Name: Prosper

Surname: Mambambo

Student Number: MMBPRO002

Introduction

The aim of this report is to establish whether Parallelism is worth it and how much efficiency and speed up is offered by Parallel Programs compared to Serial Program

Brief Description of Parallelization Algorithm

My parallelization makes use of Invoke All. Which is responsible for creating the threads and executing the tasks that are contained in the collection. My program starts off by receiving the image of any dimensions(height*width) and splits the height of the image into segments or ranges. If the area of the segment of that height multiplied by the width is smaller than the sequential cut of my program, then it executes serially other than that the program is recursive and will keep halving the height until the height meets the sequential cut off. I also have a function compute directly which contains my mean filter code or median filter code depending on which of the parallel classes I'm running. Compute directly only executes when the sequential cut off is met and it ensure that it calculates either the mean or median for that given range of the height that has been halved by recursion which met the sequential cut off.

Validation

I meanly validated my parallel program by comparing it with its serial program. For the Mean Parallel program, I compared its speed together with the Mean Serial Program. Before testing the 2 program I had a hypothesis that the Parallel Program will always be faster than the Serial Program. That was the case with the results I obtained from my Parallel Mean Program vs the Serial Mean Program where the parallel program was 3 times faster (on my Computer Architecture which is an i5 Lenovo with 8 cores) than the serial Program for the set Sequential cut off. I compared the speed of the Parallel and Serial Programs where the parallel was faster but also compared the outputs to ensure that the programs were giving me the exact same results

Output for Serial:

Time taken by Serial Mean for Window Width of 11: 2.403 seconds

Output for Parallel:

Time taken by Parallel Mean for Window Width of 11: 0.708

I also did the same for the median Parallel and Median Serial Mean and the parallel Program was faster and produced the same image output as the serial Program

Program Timing

For my Parallel Programs:

```
long startTime = System.currentTimeMillis();
forkjoinPool.invoke(medianFilterParallel);
System.out.println((System.currentTimeMillis() - startTime) / 1000.F)
```

Timed the program from when invoke is called till when invoke is done with execution both for my Mean Parallel and Mean Serial Programs.

For my Serial Programs:

```
long startTime = System.currentTimeMillis();
for (int row = 0; row < (img.getHeight() - windowWidth); row++) {
    for (int col = 0; col < (img.getWidth() - windowWidth); col++) {...}
}
System.out.println((System.currentTimeMillis() - startTime) / 1000.F);
```

Timed the for loops which do either the mean or median filter from when they start execution till when they finish executing

Establishing The Optimal Serial Threshold

The issues I encountered trying to establish the optimal serial threshold/Sequential Cut Off was that I didn't want the Threshold to be too big that would have resulted in the program running serial more times. that would make the program slow and reduce its speed. I also had to think about the sequential cut off/ optimal serial threshold to not be too small so that the program would not spend more time recurring and creating excessive amounts of threads. Too many threads being created could slow down the program making it take longer to execute

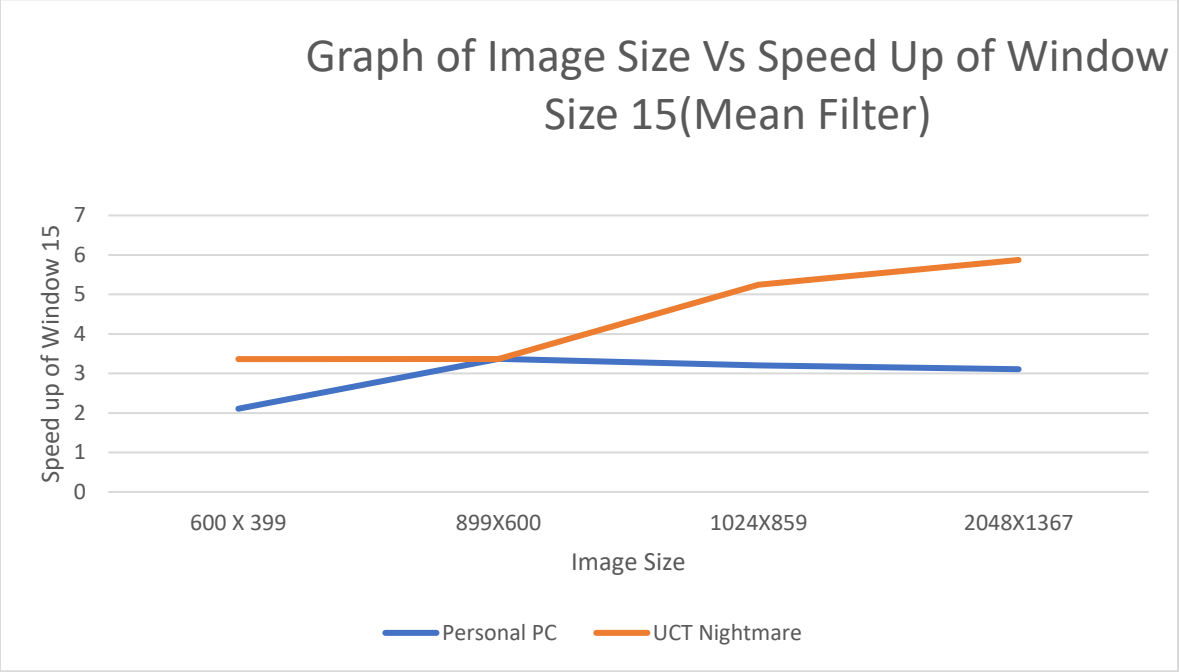
After trial and error since my program was splitting height, I decided on the optimal serial threshold of 150. so, the program would only execute serial if my height was in the range of less than 150. the serial threshold of 150 gave me the quickest time after performing trial and error with other threshold values.

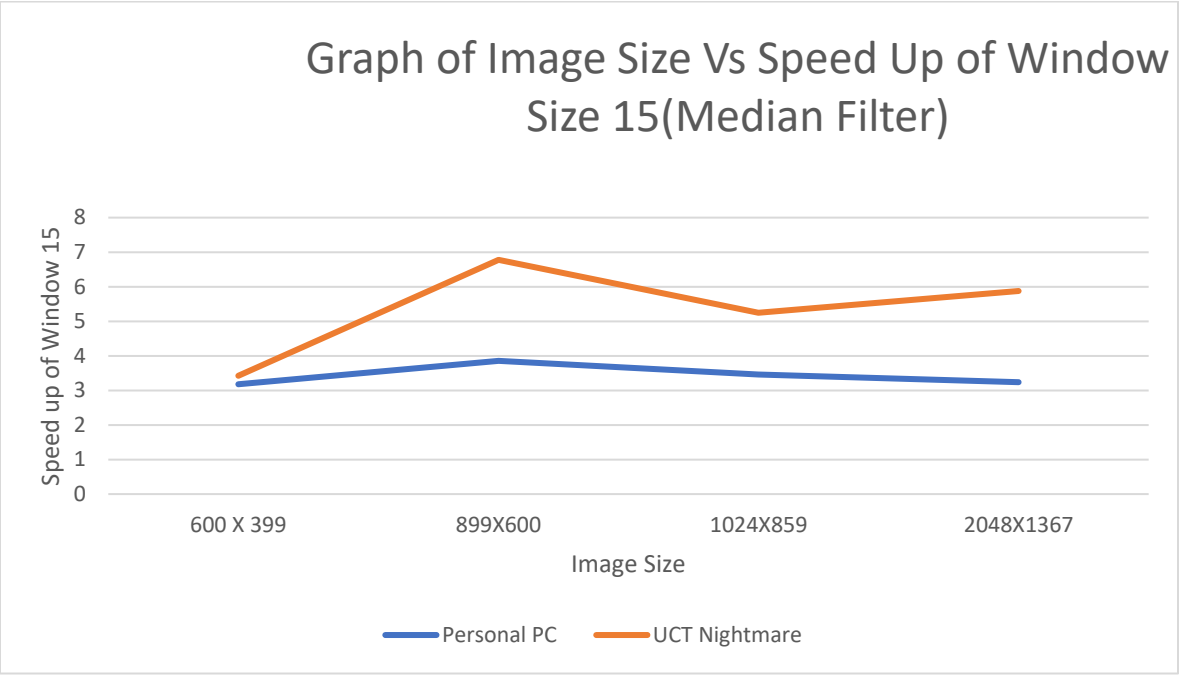
The Machine Architectures Tested On

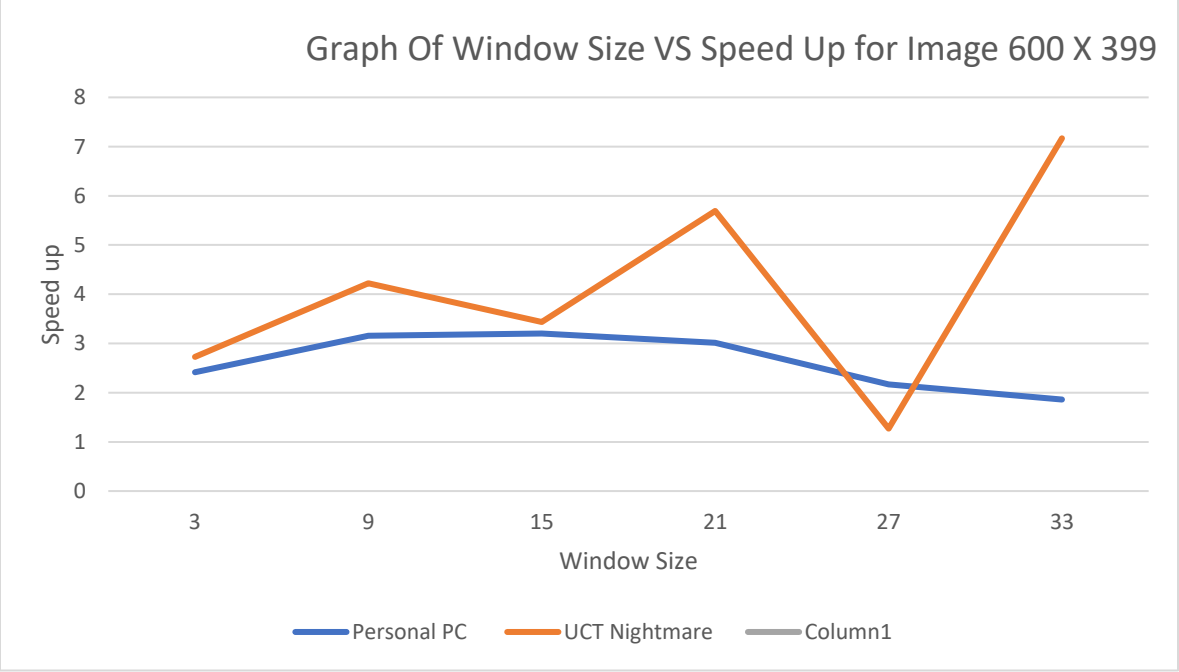
Tested my program on a i5 Lenovo with 8 cores and 20GB ram and on the UCT Computers connected to the nightmare Server with 8 cores but run faster than my computer.

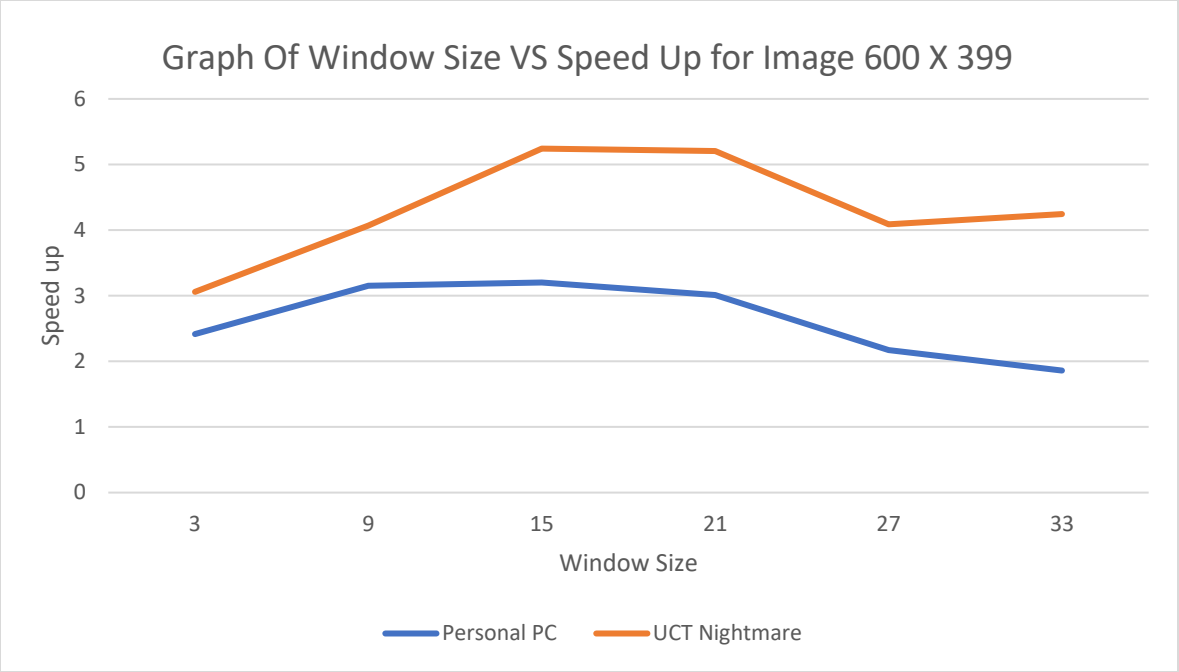
problems/difficulties encountered

- during the coding of my parallel Program for the Median Parallel Filter I concluded my Program was working as it was producing the same image as the Serial Median Filter. After timing my Serial Mean Filter and Parallel Mean Filter, I realized my Parallel code was running longer than the Serial code. This was because my sequential cut off/ Serial Threshold was greater than my picture height, so my program always ran serial due to the divided height always being greater than the sequential threshold. In fact, the coding of the height being smaller than the serial threshold was met on the first execution on my Median Filter Parallel which was inheriting from the recursive action class and no recursion was performed.
- My code was executing in less time after changing it but was displaying an image not fully filtered by the median filter that was partially filtered as my image height ranges were not fully filtering the entire image.




Graph of Image Size Vs Speed Up of Window Size 15(Median Filter)
Speed up of Window 15
8
7
6
5
4
3
2
1
0
600 X 399
899X600
1024X859
2048X1367
Image Size
Personal PC
UCT Nightmare


Graph Of Window Size VS Speed Up for Image 600 X 399
Speed up
8
7
6
5
4
3
2
1
0
3
9
15
21
27
33
Window Size
Personal PC
UCT Nightmare
Column1


Graph Of Window Size VS Speed Up for Image 600 X 399
Speed up
6
5
4
3
2
1
0
3
9
15
21
27
33
Window Size
Personal PC
UCT Nightmare

Git Usage

```
willcommits@DESKTOP-7PFOM2K:/mnt/c/Users/pmamb/Videos/MMBPRO002$ git log
commit 6c6fbc6be9159341a8fd0447ef4e7c225d052536 (HEAD -> master)
Author: willcommits <pmambambo2@gmail.com>
Date:   Sat Aug 13 00:08:51 2022 +0200

    Report and Working code

commit b268cd3e5aa4b2b8b2da6264e272aa82f3fd7704
Author: willcommits <pmambambo2@gmail.com>
Date:   Fri Aug 12 00:35:10 2022 +0200

    Working Parallel

commit f764d20e5f787a89f3dc885d666e5fd865557e28
Author: willcommits <pmambambo2@gmail.com>
Date:   Thu Aug 11 08:57:54 2022 +0200

    not working parallel

commit 1c33b357894bfdc4a2bacae241a2c63e52811c51
Author: willcommits <pmambambo2@gmail.com>
Date:   Wed Aug 10 00:17:20 2022 +0200

    WorkingSerialPrograms

commit bc4c7cb7a230861b2c758f90a9a09b03b62d5177
Author: willcommits <pmambambo2@gmail.com>
Date:   Tue Aug 9 21:36:48 2022 +0200

    First Commit-Working MeanFilter
```

Conclusion

The parallel programs on average 2 times faster than my serial Program. If the image is too small the parallel program is sometimes slower than the serial. There is no need for parallelism for a smaller dataset. The serial threshold of 150 is optimal for my Parallel Mean and Parallel Median. The median filter both the Parallel and Serial are time consuming and resource consuming as you first store to an array list then you have to sort. The mean filter is faster to execute. It's quite difficult deciding on the serial threshold as you are trying to optimize the number of threads creating to ensure that they are not too many but are just enough to make you program run faster, and you also don't want to have a serial threshold that will result in your program running in serial too much as that would make it take longer.

The UCT Computer have a higher clock speed and hence they produced faster execution time than my pc.