Alysha McCullough

Anna Mikhailenko

Jared Adams

Professor Tian

CSCD 445-040

March 21, 2023

GPU Final Project: GPU Version of HW1

For the final project we decided to replicate homework 1 on the GPU. We first started by dividing the work into three sections and distributing the work between each other. One of us worked on main/kernel, the other worked on the merge sort, and finally the last person worked on the kernel too. However, if any one of us was struggling in their sections we tried to help each other as much as we were able to.

We started the project by first reviewing the very first homework and try to figure out how to do the string parsing in a kernel. We decided to pass a global array of string lines into the kernel and pass an entire line into each thread. Each thread then figures out the different words per line. Next, using a lock, to ensure that a race condition doesn't occur, we copied all of the words in each thread to a global output array. After we figured out most of the details, we started implementing the kernel and allocated the memory for those arrays. Later the memory allocation gave us segmentation fault problems and we tried to allocate memory in three different ways. However, all of the different allocations gave the same segmentation fault.

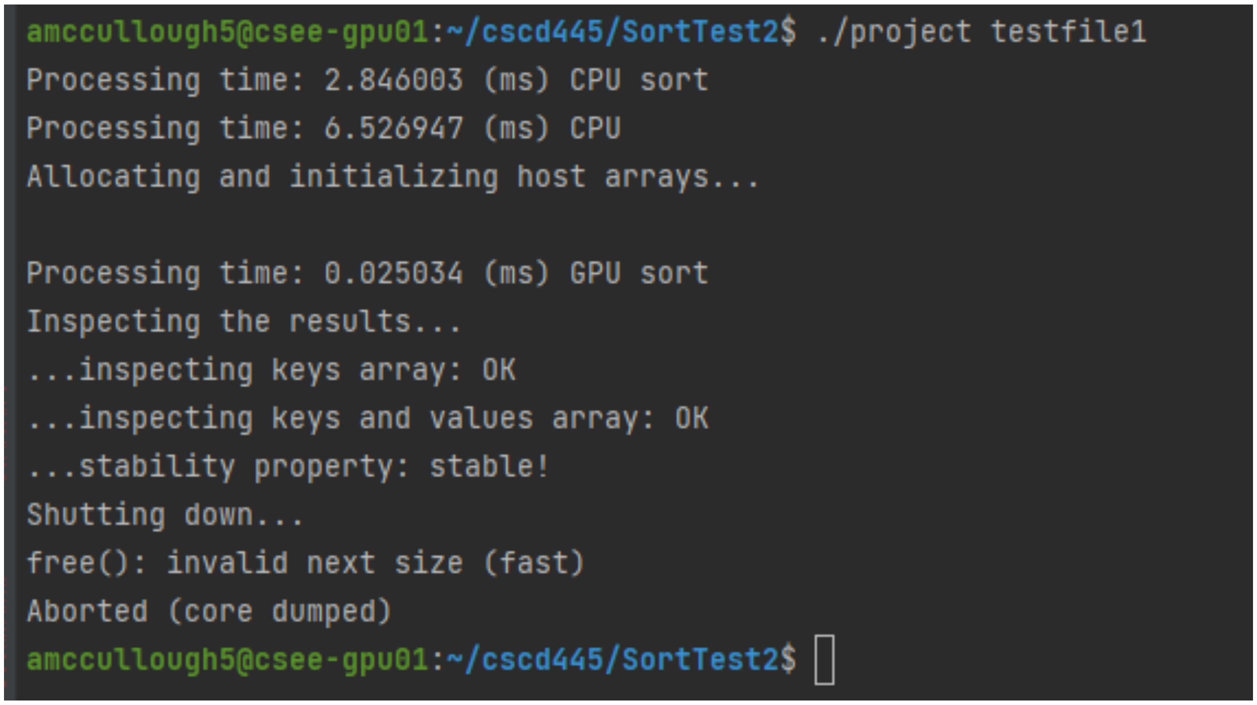
For the merge sort implementation in the kernel, we used the professor's code. We had to change some things within the professor's implementation in order for the merge sort to work with our program. Outside the merge sort we have a 2D array of words, where each column contains a single word, and an integer count array that contains the count of each word. Into the merge sort we pass in an array of column indexes of each word from the 2D array. We also input the integer array that contains the count of each word. The merge sort then sorts the two arrays according to the word count ensuring that each word count number stays with its corresponding word index. After the merge sort is finished sorting, it outputs the sorted count and indexes array. Then outside the merge sort, using the indexes array, we fix the order of the words in the 2D word array.

At this point the only thing that was needed to be done was to finish up main and connect all of the team member's sections together. However, in order for our program to work, we had to change the main file type at least three time. At one point or another our main file has been of file type .c and .cu. Finally, the only file type that allowed main to work properly was .cpp.

How To Run the Code

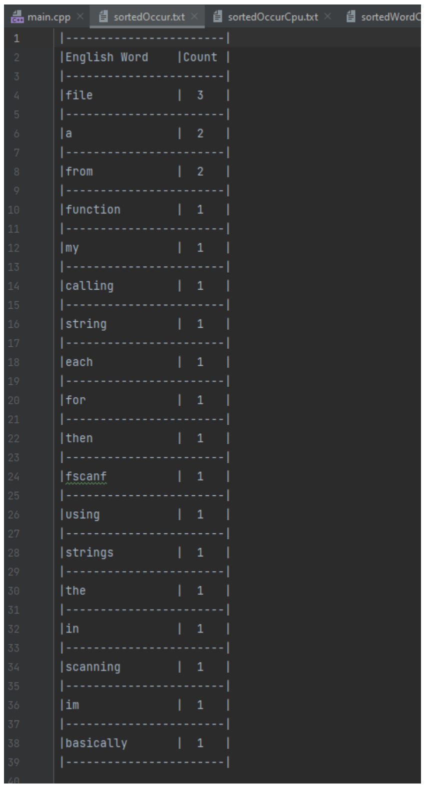
In order to compile the program, in the terminal type in "make" and then to run the program type in "./project 'name of any text file'". The command line argument should be a text file that contains words. The program then reads the file and copies it into an array in order to be used in our program.

Screen Shots of Sample Code



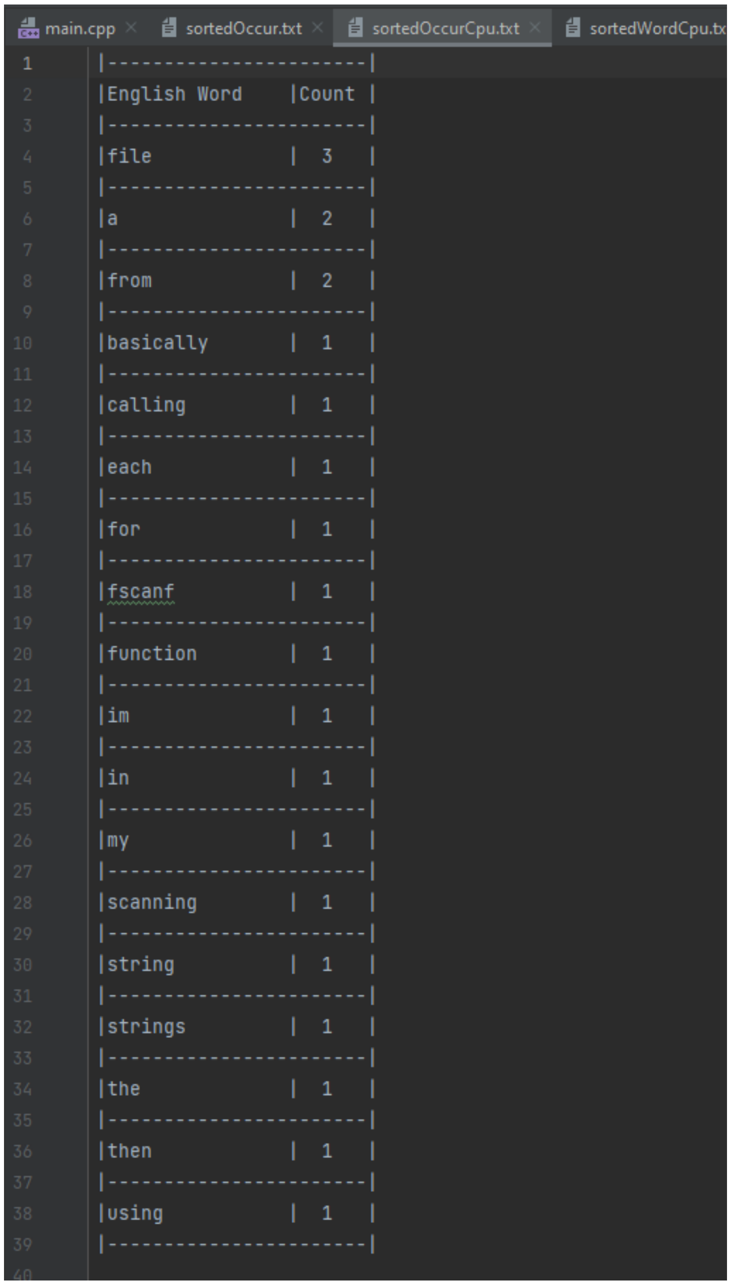
```
amccullough5@csee-gpu01:~/cscd445/SortTest2$ ./project testfile1
Processing time: 2.846003 (ms) CPU sort
Processing time: 6.526947 (ms) CPU
Allocating and initializing host arrays...

Processing time: 0.025034 (ms) GPU sort
Inspecting the results...
...inspecting keys array: OK
...inspecting keys and values array: OK
...stability property: stable!
Shutting down...
free(): invalid next size (fast)
Aborted (core dumped)
amccullough5@csee-gpu01:~/cscd445/SortTest2$
```

GPU Version output:



```
|-----------------------|
|English Word    |Count |
|-----------------------|
|file            |  3   |
|-----------------------|
|a               |  2   |
|-----------------------|
|from            |  2   |
|-----------------------|
|function        |  1   |
|-----------------------|
|my              |  1   |
|-----------------------|
|calling         |  1   |
|-----------------------|
|string          |  1   |
|-----------------------|
|each            |  1   |
|-----------------------|
|for             |  1   |
|-----------------------|
|then            |  1   |
|-----------------------|
|fscanf          |  1   |
|-----------------------|
|using           |  1   |
|-----------------------|
|strings         |  1   |
|-----------------------|
|the             |  1   |
|-----------------------|
|in              |  1   |
|-----------------------|
|scanning        |  1   |
|-----------------------|
|im              |  1   |
|-----------------------|
|basically       |  1   |
|-----------------------|
```

CPU Version output:



main.cpp sortedOccur.txt sortedOccurCpu.txt sortedWordCpu.tx

```
|-----------------------|
|English Word     |Count |
|-----------------------|
|file             |  3   |
|-----------------------|
|a                |  2   |
|-----------------------|
|from             |  2   |
|-----------------------|
|basically        |  1   |
|-----------------------|
|calling          |  1   |
|-----------------------|
|each             |  1   |
|-----------------------|
|for              |  1   |
|-----------------------|
|fscanf           |  1   |
|-----------------------|
|function         |  1   |
|-----------------------|
|im               |  1   |
|-----------------------|
|in               |  1   |
|-----------------------|
|my               |  1   |
|-----------------------|
|scanning         |  1   |
|-----------------------|
|string           |  1   |
|-----------------------|
|strings          |  1   |
|-----------------------|
|the              |  1   |
|-----------------------|
|then             |  1   |
|-----------------------|
|using            |  1   |
|-----------------------|
```

GPU Speedups

	CPU	GPU
Processing time (Sort)	2.846003	0.025034

GPU Speed up:

2.846003/ 0.025034 = 113.6855077 times faster (GPU vs. CPU sort)