CSCI-GA.2945-002 : High Performance Computing **Homework:** 4
Date: 04/17/2022
Name: Yajur Ahuja **NetID:** ya2109

The link to the github repository is `https://github.com/yajurahuja/High-Performance-Computing`
As the different GPUs have different configurations, I have in the images also included the exact commmand to run the code. Please use that to replicate the results. All of my codes have been tested in Cuda 3 (CIMS server).

4.1 Matrix-vector operations on a GPU

The below images shows Vector-vector and Matrix-vector operations operations on GPU I have put screen shots so you can also see the commands to run the same if you want to test and verify the output.

1. This was ran on Cuda1

```
[ya2109@cuda1 hw4]$ nvcc matvec.cu -Xcompiler -fopenmp && ./a.out
Testing Dot Product: CPU vs GPU
Size: 10000 Error: 1.227818e-11 CPU Time: 0.000028 GPU Time: 0.000104 Bandwidth: 2.296716 Gb/s Speedup: 0.266448
Size: 20000 Error: 1.818989e-11 CPU Time: 0.000066 GPU Time: 0.000058 Bandwidth: 8.264039 Gb/s Speedup: 1.143278
Size: 30000 Error: 4.820322e-11 CPU Time: 0.000083 GPU Time: 0.000073 Bandwidth: 9.856266 Gb/s Speedup: 1.132019
Size: 40000 Error: 7.821654e-11 CPU Time: 0.000110 GPU Time: 0.000087 Bandwidth: 10.998577 Gb/s Speedup: 1.265123
Size: 50000 Error: 1.491571e-10 CPU Time: 0.000138 GPU Time: 0.000103 Bandwidth: 11.606869 Gb/s Speedup: 1.335428
Size: 60000 Error: 7.457857e-11 CPU Time: 0.000171 GPU Time: 0.000114 Bandwidth: 12.615868 Gb/s Speedup: 1.498686
Size: 70000 Error: 2.510205e-10 CPU Time: 0.000194 GPU Time: 0.000137 Bandwidth: 12.244005 Gb/s Speedup: 1.415567
Size: 80000 Error: 1.673470e-10 CPU Time: 0.000222 GPU Time: 0.000146 Bandwidth: 13.136737 Gb/s Speedup: 1.518183
Size: 90000 Error: 1.091394e-10 CPU Time: 0.000255 GPU Time: 0.000163 Bandwidth: 13.247227 Gb/s Speedup: 1.563210


Testing Matrix Vector Product: CPU vs GPU
Size: 1000 * 1000 Error: 1.330136e-11 CPU Time: 0.003113 GPU Time: 0.003265 Bandwidth: 0.612874 Gb/s Speedup: 0.953590
Size: 3000 * 3000 Error: 1.226681e-10 CPU Time: 0.028629 GPU Time: 0.013902 Bandwidth: 1.294958 Gb/s Speedup: 2.059257
Size: 5000 * 5000 Error: 3.185505e-10 CPU Time: 0.079557 GPU Time: 0.029847 Bandwidth: 1.675379 Gb/s Speedup: 2.665485
Size: 7000 * 7000 Error: 6.464234e-10 CPU Time: 0.155812 GPU Time: 0.051146 Bandwidth: 1.916214 Gb/s Speedup: 3.046402
Size: 9000 * 9000 Error: 1.090939e-09 CPU Time: 0.257759 GPU Time: 0.078085 Bandwidth: 2.074772 Gb/s Speedup: 3.300993
Size: 11000 * 11000 Error: 1.622539e-09 CPU Time: 0.386910 GPU Time: 0.117221 Bandwidth: 2.064578 Gb/s Speedup: 3.300702
Size: 13000 * 13000 Error: 2.272827e-09 CPU Time: 0.536700 GPU Time: 0.157068 Bandwidth: 2.152023 Gb/s Speedup: 3.417004
Size: 15000 * 15000 Error: 3.108198e-09 CPU Time: 0.717954 GPU Time: 0.217967 Bandwidth: 2.064600 Gb/s Speedup: 3.293863
Size: 17000 * 17000 Error: 3.966306e-09 CPU Time: 0.926198 GPU Time: 0.343770 Bandwidth: 1.681407 Gb/s Speedup: 2.694238
Size: 19000 * 19000 Error: 4.873073e-09 CPU Time: 1.182788 GPU Time: 0.502584 Bandwidth: 1.436613 Gb/s Speedup: 2.353413
[ya2109@cuda1 hw4]$
```

Figure 4.1.1: Vector-vector and Matrix-vector operations

2. This was ran on Cuda2

```
[ya2109@cuda2 hw4]$ clear

[ya2109@cuda2 hw4]$ nvcc -arch=sm_61 matvec.cu -Xcompiler -fopenmp && ./a.out
Testing Dot Product: CPU vs GPU
Size: 10000 Error: 1.227818e-11 CPU Time: 0.000028 GPU Time: 0.000055 Bandwidth: 4.369675 Gb/s Speedup: 0.506500
Size: 20000 Error: 1.818989e-11 CPU Time: 0.000055 GPU Time: 0.000049 Bandwidth: 9.870451 Gb/s Speedup: 1.137981
Size: 30000 Error: 4.820322e-11 CPU Time: 0.000083 GPU Time: 0.000062 Bandwidth: 11.605232 Gb/s Speedup: 1.335053
Size: 40000 Error: 7.821654e-11 CPU Time: 0.000111 GPU Time: 0.000078 Bandwidth: 12.244277 Gb/s Speedup: 1.411192
Size: 50000 Error: 1.491571e-10 CPU Time: 0.000138 GPU Time: 0.000094 Bandwidth: 12.761751 Gb/s Speedup: 1.469962
Size: 60000 Error: 7.457857e-11 CPU Time: 0.000166 GPU Time: 0.000103 Bandwidth: 14.013642 Gb/s Speedup: 1.613496
Size: 70000 Error: 2.510205e-10 CPU Time: 0.000193 GPU Time: 0.000125 Bandwidth: 13.474281 Gb/s Speedup: 1.550537
Size: 80000 Error: 1.673470e-10 CPU Time: 0.000221 GPU Time: 0.000134 Bandwidth: 14.379976 Gb/s Speedup: 1.654266
Size: 90000 Error: 1.091394e-10 CPU Time: 0.000249 GPU Time: 0.000151 Bandwidth: 14.338536 Gb/s Speedup: 1.649934


Testing Matrix Vector Product: CPU vs GPU
Size: 1000 * 1000 Error: 1.330136e-11 CPU Time: 0.003122 GPU Time: 0.001590 Bandwidth: 1.258753 Gb/s Speedup: 1.963915
Size: 3000 * 3000 Error: 1.226681e-10 CPU Time: 0.028376 GPU Time: 0.009108 Bandwidth: 1.976680 Gb/s Speedup: 3.115552
Size: 5000 * 5000 Error: 3.185505e-10 CPU Time: 0.079229 GPU Time: 0.022056 Bandwidth: 2.267204 Gb/s Speedup: 3.592208
Size: 7000 * 7000 Error: 6.464234e-10 CPU Time: 0.154933 GPU Time: 0.040472 Bandwidth: 2.421596 Gb/s Speedup: 3.828155
Size: 9000 * 9000 Error: 1.090939e-09 CPU Time: 0.256141 GPU Time: 0.064488 Bandwidth: 2.512250 Gb/s Speedup: 3.971943
Size: 11000 * 11000 Error: 1.622539e-09 CPU Time: 0.382039 GPU Time: 0.093991 Bandwidth: 2.574843 Gb/s Speedup: 4.064650
Size: 13000 * 13000 Error: 2.272827e-09 CPU Time: 0.533371 GPU Time: 0.128598 Bandwidth: 2.628453 Gb/s Speedup: 4.147595
Size: 15000 * 15000 Error: 3.108198e-09 CPU Time: 0.709497 GPU Time: 0.169724 Bandwidth: 2.651451 Gb/s Speedup: 4.180296
Size: 17000 * 17000 Error: 3.966306e-09 CPU Time: 0.911849 GPU Time: 0.215963 Bandwidth: 2.676466 Gb/s Speedup: 4.222254
Size: 19000 * 19000 Error: 4.873073e-09 CPU Time: 1.153083 GPU Time: 0.266035 Bandwidth: 2.714002 Gb/s Speedup: 4.334330
[ya2109@cuda2 hw4]$
```

Figure 4.1.2: Vector-vector and Matrix-vector operations

3. This was ran on Cuda3

```
[ya2109@cuda3 hw4]$ nvcc -arch=sm_61 matvec.cu -Xcompiler -fopenmp && ./a.out
Testing Dot Product: CPU vs GPU
Size: 10000 Error: 1.227818e-11 CPU Time: 0.000054 GPU Time: 0.000089 Bandwidth: 2.696630 Gb/s Speedup: 0.608281
Size: 20000 Error: 1.818989e-11 CPU Time: 0.000108 GPU Time: 0.000047 Bandwidth: 10.201696 Gb/s Speedup: 2.286264
Size: 30000 Error: 4.820322e-11 CPU Time: 0.000152 GPU Time: 0.000063 Bandwidth: 11.512996 Gb/s Speedup: 2.435158
Size: 40000 Error: 7.821654e-11 CPU Time: 0.000203 GPU Time: 0.000072 Bandwidth: 13.327414 Gb/s Speedup: 2.816083
Size: 50000 Error: 1.491571e-10 CPU Time: 0.000241 GPU Time: 0.000090 Bandwidth: 13.363031 Gb/s Speedup: 2.680457
Size: 60000 Error: 7.457857e-11 CPU Time: 0.000296 GPU Time: 0.000099 Bandwidth: 14.482698 Gb/s Speedup: 2.976305
Size: 70000 Error: 2.510205e-10 CPU Time: 0.000344 GPU Time: 0.000108 Bandwidth: 15.496439 Gb/s Speedup: 3.174547
Size: 80000 Error: 1.673470e-10 CPU Time: 0.000365 GPU Time: 0.000127 Bandwidth: 15.087935 Gb/s Speedup: 2.870417
Size: 90000 Error: 1.091394e-10 CPU Time: 0.000418 GPU Time: 0.000137 Bandwidth: 15.813745 Gb/s Speedup: 3.063833


Testing Matrix Vector Product: CPU vs GPU
Size: 1000 * 1000 Error: 1.330136e-11 CPU Time: 0.004566 GPU Time: 0.001468 Bandwidth: 1.362722 Gb/s Speedup: 3.109797
Size: 3000 * 3000 Error: 1.226681e-10 CPU Time: 0.029079 GPU Time: 0.008297 Bandwidth: 2.169933 Gb/s Speedup: 3.504940
Size: 5000 * 5000 Error: 3.185505e-10 CPU Time: 0.080007 GPU Time: 0.020361 Bandwidth: 2.455927 Gb/s Speedup: 3.929444
Size: 7000 * 7000 Error: 6.464234e-10 CPU Time: 0.156627 GPU Time: 0.037613 Bandwidth: 2.605697 Gb/s Speedup: 4.164221
Size: 9000 * 9000 Error: 1.090939e-09 CPU Time: 0.258353 GPU Time: 0.060080 Bandwidth: 2.696542 Gb/s Speedup: 4.300127
Size: 11000 * 11000 Error: 1.622539e-09 CPU Time: 0.384262 GPU Time: 0.087927 Bandwidth: 2.752418 Gb/s Speedup: 4.370254
Size: 13000 * 13000 Error: 2.272827e-09 CPU Time: 0.535174 GPU Time: 0.120377 Bandwidth: 2.807957 Gb/s Speedup: 4.445822
Size: 15000 * 15000 Error: 3.108198e-09 CPU Time: 0.734391 GPU Time: 0.158810 Bandwidth: 2.833667 Gb/s Speedup: 4.624335
Size: 17000 * 17000 Error: 3.966306e-09 CPU Time: 0.928430 GPU Time: 0.202638 Bandwidth: 2.852460 Gb/s Speedup: 4.581717
Size: 19000 * 19000 Error: 4.873073e-09 CPU Time: 1.158135 GPU Time: 0.252532 Bandwidth: 2.859116 Gb/s Speedup: 4.586086
[ya2109@cuda3 hw4]$
```

Figure 4.1.3: Vector-vector and Matrix-vector operations

In case of vector * vector multiplication (dot product) I implemented using reduction and shared memory so here we see that the bandwidth is pretty high. We do get a speed up for matrix vector product.

4.1.1 Extra

I wanted to see the bandwidth using shared memory and reduction in matrix vector multiplication so I wrote the extra code. And tested it on smaller matrix sizes and saw a high bandwidth and speedup.(I did not test it thoroughly so it could be a bit buggy but works on small input). I ran it on Cuda3. I have commented it in the main function.

```
[ya2109@cuda3 hw4]$ nvcc -arch=sm_61 matvec.cu -Xcompiler -fopenmp && ./a.out


Testing Matrix Vector Product: CPU vs GPU (Blocking)
Size: 1000 * 1000 Error: 1.611227e-10 CPU Time: 0.007286 GPU Time: 0.000177 Bandwidth: 11.300807 Gb/s Speedup: 41.147722
Size: 2000 * 2000 Error: 8.481607e-10 CPU Time: 0.016967 GPU Time: 0.000200 Bandwidth: 40.030013 Gb/s Speedup: 84.878621
Size: 3000 * 3000 Error: 2.722686e-09 CPU Time: 0.028626 GPU Time: 0.000362 Bandwidth: 49.766547 Gb/s Speedup: 79.132149
Size: 4000 * 4000 Error: 4.903086e-09 CPU Time: 0.050887 GPU Time: 0.000589 Bandwidth: 54.310625 Gb/s Speedup: 86.355765
Size: 5000 * 5000 Error: 9.968517e-09 CPU Time: 0.079474 GPU Time: 0.000911 Bandwidth: 54.897642 Gb/s Speedup: 87.249945
[ya2109@cuda3 hw4]$
```

Figure 4.1.4: Matrix-vector operations

4.2 2D Jacobi method on a GPU

Ran the Jacobi on Cuda 3

```
[ya2109@cuda3 hw4]$ nvcc -arch=sm_61 jacobi2D.cu -Xcompiler -fopenmp && ./a.out
N: 100, Iterations: 1000, Total Error: 0.000000e+00, Max residual(any elem): 2.450740e-05, MaxCPUGPUerror(any elem): 0.000000e+00, CPUTime: 0.159800, GPUTime: 0.024462, Speed Up: 6.532648
N: 200, Iterations: 1000, Total Error: 0.000000e+00, Max residual(any elem): 6.187966e-06, MaxCPUGPUerror(any elem): 0.000000e+00, CPUTime: 0.386289, GPUTime: 0.065290, Speed Up: 5.916500
N: 300, Iterations: 1000, Total Error: 0.000000e+00, Max residual(any elem): 2.759351e-06, MaxCPUGPUerror(any elem): 0.000000e+00, CPUTime: 0.872702, GPUTime: 0.132959, Speed Up: 6.563704
N: 400, Iterations: 1000, Total Error: 1.127725e-03, Max residual(any elem): 2.052319e-06, MaxCPUGPUerror(any elem): 4.862350e-07, CPUTime: 1.512756, GPUTime: 0.223940, Speed Up: 6.755168
N: 500, Iterations: 1000, Total Error: 1.745258e-03, Max residual(any elem): 1.493100e-06, MaxCPUGPUerror(any elem): 6.941721e-07, CPUTime: 2.423547, GPUTime: 0.339891, Speed Up: 7.130367
N: 600, Iterations: 1000, Total Error: 8.717357e-03, Max residual(any elem): 1.041445e-06, MaxCPUGPUerror(any elem): 2.115161e-06, CPUTime: 3.611124, GPUTime: 0.484177, Speed Up: 7.458272
N: 700, Iterations: 1000, Total Error: 1.186813e-02, Max residual(any elem): 7.630658e-07, MaxCPUGPUerror(any elem): 2.174757e-06, CPUTime: 4.667288, GPUTime: 0.652842, Speed Up: 7.149183
N: 800, Iterations: 1000, Total Error: 6.105589e-01, Max residual(any elem): 5.906144e-07, MaxCPUGPUerror(any elem): 2.703850e-05, CPUTime: 6.205038, GPUTime: 0.846588, Speed Up: 7.329470
N: 900, Iterations: 1000, Total Error: 9.136953e-03, Max residual(any elem): 4.623291e-07, MaxCPUGPUerror(any elem): 9.792845e-07, CPUTime: 7.851136, GPUTime: 1.066194, Speed Up: 7.363700
[ya2109@cuda3 hw4]$
```

Figure 4.2.5: Jacobi2D

We see a speed up in with the size of N. To print the residuals for each iteration, uncomment the line 118 in the code. To print the errors element wise, 126. This will show that most elements have very high precision when comparing it to the CPU implementation.

4.3 Update on final project

We were parallelizing graph algorithms using OpenMP We have implemented the interface which we have tested with large graphs also. It works in parallel and we have started implementing graph algorithms based on the interface. What is mainly left is implementing the graph algorithms using the interface and analyzing them for large data sets. The interface was the main challenge as that is what is being parallelized. Now the graph algorithms when implemented should be more efficient and better in performance.

We also wanted to let you know that we would not be implementing these on the GPU as after studying GPU computing, we have realized that GPU is not useful as there is no floating point operations in our project which is required to be run in parallel.