Advanced Computer Architecture GPU (CUDA) Vector Reduction

Part A: Vector Reduction Code

Please see attached .cu file. Once compiled, program is run using the following command:

```
./vectorReduce --size xxxx --blocksize yyyy
```

with optional --size and --blocksize flags to change the program's input size and kernel blocksize.

Part B: Timing Routine Results

Please see attached Excel file VectorReduceResults, containing tables and graphs of all program executions. The output of all executions can be see in the vectorReduceResultsOutput file. The graphed output can be seen below in Fig. 1

In the cases of smaller inputs (<10000) the CPU execution time was actually faster than the total GPU time, which includes the time cost of transferring data to and from the GPU. In the case of large data sets, the overall speedup in execution time of the GPU over the CPU became significant. For instance, in the case of an input size of 2000000 with a block size of 256, the CPU took 7.426ms to complete while the GPU was half that, taking only 4.817ms.

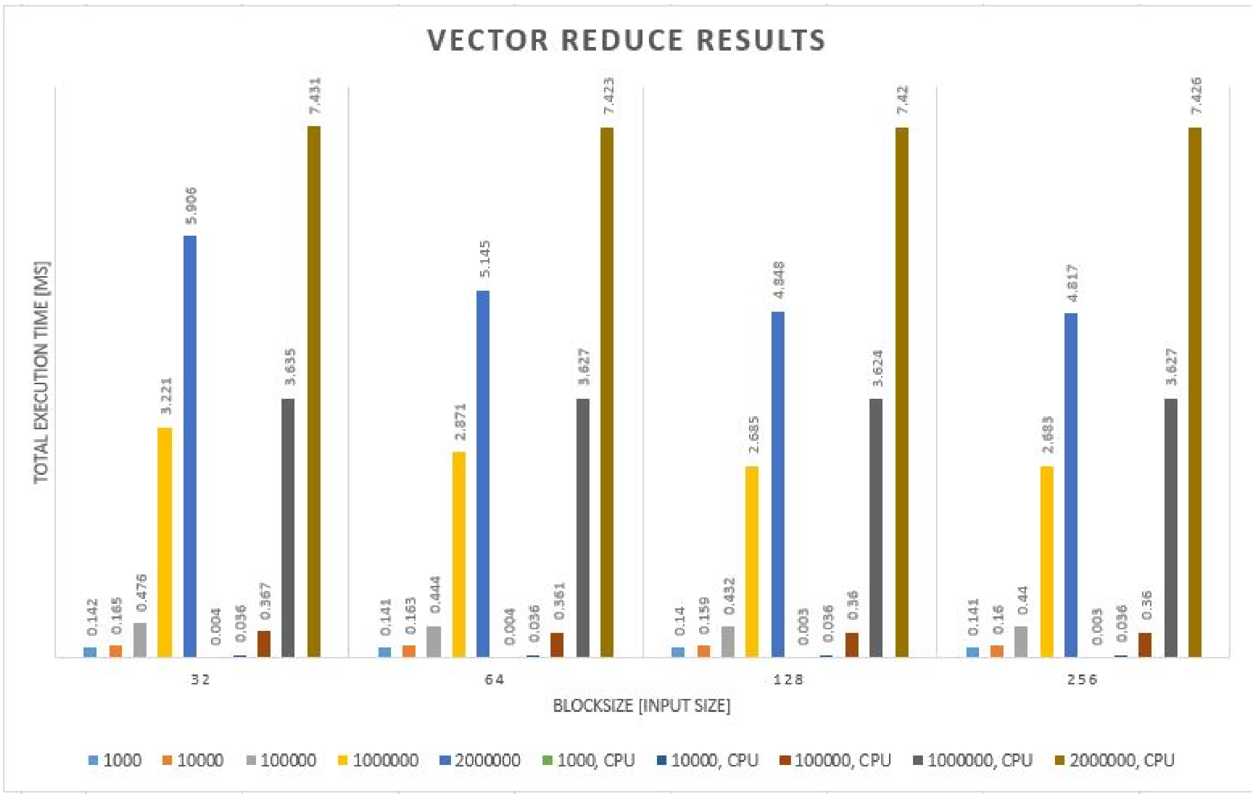

Fig. 1 Output from Part B

Part C: Atomic Function Timing Results

Please see attached Excel file VectorReduceResults, containing tables and graphs of all program executions. The output of all executions can be see in the vectorReduceResultsOutput file. The graphed output can be seen below in Fig. 2

In order to turn on the `atomicAdd` function, there is a definition at the top of the code that must be un-commented out:

```
    //#define ATOMIC_ADD_FUNCTION  // define when doing part 4 of the
assignment
```

From the output, there is no conclusive increase in using the `atomicAdd` function. In some cases, there was a large speedup (4.4% speedup in the case of a 1000-size input and 128 block size) while in other cases there was a slowdown (-0.69% slowdown in the case of a 1000000-size input and 64 block size).

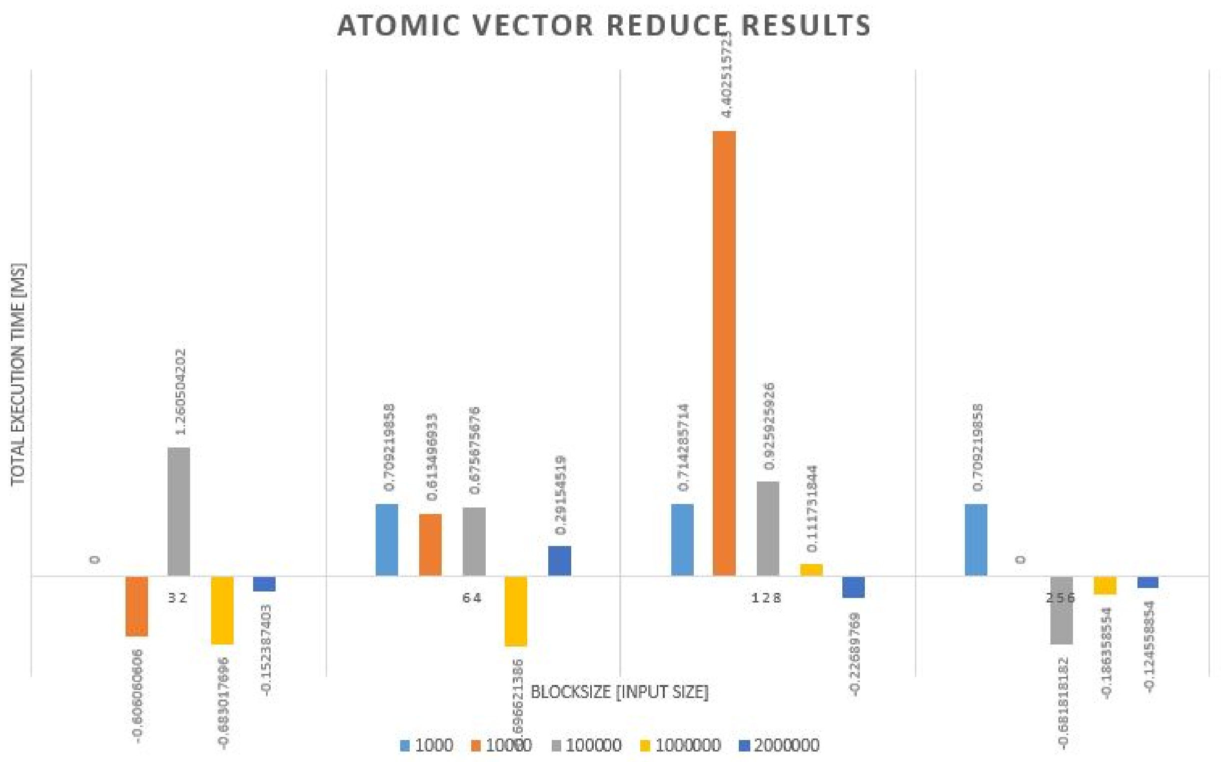

Fig. 2 Output for Part C

Part D: 'If' Statement Evaluation

Please see attached Excel file VectorReduceResults, containing tables and graphs of all program executions. The output of all executions can be see in the vectorReduceResultsOutput file. The graphed output can be seen below in Fig. 3

In order to turn off the `if` statement execution, there is a definition at the top of the code that must be un-commented out:

```
//#define ALWAYS_EXECUTE   // define when doing part 5 of the assignment
```

From the output, there is no conclusive increase in removing the '`if`' statement execution. The largest speedup was only 1.095% (input size = 262144), while there was an almost equal slowdown of -1.54% (input size = 1024). There were two cases where there was no measurable increase or decrease in the execution time (input size = 4096 and 16384) while the remaining case resulted in only a very meager speedup of 0.24% (input size = 1048576).

This was executed using the original kernel from Part A, not the modified kernel from Part C.

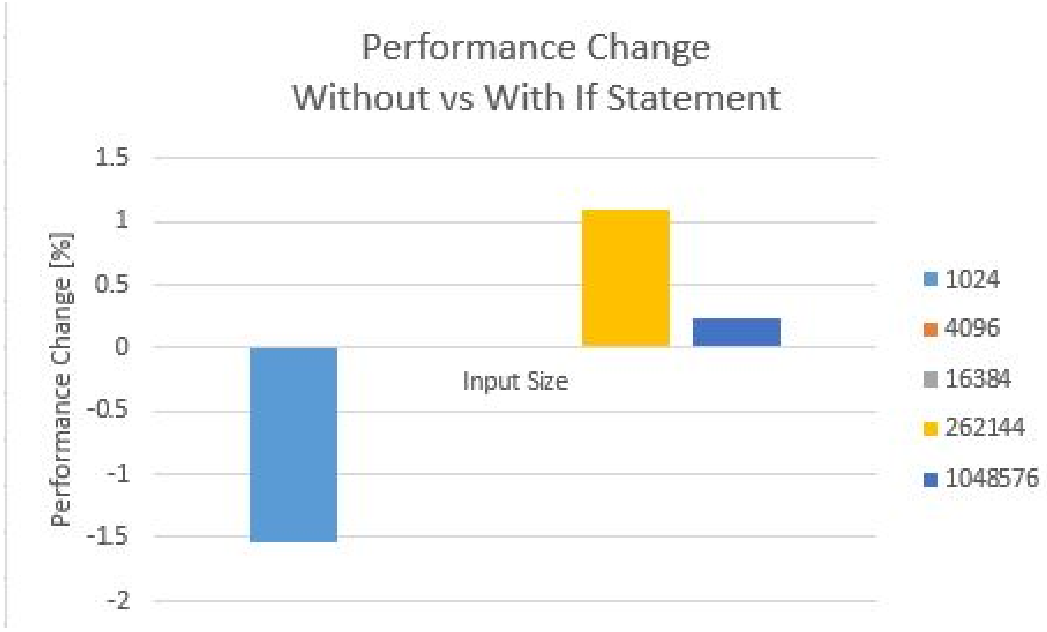

Fig. 3 Output for Part D