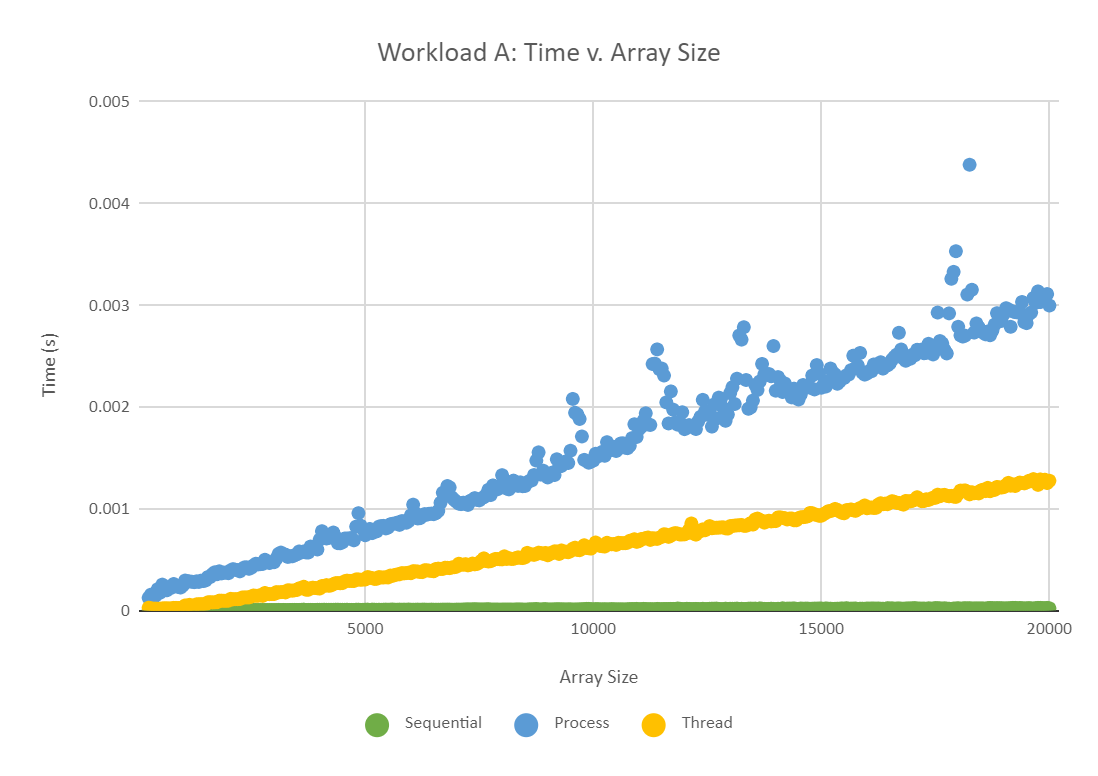
Spooky Searching: Results

# Part 1: Experimenting with Array Size

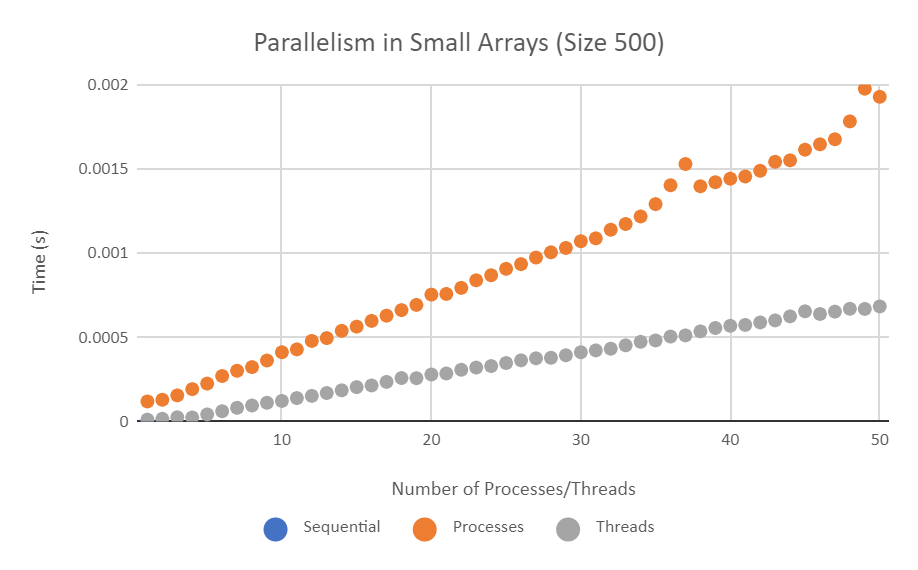
## Workload A



* In this graph, we can see that sequential search is by far the fastest way to search an array at larger sizes. We can see that multithreading is slower than sequential search, but is still much faster than multiprocessing. We also see that sequential search and multithreading follow their trends with consistency, however, multiprocessing develops outliers when the array size gets larger. The array size getting larger also affects the number of processes, which is probably the cause of the slow down for multithreading and multiprocessing.

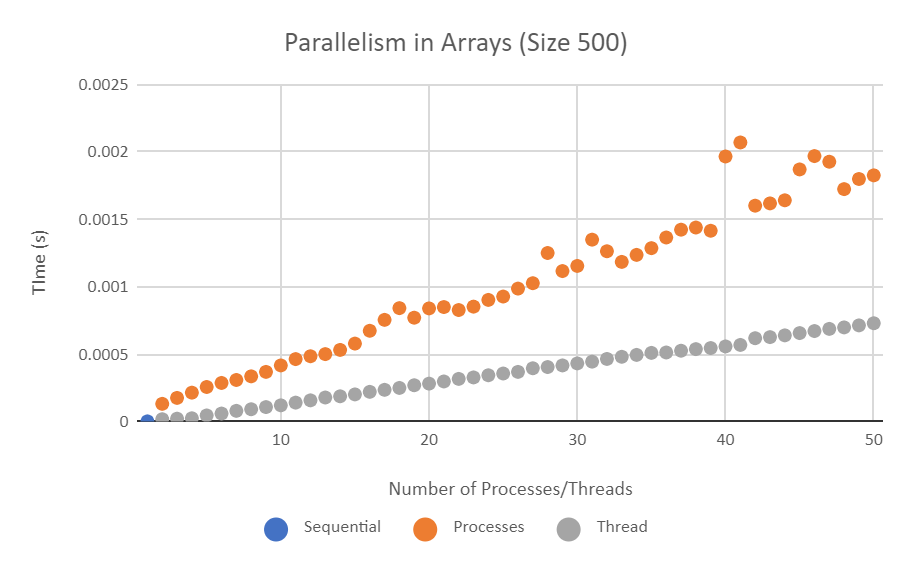
# Part 2: Experimenting with Number of Branches

## Workload B



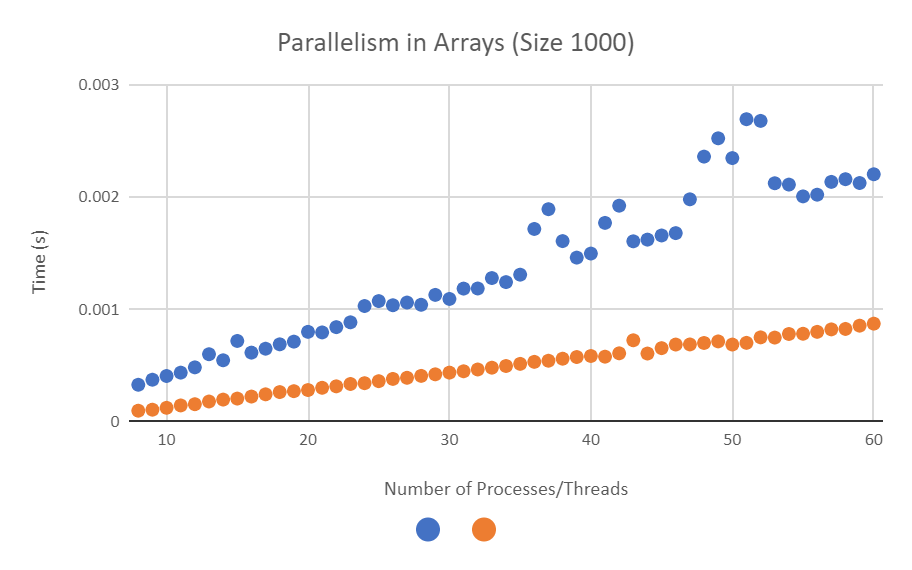
* In workload B, we see that by adding more threads and processes, the time it takes to search the array takes longer. We can also see processes is slightly slower than threads. We also see that the rate of increase for both in this workload appears to be linear.

## Workload C



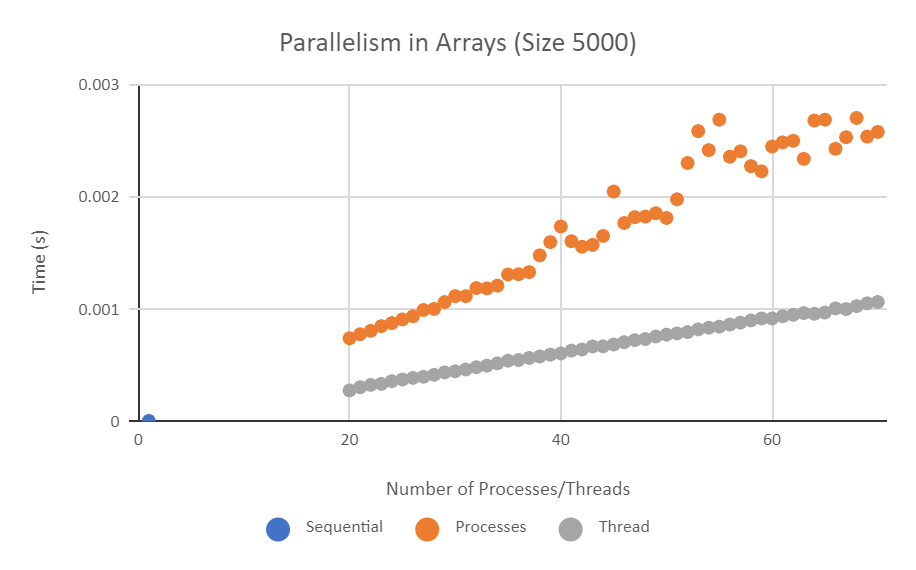
* In workload C, we continue to see that more threads and processes will slow down the time it takes to search. In this case, we still see that processes is slower than threads. What is different about this case is that we start seeing outliers in processes when the number of processes reaches around 40.

## Workload D



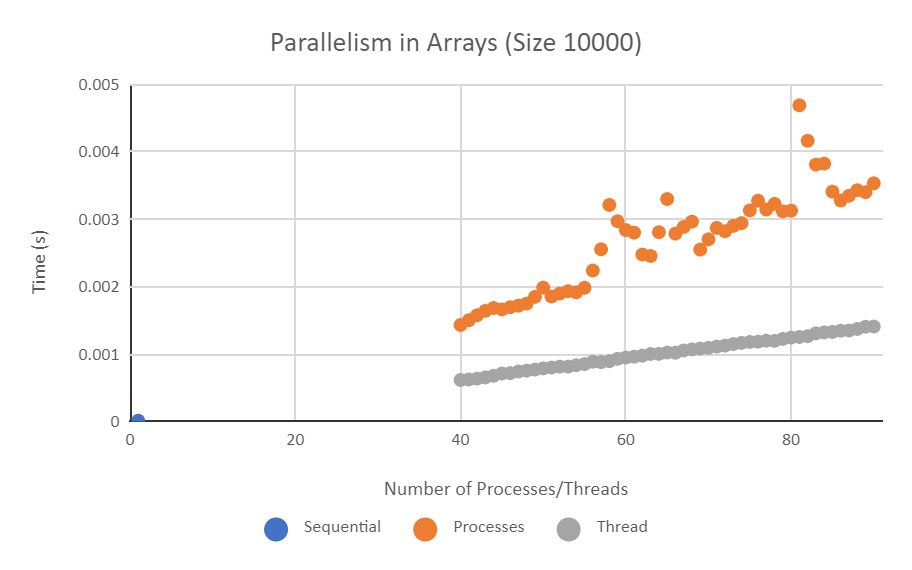
* In workload D, we continue seeing the trend of how more threads/processes equates to longer time to search. Thread continues to stay very linear with no outliers, regardless of the change in size and number of threads. Processes gets outliers at around 40-50 processes, but appears to stabilize in consistency at around 55.

## Workload E



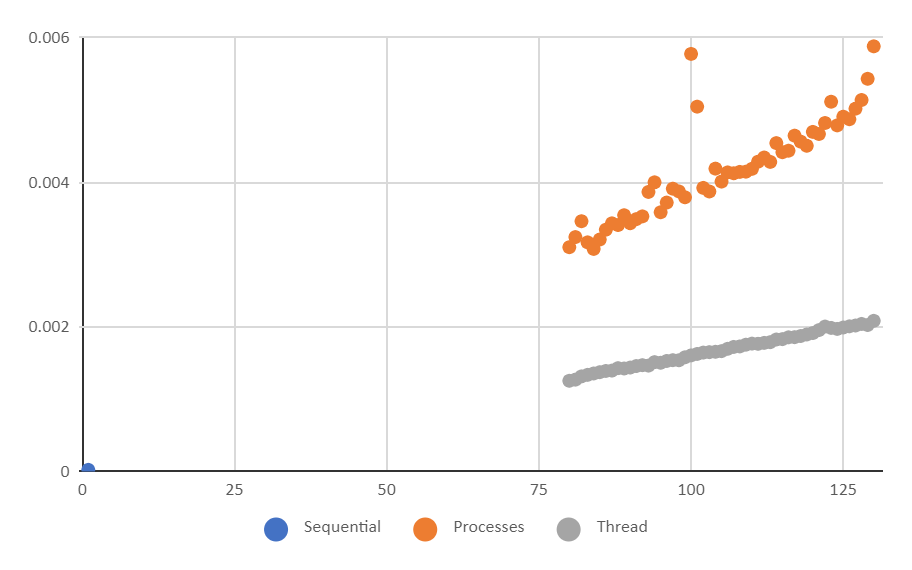
* Workload E is similar to the previous workloads in that thread is faster than processes, and that threading still appears to be consistently linear. What we can learn from this workload is that the trend becomes less consistent when you start getting to a higher number of processes.

## Workload F



* For Workload F, we still see that the trend for threads stays consistent. We start seeing that processes start to develop big outliers when we reach around 80 processes. It highlights the inconsistency of processes and how they don’t work very well when you run that many processes simultaneously.

## Workload G



* After looking at workload G, it is fair to assume that threads will consistently follow a linear trend at a given size with a variable amount of threads. We can also see that processes are not very consistent in that tests may take much longer, even if there are a fewer amount of processes taking place. The amount of time it takes also increases at a faster rate when compared to threads.