Programming Assignment #2

Author: Yu Sheng Lin   Instructor: Wei Chao Chen

April 11, 2017

1 Goals

You have to

1. Learn how to use the Thrust library.
2. Learn how to implement some parallel algorithms on GPU.

in this assignment.

2 Requirements

This is a two-part fixed assignment. You are asked to implementation the same functionality in both part. However, in part I you are only allowed to use existing library while in part II the efficiency of your code is also considered during grading.

In this assignment you are required to count the position of each character inside the word it belongs to. Listing 1. provides a sample input and output.

```
gpu qq  a hello   sonoda (input)
1230120010123450001234560 (output)
```

Listing 1: Example: Count the Position in Words.

You may easily come up with an $O(n)$ sequential algorithm and an $O(nk)$ parallel algorithm, where n is the length of the input and k is the maximum length of a word.

The input is generated in a pseudo-random manner, while we will use different random seed during grading. You can assume that $k = 500$, $n \approx 4 \times 10^7$ and the input only contains characters [a-z] and we use linebreak '\n' as the spaces.

You have to implement a function whose signature is Listing 2.. All pointers are device pointers and text_size is the n.

```
void CountPosition1(const char *text, int *pos, int text_size);
void CountPosition2(const char *text, int *pos, int text_size);
```

Listing 2: The function signature of part I.

We also notice that gridDim.x cannot exceed $2^{17} = 130172$ if you don't use -arch sm_30 compile flag.

2.1 Part I: Using the Thrust Library (40pts)

Thrust is a useful API including many common parallel computing patterns. and in this part you should only include

- Declaration of native and `thrust::*` types.
- A few structs with call operator might be required.
- `thrust::*` and native CUDA functions `cudaFree` (namely `__global__` functions are not allowed)

in `CountPosition1`.

Here are some hints:

- You will need the document https://thrust.github.io/doc/modules.html
- I have already included some necessary headers.
- If you write more than 10 lines, then you are probably wrong.

2.2 Part II: Using the Thrust Library (40pts)

In part II, you have to implement the same functionality from scratch, and no external API and library is allowed. We provide some hints about the $O(n \ln k)$ algorithm in a separate PDF while it's not the only solution, and you can decide whether to read them by yourself.

According to our experience, when $k = 500$, $O(n \ln k)$ and $O(nk)$ might have comparable speed. To achieve the best efficiency and outperform your classmate, we also challenge you to implement both and compare them if possible.

3 Submission

- The submission deadline is 2016/3/28 midnight (Mon.).
- The efficiency of part I will NOT be considered during grading, but if you break the rules listed in part 2.1, you will get 0pt.
- The efficiency of part II will also be considered during grading, and pure CPU implementations would be disqualified.
- You can only modify `lab2/counting.cu`, and we will only copy this file from your repo. At most 15pt bonus is given.
- The compile flags are `--std=c++11 -arch sm_30 -O2`. Although we will test your code on a Linux machine with one GTX 970, you should not use platform dependent libraries.
- Please also refer to assignment #0 for more details.