Program Assignment Write up

jcui71, haorui
Jiahao Cui, Haorui Guo

- Saxpy

Question 1:

```
---------------------------------------------------------
Running 3 timing tests:
Cuda Execution time: 2.624 ms
Effective BW by CUDA saxpy: 385.300 ms          [2.901 GB/s]
Cuda Execution time: 0.023 ms
Effective BW by CUDA saxpy: 272.400 ms          [4.103 GB/s]
Cuda Execution time: 0.022 ms
Effective BW by CUDA saxpy: 272.475 ms          [4.102 GB/s]
```

As shown above, the cuda execution time for saxpy is initially 2.624ms. On later execution it reached 0.023ms possible with cache optimization. The runtime is significantly faster than CPU execution which was around 10ms.

Question 2:

The memory bandwidth reaches 4.1GB/s which is significantly lower than the Nvdiva T4's 300GB/s bandwidth. The effective bandwith is limited by the 4GB/s AWS memory bandwidth.

Render

```
-------------------------------------------------------------------------
| Scene Name      | Ref Time (T_ref) | Your Time (T)   | Score           |
-------------------------------------------------------------------------
| rgb             | 0.2441           | 0.276           | 12              |
| rand10k         | 2.7236           | 3.3155          | 11              |
| rand100k        | 26.0338          | (F)             | 0               |
| pattern         | 0.4008           | 0.3117          | 12              |
| snowsingle      | 22.2853          | 23.5738         | 12              |
| biglittle       | 14.877           | (F)             | 0               |
-------------------------------------------------------------------------
|                                  | Total score:    | 47/72           |
-------------------------------------------------------------------------
```

Our solution decomposes the render tasks into calculating the color and opacity of each pixel independently. It assigns one thread to each pixel and attempts to find all the circles that overlap with a given pixel in the original order. Intially, we brute force iterate over all circles for each pixel and check if they are overlapped with the current pixel. This doesn't require synchronization between threads because each thread is independently checking circles. It generates correct results on all scenes. However, this doesn't take the potential to parallel the circle calculation and is much slower than the reference time.

Then we explored the opportunities to find the potential candidate circles for a pixel using parallelization. We first divided the image into pieces according to the block size(each pixel for each thread). Each thread inside a block is assigned to a portion of the total circles in the

image and check if they overlap with the current block. We leverage the provided methods in circleBoxTest.cu_inl file. If a circle overlaps with the current block then they must overlap with certain pixels in the block, we will add that to the current block's circle list and make definitive check with individual pixels later. After doing so, we find a list of candidate circles for each block. Then we apply the similar approach to each pixel thread, instead of checking for all circles now, we sequentially check the candidate circles computed by the last step. The automacity is also achieved in the approach since we maintain the original order of circle array. The performance becomes significant faster than.