**Class:** Final Year (Computer Science and Engineering)

**Year:** 2022-23 **Semester:** 1

**Course:** High Performance Computing Lab

**Practical No. 9**

**Exam Seat No:**

2019BTECS00033 – Teknath Krishna Jha

**Title of practical:**

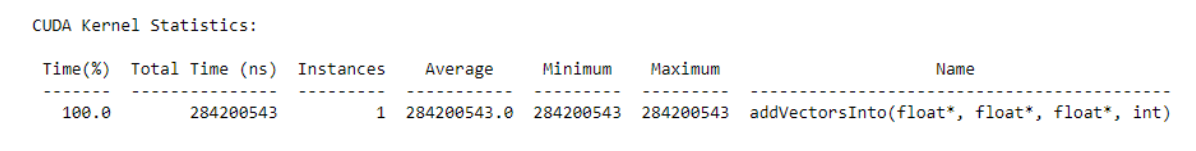
CUDA programming

**Problem Statement 1:**

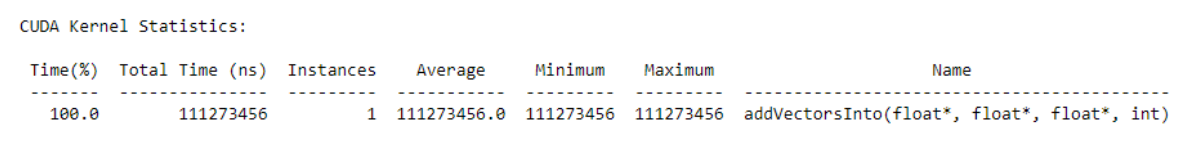
Implement Vector-Vector addition using CUDA C. State and justify the speedup using different size of threads and blocks.

**Output 1:**

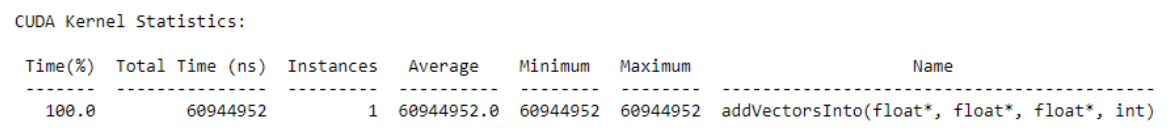
**<<<1,1>>>**

****

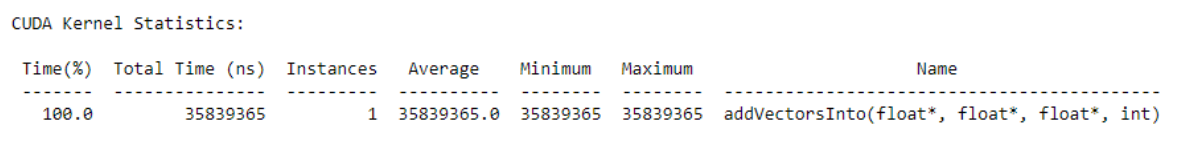
**<<<1,8>>>**

****

**<<<1,16>>>**

****

**<<<1,32>>>**

****

**For Sequential :**

Operating System Runtime API Statistics:

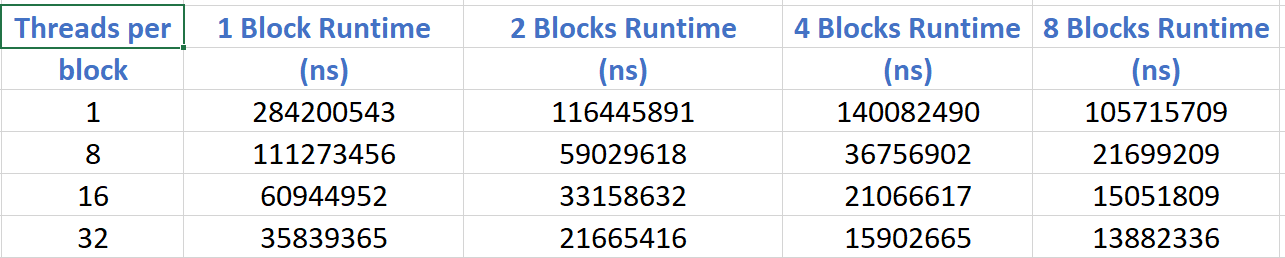
Time(%) Total Time (ns) Num Calls Average

------- --------------- --------- ---------

57.5 5025 1 284200543

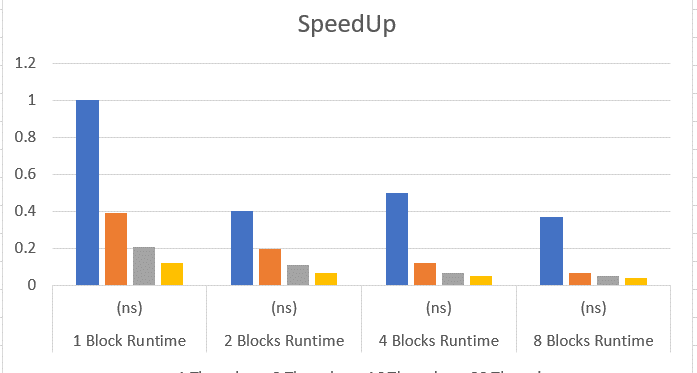
**Speedup :**

**Observation Tp time :**

****

**Calculation of speedup : speedup = Tp/Ts**

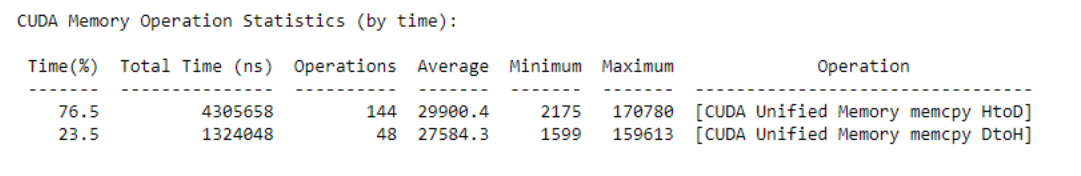
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Threads per** | **1 Block Runtime** | **2 Blocks Runtime** | **4 Blocks Runtime** | **8 Blocks Runtime** |
| **block** | **(ns)** | **(ns)** | **(ns)** | **(ns)** |
| 1 Thread | 1 | 0.4 | 0.5 | 0.37 |
| 8 Thread | 0.39 | 0.2 | 0.12 | 0.07 |
| 16 Thread | 0.21 | 0.11 | 0.07 | 0.05 |
| 32 Thread | 0.12 | 0.07 | 0.05 | 0.04 |



**Final comment :**

**Memory Migration in Unified Memory:**

Copying memory from Host to Device takes almost same time than Device to Host. But the time required for these operations is pretty much constant across changed kernel configuration.

Proof : 

**Most time consuming part:**

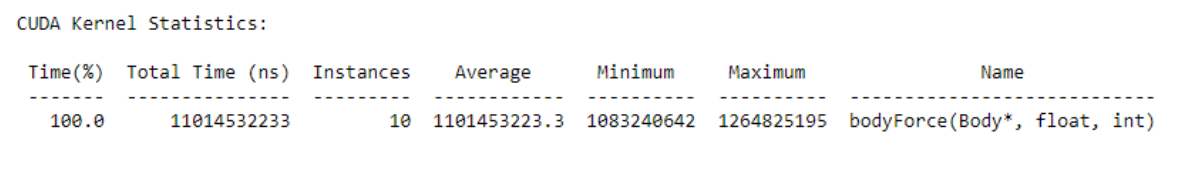
Allocation of memory by cudaMallocManaged takes largest time in the entire program in most cases

**Problem Statement 2:**

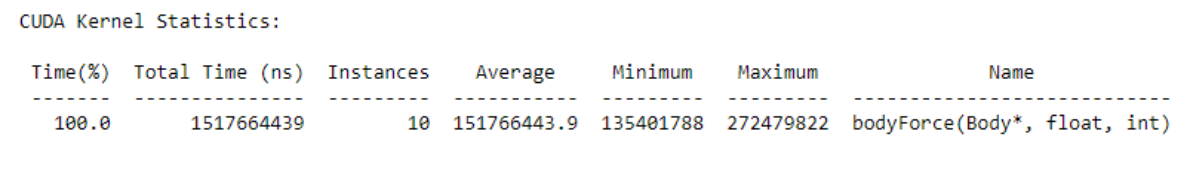
Implement N-Body Simulator using CUDA C. State and justify the speedup using different size of threads and blocks.

**Screenshot 2:**

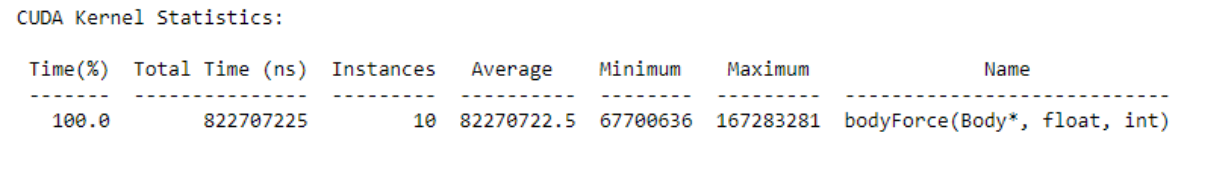
**<<<1,1>>>**

****

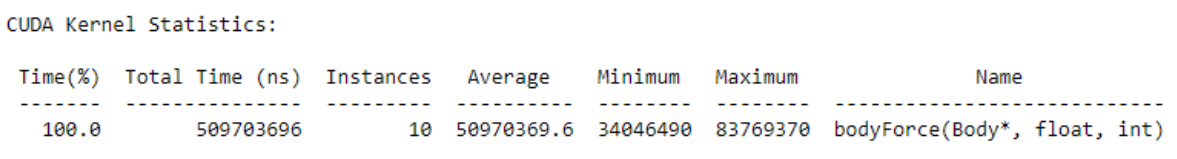
**<<<1,8>>>**

****

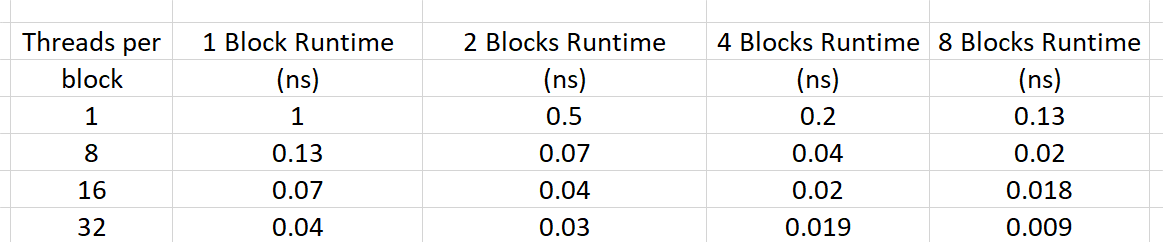
**<<<1,16>>>**

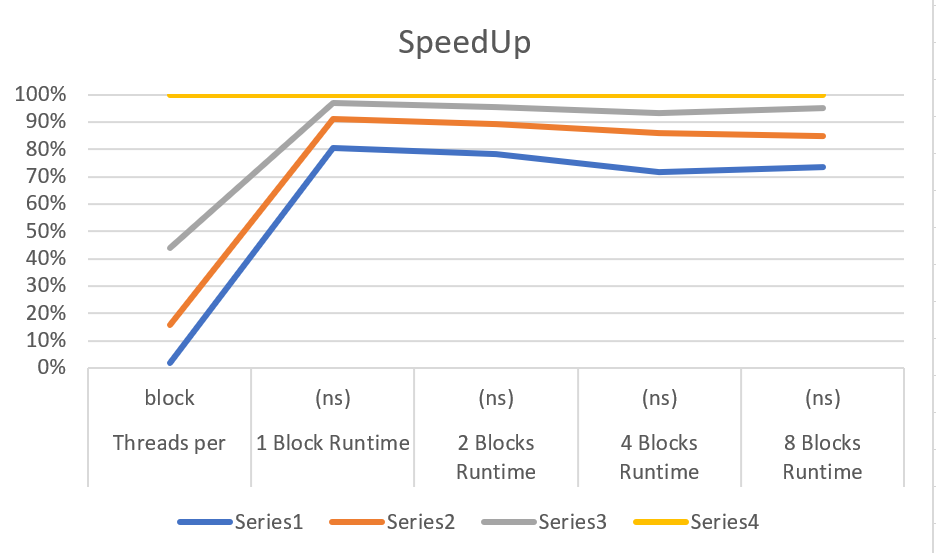
****

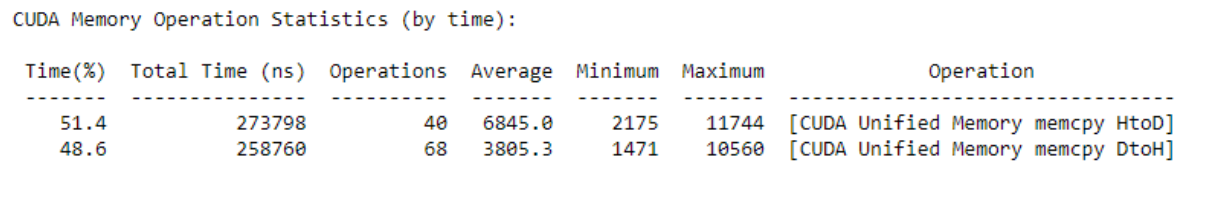
**<<<1,32>>>**

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Threads per | 1 Block Runtime | 2 Blocks Runtime | 4 Blocks Runtime | 8 Blocks Runtime |
| block | (ns) | (ns) | (ns) | (ns) |
| 1 | 11014532233 | 5563155974 | 2868550980 | 1498958754 |
| 8 | 1517664439 | 830366506 | 484274927 | 308540581 |
| 16 | 822707225 | 489260562 | 298343463 | 203863012 |
| 32 | 509703696 | 331265419 | 210476249 | 106152724 |
| SpeedUp calculation of above observation table : |  |  |  |  |



****

****

**Execution time:**

As expected, the execution time of kernel decreases as the total threads (blocks \* threads) increases. Total time taken by program also follows a similar trend.

**Memory allocation and deallocation:** All instances seem to follow random fluctuations since the same amount of memory is being allocated outside any parallel region. But is pretty much constant

**Github Link:**

<https://github.com/Teknath-jha/HPC-LAB-2019BTECS00033/tree/main/Assignment-9>