Name: Gandhi Meet Vipul
PRN No: 2020BTECS00112

High Performance Computing Lab

Practical No. 8

Title of practical: Implementation of Vector-Vector addition & N-Body Simulator using CUDA C

Problem Statement 1:

Implement Vector-Vector addition using CUDA C. State and justify the speedup using different size of threads and blocks.

Screenshots:

a. <<<1,1>>>

```
CUDA API Statistics:

 Time(%)  Total Time (ns)  Num Calls    Average      Minimum     Maximum           Name
 -------  ---------------  ---------  ------------  ----------  ----------  --------------------
    89.1       2070917912          1  2070917912.0  2070917912  2070917912  cudaDeviceSynchronize
    10.4        242855025          3    80951675.0       30158   242733210  cudaMallocManaged
     0.5         11048723          3     3682907.7     3633039     3774470  cudaFree
     0.0           107264          1      107264.0      107264      107264  cudaLaunchKernel


CUDA Kernel Statistics:

 Time(%)  Total Time (ns)  Instances    Average      Minimum     Maximum                     Name
 -------  ---------------  ---------  ------------  ----------  ----------  ----------------------------------------
   100.0       2070966675          1  2070966675.0  2070966675  2070966675  addVectorsInto(float*, float*, float*, int)
```

b. <<<1,64>>>

```
CUDA API Statistics:

 Time(%)  Total Time (ns)  Num Calls    Average     Minimum    Maximum          Name
 -------  ---------------  ---------  -----------  ---------  ---------  --------------------
    49.8        225530532          3   75176844.0      17106  225468487  cudaMallocManaged
    47.7        215892547          1  215892547.0  215892547  215892547  cudaDeviceSynchronize
     2.5         11114074          3    3704691.3    3642477    3796793  cudaFree
     0.0            30477          1      30477.0      30477      30477  cudaLaunchKernel


CUDA Kernel Statistics:

 Time(%)  Total Time (ns)  Instances    Average     Minimum    Maximum                     Name
 -------  ---------------  ---------  -----------  ---------  ---------  ----------------------------------------
   100.0        215879998          1  215879998.0  215879998  215879998  addVectorsInto(float*, float*, float*, int)
```

c. <<<1,128>>>

```
CUDA API Statistics:

 Time(%)  Total Time (ns)  Num Calls    Average     Minimum    Maximum          Name
 -------  ---------------  ---------  -----------  ---------  ---------  --------------------
    59.6        225311780          3   75103926.7      24017  225205463  cudaMallocManaged
    37.1        140300558          1  140300558.0  140300558  140300558  cudaDeviceSynchronize
     3.2         12118062          3    4039354.0    3983222    4148977  cudaFree
     0.0            49955          1      49955.0      49955      49955  cudaLaunchKernel


CUDA Kernel Statistics:

 Time(%)  Total Time (ns)  Instances    Average     Minimum    Maximum                     Name
 -------  ---------------  ---------  -----------  ---------  ---------  ----------------------------------------
   100.0        140285660          1  140285660.0  140285660  140285660  addVectorsInto(float*, float*, float*, int)
```

d. <<<2,64>>>

```
CUDA API Statistics:

 Time(%)  Total Time (ns)  Num Calls    Average     Minimum    Maximum           Name
 -------  ---------------  ---------  -----------  ---------  ---------  ---------------------
    54.7        223015874          3   74338624.7      37099  222849819  cudaMallocManaged
    40.5        165007654          1  165007654.0  165007654  165007654  cudaDeviceSynchronize
     4.7         19290938          3    6430312.7    6342114    6601342  cudaFree
     0.0           163465          1     163465.0     163465     163465  cudaLaunchKernel


CUDA Kernel Statistics:

 Time(%)  Total Time (ns)  Instances    Average     Minimum    Maximum                    Name
 -------  ---------------  ---------  -----------  ---------  ---------  ----------------------------------------
   100.0        165077880          1  165077880.0  165077880  165077880  addVectorsInto(float*, float*, float*, int)
```

e. <<<2,128>>>

```
CUDA API Statistics:

 Time(%)  Total Time (ns)  Num Calls    Average     Minimum    Maximum           Name
 -------  ---------------  ---------  -----------  ---------  ---------  ---------------------
    64.6        228493155          3   76164385.0      36231  228357618  cudaMallocManaged
    30.2        106741694          1  106741694.0  106741694  106741694  cudaDeviceSynchronize
     5.2         18491462          3    6163820.7    6061214    6253149  cudaFree
     0.0            41798          1      41798.0      41798      41798  cudaLaunchKernel


CUDA Kernel Statistics:

 Time(%)  Total Time (ns)  Instances    Average     Minimum    Maximum                    Name
 -------  ---------------  ---------  -----------  ---------  ---------  ----------------------------------------
   100.0        106726488          1  106726488.0  106726488  106726488  addVectorsInto(float*, float*, float*, int)
```

f. <<<4,64>>>

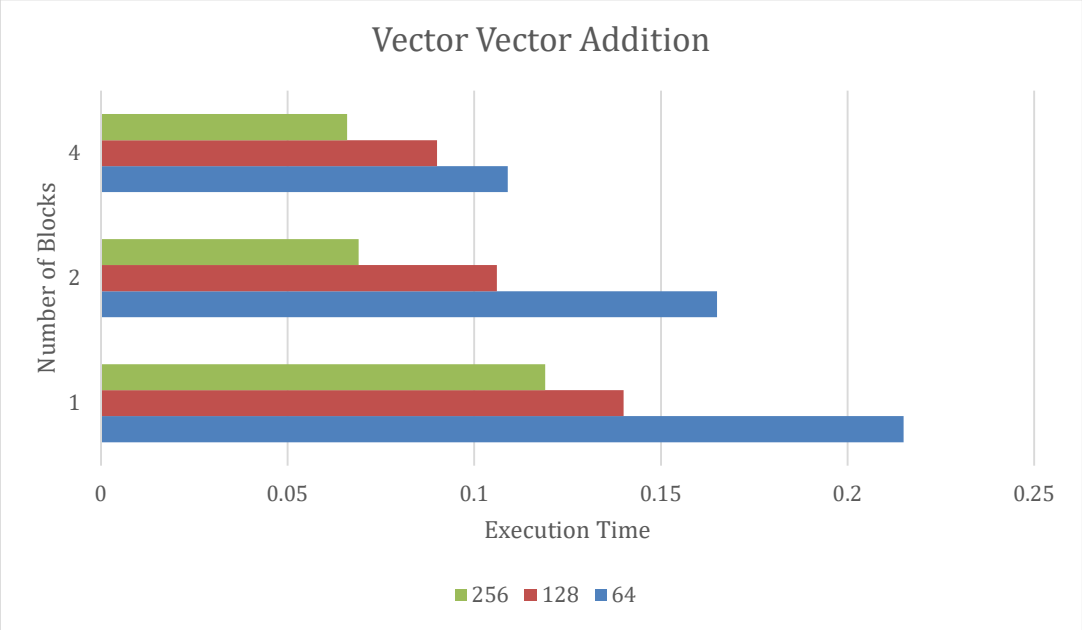
```
CUDA API Statistics:

 Time(%)  Total Time (ns)  Num Calls    Average     Minimum    Maximum           Name
 -------  ---------------  ---------  -----------  ---------  ---------  ---------------------
    70.0        297804934          3   99268311.3      18898  297725683  cudaMallocManaged
    25.8        109763552          1  109763552.0  109763552  109763552  cudaDeviceSynchronize
     4.2         17874379          3    5958126.3    5917572    6020290  cudaFree
     0.0            32427          1      32427.0      32427      32427  cudaLaunchKernel


CUDA Kernel Statistics:

 Time(%)  Total Time (ns)  Instances    Average     Minimum    Maximum                    Name
 -------  ---------------  ---------  -----------  ---------  ---------  ----------------------------------------
   100.0        109751222          1  109751222.0  109751222  109751222  addVectorsInto(float*, float*, float*, int)
```

Output:



Blocks \ Threads	64	128	256
1	0.215	0.14	0.119
2	0.165	0.106	0.069
4	0.109	0.09	0.066

Problem Statement 2:

Implement N-Body Simulator using CUDA C. State and justify the speedup using different size of threads and blocks.

Screenshots:

```
In [17]: !nvcc -std=c++11 -o nbody 09-nbody/01-nbody.cu
```

It is highly recommended you use the profiler to assist your work. Execute the following cell to generate a report file:

```
In [18]: !nsys profile --stats=true --force-overwrite=true -o nbody-report ./nbody
```

```
Warning: LBR backtrace method is not supported on this platform. DWARF backtrace method will be used.
WARNING: The command line includes a target application therefore the CPU context-switch scope has been set to process-tree.
Collecting data...
21.620 Billion Interactions / second
Processing events...
Saving temporary "/tmp/nsys-report-653b-3688-e94d-efea.qdstrm" file to disk...

Creating final output files...
Processing [==========================================================100%]
Saved report file to "/tmp/nsys-report-653b-3688-e94d-efea.qdrep"
Exporting 1109 events: [=================================================100%]

Exported successfully to
/tmp/nsys-report-653b-3688-e94d-efea.sqlite


CUDA API Statistics:

 Time(%)  Total Time (ns)  Num Calls    Average     Minimum    Maximum           Name
 -------  ---------------  ---------  -----------  ---------  ---------  ----------------------
    97.3        294102449          1  294102449.0  294102449  294102449  cudaMalloc
     2.5          7589096         20     379454.8       3461     753403  cudaDeviceSynchronize
     0.1           183892          1     183892.0     183892     183892  cudaFree
     0.1           155654         20       7782.7       3744      28941  cudaLaunchKernel
     0.0           105842          2      52921.0      44477      61365  cudaMemcpy



CUDA Kernel Statistics:

 Time(%)  Total Time (ns)  Instances  Average   Minimum  Maximum                 Name
 -------  ---------------  ---------  --------  -------  -------  ------------------------------------
    99.4          7477098         10  747709.8   746689   748449  bodyForce(Body*, float, int)
     0.6            48320         10    4832.0     4704     5376  integratePosition(Body*, float, int)
```

```
In [19]: from assessment import run_assessment
```

Execute the following cell to run and assess `nbody`:

```
In [20]: run_assessment()
```

```
Running nbody simulator with 4096 bodies
----------------------------------------

Application should run faster than 0.9s
Your application ran in: 0.2122s
Your application reports  21.942 Billion Interactions / second

Your results are correct

Running nbody simulator with 65536 bodies
-----------------------------------------

Application should run faster than 1.3s
Your application ran in: 0.5410s
Your application reports  114.210 Billion Interactions / second

Your results are correct

Congratulations! You passed the assessment!
See instructions below to generate a certificate, and see if you can accelerate the simulator even more!
```

Output:

For 4096 bodies: 0.2122 secs

For 65536 bodies: 0.5410 secs