Lab 3

CMPUT 398

Objective

In this lab you will implement a parallel merge.

This lab will be submitted as one zipped file through eclass. Details for submission are at the end of the lab.

The Algorithm

Given inputs A and B as follows:

A = [0 2 4 10]
B = [1 5 7 9]

C should be the following:

C = [0 1 2 4 5 7 9 10]

To merge the two arrays in parallel we need to find what index each element should be in C. The following table shows the index in C each element should be.

A's index in C	0	2	3	7
A	0	2	4	10
B	1	5	7	9
B's index in C	1	4	5	6

How can we calculate the correct index? Let's look at the 4 in A. We know it has two elements that come before it based of its index in A. Let i denote the index of the 4 in A. Now we need to find how many elements come before it in B. To do this we can run binary search. Doing this we realize one element comes before it. Let j be the index the value, in this example it's 4, should be placed in B. Therefore, in this example we have the following:

i = 2
j = 1
A[i] = 4
C[i+j] = 4

The same algorithm can be applied to the elements in B.

If you need materials on Binary Search, a good article can be found at https://www.khanacademy.org/computing/computer-science/algorithms/binary-search/a/binary-search

Instructions: Parallel Merge with Binary Search

Edit the code where the TODOs are specified and perform the following:

- Implement the parallel merge kernel
- Implement a binary search routine on the GPU

The parallel merge kernel will be given two input arrays A and B. Your goal is to merge the two input arrays and produce an output array C. You can assume the following:

- A and B are both size N
- C is size 2N
- Both A and B are sorted arrays

The binary search routine is used as a helper function for the parallel merge kernel.

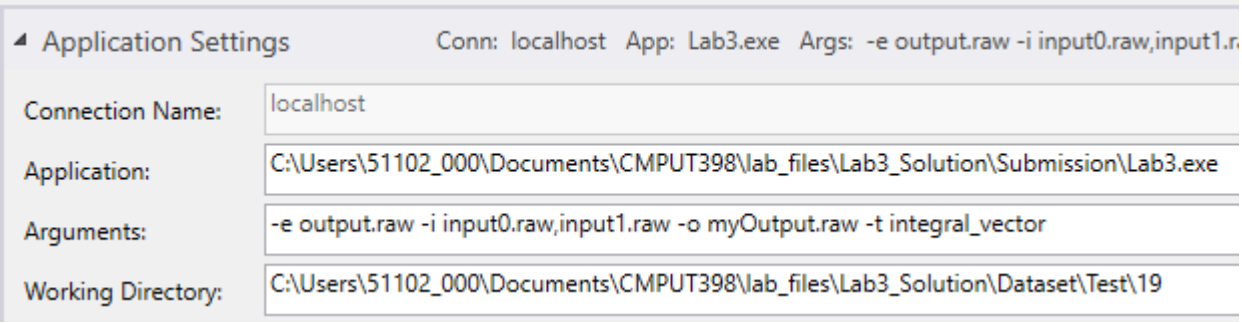
Instructions: Parallel Merge with Linear/Sequential Search

Now, implement a simple sequential search algorithm in the last TODO and replace the call to your binary search in merge sort kernel with sequential search.

After you make sure both your merge with binary search and with sequential search are correct by running the tests, do a performance analysis using NSIGHT to compare the run time of Merge using Binary Search and Merge using sequential search on **test case 19**. Save the screenshots of that show run times of the two algorithms to files "binary.png" and "sequential.png" in the Submission folder. Then, answer the question: which of them run faster and why? Put it in the Submission folder with the file name "compare.txt".

The crop shows what you should put in NSIGHT:

Arguments: -e output.raw -i input0.raw,input1.raw -o myOutput.raw -t integral_vector



Local Setup Instructions

Steps:

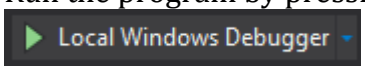
1. Download "Lab3.zip".
2. Unzip the file.
3. Open the Visual Studios Solution in Visual Studios 2013.
4. Build the project. Note the project has three configurations.

a. Debug

b. Submission

But make sure you have the "Submission" configuration selected when you finally submit.

5. Run the program by pressing the following button:



Running the "Submission" configuration from Visual Studios will throw a runtime error as there are no arguments being passed.

Testing

While working on the lab you can test your code by running the program with the "Debug" configuration selected. This will run one of the tests located in "Dataset/Test". You can also run the test script after you build with the "Submission" configuration selected. For more information, read the README located in the zip.

Note that in tests 0 to 9 the input arrays A and B don't intersect. However, in tests 10 to 19 the input arrays A and B can intersect. This can give you a hint to what might be going wrong if you are only passing the first 10 tests.

Questions

1. How many times is the binary search called? EXPLAIN.
2. Does this algorithm in the current state work if A and B are not sorted? EXPLAIN.
3. What is the best-case complexity of a merge algorithm written on the CPU? EXPLAIN.
4. What common sorting algorithm could benefit from having a parallel merge?
5. If arrays A and B don't have unique elements or the intersect of A and B is not empty. Is there an issue with doing the exact same process with A as B or vice versa? If so what is the issue and how do you fix it?

Submission

After you build your project with the "Submission" configuration selected (see above) you will see a new folder called "Submission" in the project directory. If you open the folder you will see your executable. Make sure you add your questions in the previous section as a text file. Also, the two image files "binary.png" and "sequential.png" as well as the file "compare.txt" should be in the folder. This is just like you did for previous labs. Finally zip the folder "Submission" and upload the zipped folder to eclass.

Note: You must submit files where in the code uses Binary Search, not Sequential Search

If you don't see this folder then one of two things happened. First your build could have failed and you should check for any errors in your build log in Visual Studios. Also you

might have had the “Debug” configuration selected (see above). Make sure the “Submission” configuration is selected.

Mark Breakdown

It is important that you **do not** add or remove any print statements or wb functions. We use them for part of your grading so if you make changes the marking script could fail.

Part	Breakdown (%)
Questions	20 %
Parallel Merge with Binary Search	60 %
Parallel Merge with Sequential Search	10 %
Compare the run time of the two algorithms	10 %