Homework Week 6

CS 5334

Dr. Shirley Moore

Oscar Galindo

|  |  |  |
| --- | --- | --- |
| Program | Tutotial # | Execution Time |
| Saxpy.cu | 1 | 2ms |
| Add.cu (Single Thread) | 2 | 116ms |
| Add.cu (Single Block) | 2 | 3.216ms |
| Add.cu (Blocks) | 2 | 2.65ms |
| Add\_grid.cu (Initialized in the CPU which causes page faults). | 3 | 0.280s |
| Add\_grid.cu(With data init in kernel) | 3 | 33ms |
| Calculating effective bandwith from program in tutorial 1 (output 768.976192). | 4 | 40ms |

Runtimes of Tutorials

\*Comment: All runtimes were obtained by running nvprof as the “time” routine takes into consideration the communication time between the host and the device.

For these experiments, we notice the importance of not just taking advantage of simple threading but also of memory segmentation into blocks that can be handled by the threads of the GPU. We see the decrease in the time that the computation requires to complete when we introduce a block of threads, and we see an impressive decay in the amount of time taken when we introduce more than one block when compared to the sequential/1 thread execution. Furthermore, the locality of data becomes very important in tutorial 3 as it shows that not controlling the page faults between the device and the host add a “way too heavy” overhead, during which the bus between the device and the host halts the communication of results to make copies of the data used. It is easy to imagine how unbearable a big computation would become if this phenomenon was not handled properly.

|  |  |  |
| --- | --- | --- |
| Program | Tutotial # | Execution Time |
| Jacobi converge (Serial) | 1 | 6.69s |
| Jacobi converge (GPU) | 1 | 0.26s |
| Jacobi Converge Extended Matrix (2000,10000) | 2 | 0.77s |
| Jacobi Converge Extended Matrix (4000,20000) | 3 | 3.679s |
| Jacobi Convergence (1000, 5000) and manual memory control | 4 | 8.55s |
| Jacobi Convergence (1000, 5000) and manual memory control with added copyin | 5 | 0.27s |
| Jacobi Convergence (1000, 5000) and manual memory control with added copyin/async | 6 | 0.44s |
| Jacobi Convergence (2000, 10000) and manual memory control with added copyin/async | 6 | 0.864s |
| Jacobi Convergence (4000, 20000) and manual memory control with added copyin/async | 6 | 3.68s |

Open ACC

For these tutorials we observe the advantages of the massive parallelism that the GPU can support vs. CPU parallelism. Furthermore, we observe that the order of saving is around order of magnitude (i.e., instead of seconds it takes fraction of a second). Additionally, we see how a manual manipulation of memory, just as in the previous set of tutorials, increases overhead if the communication between the host and the device is too continuous. Moreover, we see the improvements because of the inclusion of directives that implicitly declare the sizes of the memory regions that will be used during the executions. Finally, we see how an asynchronous execution of parallel parts of the work does help to decrease the amount of time where no instructions of the kernel are being executed by the GPU, which is an ideal improvement. All these improvements translate into savings of orders of magnitude depending on the size and the range of computations that are defined during execution time.