Homework 3: An Evaluation of Thread Synchronization Methods

All code as well as the “Readme” file may be found at https://github.com/TarbFela/COS331_HW3. They are also attached as a “.zip” file in the submission. Tables and a graph are included in this document on the pages below. The answers to the assignment questions are enumerated in the following paragraphs.

Question 1:
I believe in race conditions; when a thread or process attempts to update a value in memory, it will copy, modify, and write. If another process does the same during this time, the effects of the new modification will not be as if both were made; one will be cancelled out.
Question 2:
Test and Set uses CPU time to check on the status and the mutex waits to be **told** that the status is writable. The CPU can therefore spend its clock cycles on the math that needs to happen for the variable to be freed up.
Question 3:
As before, more busy waiting generally increases compute time. Additionally, more threads “claiming” the variable means a high likelihood of the variable being “claimed” when another thread goes to check it. So, the wait time increases as well.
Question 4:
This method actually seems to have been slower than any of the others and by quite a margin. This is likely a result of me not fully understanding how the `clock()` function works when threads are instantiated. Perhaps I should be timing the process externally with a shell command like “time”.

Tables & Graph

Table I: No Synchronization

Threads	Points	Pi	Time
2	5E+08	3.14158	11.058
8	5E+08	3.14153	11.884
16	5E+08	3.14164	11.914
2	5E+07	3.14173	1.120
8	5E+07	3.14148	1.246
16	5E+07	3.14148	1.257
2	1E+06	3.14138	0.034
8	1E+06	3.14225	0.023
16	1E+06	2.94655	0.031

Table II: Allocating Memory

Threads	Points	Pi	Time
2	5E+08	3.14156	11.219
8	5E+08	3.14168	15.508
16	5E+08	3.14156	15.682
2	5E+07	3.14188	1.148
8	5E+07	3.14155	1.592
16	5E+07	3.14155	1.634
2	1E+06	3.14271	0.032
8	1E+06	3.13934	0.039
16	1E+06	3.13916	0.033

Table III: Mutex Synchronization

Threads	Points	Pi	Time
2	5E+08	3.14163	11.050
8	5E+08	3.14163	11.872
16	5E+08	3.14155	11.979
2	5E+07	3.14213	1.122
8	5E+07	3.14155	1.247
16	5E+07	3.14104	1.236
2	1E+06	3.14092	0.032
8	1E+06	3.14181	0.031
16	1E+06	3.14381	0.024

Table IV: Test-and-Set Synchronization

Threads	Points	Pi	Time
2	5E+08	3.14165	11.018
8	5E+08	3.14158	11.931
16	5E+08	3.14151	11.896
2	5E+07	3.14164	1.128
8	5E+07	3.14149	1.271
16	5E+07	3.14126	1.247
2	1E+06	3.13861	0.033
8	1E+06	3.14290	0.026
16	1E+06	3.14102	0.031

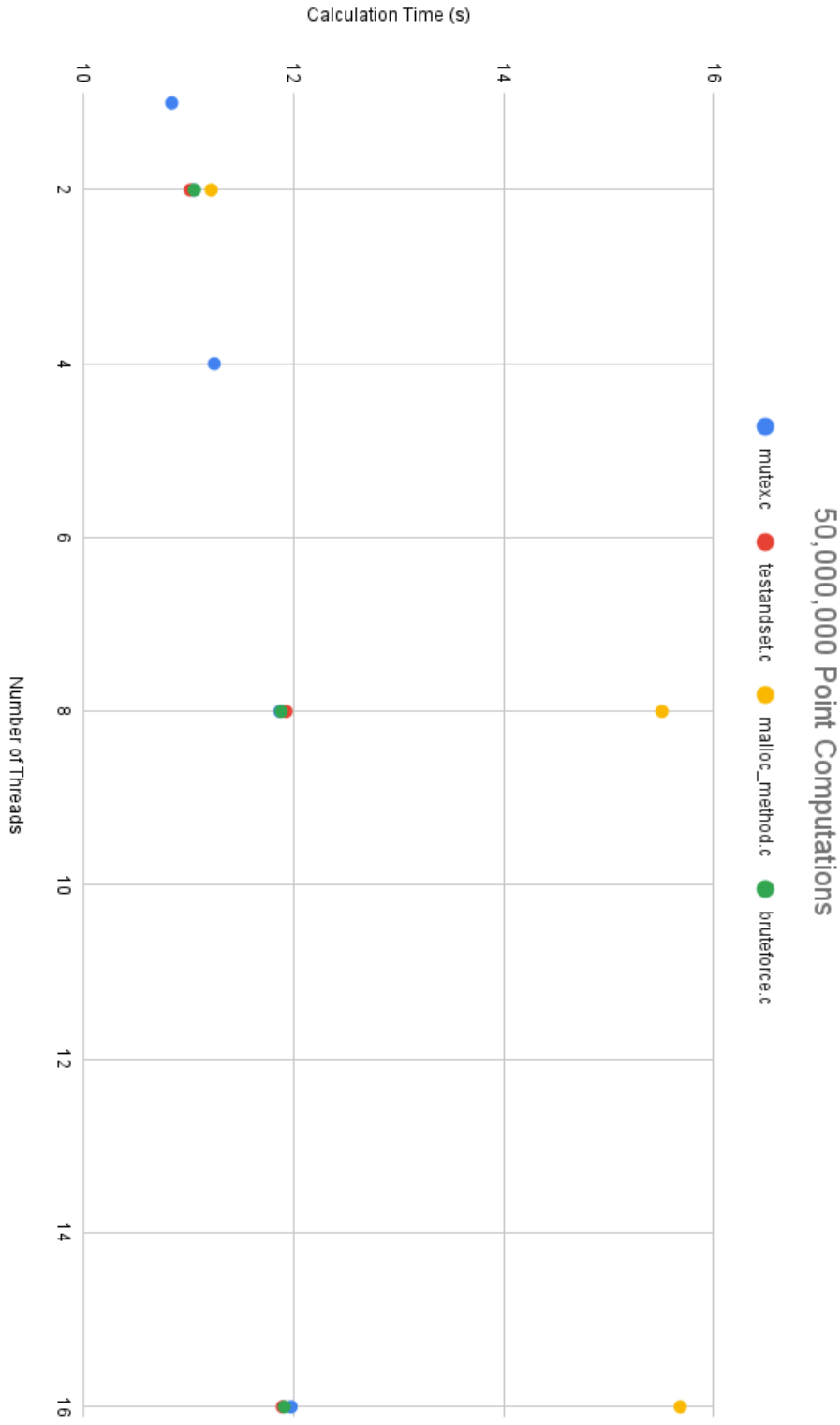

Fig. 1. Graph of execution times for various synchronization programs against the number of threads being used.