Rutgers CS 01:198:214 Systems Programming
Professor John-Austen Francisco
Authors: Anthony Siluk (ars373) & Alexander Goodkind (amg540)
Due: 11/22/2019

Extra credit: Context Switching Performance Averages

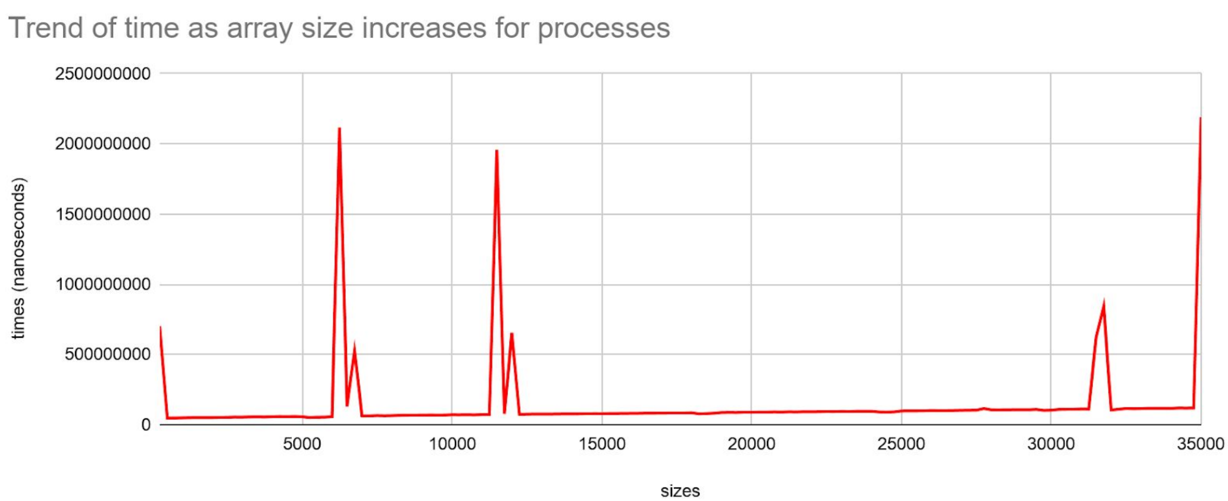
For thread:
Total number of voluntary and involuntary context switches: 69872792207
Average time per context switch: 560441ns

For proc:
Total number of voluntary and involuntary context switches: 58309
Average time per context switch: 2250011ns

We can see that because threads took less time on average, they were able to achieve more context switches, which may explain why they were faster than proc's.

Processes:

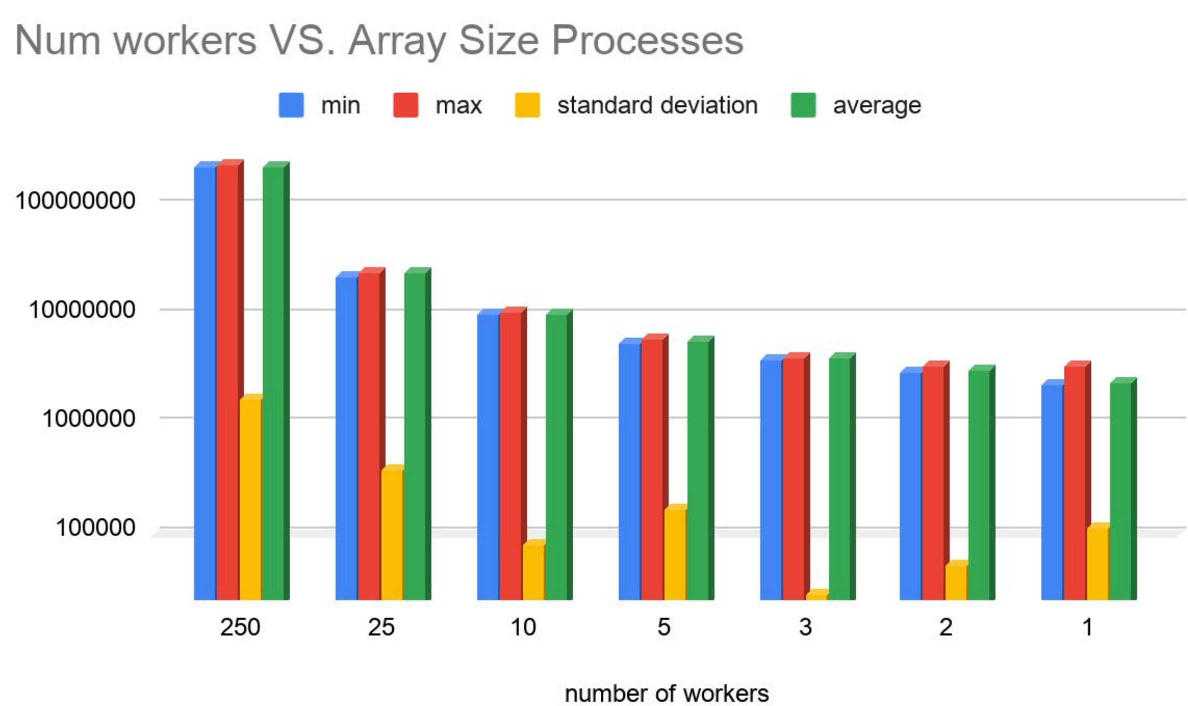
Test A:



In test A we see some odd spikes, this is most likely the operating system scheduler causing the processes to wait for an unrelated process before they can execute.

Minimum: 42358516ns
Maximum: 119677297ns
Standard Deviation: 20621415ns
Average: 83253356ns

Test C:



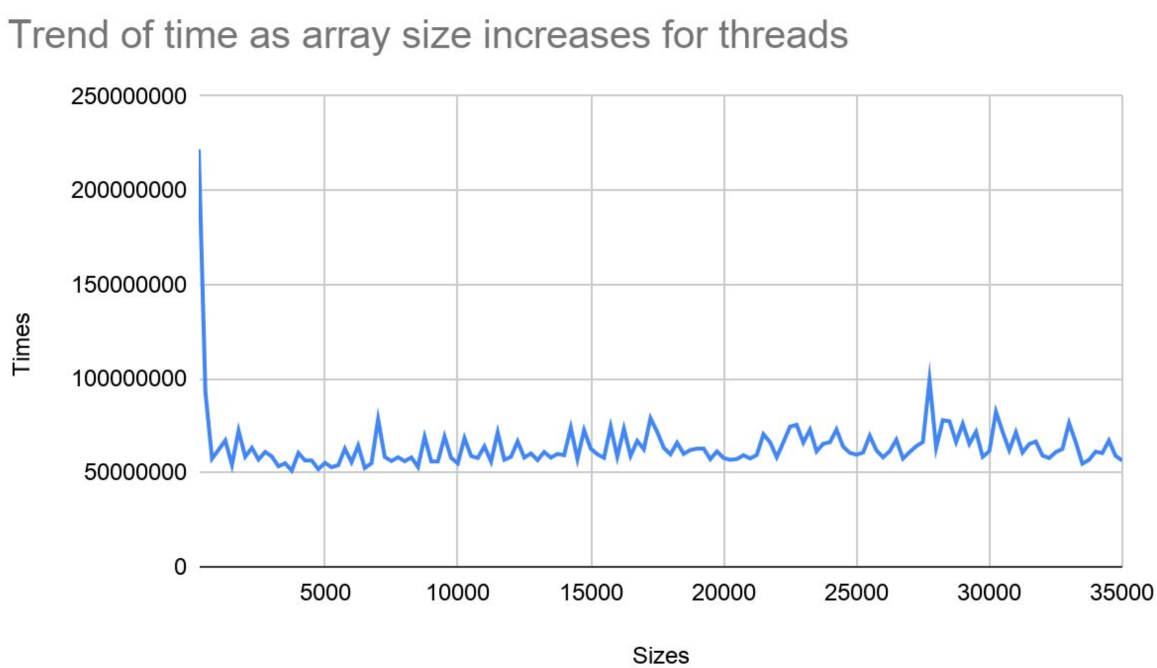
While the average runtime decreases, as well as the max and min runtimes, the three almost remain equal throughout. This again is most likely due to the fact that processes, unlike threads, have to copy and paste themselves into new regions of memory whenever a child is created.

number of workers	250	25	10	5	3	2	1
min	85095026	8644958	3161874	1283604	813941	734276	552474
max	160531621	1864405	5120365	3183815	1197450	1149860	859773
standard deviation	11625022	1666098	398250	251616	74249	81693	45067

average	107275658	10284963	3625085	1482797	986809	895929	667058

Threads:

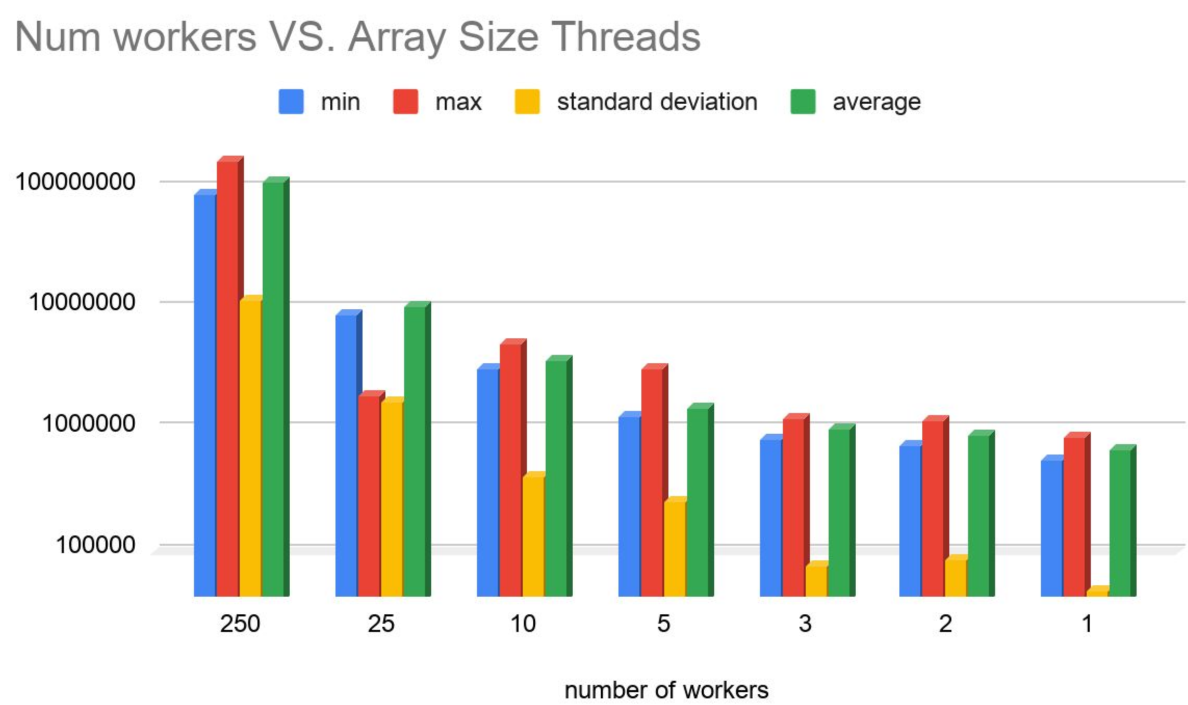
Test A:



The spike at the beginning probably a fluke and can be safely ignored, we see that as the size of the array increases the time it takes to search for the random target is relatively unchanged (except for a very slight increase).

Minimum: 48683248ns
Maximum: 237466678ns
Standard Deviation: 17272640ns
Average: 64892495ns

Test C:

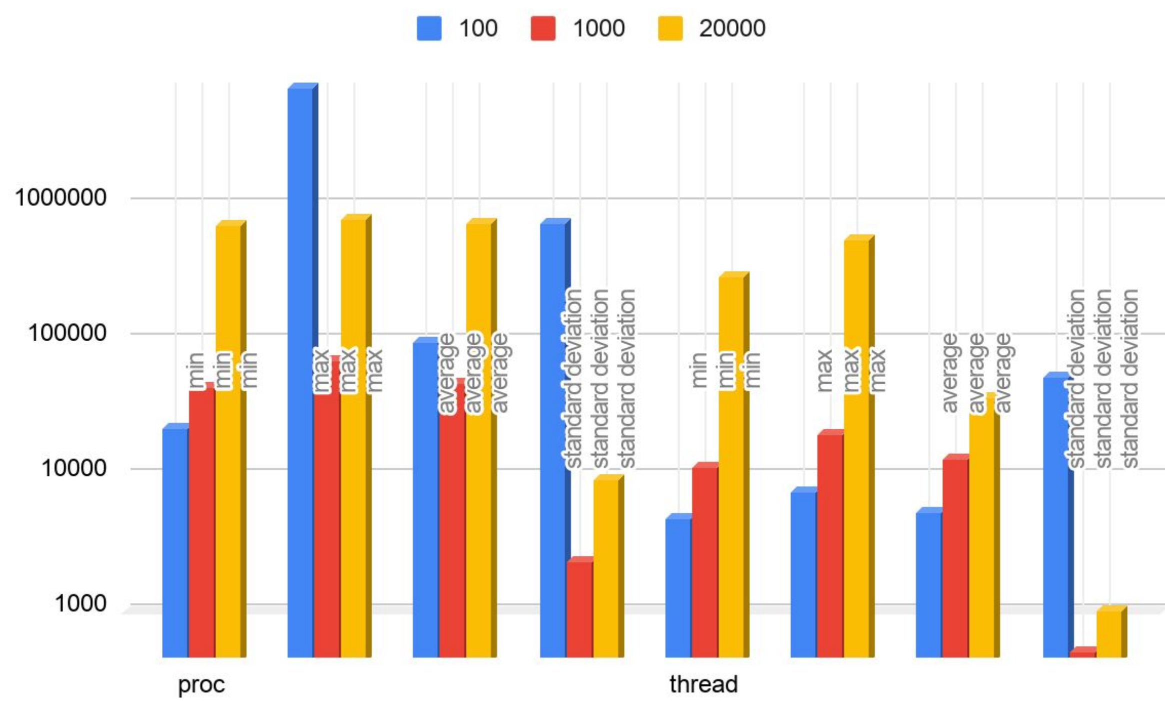


For threads, as one can see, the average workload tends to decrease as the number of parallel threads decreases. This is most likely due to the fact that it is faster to conduct a linear search in one process, rather than calculate a division of labor amongst multiple threads and have each of those search a portion of the array in parallel.

number of workers	250	25	10	5	3	2	1
min	226597554	22083859	10083719	5432733	3895322	2926586	2292734
max	235305596	24460143	10517395	5975565	4068442	3379466	3456314
standard deviation	1659417	380602	79784	167934	26794	50310	110919
average	231214345	24039649	10333169	5752446	3992666	3065410	2399426

Processes VS. Threads

Test B:



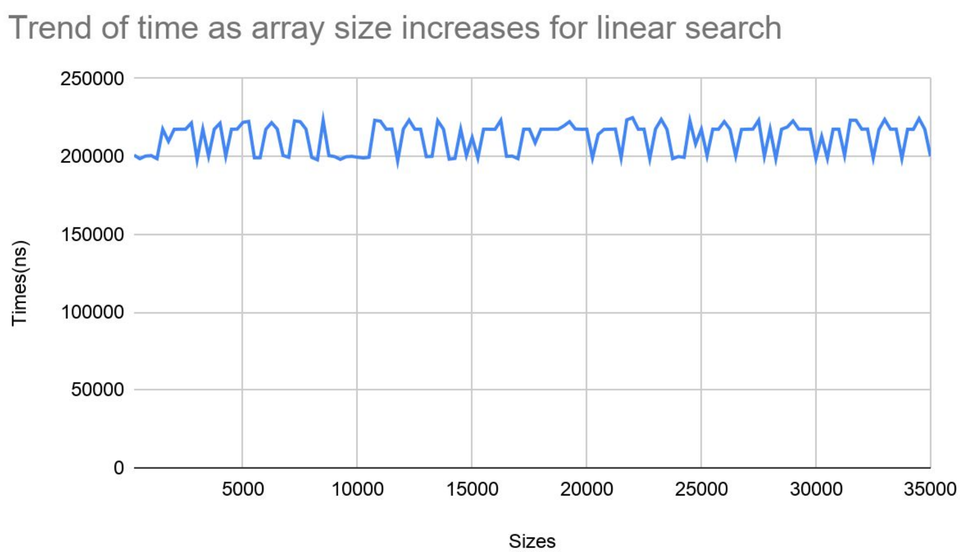
As one can see from this graph, the min, max, and average run times for threads tend to be better than those of processes. This is most likely due to the fact that threads all share the same space in memory, whereas processes have to effectively copy and paste themselves into new regions of memory, thus increasing the runtime.

	size	100	1000	20000
	min	2170773	4377427	68187570
	max	708443959	6823375	75325422
	average	9328120	4669991	71864604
proc	standard deviation	70263796	223999	899333

	min	466901	1135329	28712223
	max	727285	1930748	52843695
	average	522026	1275666	3630756
thread	standard deviation	5250469	47721	98715

Linear Search:

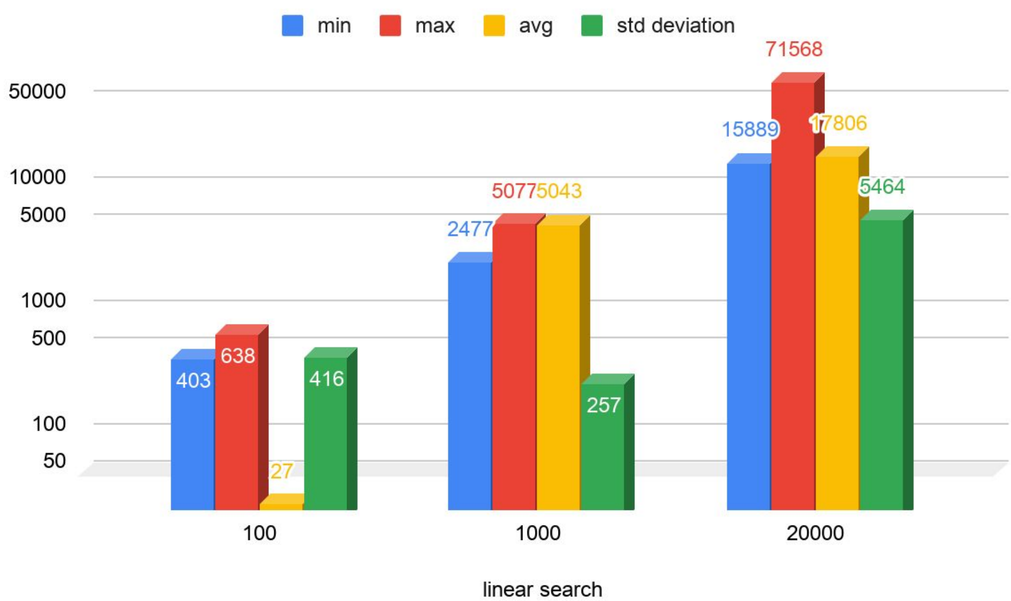
Test A:



The time for a linear search is a consistent as it is high due to the fact that, while the size of the array is either increasing or decreasing, it is still one process. As a result, the jumps in the graph are rather small when compared to a process or a thread search, which has to rely on the results of the process or threads that are running parallel to it.

Minimum: 197216ns
Maximum: 224896ns
Standard Deviation: 9472ns
Average: 212019n

Test B:



Similar to what was said in Test A for linear search, the results remain consistent even more so, now that the array sizes are fixed.

linear search			
sizes	100	1000	20000
min	403	2477	15889
max	638	5077	71568
avg	27	5043	17806
std deviation	416	257	5464

Test C:

There is no test C because there is simply only 1 “worker” as that is the definition of a linear search.