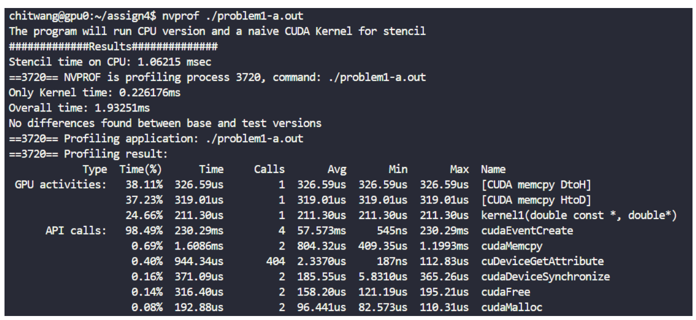
Name: Chitwan Goel
Roll No. 210295

CS610 Assignment 4

1 Problem 1

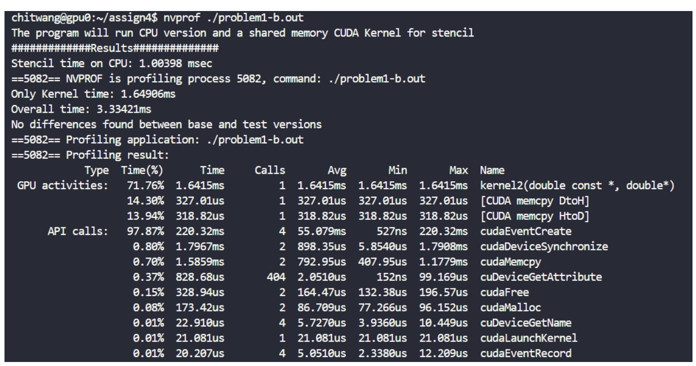
The directory `problem1-dir` contains the source files. Just run the script `run.sh`, it will generate the executables `problem1-a.out`, `problem1-b.out`, `problem1-c.out`, `problem1-d.out` and `problem1-e.out`. Also, it will run the executables using `nvprof`. The following images show the outputs of `nvprof` on various kernels on `gpu0`. `nvprof` works on `gpu0`, so `gpu0` was used for this problem.

```
chmod +x run.sh
./run.sh
```



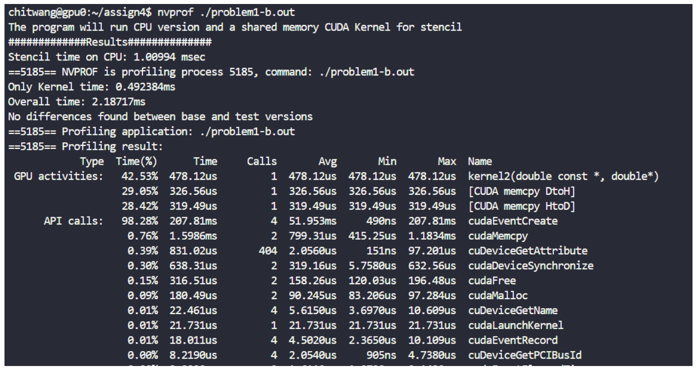
```
chitwang@gpu0:~/assign4$ nvprof ./problem1-a.out
The program will run CPU version and a naive CUDA Kernel for stencil
############Results##############
Stencil time on CPU: 1.06215 msec
==3720== NVPROF is profiling process 3720, command: ./problem1-a.out
Only Kernel time: 0.226176ms
Overall time: 1.93251ms
No differences found between base and test versions
==3720== Profiling application: ./problem1-a.out
==3720== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   38.11%  326.59us         1  326.59us  326.59us  326.59us  [CUDA memcpy DtoH]
                   37.23%  319.01us         1  319.01us  319.01us  319.01us  [CUDA memcpy HtoD]
                   24.66%  211.30us         1  211.30us  211.30us  211.30us  kernel1(double const *, double*)
      API calls:   98.49%  230.29ms         4  57.573ms     545ns  230.29ms  cudaEventCreate
                    0.69%  1.6086ms         2  804.32us  409.35us  1.1993ms  cudaMemcpy
                    0.40%  944.34us       404  2.3370us     187ns  112.83us  cuDeviceGetAttribute
                    0.16%  371.09us         2  185.55us  5.8310us  365.26us  cudaDeviceSynchronize
                    0.14%  316.40us         2  158.20us  121.19us  195.21us  cudaFree
                    0.08%  192.88us         2  96.441us  82.573us  110.31us  cudaMalloc
```

Figure 1: Naive CUDA Implementation



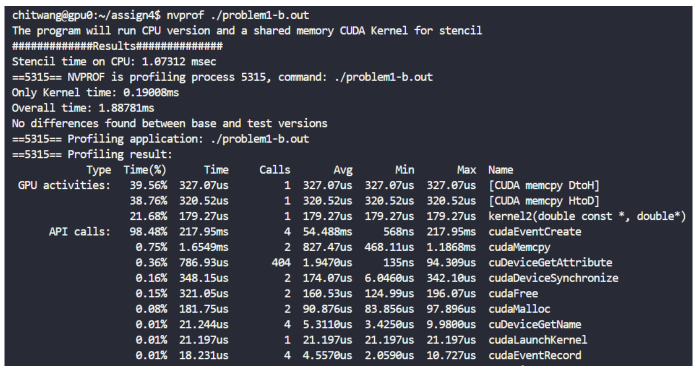
```
chitwang@gpu0:~/assign4$ nvprof ./problem1-b.out
The program will run CPU version and a shared memory CUDA Kernel for stencil
############Results##############
Stencil time on CPU: 1.00398 msec
==5082== NVPROF is profiling process 5082, command: ./problem1-b.out
Only Kernel time: 1.64906ms
Overall time: 3.33421ms
No differences found between base and test versions
==5082== Profiling application: ./problem1-b.out
==5082== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   71.76%  1.6415ms         1  1.6415ms  1.6415ms  1.6415ms  kernel2(double const *, double*)
                   14.30%  327.01us         1  327.01us  327.01us  327.01us  [CUDA memcpy DtoH]
                   13.94%  318.82us         1  318.82us  318.82us  318.82us  [CUDA memcpy HtoD]
      API calls:   97.87%  220.32ms         4  55.079ms     527ns  220.32ms  cudaEventCreate
                    0.80%  1.7967ms         2  898.35us  5.8540us  1.7908ms  cudaDeviceSynchronize
                    0.70%  1.5859ms         2  792.95us  407.95us  1.1779ms  cudaMemcpy
                    0.37%  828.68us       404  2.0510us     152ns  99.169us  cuDeviceGetAttribute
                    0.15%  328.94us         2  164.47us  132.38us  196.57us  cudaFree
                    0.08%  173.42us         2  86.709us  77.266us  96.152us  cudaMalloc
                    0.01%  22.910us         4  5.7270us  3.9360us  10.449us  cuDeviceGetName
                    0.01%  21.081us         1  21.081us  21.081us  21.081us  cudaLaunchKernel
                    0.01%  20.207us         4  5.0510us  2.3380us  12.209us  cudaEventRecord
```

Figure 2: Shared Memory of TILE = 1



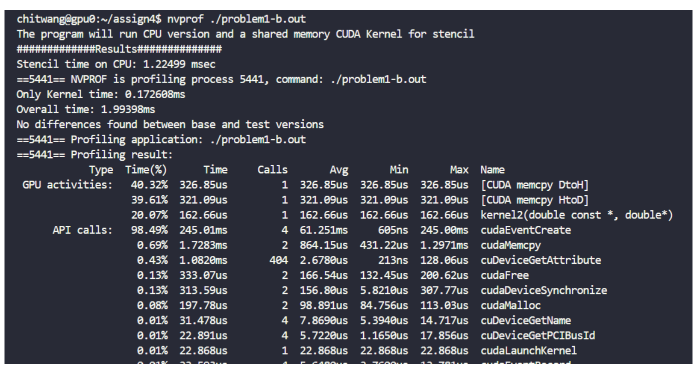
```
chitwang@gpu0:~/assign4$ nvprof ./problem1-b.out
The program will run CPU version and a shared memory CUDA Kernel for stencil
############Results#############
Stencil time on CPU: 1.00994 msec
==5185== NVPROF is profiling process 5185, command: ./problem1-b.out
Only Kernel time: 0.492384ms
Overall time: 2.18717ms
No differences found between base and test versions
==5185== Profiling application: ./problem1-b.out
==5185== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   42.53%  478.12us         1  478.12us  478.12us  478.12us  kernel2(double const *, double*)
                   29.05%  326.56us         1  326.56us  326.56us  326.56us  [CUDA memcpy DtoH]
                   28.42%  319.49us         1  319.49us  319.49us  319.49us  [CUDA memcpy HtoD]
      API calls:   98.28%  207.81ms         4  51.953ms     490ns  207.81ms  cudaEventCreate
                    0.76%  1.5986ms         2  799.31us  415.25us  1.1834ms  cudaMemcpy
                    0.39%  831.02us       404  2.0560us     151ns  97.201us  cuDeviceGetAttribute
                    0.30%  638.31us         2  319.16us  5.7580us  632.56us  cudaDeviceSynchronize
                    0.15%  316.51us         2  158.26us  120.03us  196.48us  cudaFree
                    0.09%  180.49us         2  90.245us  83.206us  97.284us  cudaMalloc
                    0.01%  22.461us         4  5.6150us  3.6970us  10.609us  cuDeviceGetName
                    0.01%  21.731us         1  21.731us  21.731us  21.731us  cudaLaunchKernel
                    0.01%  18.011us         4  4.5020us  2.3650us  10.109us  cudaEventRecord
                    0.00%  8.2190us         4  2.0540us     905ns  4.7380us  cuDeviceGetPCIBusId
```

Figure 3: Shared Memory of TILE = 2



```
chitwang@gpu0:~/assign4$ nvprof ./problem1-b.out
The program will run CPU version and a shared memory CUDA Kernel for stencil
############Results#############
Stencil time on CPU: 1.07312 msec
==5315== NVPROF is profiling process 5315, command: ./problem1-b.out
Only Kernel time: 0.19008ms
Overall time: 1.88781ms
No differences found between base and test versions
==5315== Profiling application: ./problem1-b.out
==5315== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   39.56%  327.07us         1  327.07us  327.07us  327.07us  [CUDA memcpy DtoH]
                   38.76%  320.52us         1  320.52us  320.52us  320.52us  [CUDA memcpy HtoD]
                   21.68%  179.27us         1  179.27us  179.27us  179.27us  kernel2(double const *, double*)
      API calls:   98.48%  217.95ms         4  54.488ms     568ns  217.95ms  cudaEventCreate
                    0.75%  1.6549ms         2  827.47us  468.11us  1.1868ms  cudaMemcpy
                    0.36%  786.93us       404  1.9470us     135ns  94.309us  cuDeviceGetAttribute
                    0.16%  348.15us         2  174.07us  6.0460us  342.10us  cudaDeviceSynchronize
                    0.15%  321.05us         2  160.53us  124.99us  196.07us  cudaFree
                    0.08%  181.75us         2  90.876us  83.856us  97.896us  cudaMalloc
                    0.01%  21.244us         4  5.3110us  3.4250us  9.9800us  cuDeviceGetName
                    0.01%  21.197us         1  21.197us  21.197us  21.197us  cudaLaunchKernel
                    0.01%  18.231us         4  4.5570us  2.0590us  10.727us  cudaEventRecord
```

Figure 4: Shared Memory of TILE = 4



```
chitwang@gpu0:~/assign4$ nvprof ./problem1-b.out
The program will run CPU version and a shared memory CUDA Kernel for stencil
############Results#############
Stencil time on CPU: 1.22499 msec
==5441== NVPROF is profiling process 5441, command: ./problem1-b.out
Only Kernel time: 0.172608ms
Overall time: 1.99398ms
No differences found between base and test versions
==5441== Profiling application: ./problem1-b.out
==5441== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   40.32%  326.85us         1  326.85us  326.85us  326.85us  [CUDA memcpy DtoH]
                   39.61%  321.09us         1  321.09us  321.09us  321.09us  [CUDA memcpy HtoD]
                   20.07%  162.66us         1  162.66us  162.66us  162.66us  kernel2(double const *, double*)
      API calls:   98.49%  245.01ms         4  61.251ms     605ns  245.00ms  cudaEventCreate
                    0.69%  1.7283ms         2  864.15us  431.22us  1.2971ms  cudaMemcpy
                    0.43%  1.0820ms       404  2.6780us     213ns  128.06us  cuDeviceGetAttribute
                    0.13%  333.07us         2  166.54us  132.45us  200.62us  cudaFree
                    0.13%  313.59us         2  156.80us  5.8210us  307.77us  cudaDeviceSynchronize
                    0.08%  197.78us         2  98.891us  84.756us  113.03us  cudaMalloc
                    0.01%  31.478us         4  7.8690us  5.3940us  14.717us  cuDeviceGetName
                    0.01%  22.891us         4  5.7220us  1.1650us  17.856us  cuDeviceGetPCIBusId
                    0.01%  22.868us         1  22.868us  22.868us  22.868us  cudaLaunchKernel
```

Figure 5: Shared Memory of TILE = 8

```
chitwang@gpu0:~/assign4$ nvprof ./problem1-c.out
The program will run CPU version and a shared memory CUDA Kernel using only limited number of worker threads a
#############Results##############
Stencil time on CPU: 1.07479 msec
==2809== NVPROF is profiling process 2809, command: ./problem1-c.out
Only Kernel time: 0.348672ms
Overall time: 2.06774ms
No differences found between base and test versions
==2809== Profiling application: ./problem1-c.out
==2809== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   38.55%  326.21us         1  326.21us  326.21us  326.21us  [CUDA memcpy DtoH]
                   37.69%  318.95us         1  318.95us  318.95us  318.95us  [CUDA memcpy HtoD]
                   23.76%  201.03us         1  201.03us  201.03us  201.03us  kernel2(double const *, double*)
      API calls:   98.35%  238.13ms         4  59.532ms     521ns  238.12ms  cudaEventCreate
                    0.78%  1.8849ms         2  942.44us  681.01us  1.2039ms  cudaMemcpy
                    0.42%  1.0217ms       404  2.5290us     152ns  196.23us  cuDeviceGetAttribute
                    0.20%  496.26us         2  248.13us  5.7970us  490.47us  cudaDeviceSynchronize
                    0.13%  319.96us         2  159.98us  123.13us  196.83us  cudaFree
                    0.08%  185.79us         2  92.896us  83.127us  102.67us  cudaMalloc
                    0.01%  23.346us         4  5.8360us  3.7440us  11.066us  cuDeviceGetName
```

Figure 6: Shared Memory + Loop Transformations

```
chitwang@gpu0:~/assign4$ nvprof ./problem1-d.out
The program will run CPU version and a pinned + shared memory CUDA Kernel for stencil
#############Results##############
==2914== NVPROF is profiling process 2914, command: ./problem1-d.out
Stencil time on CPU: 1.13297 msec
Only Kernel time: 0.306848ms
Overall time: 1.00947ms
No differences found between base and test versions
==2914== Profiling application: ./problem1-d.out
==2914== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   38.42%  327.91us         1  327.91us  327.91us  327.91us  [CUDA memcpy HtoD]
                   37.88%  323.33us         1  323.33us  323.33us  323.33us  [CUDA memcpy DtoH]
                   23.70%  202.27us         1  202.27us  202.27us  202.27us  kernel2(double const *, double*)
      API calls:   98.55%  231.27ms         2  115.64ms  786.61us  230.48ms  cudaHostAlloc
                    0.41%  960.37us       404  2.3770us     171ns  117.11us  cuDeviceGetAttribute
                    0.33%  781.54us         2  390.77us  340.93us  440.61us  cudaMemcpy
                    0.32%  739.87us         2  369.93us  301.67us  438.19us  cudaFreeHost
                    0.14%  321.42us         2  160.71us  121.04us  200.38us  cudaFree
                    0.13%  293.65us         2  146.82us  6.0940us  287.55us  cudaDeviceSynchronize
                    0.08%  187.15us         2  93.576us  82.997us  104.16us  cudaMalloc
                    0.02%  35.600us         4  8.9000us  1.2120us  30.713us  cuDeviceGetPCIBusId
                    0.01%  29.382us         4  7.3450us  4.6630us  14.615us  cuDeviceGetName
```

Figure 7: Shared Memory + Loop Transformations + Pinned Memory

```
chitwang@gpu0:~/assign4$ nvprof ./problem1-e.out
The program will run CPU version and a UVM + shared memory CUDA Kernel for stencil
#############Results##############
==3028== NVPROF is profiling process 3028, command: ./problem1-e.out
Stencil time on CPU: 1.07789 msec
Kernel time: 1.94781ms
No differences found between base and test versions
==3028== Profiling application: ./problem1-e.out
==3028== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:  100.00%  1.9399ms         1  1.9399ms  1.9399ms  1.9399ms  kernel2(double const *, double*)
      API calls:   98.58%  221.86ms         2  110.93ms  15.143us  221.84ms  cudaMallocManaged
                    0.91%  2.0514ms         1  2.0514ms  2.0514ms  2.0514ms  cudaDeviceSynchronize
                    0.37%  825.55us       404  2.0430us     157ns  98.287us  cuDeviceGetAttribute
                    0.11%  242.57us         2  121.28us  72.881us  169.69us  cudaFree
                    0.01%  21.429us         1  21.429us  21.429us  21.429us  cudaLaunchKernel
                    0.01%  21.357us         4  5.3390us  3.7070us  9.5510us  cuDeviceGetName
                    0.01%  13.438us         2  6.7190us  2.6320us  10.806us  cudaEventRecord
                    0.01%  11.901us         2  5.9500us     739ns  11.162us  cudaEventCreate
                    0.00%  8.6490us         4  2.1620us  1.0260us  4.8470us  cuDeviceGetPCIBusId
                    0.00%  2.0250us         1  2.0250us  2.0250us  2.0250us  cudaEventElapsedTime
                    0.00%  1.8720us         8     234ns     168ns     483ns  cuDeviceGet
                    0.00%  1.4740us         4     368ns     263ns     599ns  cuDeviceTotalMem
```

Figure 8: Shared Memory + Loop Transformations + UVM

Observations from the figure:

- `Only kernel` time with shared memory decreases with increase in `TILE` size, which is expected as well. Using `TILE` of size 1 is worst, which should indeed be, since there is only an extra loading cost involved without any benefit.
- Using `pinned memory`, the time taken by the `Memcpy` call has reduced to 781 μs in Figure 7 from around 1.7ms in figures above it, which is also expected.
- In the last figure, there is no `Memcpy` in the profiler results, which is also expected since no explicit copy is done using UVM. The total time using UVM is slightly more than other, which is also explained due to the page migrations incurred in UVM.

- Shared memory using Loops by keeping limited number of threads performs worse than one-one mapping, since each thread now has to do larger amount of work. Also, even using `#pragma unroll` is not able to beat the one-one mapping version. No scope of loop permutation was observed for the scenario.

2 Problem 2

The directory `problem2-dir` contains the source files. Just run the script `run.sh`, it will generate `problem2.out`. Running this executable will print the `thrust` time as well as my kernel's time.

```
chmod +x run.sh
./run.sh
./problem2.out
```

Below table shows the results when both were executed on array of size 2^{20}.

Version	Time (ms)
Thrust	1.42
My Kernel	3.20

Table 1: Performance Comparison of three versions (Data taken on `gpu3` with 2^{20} sized array)

3 Problem 3

The directory `problem3-dir` contains the source files as well as the files `disp.txt` and `grid.txt`. Just run the script `run.sh`, it will generate the executables `problem3-v0.out`, `problem3-a.out`, `problem3-b.out`, `problem3-c.out` and `problem3-d.out`. After generating the executables, it will execute them one by one and print the stats on the console. Also, the output files will be generated as `results-v0.txt` for the baseline implementation and `results-v<i>.txt` for `i` in {`a,b,c,d`} representing the 4 kernel implementations. The script also runs the `diff` command for each of the 4 CUDA implementations comparing them with the baseline output file.

```
chmod +x run.sh
./run.sh
```

Here is a brief description of the 4 CUDA implementations: In all the implementations, chunking of the large iteration space was done to ensure that memory for output can be allocated safely. Let's say there are `n` iterations which satisfy the constraints in a particular chunk. Then the kernel stores those iteration numbers (linearized) in the first `n` positions in the output buffer. After that the CPU implementation sorts the `n` positions and writes to the file `x_array` by calculating it using the linearized iteration number.

- `problem3-a` is the naive version
- `problem3-b` is the version with optimisations applied, like loop unrolling, `prefetch` and `memadvise`. Launching parallel kernels on multiple streams was also tried, but performed worse. So, it was removed from the submitted code.
- `problem3-c` is similar to `problem3-a`, just implemented using `UVM`.
- `problem3-d` is the `Thrust` implementation.

The table below lists the statistics for different versions.

Version	Time (s)
CPU	616
Naive	76.4
UVM	77.9
UVM + optimisations	74.8
Thrust	88.7

Table 2: Performance Comparison of three versions (Data taken on `gpu3`)

4 Problem 4

The directory `problem4-dir` contains the source files. Just run the script `run.sh`, it will generate `problem4.out`. Running this executable will print the timing results for cpu as well as basic and optimised kernels for both 2D and 3D convolution.

```
chmod +x run.sh
./run.sh
./problem4.out
```

Below images show the results when both were executed on array of various sides and `TILE` size equal to 8.

```
Executing...
######### 2D Convolution ##########
CPU convolution time: 0.0641346 ms
Basic Kernel time: 0.004096ms
Basic Kernel time including memory transfers: 4.7159ms
No differences found between CPU and GPU results
Optimized Kernel time: 0.004096ms
Optimized Kernel time including memory transfers: 4.62362ms
No differences found between CPU and GPU results
######### 3D Convolution ##########
CPU convolution time: 21.4849 ms
Basic Kernel time: 5.3049ms
Basic Kernel time including memory transfers: 6.12013ms
No differences found between CPU and GPU results
Optimized Kernel time: 3.66666ms
Optimized Kernel time including memory transfers: 4.44986ms
No differences found between CPU and GPU results
```

Figure 9: Results of various convolutions for N = 64

```
Executing...
######### 2D Convolution ##########
CPU convolution time: 0.255108 ms
Basic Kernel time: 0.021504ms
Basic Kernel time including memory transfers: 0.123232ms
No differences found between CPU and GPU results
Optimized Kernel time: 0.007264ms
Optimized Kernel time including memory transfers: 0.092992ms
No differences found between CPU and GPU results
######### 3D Convolution ##########
CPU convolution time: 162.609 ms
Basic Kernel time: 0.418368ms
Basic Kernel time including memory transfers: 4.61462ms
No differences found between CPU and GPU results
Optimized Kernel time: 1.09146ms
Optimized Kernel time including memory transfers: 5.15443ms
No differences found between CPU and GPU results
```

Figure 10: Results of various convolutions for N = 128

```
Executing...
######### 2D Convolution ##########
CPU convolution time: 3.80898 ms
Basic Kernel time: 0.014624ms
Basic Kernel time including memory transfers: 0.790752ms
No differences found between CPU and GPU results
Optimized Kernel time: 0.050752ms
Optimized Kernel time including memory transfers: 0.799392ms
No differences found between CPU and GPU results
######### 3D Convolution ##########
CPU convolution time: 10367.8 ms
Basic Kernel time: 26.4457ms
Basic Kernel time including memory transfers: 530.882ms
No differences found between CPU and GPU results
Optimized Kernel time: 69.4903ms
Optimized Kernel time including memory transfers: 308.005ms
No differences found between CPU and GPU results
```

Figure 11: Results of various convolutions for N = 512