

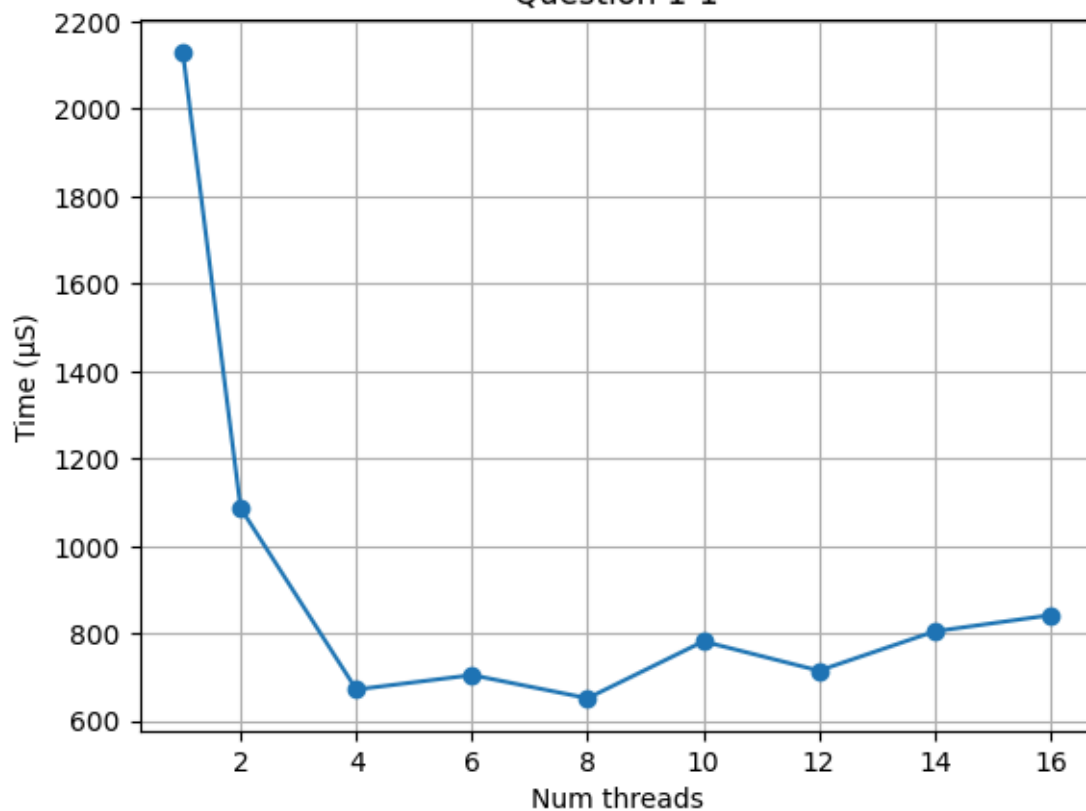
CSD204 - OS - Lab04

Lalit Maurya, 2310110164

Question 01

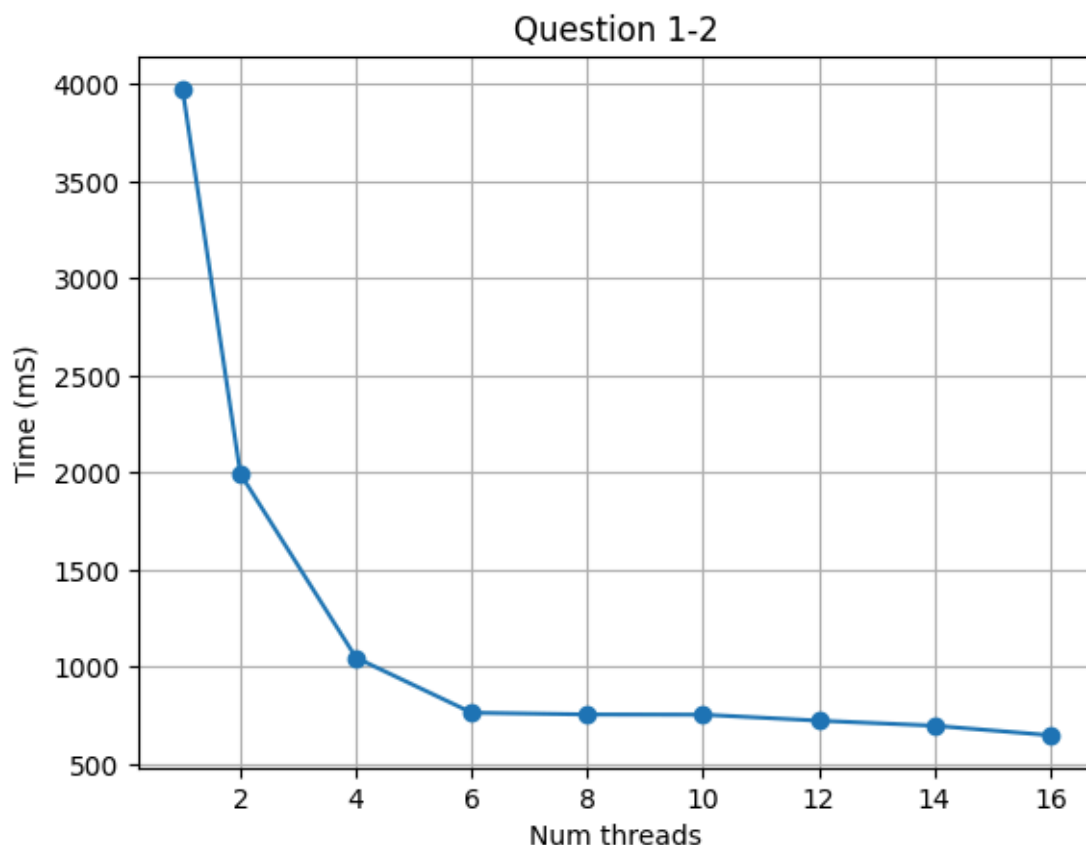
```
..D204-OS/lab04
~/devEnv/snu/CSD204-OS/lab04 main ?2
> make run q01_1
✓ 1 calculated accurate sum in 2130µS
✓ 2 calculated accurate sum in 1087µS
✓ 4 calculated accurate sum in 672µS
✓ 6 calculated accurate sum in 705µS
✓ 8 calculated accurate sum in 652µS
✓ 10 calculated accurate sum in 782µS
✓ 12 calculated accurate sum in 715µS
✓ 14 calculated accurate sum in 805µS
✓ 16 calculated accurate sum in 842µS
make: 'q01_1' is up to date.
```

Question 1-1



Here we can see that while a greater number of threads results in lower execution times, there's a limit to this. Due to the overhead of creating more threads, the execution time rises when more threads are created after a certain point. To better visualize the effect of more threads, I wrote another program that sums up numbers from 0 to 10^{15} . Here is the output for the same:

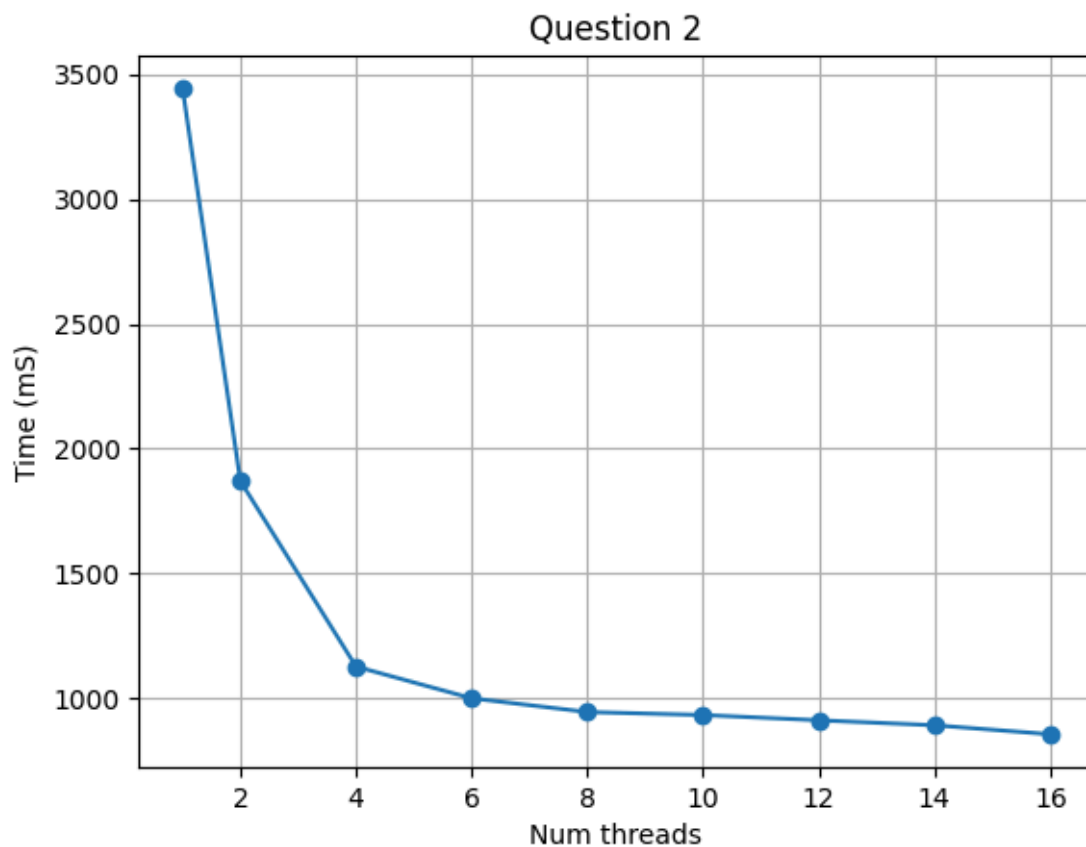
```
~/devEnv/snu/CSD204-OS/lab04 main 72
> make run q01_2
✓ 1 Threads calculated accurate sum in 3975mS
✓ 2 Threads calculated accurate sum in 1994mS
✓ 4 Threads calculated accurate sum in 1047mS
✓ 6 Threads calculated accurate sum in 764mS
✓ 8 Threads calculated accurate sum in 755mS
✓ 10 Threads calculated accurate sum in 754mS
✓ 12 Threads calculated accurate sum in 723mS
✓ 14 Threads calculated accurate sum in 697mS
✓ 16 Threads calculated accurate sum in 648mS
make: 'q01_2' is up to date.
```



Now we can clearly see the decrease in execution time with an increase in number of threads.

Question 02

```
~/devEnv/snu/CSD204-05/lab04 main ?2
make run q02
✓ 1 Threads successfully sorted array in 3445mS
✓ 2 Threads successfully sorted array in 1870mS
✓ 4 Threads successfully sorted array in 1126mS
✓ 6 Threads successfully sorted array in 999mS
✓ 8 Threads successfully sorted array in 944mS
✓ 10 Threads successfully sorted array in 932mS
✓ 12 Threads successfully sorted array in 911mS
✓ 14 Threads successfully sorted array in 891mS
✓ 16 Threads successfully sorted array in 855mS
make: 'q02' is up to date.
```



The size of the array I am sorting is 10,000,000 (10 million) to avoid the same problem as in question1. We see the exact thing we expect, which is a decrease in execution times when the number of threads increases.

Question 03

```
~/devEnv/snu/CSD204-OS/lab04 main ?2
> make run q03
Expected | Without Lock | With Lock
-----+-----+-----
20000000 | 1269670 | 20000000
make: 'q03' is up to date.
```

The output is clear, a lack of mutex locks leads to wildly inaccurate results while a proper usage yields a perfect output.