	442
6	259	449
18	310	506
54	548	687

The effect of number of reducers to time efficiency-1 mapper.

Table 2.4

Nodes	Stripes(secs)	Pairs(secs)
2	330	446
4	309	433
6	301	430
8	297	421
10	279	427

The effect of number of slave nodes to time efficiency-data size 101mb.

Table 2.5

Nodes	Stripes(secs)	Pairs(secs)
2	658	982
4	462	898
6	454	901
8	431	549
10	426	542

The effect of number of slave nodes to time efficiency-data size 407mb.