Programming Assignment #2

1. Hardware

The machine used for compilation, execution, and benchmarking was the 2021 Macbook Pro with the Apple M1 pro chip.

- 8-core CPU with 6 performance cores and 2 efficiency cores
- 14-core GPU
- 16-core Neural Engine
- 200GB/s max memory bandwidth

2. Operating System and Compilation:

The operating system was MacOS Monterrey 12.5.

The gcc provided by apple did not support openmp so I had to install another version with homebrew. The optimization flag -O3 was used to make the performance comparable to that of an intel compiler. Using this flag meant that there was a smaller difference between the timings of the combined kernel and the individual kernels. However, the percentage change between the timings remained approximately the same.

Commands used for compilation and execution:

```
g++-12 -fopenmp -O3 *.cpp
./a.out
```

3. Code Changes

Timing kernels:

A Timer object was made for each occurrence of a kernel. For example, for the individual kernels, each occurrence of ComputeLaplacian() had its own timer object. For the combined kernel code, each combined kernel had its own timer. The timer methods Start, Restart and Pause were used. A method called getElapsedTime was also defined in “Timer.h” to get all the timing information after executing conjugate gradients algorithm. The information was then formatted and printed out. The kernels that did not execute inside the for loop did not have a “per iteration” section when printing out their timing information.

The total duration of the algorithm was also timed. This timer was started after the intialization and stopped immediately after the Saxpy operation on line 10 is executed. This total duration is printed out along with the sum of the times from each kernel to visually confirm that the timings are accurate.

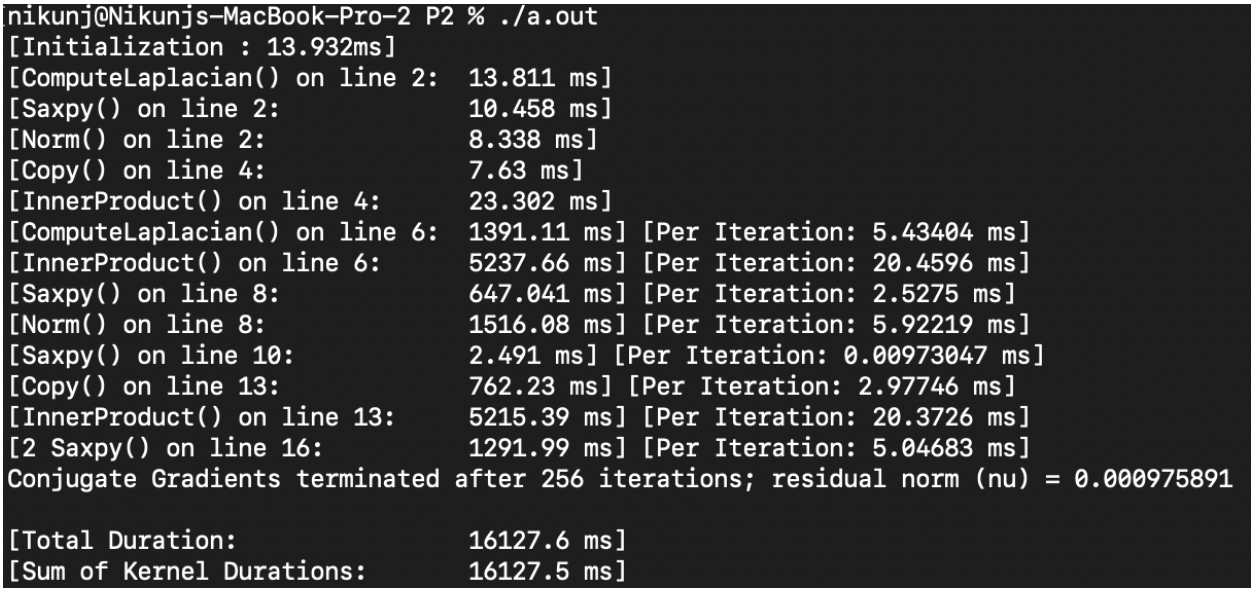
Combining Kernels:

Two new files called “CombinedKernels.h” and “CombinedKernels.cpp” were created. The header was included in the “ConjugateGradients.cpp” file. There were 5 combined kernels for 6 lines. The combined kernel for lines 4 and 13 were the same.

Each combined kernel aimed to improve performance by executing all the individual kernels in a single pass. The code for the saxpy kernel was also optimized with openmp since the special case of arguments being aliases of each other was being dealt with individually.

4. Performance Figures

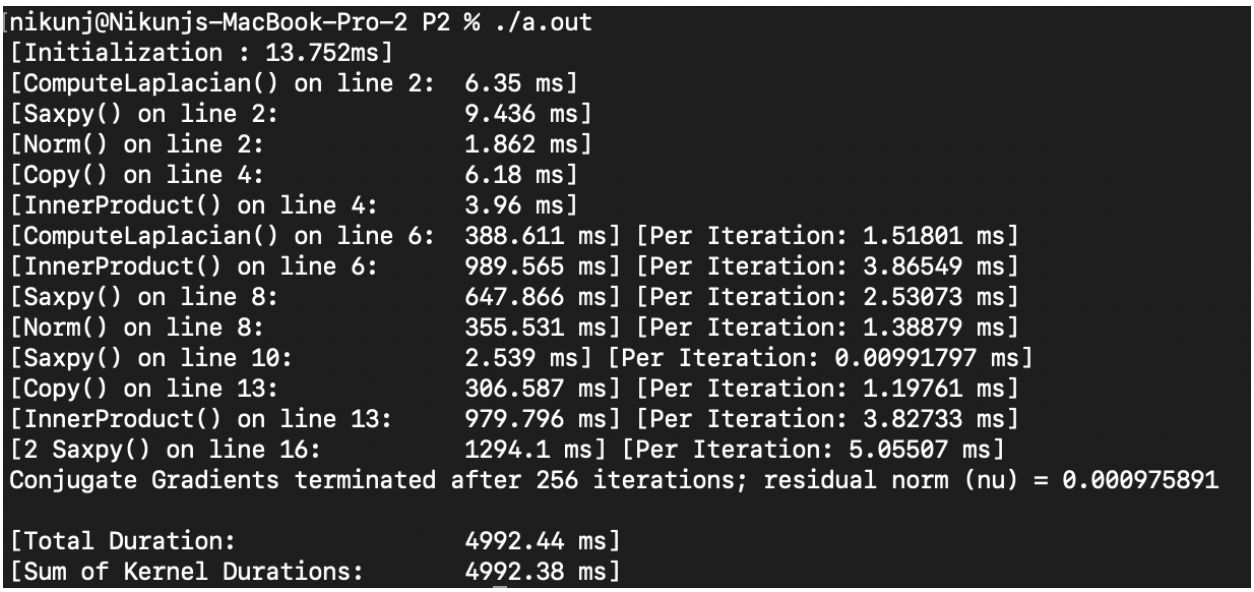
Figure 1: Serial Run, No change to kernels



```
nikunj@Nikunjs-MacBook-Pro-2 P2 % ./a.out
[Initialization : 13.932ms]
[ComputeLaplacian() on line 2:  13.811 ms]
[Saxpy() on line 2:             10.458 ms]
[Norm() on line 2:              8.338 ms]
[Copy() on line 4:              7.63 ms]
[InnerProduct() on line 4:      23.302 ms]
[ComputeLaplacian() on line 6:  1391.11 ms] [Per Iteration: 5.43404 ms]
[InnerProduct() on line 6:      5237.66 ms] [Per Iteration: 20.4596 ms]
[Saxpy() on line 8:             647.041 ms] [Per Iteration: 2.5275 ms]
[Norm() on line 8:              1516.08 ms] [Per Iteration: 5.92219 ms]
[Saxpy() on line 10:            2.491 ms] [Per Iteration: 0.00973047 ms]
[Copy() on line 13:             762.23 ms] [Per Iteration: 2.97746 ms]
[InnerProduct() on line 13:     5215.39 ms] [Per Iteration: 20.3726 ms]
[2 Saxpy() on line 16:          1291.99 ms] [Per Iteration: 5.04683 ms]
Conjugate Gradients terminated after 256 iterations; residual norm (nu) = 0.000975891

[Total Duration:                16127.6 ms]
[Sum of Kernel Durations:       16127.5 ms]
```

Figure 2: Parallel Run, No change to kernels



```
nikunj@Nikunjs-MacBook-Pro-2 P2 % ./a.out
[Initialization : 13.752ms]
[ComputeLaplacian() on line 2:  6.35 ms]
[Saxpy() on line 2:             9.436 ms]
[Norm() on line 2:              1.862 ms]
[Copy() on line 4:              6.18 ms]
[InnerProduct() on line 4:      3.96 ms]
[ComputeLaplacian() on line 6:  388.611 ms] [Per Iteration: 1.51801 ms]
[InnerProduct() on line 6:      989.565 ms] [Per Iteration: 3.86549 ms]
[Saxpy() on line 8:             647.866 ms] [Per Iteration: 2.53073 ms]
[Norm() on line 8:              355.531 ms] [Per Iteration: 1.38879 ms]
[Saxpy() on line 10:            2.539 ms] [Per Iteration: 0.00991797 ms]
[Copy() on line 13:             306.587 ms] [Per Iteration: 1.19761 ms]
[InnerProduct() on line 13:     979.796 ms] [Per Iteration: 3.82733 ms]
[2 Saxpy() on line 16:          1294.1 ms] [Per Iteration: 5.05507 ms]
Conjugate Gradients terminated after 256 iterations; residual norm (nu) = 0.000975891

[Total Duration:                4992.44 ms]
[Sum of Kernel Durations:       4992.38 ms]
```

Figure 3: Serial Run, Combined Kernels

```
nikunj@Nikunjs-MacBook-Pro-2 P2 % ./a.out
[Initialization : 14.44ms]
[Combined Kernel on line 2:     34.341 ms]
[Combined Kernel on line 4:     27.093 ms]
[Combined Kernel on line 6:     5255.82 ms] [Per Iteration: 20.5305 ms]
[Combined Kernel on line 8:     2521.49 ms] [Per Iteration: 9.84957 ms]
[Saxpy() on line 10:            2.506 ms] [Per Iteration: 0.00978906 ms]
[Combined Kernel on line 13:    5206.92 ms] [Per Iteration: 20.3395 ms]
[Combined Kernel on line 16:    1043.38 ms] [Per Iteration: 4.0757 ms]
Conjugate Gradients terminated after 256 iterations; residual norm (nu) = 0.000975891

[Total Duration:                14091.6 ms]
[Sum of Kernel Durations:       14091.5 ms]
```

Figure 4: Parallel Run, Combined Kernels

```
nikunj@Nikunjs-MacBook-Pro-2 P2 % ./a.out
[Initialization : 15.708ms]
[Combined Kernel on line 2:     11.851 ms]
[Combined Kernel on line 4:     7.285 ms]
[Combined Kernel on line 6:     1200.86 ms] [Per Iteration: 4.69086 ms]
[Combined Kernel on line 8:     551.062 ms] [Per Iteration: 2.15259 ms]
[Saxpy() on line 10:            2.522 ms] [Per Iteration: 0.00985156 ms]
[Combined Kernel on line 13:    997.5 ms] [Per Iteration: 3.89648 ms]
[Combined Kernel on line 16:    526.219 ms] [Per Iteration: 2.05554 ms]
Conjugate Gradients terminated after 256 iterations; residual norm (nu) = 0.000975891

[Total Duration:                3297.38 ms]
[Sum of Kernel Durations:       3297.3 ms]
```

5. Comments

A significant improvement was seen in both the serial and parallel cases when combining the kernels. Each of the combined kernels showed a better timing than the sum of its individual counterparts. In particular, a big improvement was seen in the combined kernels that involved the Saxpy operation because it was now optimized with openmp.

As a result, there was a 12.5% improvement for the serial run (Figures 1 and 3) and a 34% improvement for the parallel run (Figures 2 and 4). A majority of this disparity can be credited to the Saxpy operations being optimized in the combined kernels version.

This would be seen clearly in parallel execution and not serial execution because executing on 1 thread does not allow the major benefits of the openmp optimization.

Reducing the memory usage by executing a single pass in combined kernels significantly improved performance.