TDT4200 - Problem Set 5

Nicolai Hollup Brand

October 17, 2024

Note on the Implementation

I get floating point precision errors on five of the pixels when running on the GPU. As far I can gather, the Turing architecture does include FP64 functional units on each streaming processors. However, according to an NVIDIA white paper on the Turing Architecture, these FP64 FU's have a TFLOP rate 1/32nd of the TFLOP rate of the FP32 FU's. There is also a lot less FP64 hardware. I am left somewhat puzzled why it appears to compute using FP32's since FP64 support exists, but it probably has something to do with the worse performance. In any case, when changing the 'my_complex_t' to use floats and not doubles, I get matching results from the CPU and GPU. I hope this is fine.

Source `https://images.nvidia.com/aem-dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf` .

Theory Questions

1)

I get a massive speedup on the GPU. Informally, running the program on the CPU takes around 5150 ms. On the GPU, the calculation takes around 45 ms, and copying the result over to the CPU takes another 15 ms.

The increase in performance can be attributed to the huge amount of parallelisation we can exploit for the mandelpbrot set problem. Calculating the mandelbrot set has two nice properties. 1. Every calculation is independent of all other calculations. 2. The only information a calculations needs are its coordinates within the image, meaning no data copy from the CPU/host is necessary.

Since we can spin up an incredible amount of execution threads on the GPU, we can exploit this parallelism to a very large degree.

2)

The major and minor version of the CUDA device is '7.5', meaning it uses the Turing Architecture. To figure out exactly what GPU's are available, I ran 'scontrol show nodes | grep Gres'. The snotra system appears to have two different kinds of GPU's: GTX 1080 Ti's, and Tesla T4's. Both use the Turing architecture.

In the code I print how many streaming processors the allocated GPU has. The result was 40. This matches with what I could find online about the

T4. The 1080 Ti has 28 SMs. Therefore, I am quite sure I was running the program on the Tesla T4.

3)

SIMD means single instruction, multiple data. It means that for one instruction, for example an add instruction, instead of operating on two operands to produce one result, we can operate on an array of data and produce multiple results simultaneously. While SIMD executes simultaneously on different data, it is bound to always execute the same instruction. So no branching per unit.

SPMD means single program multiple data. Here, multiple instances of the same program can run independently. This model is more flexible and allows for branching, meaning the control flow can diverge.

According to the course material slides, CUDA is SIMT (Single Instruction Multiple Treads). In SIMT, single instructions are executed, but using multiple sets of register. Furthermore and similar to SPMD, branching is allowed.