**Assignment 3 Report**

**CMPS 270: Software Construction**

**Osama Iskandarani**

**Instructor: Dr. Rida Assaf**

**11/20/2022**

**Laptop Specs: TUF Gaming FX705GM, 16 GB RAM, 6 GB Nvidia Geforce 1060, Core i7 8th Gen**

**As the number of threads increase, the number of iterations tend to be inaccurate and thus lead to an inaccurate measurement of the counter. This can be seen when we had arrays of size 100, as the threads increase to 8, 16, etc., the number of iterations decrease from 10000 to 9600 to 6400, with the accurate count of ones decreasing from 100 to 0,1,2, etc. Also, when we increase the array size (e.g., to 10000), we can see that this nearly the same case. When we increase the number of threads from 1,2,4, to 16, we can see that the number of iterations dropped from 1000000 to 999453, with the number of accurate count of ones dropping from 100 to 11. Another experiment of this case gave us an accurate number of counts (100), but this time, the number of iterations was the true one, (i.e., 1000000). We can also notice that as the number of iterations decrease, the time taken is less than the actual one when we have the accurate number of iterations, and vice versa when we have a larger inaccurate number of iterations. This implies that the computer is skipping iterations in certain cases, and sometimes overcounting in others.**

**After the implementation of the Mutex feature (locking before the critical action, which is looping over the array and counting, then unlocking afterwards), we notice that the accurate counted ones are way higher than having the program done without the Mutex logic. However, this improvement is mostly witnessed when we have larger array sizes. For instance, when we increase the number of threads for arrays of small sizes (e.g., 100), we notice that the accurate count of ones drops sharply from 100 to 4 and 0 as the number of threads increase from 1 to 8. We can also notice that the number of iterations is less by 400 iterations than the actual one (9600 compared to 10000), with the average time taken dropping from around 1.12 seconds to 0.27 seconds. This supports our hypothesis from the previous case (race case with no Mutex implementation), where iterations are being skipped and thus result in inaccurate counting. As for arrays of large sizes (e.g., 10000, 1000000, etc.), we can notice that there’s a vast improvement where the increased number of threads do not result in the same number of errors and race conditions. This improvement varies as the size increases (i.e., for an array of size 10000, and to be tested with the number of threads being 64, it might result in iteration skipping. But for arrays of size 1000000, even if we change the number of threads to 64, the accuracy is mostly, if not always accurate). We can also see that the time taken increases compared to running the program without the mutex implementation. This is due to race condition management. As we can see in the plot below, the accuracy is higher than that of the first plot.**

**After the implementation of the private variable logic, we notice a significant improvement in the accuracy of the counted ones. This can be seen in the “Private” Excel sheet where we can increase the number of threads and the size of the array to large values and still maintain proper accuracy (e.g., array size of 1000000 and 32 threads). However, we can also witness that when utilizing many threads for arrays of small sizes, the accuracy drops significantly and can reach as low as 0. With all this consideration, we can still consider this method to be way more accurate compared to that without it as the race conditions are significantly lower due to complete (or high partial) absence of memory sharing between variables. As we can see in the plot below, the accuracy is higher than that of the first plot.**

**After the implementation of the L1 Cache’s logic in this exercise, we can see that there was also a significant increase in accuracy compared to the sequential count of ones. We must note though that the L1 Cache size of laptop is 184 KiloBytes, therefore it can not hold counts in an array of extremely large sizes. There are still some drawbacks when it comes to arrays of small sizes while having large numbers of threads. However, there is an improvement compared to previous experiments.**