ICS-E4020: Week 4 - Correlated pairs GPU

Néstor Castillo García (472081)
nestor.castillogarcia@aalto.fi

May 9, 2015

1 Correlated pairs in GPU

1.1 Description

Using CUDA, a working solution that solved the image correlation problem on the GPU was done.

1.2 Hardware

The computers had the following specifications: Intel Xeon E3-1230v2, 4 cores, 8 thread, 3,3 GHz, RAM: 16 GB, GPU: Nvidia K2000.

1.3 Performance

As the focus of this exercise was not in performance, it was not heavily optimised. Nevertheless, this GPU version did the correlated pairs task of a 4000 x 4000 image in less than 15s; faster than a double threaded version but slower than a 8-threaded version. The block size does matter in the performance, for this case the optimal block size was 8 x 8 threads as shown in figure 1.

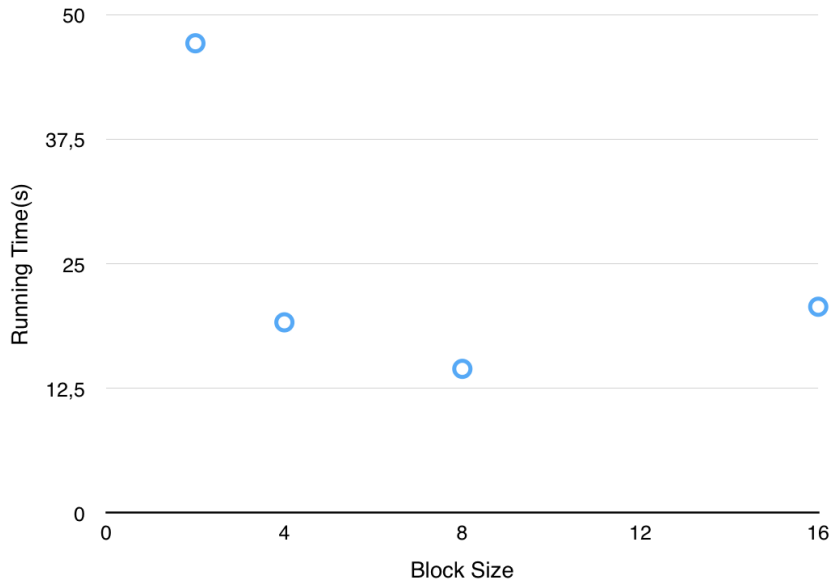

Figure 1: Time vs BlockSize (squared) in a 4000 x 4000 image

Block Size	Time(s)
2	47.165
4	19.136
8	14.465
16	20.703

Figure 2: Time vs BlockSize table in a 4000 x 4000 image

The code was profiled with nvidia profiler and it was found that the kernel execution time occupies 98,8% of the total time. Thus, the cudaMemCopy instructions occupy a small amount of time compared to the kernel execution.

==5101== Profiling application: ./cp-benchmark 4000 4000						
==5101== Profiling result:						
Time(%)	Time	Calls	Avg	Min	Max	Name
99.80%	14.2959s	1	14.2959s	14.2959s	14.2959s	correlateCall(int, int, double*, float*, int)
0.14%	19.491ms	1	19.491ms	19.491ms	19.491ms	[CUDA memcpy HtoD]
0.07%	9.7781ms	1	9.7781ms	9.7781ms	9.7781ms	[CUDA memcpy DtoH]

Figure 3: Detailed execution time in a 4000 x 4000 image

1.4 So1 resubmission

NOTE: My last submission of so1 was rejected due to a fail in make DEBUG=2. However, as stated in email conversations, it was due to a "compiler bug" so I was allowed to resumbit without losing points.

I