**Advanced Computer Architecture (HPCA)**
**HW: GPU (CUDA) Histogram and Atomics**

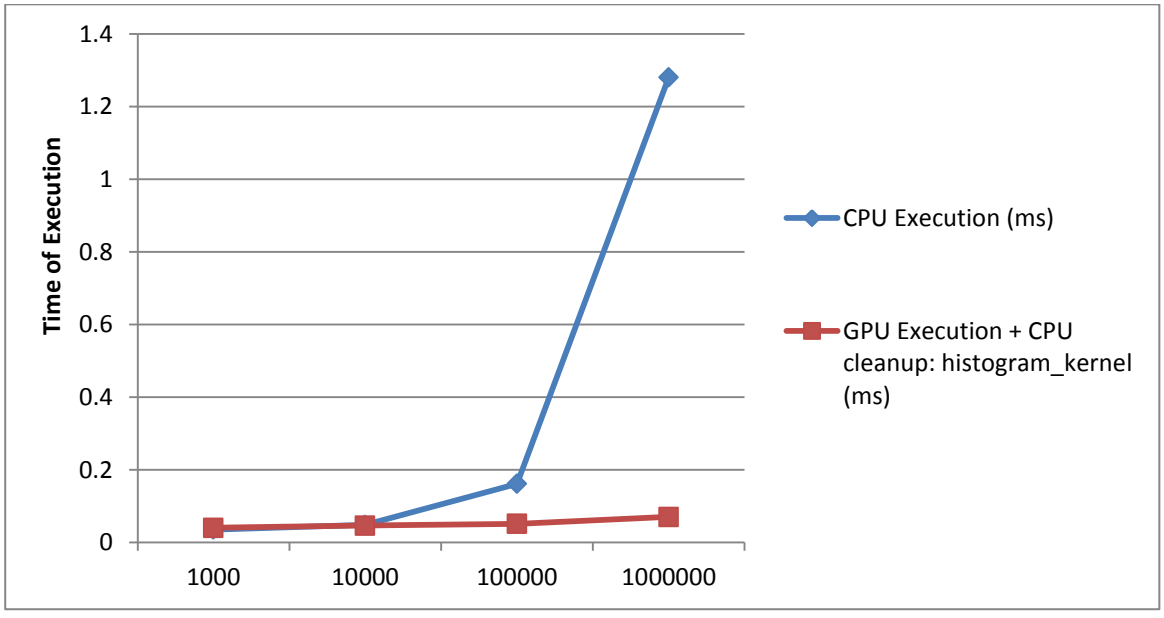
- Rutvij Girish Karkhanis

## Item: 1

CUDA program is implemented to compute a histogram of a random number sequence. A list of *N* random numbers between 0 and 64 is generated. The analysis of the histogram over CPU and GPU is shown below.

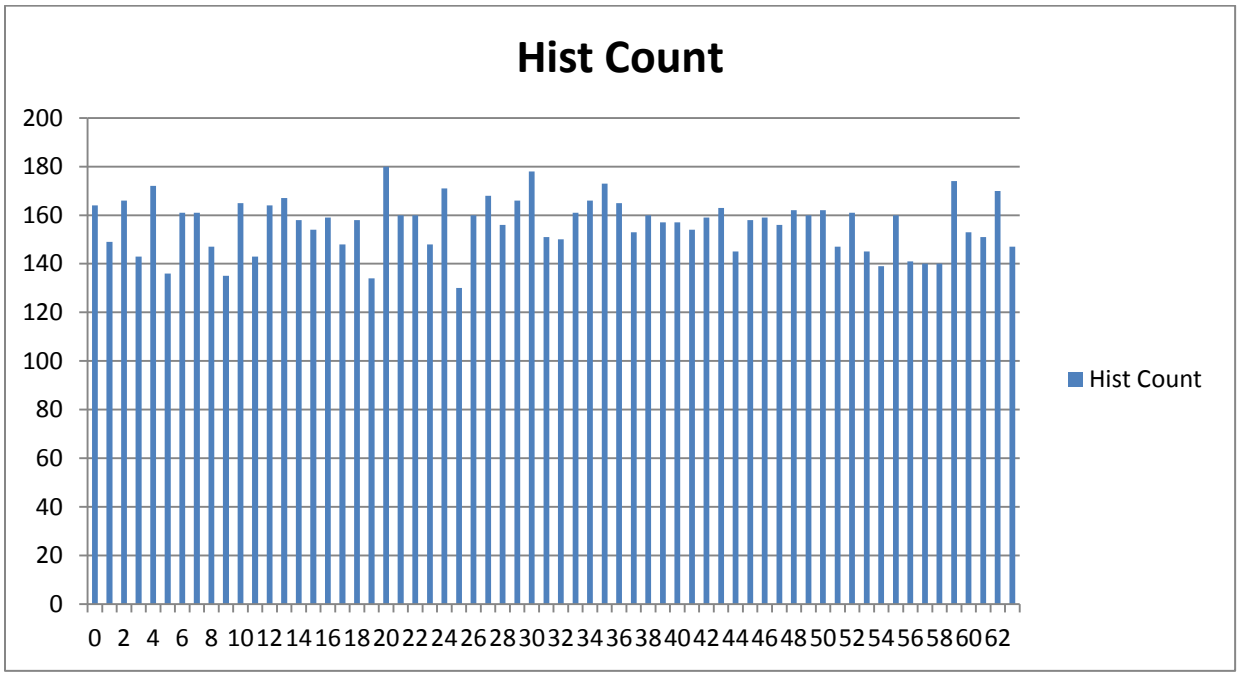
| **Array Size (N)** | **CPU Execution (ms)** | **GPU Execution + CPU cleanup: histogram_kernel (ms)** | **GPU Speedup** |
|---|---|---|---|
| 1000 | 0.035000 | 0.040000 | 0.875000 |
| 10000 | 0.048000 | 0.046000 | 1.043478 |
| 100000 | 0.161000 | 0.051000 | 3.156862 |
| 1000000 | 1.280000 | 0.070000 | 18.28571 |

From the table it is seen that the GPU is execution time is very small compared with the CPU. However the memory transfer time between CPU and GPU is considerable. Additionally, as the input array size increases the GPU speedup increases drastically.



From the above graph it is clear that the CPU execution time increases drastically for the higher size of input arrays while the GPU execution time remains very small.

Following graph shows the histogram calculated for input size 10000.

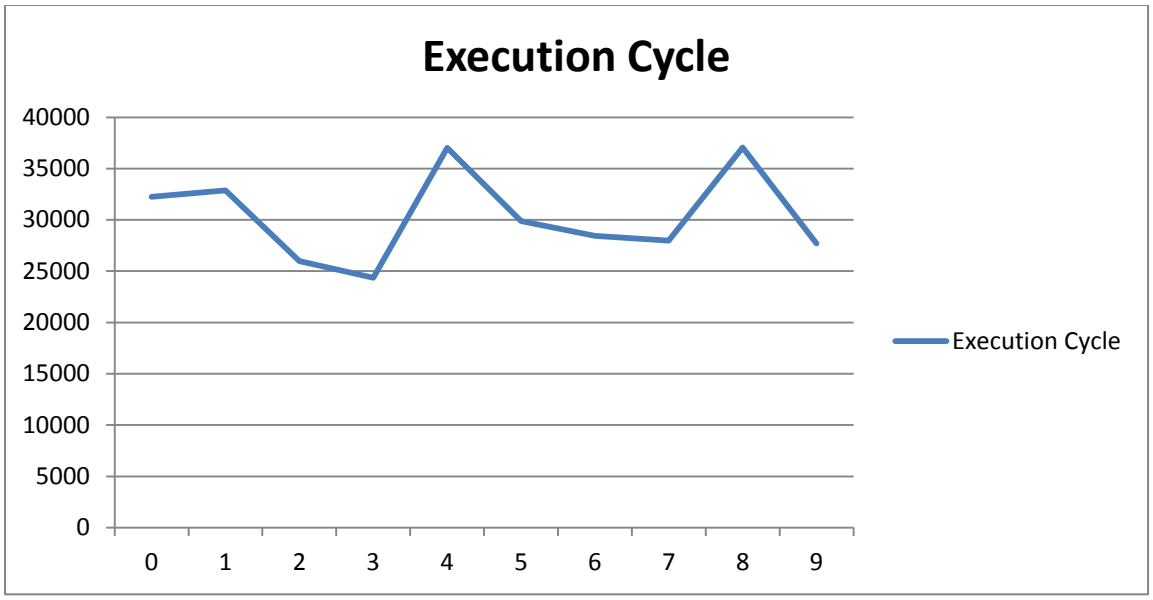



**Item: 2**

The clock function calculates the block execution time in terms of clock cycles. The execution time has been calculated and are tabulated as below:

| **Block Number** | **Execution Time (clock cycles)** |
|---|---|
| 0 | 32254 |
| 1 | 32876 |
| 2 | 25988 |
| 3 | 24374 |
| 4 | 37038 |
| 5 | 29896 |
| 6 | 28458 |
| 7 | 27974 |
| 8 | 37060 |
| 9 | 27710 |

The histogram is plotted for the execution time for respective block numbers.




The start time and the stop time for every block is show as below

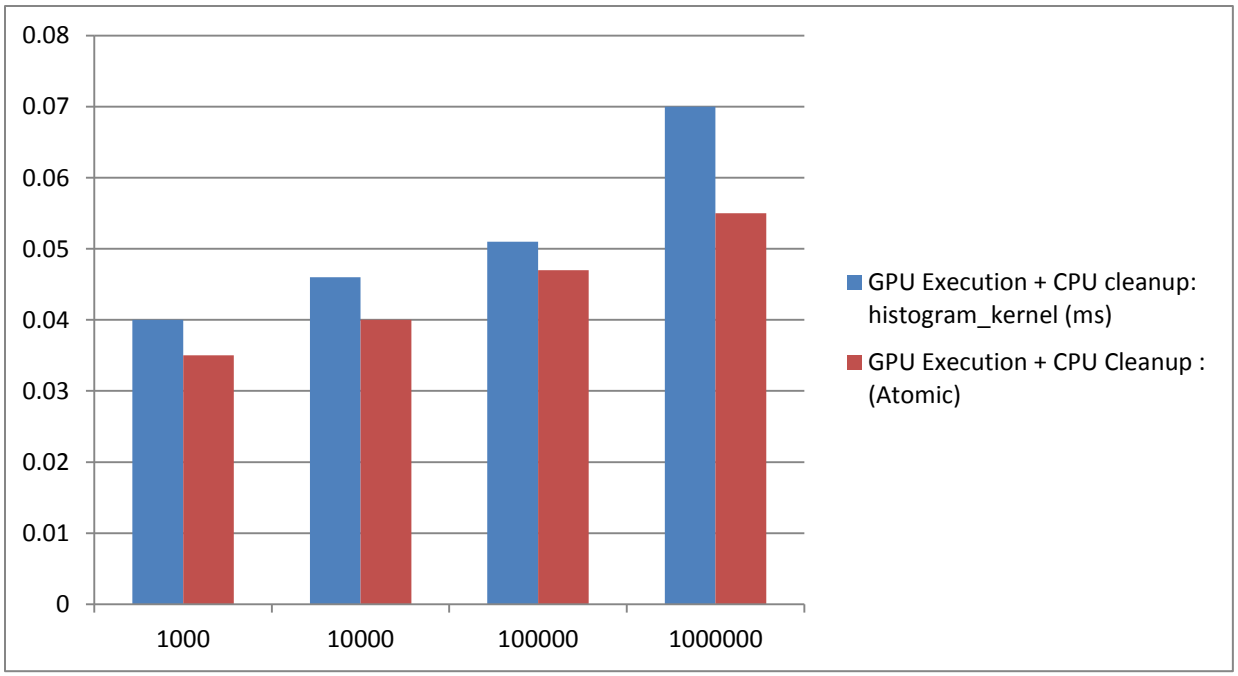
| Block Number | Start Time | Stop Time |
|---|---|---|
| 0 | 36677086 | 36709492 |
| 1 | 36677098 | 36709742 |
| 2 | 36677102 | 36703006 |
| 3 | 36677114 | 36701422 |
| 4 | 36677626 | 36714408 |
| 5 | 36677638 | 36706706 |
| 6 | 36677642 | 36705876 |
| 7 | 36677654 | 36706340 |
| 8 | 36677600 | 36714574 |
| 9 | 36677612 | 36704816 |

The longest execution time was 37060 clock cycles whereas the smallest block execution time was 24374.

**Item: 3**

The atomicAdd() function is used to calculate the histogram. Instead of using separate partial histogram, one global histogram is modified by each block to get the final result. The GPU execution time with and without atomicAdd() function is tabulated below

| **Array Size** | **GPU Execution + CPU Cleanup : Non-Atomic** | **GPU Execution + CPU Cleanup : Atomic** | **Speedup** |
|---|---|---|---|
| 1000 | 0.035000 | 0.035000 | 1 |
| 10000 | 0.040000 | 0.040000 | 1 |
| 100000 | 0.049000 | 0.047000 | 1.04 |
| 1000000 | 0.059000 | 0.055000 | 1.072 |



As seen from the above graph it is seen that the histogram execution time improves by using the atomicAdd() function.

**Item:4**

Even if use of atomicAdd increases the overall performance, it adds little overhead over the only GPU execution. The GPU execution time are analyzed as below.

| Array Size | GPU Execution Time : Non-Atomic (ms) | GPU Execution Time : Atomic (ms) | Overhead due to atomic add operation |
|---|---|---|---|
| 1000 | 0.035000 | 0.035000 | 0.000 |
| 10000 | 0.040000 | 0.040000 | 0.000 |
| 100000 | 0.049000 | 0.047000 | 0.002 |
| 1000000 | 0.053000 | 0.055000 | 0.002 |