Lebanese American University

School of Arts and Sciences

Department of Computer Science & Mathematics

CSC447: Parallel Programming for Multicore and Cluster Systems

Lab #2: Data Race

Karim El Jammal - 201903088

March 16, 2022

<u>Note:</u> The machine used has a Intel(R) Core(TM) i9-9900 CPU @ 3.10GHz 3.10 GHz processor and 32.0 GB of installed RAM

1.
   - Array size 1000:
     - Number of 3's: 108
     - Time it took: 0.000000 s
   - Array size 10000:
     - Number of 3's: 990
     - Time it took: 0.000000 s
   - Array size 100000:
     - Number of 3's: 10042
     - Time it took: 0.000000 s
   - Array size 10000000:
     - Number of 3's: 1001302
     - Time it took: 0.025000 s

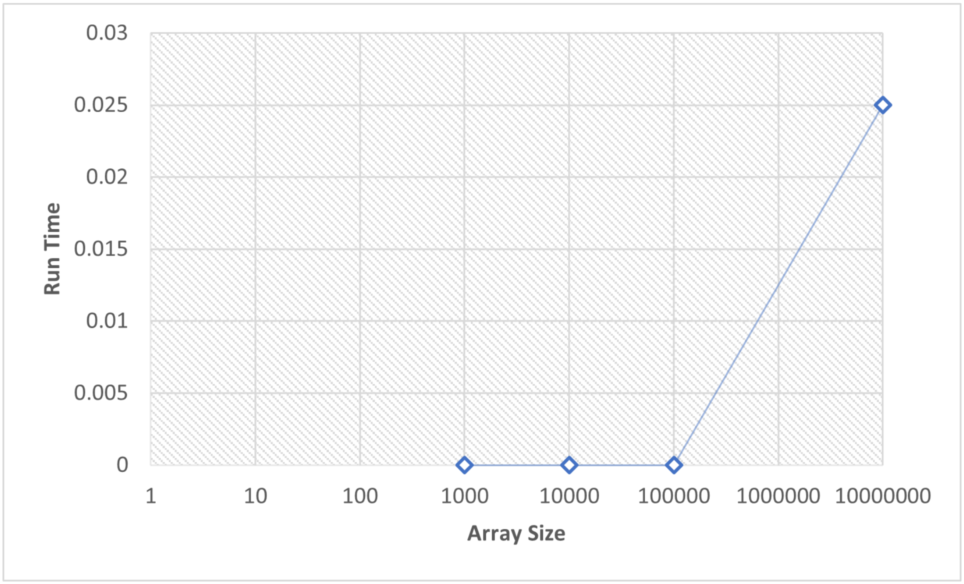

Figure 1: Plot showing the run time versus array size

2. The below results were obtained by running the code on an array of length 2^{20}

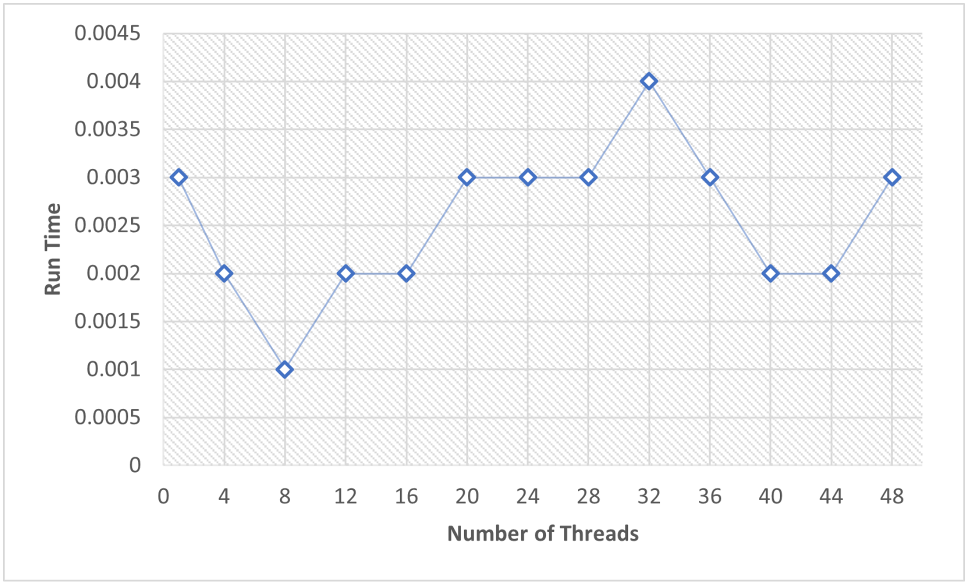

Figure 2: Plot showing the run time versus number of threads

Without padding or locks, the number of 3's counted for each thread were inconsistent with the correct output (run in serial mode).

3. The below results were obtained by running the code on an array of length 2^{20}

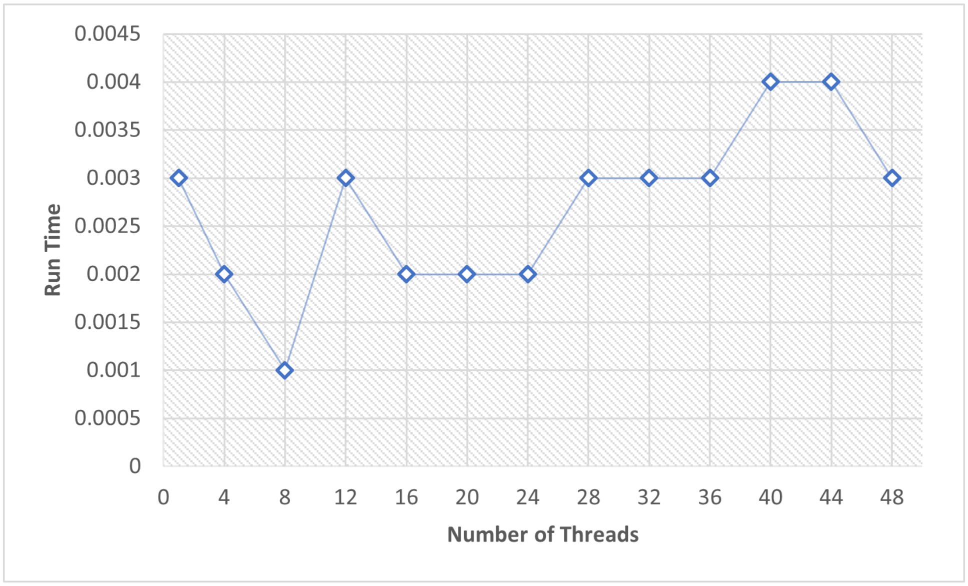

Figure 3: Plot showing the run time versus number of threads

Without padding, the number of 3's counted for each thread were consistent with the desired output (run in serial mode). However, computation took longer time.

4. The below results were obtained by running the code on an array of length 2^{20}

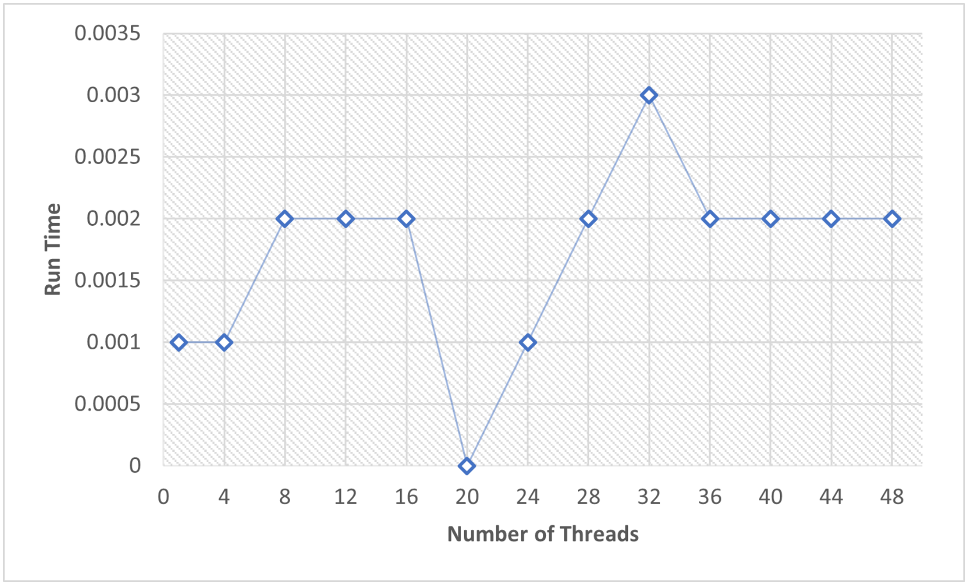

Figure 4: Plot showing the run time versus number of threads

With padding and locks, the number of 3's counted for each thread were consistent with the desired output (run in serial mode). In addition there was a significant decrease in time compared with when there was no padding.

5. When we added padding and locks to our code, we realize that the output is correct and it took much less time. This is due to the fact that we have dealt with both data race and false sharing condition. In this way, each thread has its own private counter and cache line. Threads wont have to worry about synchronization, hence the decrease in time. However, we can still observe that the run time in serial code is equal or in some cases even better than the one is parallel.