Student Name: Yogesh Haresh Bojja
Student Number: s3789918

Step1 - Folder **s3789918_S2_2020/target** has "**s3789918_S2_2020-0.0.1-SNAPSHOT.jar**"

Step 2 - Upload "**s3789918_S2_2020-0.0.1-SNAPSHOT.jar**" to **Hue**

Step 3 - Copy "s3789918_S2_2020-0.0.1-SNAPSHOT.jar" to EMR node by executing "**hadoop fs -copyToLocal <HDFS jar path> ~/**"

Step 4 - Create a folder with files **RMIT**, **MELBOURNE** and **3littlepigs** in it, this will be your Input folder.

input_path: path of Input folder

output_path: path of desired output folder.

Task 1 execution: hadoop jar s3789918_S2_2020-0.0.1-SNAPSHOT.jar edu.rmit.cosc2367.s3789918_S2_2020.Yogesh_task1 <input_path> <output_path>

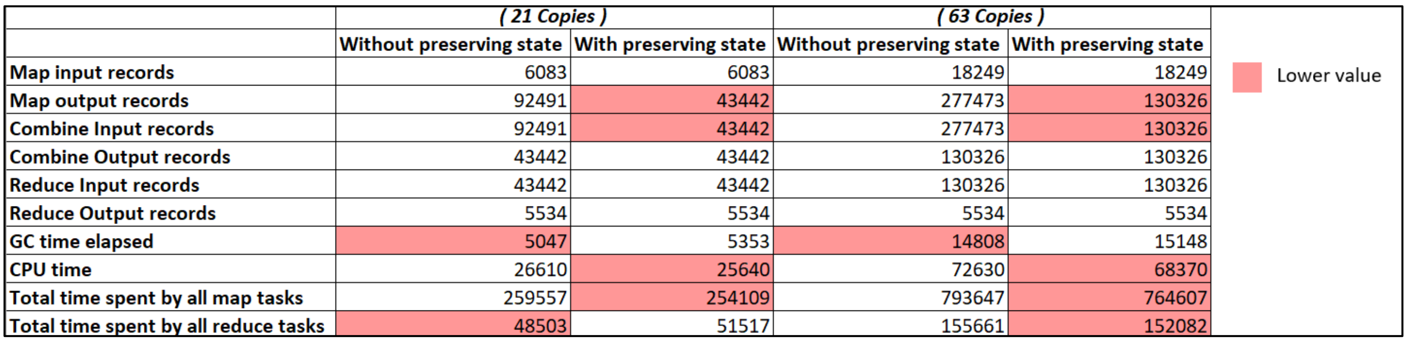
Task 2 execution: hadoop jar s3789918_S2_2020-0.0.1-SNAPSHOT.jar edu.rmit.cosc2367.s3789918_S2_2020.Yogesh_task2 <input_path> <output_path>

Task 3 execution: hadoop jar s3789918_S2_2020-0.0.1-SNAPSHOT.jar edu.rmit.cosc2367.s3789918_S2_2020.Yogesh_task3 <input_path> <output_path>

Task 4 :

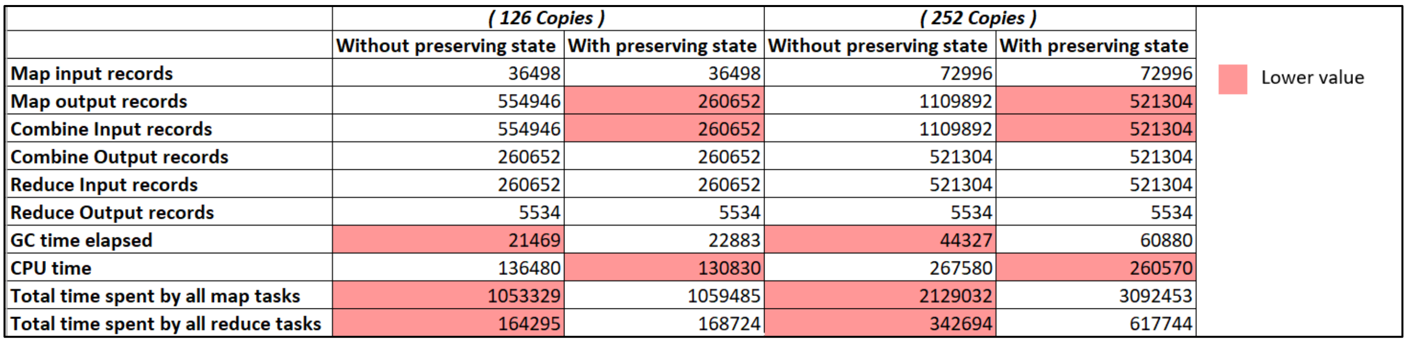
Case 1: Increase Number of documents.

There are three ways to implement MAP-REDUCE. First way is to implement it traditionally, where we send all the <KEY-VALUE> pairs on network i.e., to reducer or combiner and this method is costly. To solve this problem 'In-mapper combining" is done which again is implemented with or without preserving state. In task 2 of the assignment, we have implemented In-mapper combining by without preserving state whereas in task 3 it is implemented with preserving state. The basic difference between both is that in "without preserving state" associative array is stored locally per document per mapper and on the other hand in "with preserving state" associative array is stored globally for all the documents per mapper. From image1 and image 2, number of <KEY-VALUE> pairs emitted after Mapper task, and which is given as an input to combiner task is more when state is not preserved and less when state is preserved. Thus, we can say that In-mapper combining with preserving state basically does the same work as combiner and this can be seen in image1/2 i.e., Combine output records = Map output records for "with preserving state". If combiner task was not present, then the <KEY-VALUE> pairs from Map output records would be given to reducer task directly instead of combiner.



	(21 Copies)		*(63 Copies)*		
	Without preserving state	**With preserving state**	**Without preserving state**	**With preserving state**	
Map input records	6083	6083	18249	18249	🟥 Lower value
Map output records	92491	43442	277473	130326	
Combine Input records	92491	43442	277473	130326	
Combine Output records	43442	43442	130326	130326	
Reduce Input records	43442	43442	130326	130326	
Reduce Output records	5534	5534	5534	5534	
GC time elapsed	5047	5353	14808	15148	
CPU time	26610	25640	72630	68370	
Total time spent by all map tasks	259557	254109	793647	764607	
Total time spent by all reduce tasks	48503	51517	155661	152082	

(Image1)



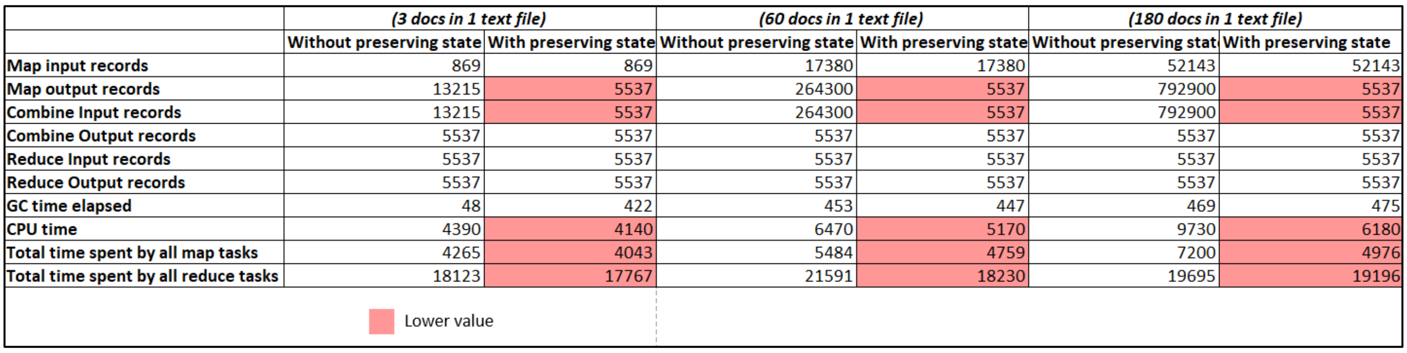
	(126 Copies)		*(252 Copies)*	
	Without preserving state	**With preserving state**	**Without preserving state**	**With preserving state**
Map input records	36498	36498	72996	72996
Map output records	554946	260652	1109892	521304
Combine Input records	554946	260652	1109892	521304
Combine Output records	260652	260652	521304	521304
Reduce Input records	260652	260652	521304	521304
Reduce Output records	5534	5534	5534	5534
GC time elapsed	21469	22883	44327	60880
CPU time	136480	130830	267580	260570
Total time spent by all map tasks	1053329	1059485	2129032	3092453
Total time spent by all reduce tasks	164295	168724	342694	617744

🟥 Lower value

(Image2)

GC time elapsed shows the time required to execute Garbage Collector. From image1 and image2 it is seen that GC time elapsed is more in case of In-mapper combining without preserving state. CPU time required for In-mapper combining without preserving state is more than that of with preserving state. From image1, for 21 and 63 copies In-mapper combining with preserving state spends less time for its mapping tasks i.e., “Total time spent by all map tasks” but as number of copies increases(shown in image 2) In-mapper combining without preserving state performs better and takes less time. Time taken for all reducer task i.e., “Total time spent by all reduce tasks” gets better for In-mapper combining without preserving state as the number of copies given to Hadoop increases. Thus we can conclude that if the number of copies is large its better to go with “In-mapper combining without preserving the state” and if number of documents are less then its better to select “In-mapper combining with preserving the state”.

Case 2: Increase size of a single document.



	(3 docs in 1 text file)		*(60 docs in 1 text file)*		*(180 docs in 1 text file)*	
	Without preserving state	**With preserving state**	**Without preserving state**	**With preserving state**	**Without preserving stat**	**With preserving state**
Map input records	869	869	17380	17380	52143	52143
Map output records	13215	5537	264300	5537	792900	5537
Combine Input records	13215	5537	264300	5537	792900	5537
Combine Output records	5537	5537	5537	5537	5537	5537
Reduce Input records	5537	5537	5537	5537	5537	5537
Reduce Output records	5537	5537	5537	5537	5537	5537
GC time elapsed	48	422	453	447	469	475
CPU time	4390	4140	6470	5170	9730	6180
Total time spent by all map tasks	4265	4043	5484	4759	7200	4976
Total time spent by all reduce tasks	18123	17767	21591	18230	19695	19196

🟥 Lower value

(Image 3)

In this case its seen that as we go on increasing the content in the document In-mapper combining with preserving state performs better in case of CPU time, Time taken for all map/reduce tasks. Even though it performs better there is high possibility of associative array crashing if document size is very large.