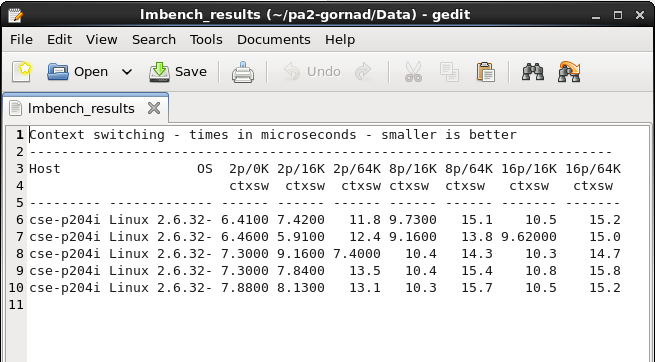
3.2.1)



Empirical average and variance of kernel threads performance:

Average = (6.4100+6.4600+7.3000+7.3000+7.8800)/5 = 7.07 µs

Variance = 0.3924 (using excel)

The number of context switches so far is: 1

The time taken by the context switch is: 2025 = 2.025 µs

The number of context switches so far is: 2

The time taken by the context switch is: 2095 = 2.095 µs

The number of context switches so far is: 3

The time taken by the context switch is: 1886 = 1.886 µs

The number of context switches so far is: 4

The time taken by the context switch is: 2374 = 2.374 µs

The number of context switches so far is: 5

The time taken by the context switch is: 2095 = 2.095 µs

The number of context switches so far is: 6

The time taken by the context switch is: 2165 = 2.165 µs

The number of context switches so far is: 7

The time taken by the context switch is: 1886 = 1.886 µs

The number of context switches so far is: 8

The time taken by the context switch is: 2025 = 2.025 µs

The number of context switches so far is: 9

The time taken by the context switch is: 1816 = 1.816 µs

The number of context switches so far is: 10

The time taken by the context switch is: 2025 = 2.025 µs

The number of context switches so far is: 11

The time taken by the context switch is: 1257 = 1.257 µs

Empirical average and variance:

Average = (2025 + 2095 + 1886 + 2374 + 2095 + 2165 + 1886 + 2025 + 1816 + 2025 + 1257)/11 = 1968 ns = 1.968 µs

Variance = 78772.29 (using excel)

3.2.2)

|  |  |
| --- | --- |
| Kernel Threads Sleeping() | User Threads Sleeping() |
| 17463847 | 17632545 |
| 16270248 | 19306493 |
| 17177640 | 22372854 |
| 17669386 | 28898776 |
| 17209392 | 43898551 |
| 17069643 | 72135502 |

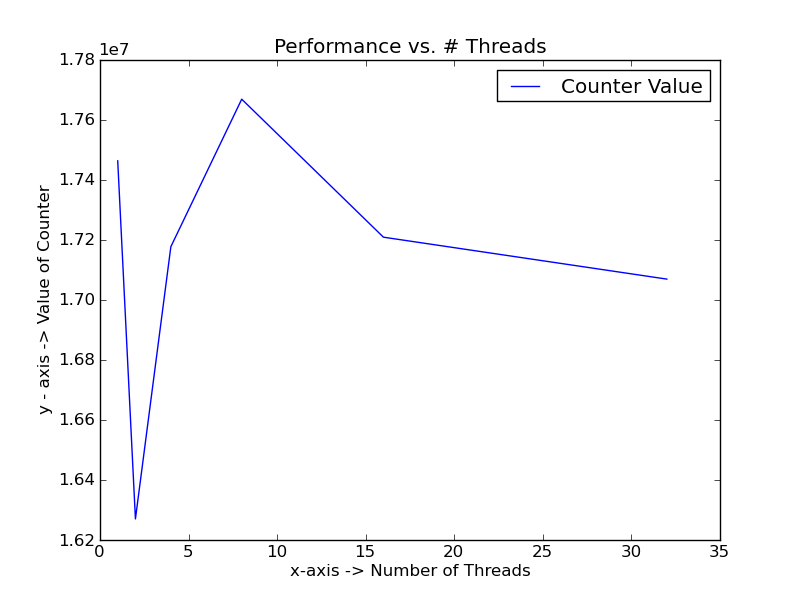


Image 1: Kernel threads sleeping()

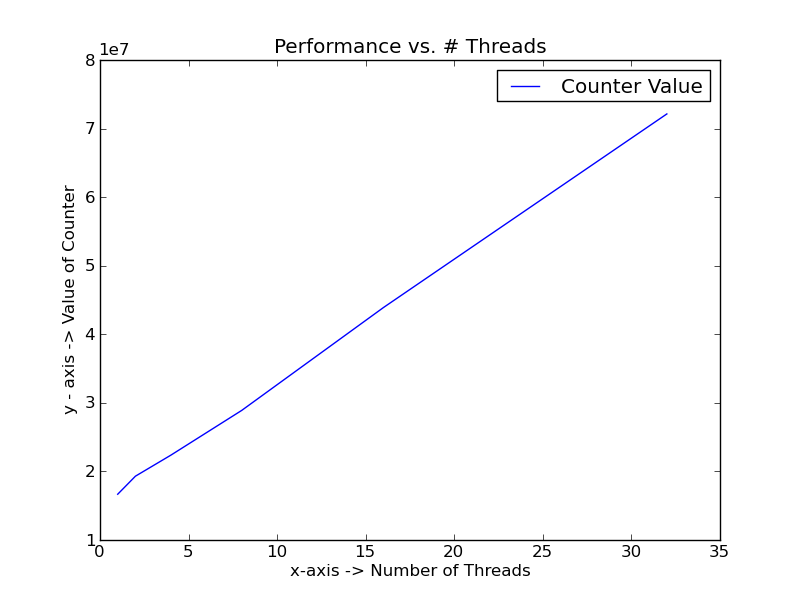


Image 2: User threads sleeping()

Note: Both user and kernel thread tests were tested for an increasing number of threads similar to 3.2.3 to produce more reliable results.

For kernel threads, we notice that performance is extremely unreliable. This is caused by the expensive context switching times coupled with occupied system resources and scheduling. For user level threads, we notice a consistent increase of performance that is due to the relatively inexpensive context switching times of user threads.

3.2.3)

|  |  |
| --- | --- |
| Kernel Threads Counter() | User Threads Counter() |
| 17329426 | 17656769 |
| 18020844 | 18867391 |
| 17065985 | 22594306 |
| 16195997 | 29240851 |
| 16545564 | 43117937 |
| 17784652 | 71651928 |

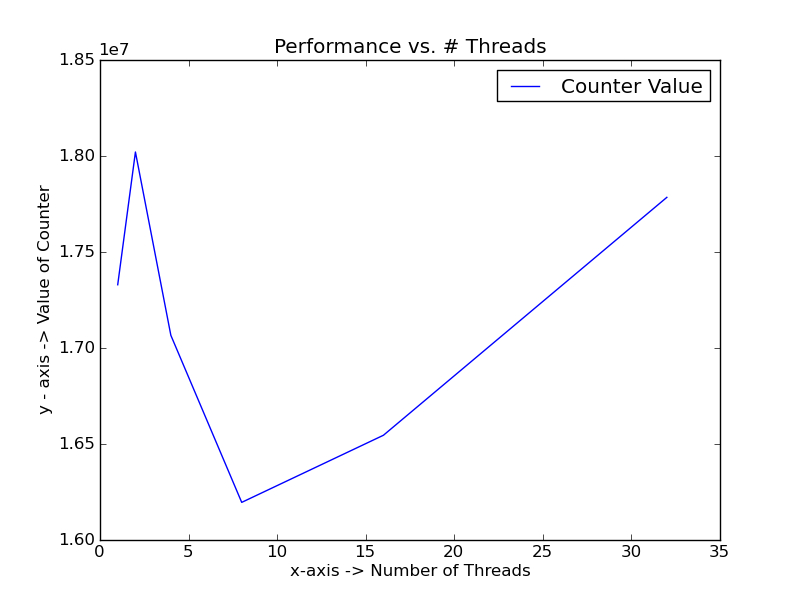


Image 3: Kernel threads counter()

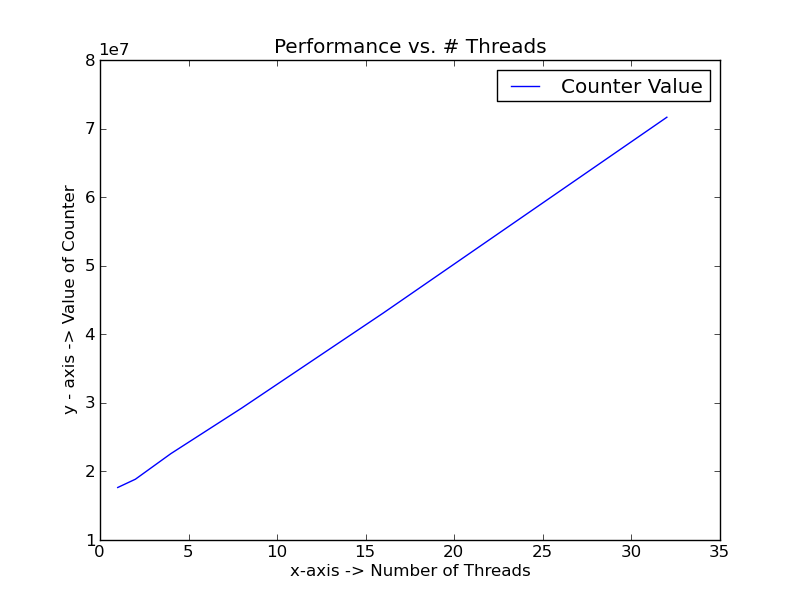


Image 4: User threads counter()

Since the counter() function is less expensive for kernel threads, we do not see as much inconsistency as with the sleeping() function. For both graphs, we notice an increase of work done in relation to the number of threads. There is a correlation between the number of cores in the system and the number of threads. This is explained by parallel processing. More cores mean more threads can be handled efficiently leading to an increased performance.