ICS-E4020: Week 4 - Correlated pairs GPU

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

May 10, 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.
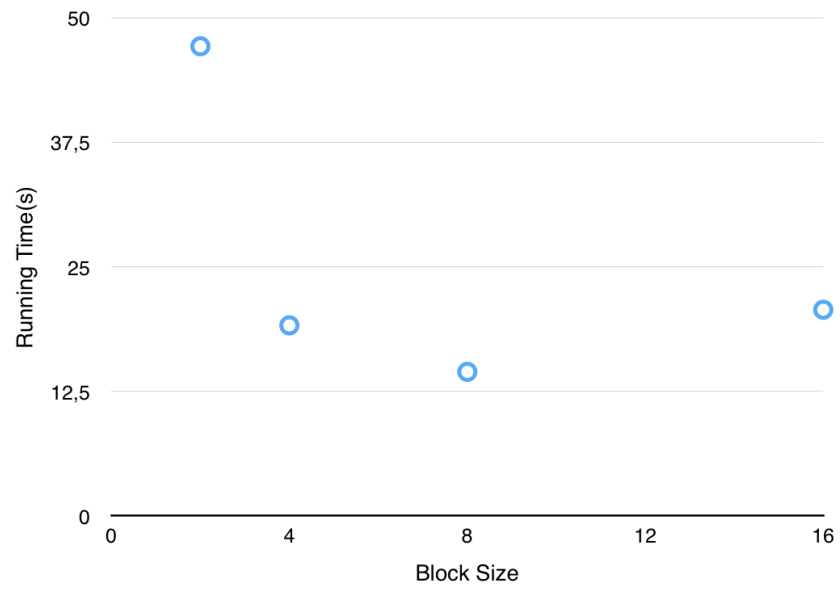
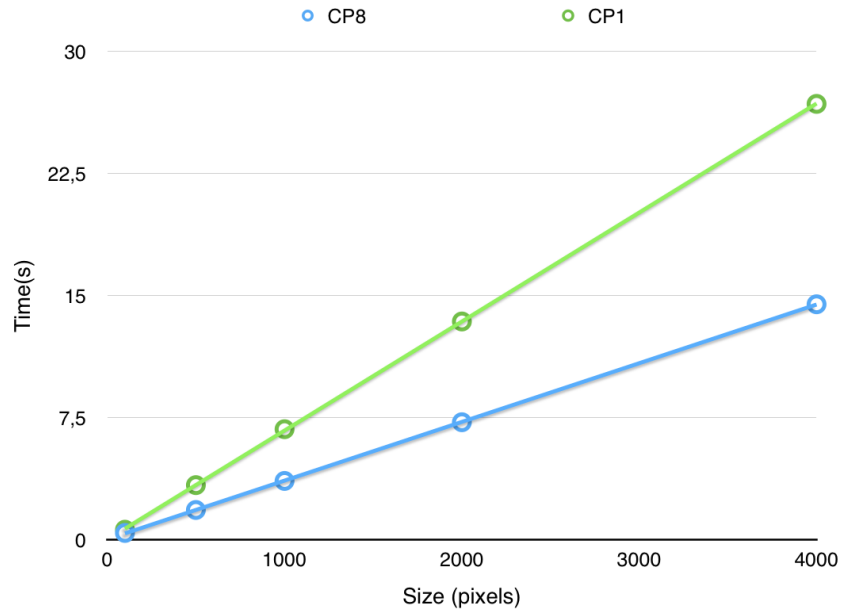Figure 1: Time vs BlockSize (squared) in a 4000 x 4000 image

Figure 2: Single threaded version vs GPU implementation in a 4000 * N image

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

Figure 3: 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.

Figure 4: Detailed execution time in a 4000 x 4000 image

```
% make benchmark
cp      1       1       0.046
cp      1       10      0.033
cp      1       100     0.033
cp      1       500     0.032
cp      1       1000    0.032
cp      1       1500    0.030
cp      10      1       0.034
cp      10      10      0.033
cp      10      100     0.033
cp      10      500     0.033
cp      10      1000    0.031
cp      10      1500    0.038
cp      100     1       0.032
cp      100     10      0.031
cp      100     100     0.035
cp      100     500     0.036
cp      100     1000    0.039
cp      100     1500    0.039
cp      500     1       0.034
cp      500     10      0.033
cp      500     100     0.041
cp      500     500     0.065
cp      500     1000    0.098
cp      500     1500    0.122
cp      1000    1       0.040
cp      1000    10      0.036
cp      1000    100     0.062
cp      1000    500     0.151
cp      1000    1000    0.265
cp      1000    1500    0.378
cp      1500    1       0.031
cp      1500    10      0.031
cp      1500    100     0.084
cp      1500    500     0.282
cp      1500    1000    0.537
cp      1500    1500    0.795
```

Figure 5: Benchmark results

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