Test 1: LaplacianStencil\_0\_0

Maximum Threads possible = 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Run iteration/ num of threads | Threads = 1 | Threads = 2 | Threads = 3 | Threads = 4 |
| iteration 1 | 2000.75ms | 1017.22ms | 695.872ms | 537.731ms |
| iteration 2 | 1552.5ms | 775.542ms | 526.67ms | 411.383ms |
| iteration 3 | 1550.72ms | 776.352ms | 526.091ms | 449.335ms |
| iteration 4 | 1549.39ms | 774.021ms | 526.653ms | 416.915ms |
| iteration 5 | 1550.4ms | 777.596ms | 526.558ms | 408.184ms |
| iteration 6 | 1550.37ms | 781.226ms | 527.16ms | **405.345ms** |
| iteration 7 | 1550.17ms | 780.577ms | 527.228ms | 411.666ms |
| iteration 8 | 1549.86ms | 774.971ms | 526.641ms | 406.143ms |
| iteration 9 | 1550.76ms | 770.959ms | 526.328ms | 405.903ms |
| iteration 10 | 1550.68ms | 769.897ms | 526.593ms | 409.567ms |

Best performance: (4 threads) **405.345ms**

Test 1: LaplacianStencil\_0\_1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Run iteration/ size | Size = 16K x 16K | Size = 8K x 8K | Size = 4K x 4K | Size = 2K x 2K |
| iteration 1 | 1830.41ms | 449.442ms | 116.314ms | 37.3346ms |
| iteration 2 | 1333.4ms | 324.049ms | 80.5301ms | 20.1789ms |
| iteration 3 | 1331.43ms | 323.151ms | 80.5584ms | 20.1136ms |
| iteration 4 | 1331.62ms | 323.21ms | 80.8332ms | 20.1338ms |
| iteration 5 | 1323.41ms | 324.193ms | 80.6698ms | 20.0842ms |
| iteration 6 | 1326.75ms | 323.324ms | 80.4361ms | 20.1973ms |
| iteration 7 | 1301.01ms | 323.371ms | 81.6893ms | 20.095ms |
| iteration 8 | 1290.55ms | 324.093ms | 80.772ms | 20.1168ms |
| iteration 9 | 1291.59ms | 323.036ms | 80.5481ms | 20.1272ms |
| iteration 10 | 1290.56ms | 323.161ms | 80.5182ms | 20.1591ms |
| Avg Time: | 1365.07 | 336.103 | 84.2869 | 21.8541 |

16K x 16K = 2^14 \* 2^14 = 2^28 | 8K x 8K = 2^13 \* 2^13 = 2^26 | 4K x 4K = 2^12 \* 2^12 = 2^24

2K x 2K = 2^11 \* 2^11 = 2^22 | Each size has a difference factor of 2^2 = 4

The running time relative to each other is expected since each Avg is relatively 4 times slower/faster than the one after/before it. However, the difference in running time between test 00 and 01 is not expected especially when using only 1 Thread. Test 01 doesn’t use openMP, thus it will be run sequentially, which is equivalent to 1 thread. At the same time in test 00, using 4 thread means that the work will be split into four sections and each section has a size of 8K x 8K.

We have two sets of data that we can compare each other to. (Thread 1 and Size 16K x 16K | Thread 4 and size 8K x 8K)  
My only explanation is that using openMP will alter the compiler a bit and make it more **suspicion** and run a bit more **careful**. Thus, using openMP will tax the CPU with additional work which we can see in (Thread 1 and Size 16K x 16K) being ~200ms and in (Thread 4 and size 8K x 8K) being ~90ms.

Comment: This is very insignificant, but it is noticeable in our examples.