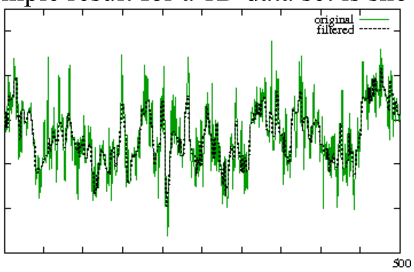
CSC2002S Tutorial 2016

Parallel Programming with the Java Fork/Join framework: 1D Median filter

Median filtering is a nonlinear digital filtering technique that is often used to remove noise from a data set. Noise reduction is usually a pre-processing step to that improves or "cleans" the data in preparation for later processing. In this method, a median filter slides over the data item by item, generating a new data set where each item is replaced by the median of neighbouring entries. An example result for a 1D data set is shown in the image below.



The size of the filter determines number of neighbours considered for the median. In this assignment, you will consider only one-dimensional data sets and filters of odd size, ranging from 3 to 21 data items.

> For example, a median filter of size 3 applied to the simple 1D array x:
> ```
> x = [2 80 6 3 1]
> ```
> produces output array y:
> ```
> y = [2 6 6 3 1]
> ```
> where each element of y is calculated as follows:
> ```
> y[1] = Median[2 80 6] = 6
> y[2] = Median[80 6 3] = 6
> y[3] = Median[6 3 1] = 3
> ```
> Note that the borders are not changed.

The naïve approach to median filtering sorts the elements within the filter window at each step to calculate the median (which will then be the middle element).
Using this naïve approach (which we will do in this assignment) the median computation can be quite expensive, especially if the filter window is large. Therefore, in this assignment you will attempt to parallelize the problem in order to speed it up. You will:

- Use the Java Fork/Join framework to parallelize the median filter operation using a divide-and-conquer algorithm.

- Evaluate your program experimentally:

    - Using high-precision timing to evaluate the run times of your serial and your parallel method, across a range of input sizes and a range of filter sizes, and

    - Experimenting with different parameters to establish the limits at which sequential processing should begin.

- Write a report that lists and explains your findings.

Note that parallel programs need to be both **correct** and **faster** than the serial versions. Therefore, you need to demonstrate both correctness and speedup for your assignment.

Assignment details and requirements

Your program will smooth one-dimensional time series data using a median filter. The filter will not smooth the endpoints.

1.1 Input and Output

Your program must take the following command-line parameters:

<data file name> <filter size (must be an odd integer >= 3)> <output file name>

You can assume that the data file starts with an integer on a single line, specifying the number of subsequent lines in the file. Successive line contain two numbers separated by spaces:

<line number> <floating point value>

We will provide you with sample input files of different sizes (on Vula), though you may also create your own.

Your program should output the cleaned 1D data set in the same format as the input file.

You will also need to display timings for parallel versus serial code, but how you output/display these is up to you (and should be reported on).

1.2 Report

You must submit an assignment report in pdf format. Your clear and concise report should contain the following:

- An *Introduction*, comprising a short description of the algorithm and why it is worth considering parallelization.

- A *Methods* section, giving a description of your approach to the solution, with details on the parallelization. This section must explain how you validated your algorithm (showed that it was correct), as well as how you timed your algorithms with different input, how you measured speedup, the machine architectures you tested the code on and interesting problems/difficulties you encountered.

- A *Results and Discussion* section, demonstrating the effect of data sizes, filter sizes and sequential limits on parallel speedup. This section should include speedup graphs and a discussion. In this section, we expect you to should answer the following questions:
  - Is it worth using parallelization (multithreading) to tackle this problem in Java?
  - For what range of data set sizes and filter sizes does your parallel program perform well?
  - What is the maximum speedup obtainable with your parallel approach? How close is this speedup to the ideal expected?
  - What is an optimal sequential cutoff for this problem? (Note that the optimal sequential cutoff can vary on dataset size and filter size.)

- A *Conclusions* section (note the plural) listing the conclusions that you have drawn from this project. What do your results tell you and how significant or reliable are they?

Please do NOT ask the lecturer for the recommended numbers of pages for this report. Say what you need to say: no more, no less.

1.3 Assignment submission requirements

- You will need to create and submit a GIT archive (see the instructions on Vula on how to do this).
- Your submission archive must consist of BOTH a technical report and your solution code and a **Makefile** for compilation.
- **Label** your assignment archive with your **student number** and the **assignment number e.g. KTTMIC004_CSC2002S_Assignment1.**
- Upload the file and **then check that it is uploaded.** It is your responsibility to check that the uploaded file is correct, as mistakes cannot be corrected after the due date.
- The usual late penalties of 10% a day (or part thereof) apply to this assignment.

Due date:

9AM ON 11TH AUGUST 2015

- **Late** submissions will be **penalized at 10%** (of the total mark) per day, or part thereof.
- The deadline for marking **queries** on your assignment is **one week after the return of your mark.** After this time, you may not query your mark.

1.4 Extensions to the assignment

If you finish your assignment with time to spare, you can attempt one of the following for extra credit:

- Compare the performance of the Fork/Join library with standard threads.
- Compare performance with a mean filter, which replaces a value with the mean of the surrounding elements.
- Implement a 2D dataset and a 2D median filter.
- Investigate and implement linear time solutions to median filtering.

1.5 Assignment marking

Your mark will be based primarily on your analysis and investigation, as detailed in the report.

Rough/General Rubric for Marking of Assignment 1	
Item	Marks
Coding correctness and style	20
Introduction	5
Methods	10
Results	50
Going the extra mile - excellence/thoroughness/extra work - e.g. additional investigations, depth of analysis, different hardware etc.	10
Conclusions	5
Total	100

Note: submitted code that does not run or does not pass standard test cases will result in a mark of zero. **Any plagiarism, academic dishonesty, or falsifying of results reported will get a mark of 0 and be submitted to the university court**.