Name: Jiaqiang Ruan; ID: jruan

1 Problem 1: SAXPY

Question: Compare and explain the difference between the results provided by two sets of timers (the timer you added and the timer that was already in the provided starter code). Are the bandwidth values observed roughly consistent with the reported bandwidths available to the different components of the machine? Hint: You should use the web to track down the memory bandwidth of an NVIDIA RTX 2080 GPU, and the maximum transfer speed of the computer's PCIe-x16 bus. It's PCIe 3.0, and a 16 lane bus connecting the CPU with the GPU.

```
---------------------------------------------------------
Found 1 CUDA devices
Device 0: GeForce RTX 2080
   SMs:        46
   Global mem: 7974 MB
   CUDA Cap:   7.5
---------------------------------------------------------
Kernel: 0.683 ms                [327.451 GB/s]
Overall: 23.834 ms              [9.378 GB/s]
Kernel: 0.684 ms                [326.955 GB/s]
Overall: 23.609 ms              [9.467 GB/s]
Kernel: 0.681 ms                [328.100 GB/s]
Overall: 23.634 ms              [9.457 GB/s]
```

Overall is the original timer which include the communication time between host memory and device memory. Kernel is my timer which is only the time that GPU uses to do the work.

The memory bandwidth of RTX 2080 is 448.0 GB/s. The maximum transfer speed of PCIe-x16 bus is 15.75 GB/s. This is roughly consistent with the reported bindwidth, since there might be other users using the server at the same time.

Reference: https://www.techpowerup.com/gpu-specs/geforce-rtx-2080.c3224; https://en.wikipedia.org/wiki/PCI

2 Problem 2:Parallel Prefix-Sum

The algorithm is the same as write-up. The BlockSize used in a single thread block is change to 1024 to boost the speed.

3 Problem 3:

Include both partners' names and Andrew Id's at the top of your write-up.

At the head of this report.

Replicate the score table generated for your solution and specify which machine you ran your code on.

```
-------------
Running tests on ghc71.ghc.andrew.cmu.edu, size = 1150, mode = cuda
-------------
Score table:
-------------
```

Scene Name	Target Time	Your Time	Score
rgb	0.1794	0.1545	12
rand10k	2.0181	1.6295	12
rand100k	18.9287	15.3535	12
pattern	0.2684	0.2297	12
snowsingle	6.5172	5.9616	12
biglittle	17.2710	13.2735	12

Describe how you decomposed the problem and how you assigned work to CUDA thread blocks and threads (and maybe even warps.)

The total image is split into square boxes. Each box is 32 pixel width and 32 pixel height (imagine the image is padded to fit this requirement). Then call a kernel function and set its gridsize to be the number of the boxes and its blocksize to be the size of each box, which is 32*32 in this case.

Inside the kernel function, we go through following procedures. Each block represents a box of the original image, each thread can viewed as one pixel of the box or one thread to parallel on some tasks. First, a TEST array is created, each element indicates whether current box inter-set with the circle. Then a PREFIX array is calculated using the value from TEST table. Then using the index from INDEX table, we can pack all circles that inter-set with current box into the front of a new array INDEX. All these work are done in parallel by viewing each thread in the block as a worker and using synchronize. Then we view each thread as one pixel on the image, by iterating the circle in INDEX array, we update a temporary color value and finally store it to the global image data. Since the number of circle could be large, they are split into parts and go over the above procedure in sequence.

Describe where synchronization occurs in your solution.

Within each iteration on circles, we first set arrays value to be zero and synchronize. Second, do the box and circle test and synchronize. Third, perform parallel prefix sum and synchronize. Forth, pack index into INDEX array and synchronize. Finally for each pixel iterates on the INDEX array and update the color. Before getting into next round of circles, we do synchronize.

What,if any,steps did you take to reduce communication requirements (e.g.,synchronization or main memory bandwidth requirements)?

I use shared memory for TEST, PREFIX, INDEX array mention above to improve communication speed. And I updated the color information on a temporary value and finally store it back to the global memory.

Briefly describe how you arrived at your final solution. What other approaches did you try along the way. What was wrong with them?

1. Inspired by map-reduce, i first try to use a map function to find out all inter-set circles for each pixel. Then in a reduce function, color each pixel. But this require large memory. Then I try to do the do one by one on each part of the image, but it takes too much time.

2. Since pixel level is not feasible, I decide to divide the image into boxes, calculated needed information and then parallel on all pixels. I tried naively iterates on all box-circle, stored them into array. I tried fetching inter-set circle one-by-one. Then I tried prefix-sum way of calculating the box-circle information. But all of these are done on global array, which I obtained the best score of 53.

3. I try to utilize more shared memory to boost the speed. And the same algorithm in 2 is used in the finally program. The fundamental idea change here is trying to view the thread in a block as a pixel and a potential parallel worker at the same time.

4 Extra Credit

My method is faster than the benchmark.