Machine Problem 3 Report

Benjamin Gabay

926006902

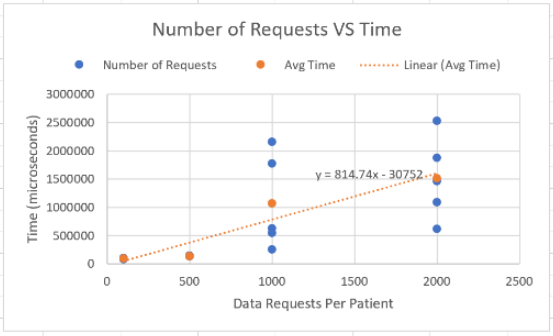
CSCE 313: Section 515

10/27/2019

I created my Bounded Buffer before reading the documentation completely, so I named my class PCBuffer instead of BoundedBuffer. After testing using valgrind I found there are a few minor memory leaks in my program that I was unable to fix but I was able to figure out somewhat where they are coming from. For every additional worker thread that I create there are 26 bytes in blocks “definitely lost”. There is also an additional 28 bytes in 2 blocks “definitely lost” elsewhere in my program. To test the time of my program under different conditions I created a time stamp at the beginning of my client (after forking), and another after all threads have been joined and channels were deleted, essentially once the program was done but before sleeping. For all my tests I used a base case of 1000 requests being made per patient, a bounded buffer size of 50, and 5 worker threads. I ran tests changing each of these parameters by different factors and recording the total process time in microseconds. After looking at my results they were different than I expected, such as being slower with more worker threads. One reason for this could be that making more threads takes more time that it saves by multiple threads working in parallel. Another reason for this could be due to the fact that I was running my program from my apartment, which required me connecting to use a VPN to connect to the University’s “Compute Server”, which could have caused a bigger delay in creating multiple threads than it would have had I actually been on campus running the program. Nonetheless I recorded my results and graphed them.

These are the results when I changed the number of requests per patient, leaving the buffer size and number of worker threads constant.

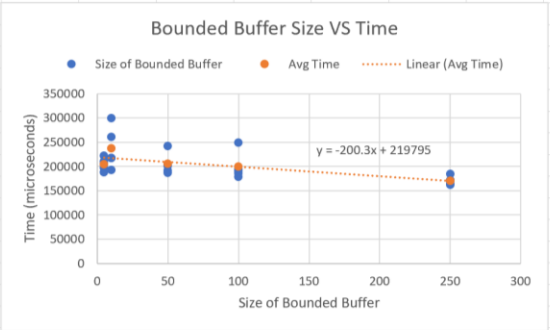
Data Requests Per Patient	Size of Bounded Buffer	Number of Worker Threads	Time (microseconds)	AVG Time for Data Set
1000	50	5	627035	
1000	50	5	543074	
1000	50	5	2160852	
1000	50	5	254002	
1000	50	5	1776791	1072350.8
500	50	5	146219	
500	50	5	128413	
500	50	5	128306	
500	50	5	130906	
500	50	5	146817	136132.2
100	50	5	89077	
100	50	5	85805	
100	50	5	98628	
100	50	5	90893	
100	50	5	79641	88808.8
2000	50	5	615394	
2000	50	5	2529616	
2000	50	5	1873736	
2000	50	5	1459490	
2000	50	5	1085579	1512763



Here we can see that time increases with the number of requests made per patient as expected.

These are the results when I changed the buffer size, leaving the number of requests per patient and number of worker threads constant.

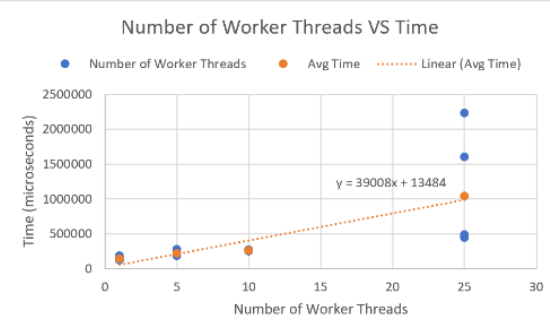
Data Requests Per Patient	Size of Bounded Buffer	Number of Worker Threads	Time (microseconds)	AVG Time for Data Set
1000	50	5	186944	
1000	50	5	203704	
1000	50	5	241846	
1000	50	5	200367	
1000	50	5	191058	204783.8
1000	10	5	261087	
1000	10	5	299853	
1000	10	5	216672	
1000	10	5	192820	
1000	10	5	216700	237426.4
1000	100	5	248598	
1000	100	5	193460	
1000	100	5	190394	
1000	100	5	185113	
1000	100	5	178322	199177.4
1000	5	5	222018	
1000	5	5	196611	
1000	5	5	203743	
1000	5	5	210311	
1000	5	5	187304	203997.4
1000	250	5	184366	
1000	250	5	171370	
1000	250	5	161265	
1000	250	5	170755	
1000	250	5	164570	170465.2



Here we can see that time slightly decreases as the buffer size increases, however it is very slight as though the buffer size has minimal effect on the speed of the program execution.

These are the results when I changed the number of worker threads, leaving the number of requests per patient and the buffer size constant.

Data Requests Per Patient	Size of Bounded Buffer	Number of Worker Threads	Time (microseconds)	AVG Time for Data Set
1000	50	5	212240	
1000	50	5	181879	
1000	50	5	276949	
1000	50	5	212659	
1000	50	5	187945	214334.4
1000	50	1	124950	
1000	50	1	123487	
1000	50	1	178026	
1000	50	1	128245	
1000	50	1	132136	137368.8
1000	50	10	262197	
1000	50	10	261717	
1000	50	10	269415	
1000	50	10	258731	
1000	50	10	247989	260009.8
1000	50	25	1601960	
1000	50	25	456125	
1000	50	25	481355	
1000	50	25	2229481	
1000	50	25	438791	1041542.4



Here we can see that time increases with the number of worker threads. I described a possible reason for this earlier.