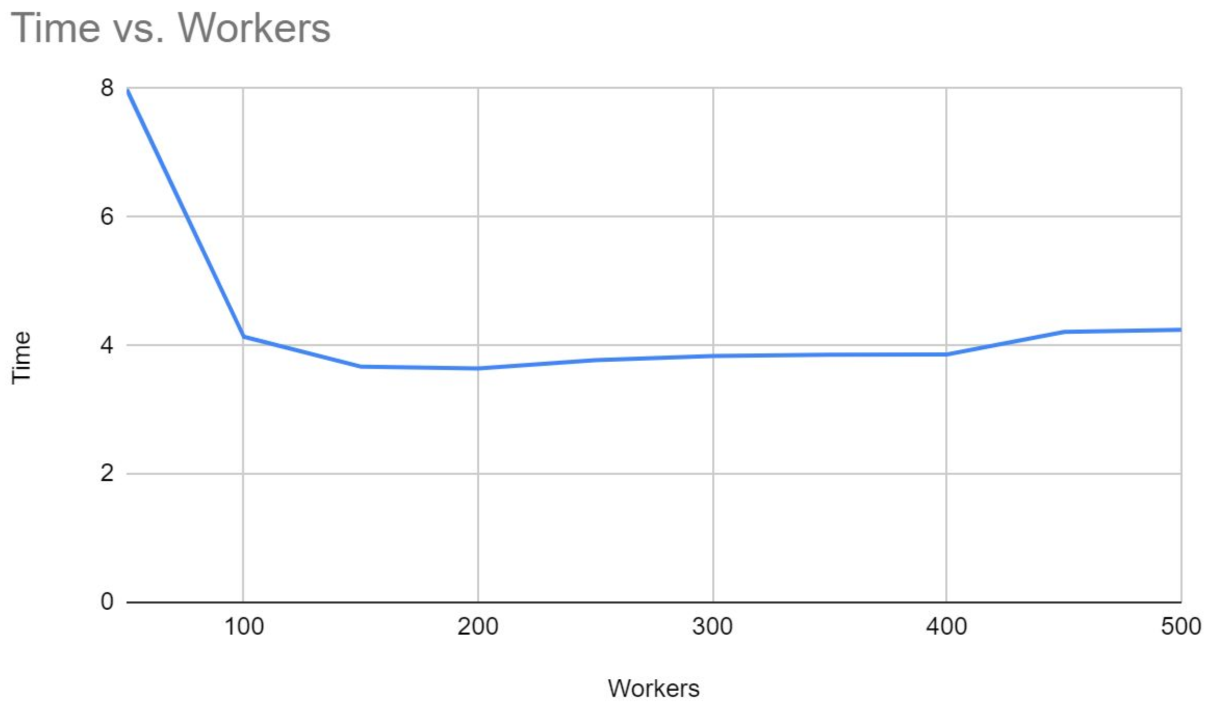
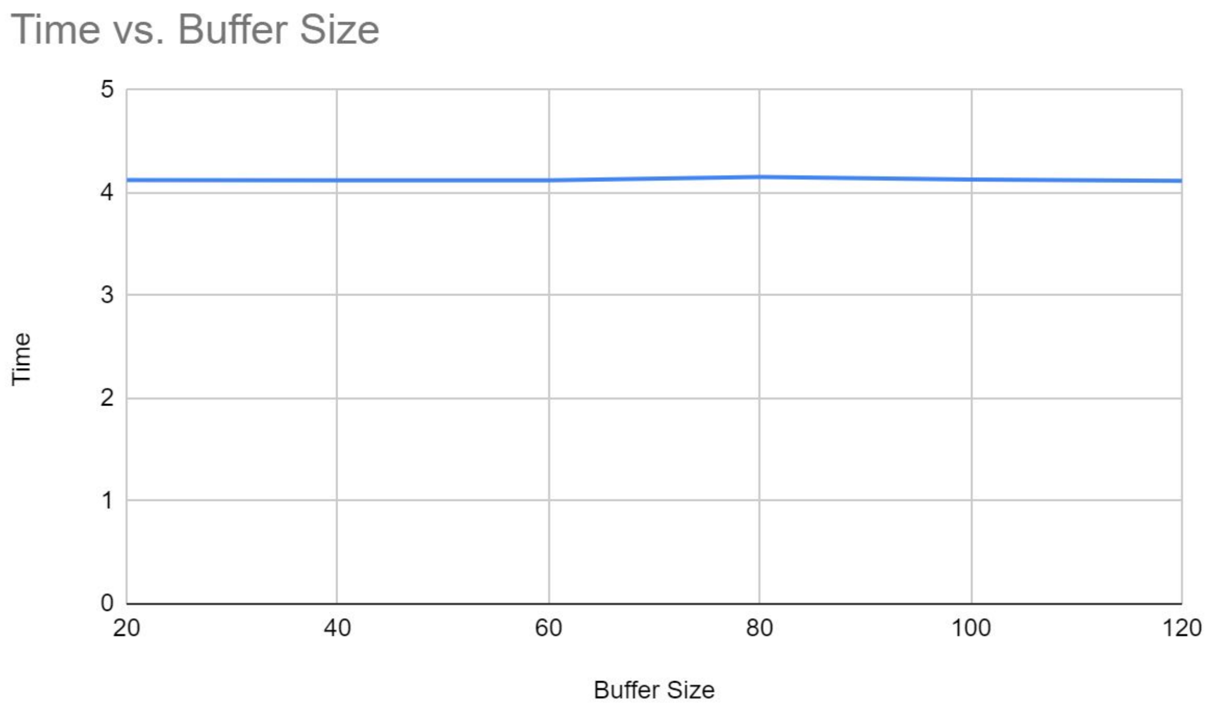
Thomas Hari Budihardjo
UIN: 126009556

PA3 Report

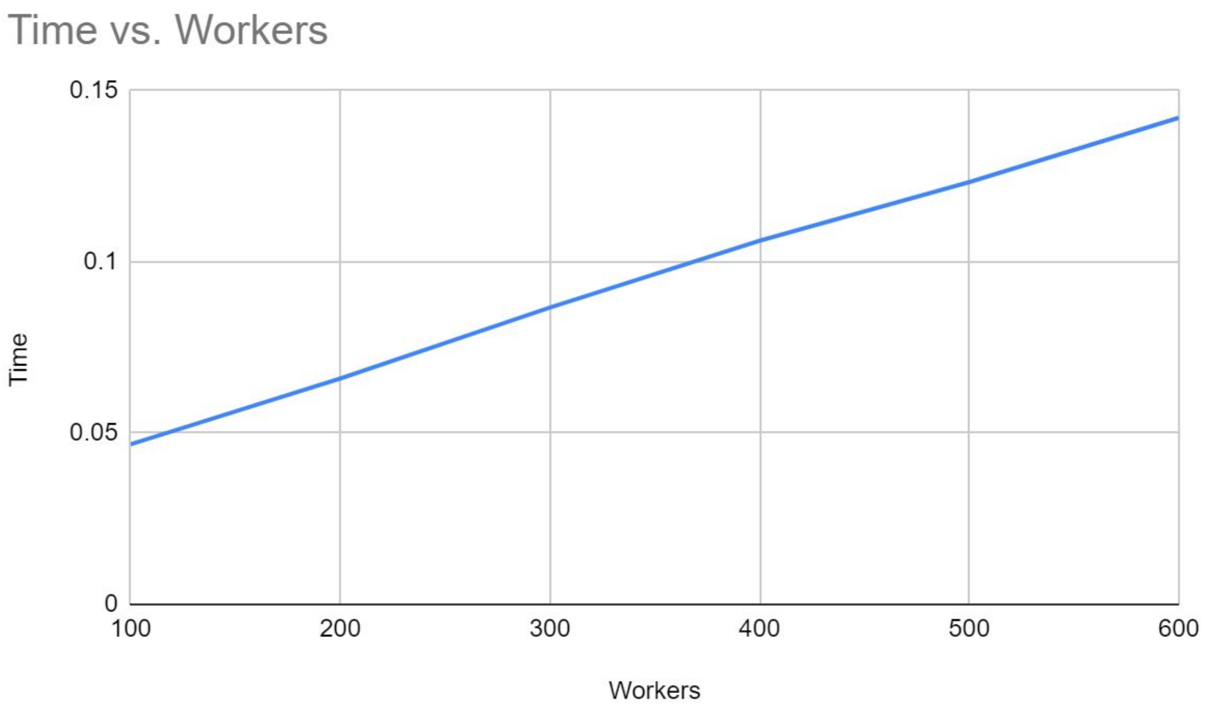
The performance changes when varying the worker thread in requesting data points. It starts slow with low amount of workers, increasing in speed as we add worker threads approaching the point of diminishing returns (which for my program occurs after 200 workers), at which point the amount of time starts increasing again. Increasing workers will initially decrease the time it takes to complete the data transfer because there will be more and more worker threads consuming the tasks in the buffer, but eventually the number of context switches that takes place due to having so many workers far outweighs the benefit of multithreading. This is due to the fact that the number of patient threads stay the same, which means the buffer will be empty almost always during the running of the program because the patient threads cannot keep up its production with the rate at which the worker threads are taking things out of the buffer.



As for varying the buffer size, the time does not vary. Increasing the buffer size does not change the balance between the patient and worker threads, which means the time it takes to complete the tasks will stay constant. We can say that this relationship is linear.



Varying the amount of workers in requesting files increases linearly. I believe this relationship will hold forever. This relationship can be explained by the fact that there is only one patient thread, which means the workers are taking things out of the buffer at a rate that is faster than the patient threads are putting it in. Adding more and more workers will only increase the amount of context switches, which will add overhead that will make longer runtime.


Time vs. Workers
Time
Workers