Austin Sura
October 1, 2018

Cloud Computing Homework A2:

Part 1:

To measure how many comparisons were able to be completed in one minute I used the given command sequence.

```
head -n2 agambiae.small.fasta >  1.fasta
timeout 60s ./swaligntool 1.fasta agambiae.small.fasta > results.dat
```

The alteration that I made was adding the timeout 60s command to the beginning of the 2nd line. Using grep on results.dat I saw that 5 comparisons were made during the 60 seconds. This was tested on student02.cse.nd.edu. 60/5 yields that each comparison takes 12 seconds.

The number of comparisons need for the small file is 31125, so I multiplied that by the fact that each comparison took 12 seconds in my baseline and my result was: 4 days, 7 hours, 45 minutes and 0 seconds, which is 6225 minutes.

This comes out to about 1037.5 hours; to complete the entire process in an hour it would take 1038 machines at my baseline pace. To do the comparisons on the entire genome of 100,000 sequences you would need 4999950000 comparisons. To complete that on a singular machine that computes 5 comparisons per minute, it would take an extremely long time: 694437 days and 12 hours. Over 1900 years!

Part 2:

Output:
The following are the top 10 matches that were found using my compareit tool.

```
Top Ten Matches:
1: sequence 1102140143903 matches 1102140176186 with a score of 1539
2: sequence 1102140177182 matches 1102140177183 with a score of 874
3: sequence 1102140167168 matches 1102140172678 with a score of 818
4: sequence 1101555423227 matches 1102140171813 with a score of 777
5: sequence 1101555423815 matches 1102140164074 with a score of 767
6: sequence 1101671610328 matches 1102140171192 with a score of 764
7: sequence 1102140170611 matches 1102140171192 with a score of 744
8: sequence 1102140171192 matches 1102140178449 with a score of 742
9: sequence 1101555423803 matches 1102140171192 with a score of 741
10: sequence 1101671610328 matches 1102140178846 with a score of 736
```

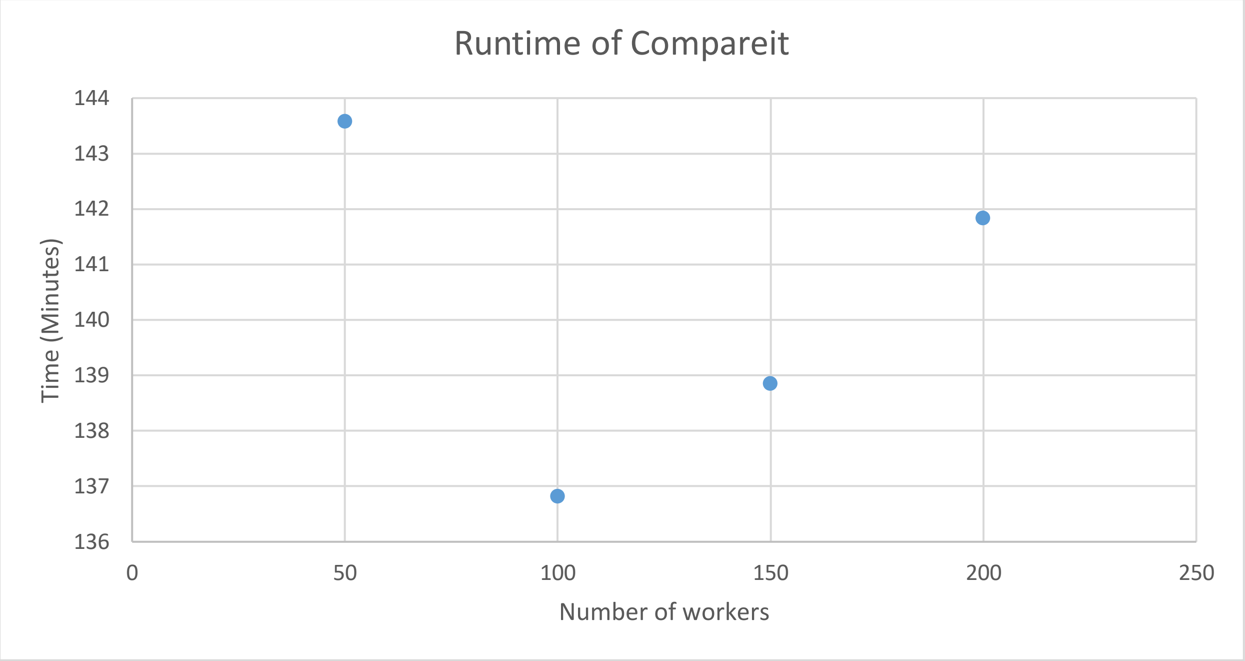
Part 3:

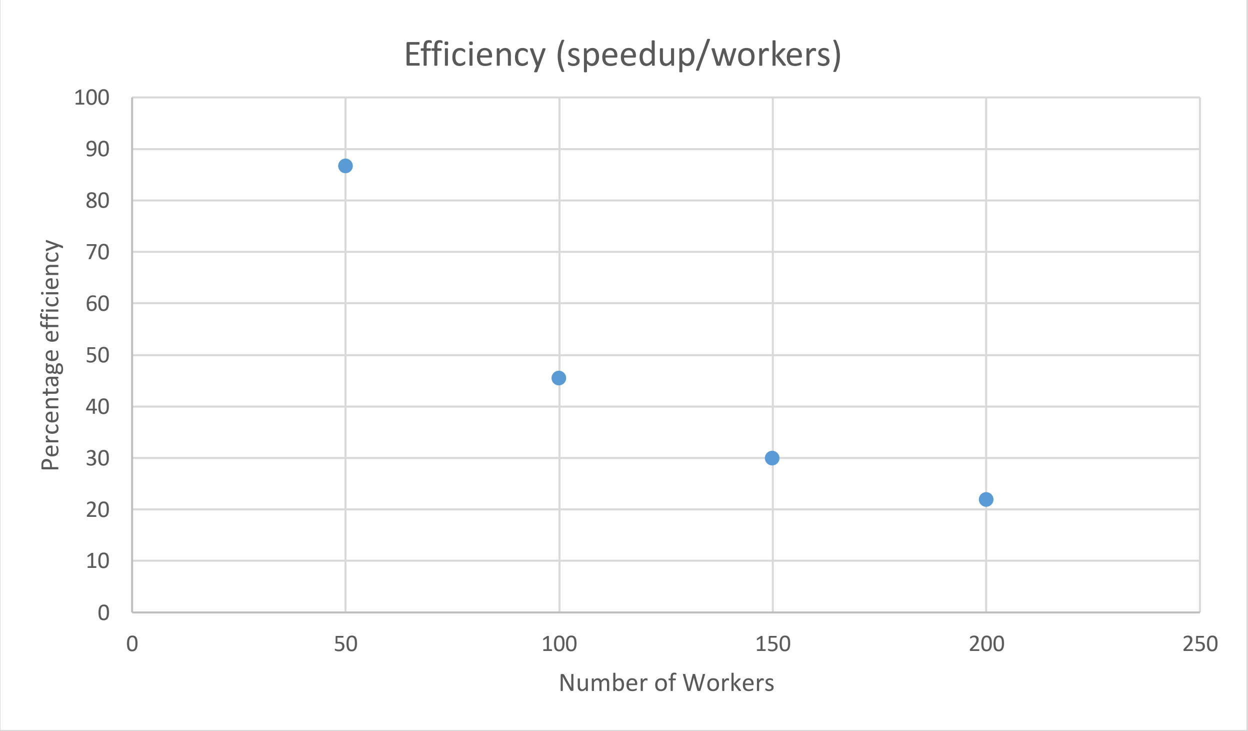
Number of Workers	**Final time**	**Speedup**	**Efficiency**
50	143m 35.909s	6225m/143m35.909s = 43.355 speedup.	Efficiency: 43.355/50 workers 86.71 %
100	136m 49.321s	6225m/136m49.321s = 45.497 speedup.	Efficiency: 45.497/100 workers 45.50 %
150	138m 50.941s	6225m/138m50.941s = 44.833 speedup.	Efficiency: 44.833/150 workers 29.89%
200	141m 50.428s	6225m/141m50.428s = 43.889 speedup.	Efficiency: 43.889/200 workers 21.94%

Explanation:

My results demonstrate that simply adding more workers does not mean faster completions as a hard and fast rule. There was a sizable speed up between 50 workers and 100 workers but then the time slowed slightly for 150 and 200 workers. There was diminishing value in adding workers and at some point it appeared to be detrimental. This output did not match my expectations; I was anticipating a nearly proportional decrease in the final time as workers were added. After discussing with some of my colleagues, I believe that the reason for the lack of speedup has to do with limitations of the master. As you add more workers and tasks to the workqueue, there is more data being processed. I theorize that as the master is handling this incoming data from the tasks returning, it slows or halts the progress of successive jobs. Another explanation could be that there was differing amounts of traffic on workqueue and condor during the slightly different times of my tests. If more people are using condor, my jobs would be submitted to workqueue at a slower pace generally. Plots of my runtime and efficiency are located below.

Charts:


Runtime of Compareit
Time (Minutes)
144
143
142
141
140
139
138
137
136
0
50
100
150
200
250
Number of workers


Efficiency (speedup/workers)
Percentage efficiency
100
90
80
70
60
50
40
30
20
10
0
0
50
100
150
200
250
Number of Workers