Coursework 2B Report

Miles Budden C1769331
BuddenM@Cardiff.ac.uk
Cardiff University - High Performance Computing (CM3103)

Contents

1 Investigation

1.1 Hardware and Software Environment

I developed and ran the application on machines in the lab. These run Linux and have the correct drivers and software for accessing and using the CUDA features of their graphics cards. I connected to these machines through ssh as it allowed me to quickly change machine for collecting data if I could see another user logged in as this could have an effect on performance. The specifications of the lab machines' hardware is as follows:

- Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz
- Nvidia RTX 2070
- 16GB RAM

To compile the code I used `nvcc` which produced an executable that could be ran directly.

1.2 Description of Experiments

I investigated the effect of the number of blur iterations on run-time of the whole program including I/O and memory allocation and subsequently the effect of the number of blur iterations on the time per blur iteration. For each measurement, I did 50 repeats and found a median as there was a relatively large deviation between results.

1.3 Results of Experiments

As can be seen in in Figure 1, there is a linear relationship between the number of blur iterations and the time taken for the program to execute. There is, however, a large overhead that is not affected by the number of blurs or parallelisation. Therefore, as can be seen in Figure 2, the time per blur iteration falls quickly as the number of blurs is increased as the overhead of the non parallel part of the program is divided over more blur iterations.

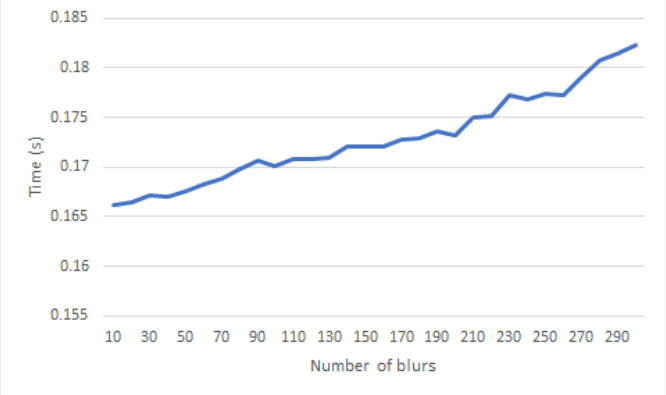

Figure 1: A graph to show how changing the number of blur iterations affects the total run time of the program.

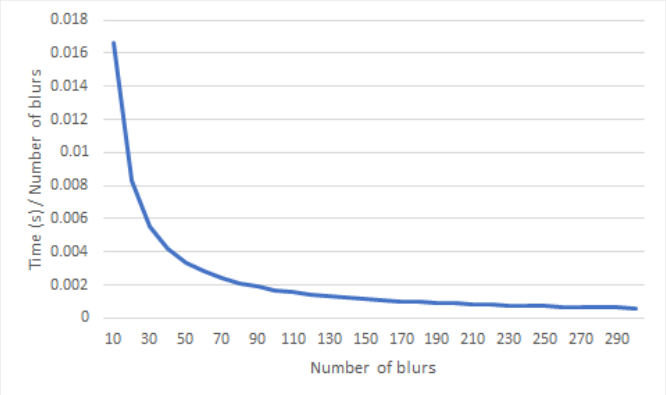

Figure 2: A graph to show how changing the number of blur iterations affects time per blur.

1.4 Discussion of Results

Figure 1 shows a simple linear correlation between the number of blur iterations and the total time of program execution. As can be seen from Figure 2 however, the more iterations of the blur function, the more efficient the code becomes where efficiency is measured as the time taken per blur iteration. As the trend can be roughly modelled by $t = \frac{x}{k}$ where t is the time taken to execute, x is a constant, and k is the number of iterations. Therefore,

$$\frac{x}{k} - \frac{x}{k-1} \xrightarrow{k\to\infty} 0$$

we can see that at first there is a large efficiency gain by increasing the number of iterations, however, as the number of iterations increase, the increase in efficiency tends to 0.

2 Conclusion

The results show that although the relationship between the number of iterations and the total execution time is linear, the large overhead of allocating memory, I/O etc. mean that only a small portion of the execution time is spent actually blurring the image. Therefore, as the number of blur iterations increases, the proportion of time spent blurring versus overhead increases. This change in proportion means that the more blur iterations, the smaller proportion of time is spend setting up and tearing down and therefore the overall time per iteration is smaller thus resulting in a greater efficiency per iteration.