COSC2637 Assignment 2

Matthew Bentham
S3923076

Task 2 – Performance Analysis

Results

Matrix size	No. of reducers	Map input records	Map output records	CPU time spent (ms)
6X6	1	18	468	8700
6X6	3	18	468	11430
6X6	6	18	468	14000
6X6	9	18	468	16680
20X20	1	60	16400	11670
20X20	3	60	16400	12780
20X20	6	60	16400	15880
20X20	9	60	16400	16650
50X50	1	150	252500	15850
50X50	3	150	252500	20280
50X50	6	150	252500	25260
50X50	9	150	252500	26780
100X100	1	300	2010000	26620
100X100	3	300	2010000	34700
100X100	6	300	2010000	43720
100X100	9	300	2010000	50740
200X200	1	600	274772357	200110
200X200	3	600	274772357	240130
200X200	6	600	274772357	237890
200X200	9	600	274772357	222350

data table.

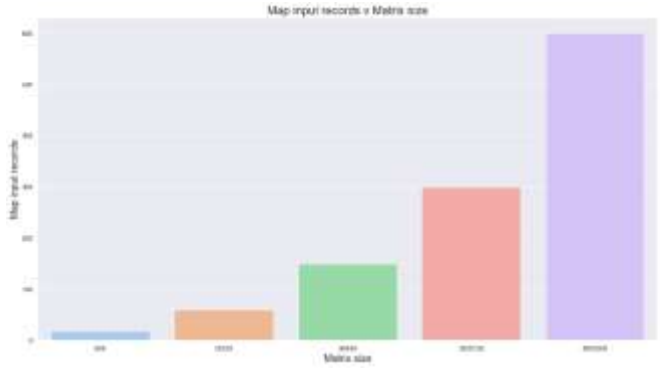

Fig 1.

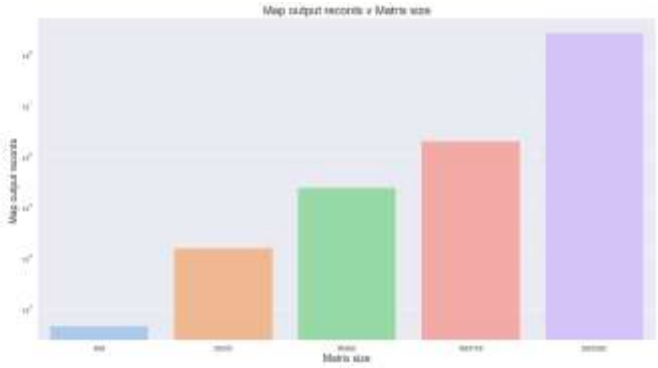

Fig 2. (Logarithmic scale)

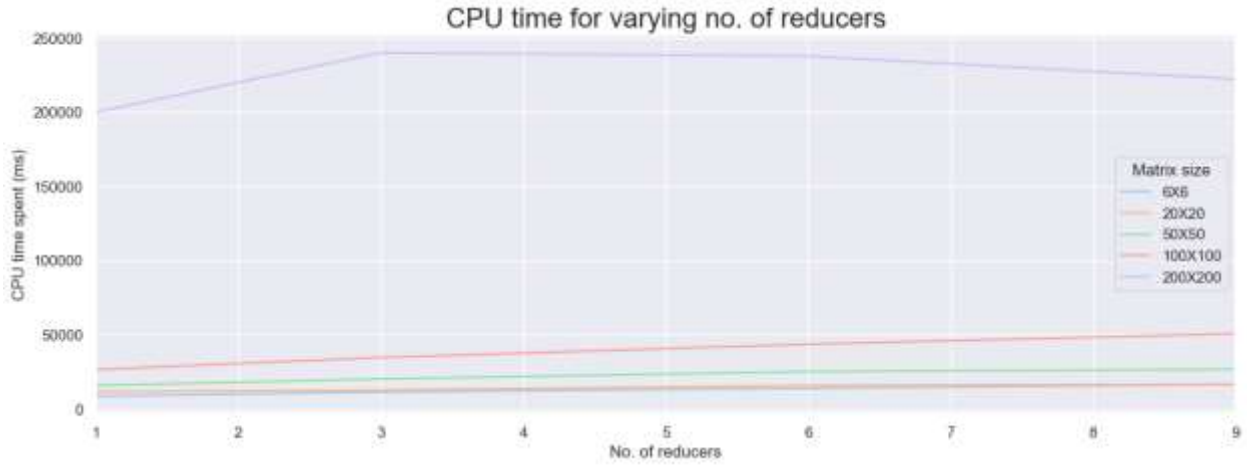

Fig 3.

B. What is the impact of the matrix size to the performance? Explain your answer based on the test results

As seen in both *fig 1.* and *fig 2.* Increasing the size of the matrix inputs also obviously increases the map input records and drastically increases the output records at an exponential rate, so much so that the y-axis needed to be scaled (logarithmic) to see all the values. Overall, as the size of the input data increases so does both the loads placed on the mapper and reducer and the amount of memory needed for allocation, all of which ultimately have their limits and will lead to a reduction in performance as size increases. Additionally, increasing the size of the mapper output also increases the amount of disk and network I/O that occurs which is a common bottleneck for MapReduce programs. *Fig 3* further support this, as for all cases, the larger input sizes have a greater CPU time. Size therefore can be seen to have a substantial effect on performance, as the 200 X 200 matrix input performed more than 3 times worse the 100 X 100 matrix which intern performed worse than all the other inputs. This suggest that CPU time has an exponential relationship with matrix size, which is more clearly visualised in *Fig 4* below showing the average cup times for each input size.

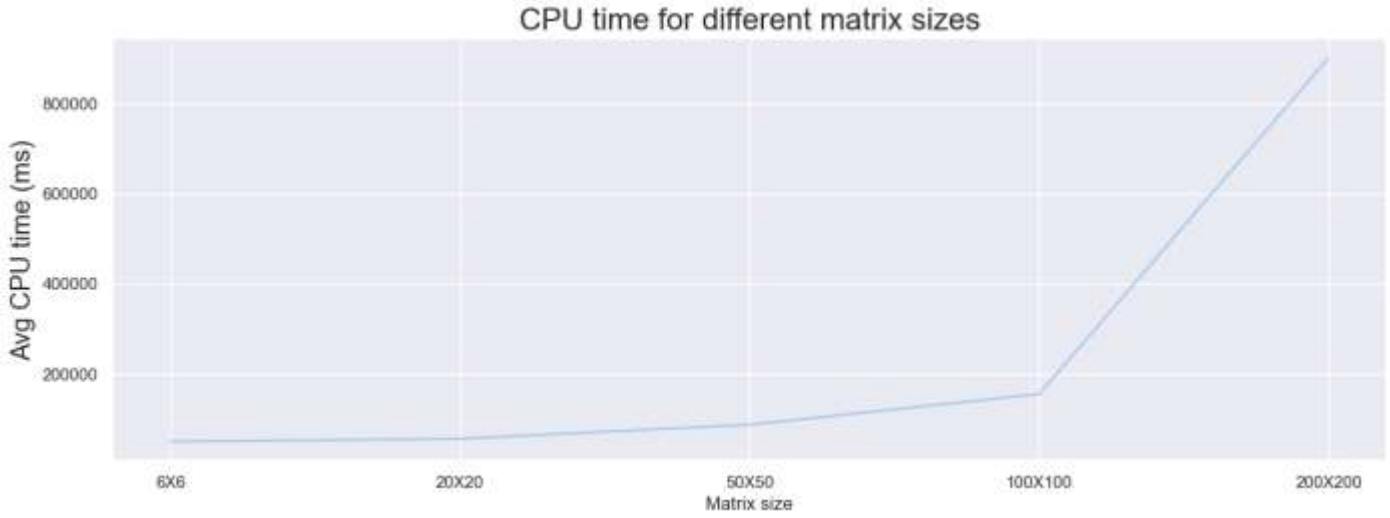

Fig 4.

C. Can more reducers always benefit the performance? Explain your answer based on the test results.

As seen in *fig 3.* The number of reduces is also heavily correlated with performance. As the number of reducers increases for all cases aside from the 200 X 200 matrix, the CPU time also increases. This is likely to have occurred, because small matrices (less than 200 rows) do not require an extensive amount of memory, therefore increasing the number of reducers from this point is likely to only hinder performance as it increases the start-up time of the reducer phase, increases the number of reducers that the data needs to be parsed too and increases the number of files need to be processed and outputted overall. The 200 x 200 matrix on the other hand, does not exhibit a linear relationship between no. of reducers and run time, however having one reducer still performs the best out of all tested cases. This could suggest that further increasing the number of reducers past nine could improve the performance overall, however further testing is needed to determine this. In all tested cases however, the added performance boost of having multiple reduce jobs doesn't outweigh its computational cost. In cases where much larger matrices are inputted (greater than 200 rows); it is safe to assume that having more than one reducer process the entire mapper output load would increase the performance significantly. Overall, more reducers **DOESN'T** always benefit performance and the number of reducers used should be determined on a case-by-case basis depending on the input and output sizes.