University of Tromsø

Assignment 3- Cuda/GPGPU

INF-3201

Camilla Stormoen

Department of Computer Science

November 4, 2016

1 Introduction

This report describes the implementation of a parallelized version of a password decoder using Cuda/OpenCL.

1.1 Requirements

- First requirement is to parallelize the pre-code using GPU
- Second requirement is to analyze the solution using the Nvidia Visual Profiler or equivalent for other GPUs

2 Technical Background

2.1 Nvidia Cuda

Cuda is a parallel computing platform created by Nvidia. It increases performance by using the GPU (Graphics Processing Unit) for parallel computing. Cuda supports programming languages such as C, C++ and Fortran.[1]

The Cuda kernel is organized as a grid of thread blocks, as shown in Figure 3.

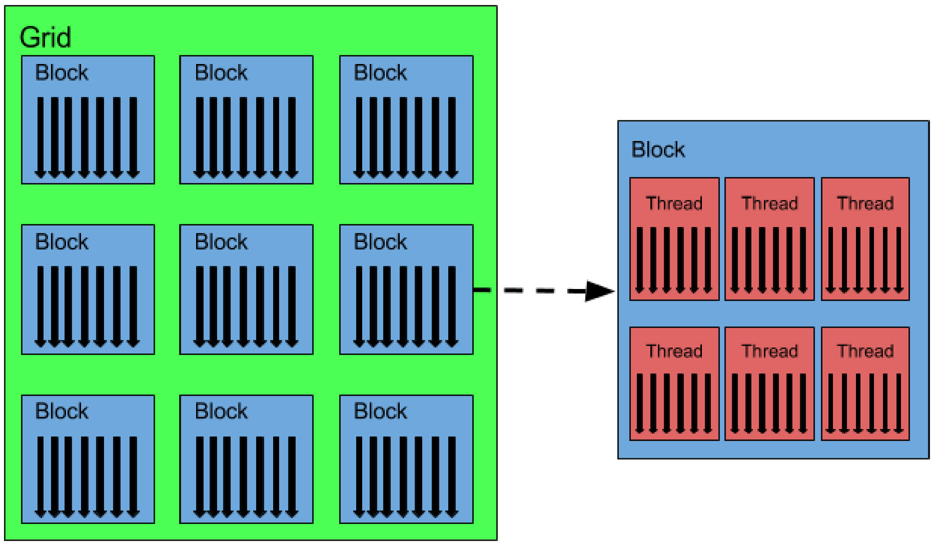


Figure 1: Shows Grid of Thread Blocks

2.2 Nvidia Visual Tool

Nvidia Visual Profiler is a performance profiling tool for feedback on optimizing C/C++ and PyCuda applications.

2.3 PyCuda

PyCUDA is Nvidias parallel computation API for Python.It is a higher level than CUDA C, but the programmer will have to write the CUDA kernel in C.[2]

3 Design

3.1 Decode Password

Each block of decoded data consists of two data points where each data point consists of 24 bits location information and 8 bits of data. When the password isn't given, the password is guessed by iteration through the printable characters for several levels. To test if the password is correct is the data decoded with the guessed password. If there is something in the plain text in the decoded data, it is assumed that the password is correct. Because of the time decoding the password, the workload is distributed on the Cuda kernel, where each thread have the same amount of work. The number of threads are limited to 10.000 threads.

4 Implementation

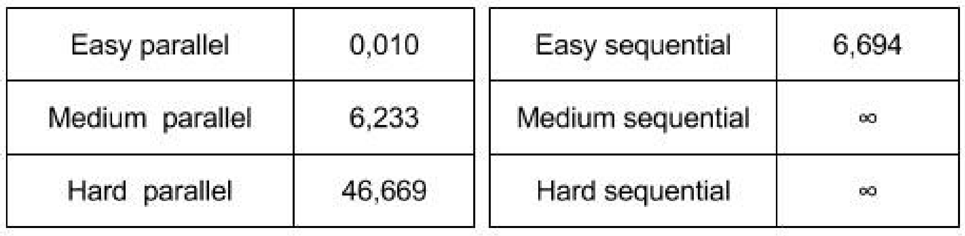
The implementation is written in Python/PyCuda and executed using the Linux terminal. The machine used is the *"ifilab95"* with 31.3 GiB Memory, *Intel@Core i/-6700 CPU @ 3.4GHz x 8 processor* and *Quadro K620/PCIe/SSE2*.
The testing and profiling is provided by the Nvidia Visual Profiler.

5 Discussion

By looking at the execution time, the performance of decoding the password is overall low. Profiling tools shows that it is decrypting bytes and reconstructing the secret that is the bottleneck for performance and execution time. Instead of having the CPU to handle these functions, it's a job for the GPU.

5.1 Evaluation

Figure 2 and Figure 3 shows execution time for both the parallel implementation and the sequential. As the Figure 2, decoding the medium or hard password would take forever and the decoding will eventually time out. Looking at the graph in Figure 3, there is an increasing in speedup and performance by parallelize the problem. This is not a suprise since running code in parallel is generally much faster and better performance than running code sequentially.



Easy parallel	0,010	Easy sequential	6,694
Medium parallel	6,233	Medium sequential	∞
Hard parallel	46,669	Hard sequential	∞

Figure 2: Shows execution time for decoding the password using both sequential code and Cuda Kernel.

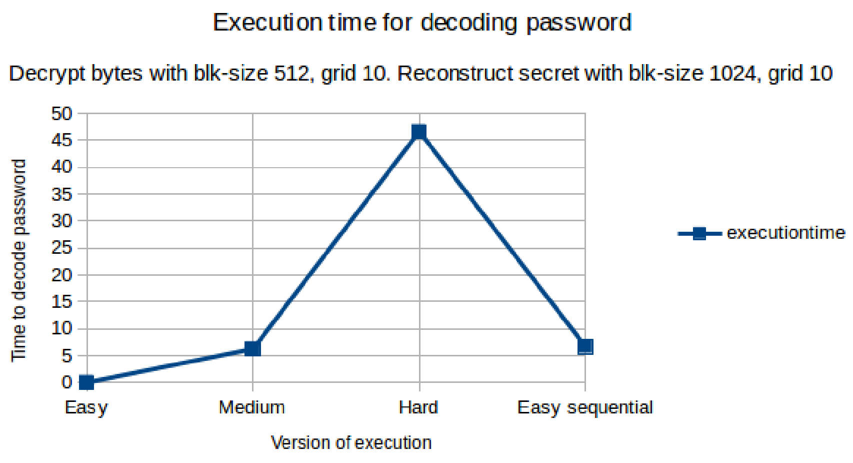


Figure 3: Shows execution time for decoding the password using both sequential code and Cuda Kernel.

There where not much difference in performance with different block-size or grid-size, and a reason for this may be that this is a smaller problem than the number of cores and threads in the kernel, so there will won't be any difference. This is also one of the output from Nvidias Profiling Tool which says that *"The multiprocessors of one or more GPUs are mostly idle"*. If the problem was big enough to fit on all cores, the performance would most likely differ from each other.

5.1.1 Improvements

The implementation is easy and it decrypts the password, however the performance could have been way better. The bottleneck in the implementation is that the data is being copied between the code and the kernel several times, which results in low performance. So by copying the data to the kernel only once, the performance would increase and execution time would decrease drasticly.

6 Conclusion

This report concludes the implementation and design of a parallelized version of a password decoder using CUDA. The performance is increasing when using parallelism instead of sequential code, and the difference is truly high.

References

[1] Wikipedia, "Cuda — wikipedia, the free encyclopedia," 2016. [Online; accessed 29-October-2016].

[2] A. Klöckner, N. Pinto, Y. Lee, B. Catanzaro, P. Ivanov, and A. Fasih, "PyCUDA and PyOpenCL: A Scripting-Based Approach to GPU Run-Time Code Generation," *Parallel Computing*, vol. 38, no. 3, pp. 157–174, 2012.