```
Running seed 2 with 1 threads and a range of 50000
The number of composite numbers in the array is 89731177/100000000
real    0m2.827s
user    0m2.737s
sys     0m0.085s

Running seed 2 with 2 threads and a range of 50000
The number of composite numbers in the array is 89731177/100000000
real    0m1.757s
user    0m2.692s
sys     0m0.076s

Running seed 2 with 4 threads and a range of 50000
The number of composite numbers in the array is 89731177/100000000
real    0m1.316s
user    0m2.911s
sys     0m0.073s

Running seed 2 with 8 threads and a range of 50000
The number of composite numbers in the array is 89731177/100000000
real    0m1.139s
user    0m3.462s
sys     0m0.089s

Running seed 2 with 16 threads and a range of 50000
The number of composite numbers in the array is 89731177/100000000
real    0m1.068s
user    0m4.128s
sys     0m0.073s

Running seed 2 with 32 threads and a range of 50000
The number of composite numbers in the array is 89731177/100000000
real    0m1.026s
user    0m3.969s
sys     0m0.070s

Running seed 2 with 64 threads and a range of 50000
The number of composite numbers in the array is 89731177/100000000
real    0m1.010s
user    0m3.730s
sys     0m0.085s

Running seed 2 with 128 threads and a range of 50000
The number of composite numbers in the array is 89731177/100000000
real    0m0.989s
user    0m3.556s
sys     0m0.082s

Running seed 2 with 256 threads and a range of 50000
The number of composite numbers in the array is 89731177/100000000
real    0m0.996s
user    0m3.556s
sys     0m0.081s
```

As we can see, we generate 100,000,000 numbers within a range of 0-50000 on seed 2. As we increase the thread count, the program will execute faster as the workload is being split amongst the threads. However, we start to notice less of a performance increase as the thread count increases; this could be since the code is being run on the school's server and the CPU simply doesn't have enough threads or the workload is being split to the point where it isn't efficient anymore. We may not be creating enough work for the threads. This increase in performance is expected, and we can see an exponential decrease in the performance gain using more threads. Overall, increasing threads and splitting workload will increase performance and decrease the run time of a program. However, we must be wary that increasing the thread count too high could lead to other bottlenecks, thus making the performance gain minimal in higher thread situations.