CSC2002S Tutorial 2018

Parallel Programming with the Java Fork/Join framework: Photosynthesis

There is a branch of Botany that is concerned with simulating vegetation patterns using computerized models. This has implications for anticipating the effect that climate change will have on vegetation. These models scale in computational complexity with the size of the input terrain, which in some cases can span entire continents. They are therefore prime candidates for acceleration using parallel programming.

One of the most important factors affecting a tree's growth is its exposure to sunlight. In this assignment you will efficiently calculate the average sunlight received by trees in a forest (and also record the sunlight exposure of individual trees).

For the purposes of this assignment the landscape is represented by a regular matrix of floating point values. Each value represents the average hours of sunlight per day falling on a 1m x 1m area of the terrain. Not all areas receive the same amount of sunlight because of folds in the landscape (see below).

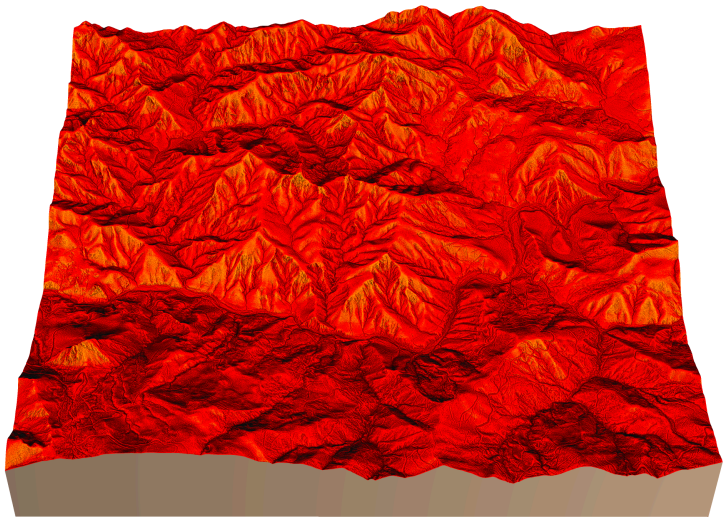


Figure 1: Average sun exposure during March for a 3km x 3km section of terrain in California. North facing slopes receive less sunlight (and are represented by a darker red in the colour map). South facing slopes tend to receive more sunlight (a bright red or orange in the colour map).

Tree canopies are represented (rather unrealistically) as a square coverage area on the terrain. For the forest you will be provided with the top left (x, y) terrain coordinate of a tree's canopy indexed from (0,0) and the extent in meters (e).

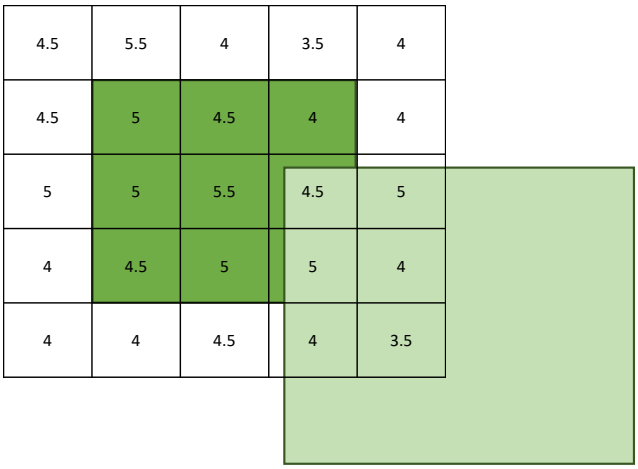
Your task is to calculate both the total sunlight exposure for each tree canopy (the sum of the sunlight values within its canopy area) and the mean sunlight exposure for all trees.

The file input format is as follows:
<terrain x size – INT> <terrain y size – INT>

<avg. hours sunlight at (0,0) – FLOAT> <avg. hours sunlight at (0, 1) – FLOAT> … <avg. hours sunlight at (xsize-1, ysize-1) – FLOAT>
<number of trees – INT>
<x corner of tree 0 – INT> <y corner of tree 0 – INT> <canopy extent of tree 0 – INT>
…
<x corner of tree (numtrees-1) – INT> <y corner of tree (numtrees-1) – INT> <canopy extent of tree (numtrees-1) – INT>

For example, a very small 5m x 5m section of terrain, with 2 trees would be encoded in the input file as:
5 5
4.5 5.5 4 3.5 4 4.5 5 4.5 4 4 5 5 5.5 4.5 5 4 4.5 5 5 4 4 4 4.5 4 3.5
2
1 1 3
2 3 4

Visually this corresponds to:



Note that some trees may extend beyond the edge of the terrain (as is the case above for the light green tree) and these parts of the canopy should be ignored. Trees may also overlap, but for the purposes of this simulation they do not occlude each other.

The file output format is as follows:
<avg. sunlight per tree – FLOAT>
<number of trees – INT>
<total sunlight tree 0 – FLOAT>
…
<total sunlight tree (numtrees-1) – FLOAT>

For the example above this would mean an output of:
34.5
2
43
26

In this assignment you will attempt to parallelize the problem in order to speed it up. You will be required to:

- Use the Java Fork/Join framework to parallelize the sunlight exposure calculations 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
    - 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 calculate the sunlight exposure of individual trees in parallel as well as the average sunlight exposure of all trees in the forest.

1.1 Input and Output

Your program must take the following command-line parameters:
<data file name> <output file name>
The format of these files must follow the specification provided above.

We will provide you with a sample input file (on Vula), though you may also create your own using random data.

1.2 Benchmarking

You must time the execution of your parallel sorts across a range of data **sizes** and **number of threads** (sequential cutoffs) and report the speedup relative to a serial implementation. The code should be tested on at least **two different machine architectures** (at least one of which must be a multi-CPU/multi-core machine).

Timing should be done at least 5 times. Use the `System.currentTimeMillis` method to do the measurements. Timing must be restricted to the sunlight exposure code and must exclude the time to read in the files and other unnecessary operations, as discussed in class. Call `System.gc()` to minimize the likelihood that the garbage collector will run during your execution (but do not call this within your timing block!).

1.3 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 aim of the project, the parallel algorithms and their expected speedup/performance.
- 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 and numbers of threads and different architectures on parallel speedup. This section should **include speedup graphs** and **a discussion**. Graphs should be clear and labelled (title and axes). In the discussion, we expect you to address the following questions:
    - Is it worth using parallelization (multithreading) to tackle this problem in Java?
    - For what range of data set 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 based on dataset size and architecture.)
    - What is the optimal number of threads on each architecture?
- A *Conclusions* (note the plural) section 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.

1.4 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:

10am on 17th September 2018

- **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.

1.5 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.
- Replace the square canopy with a circular canopy and measure the impact that this has on performance.
- Not all trees absorb sunlight with the same efficiency. Add a per tree weighting factor for the sunlight exposure. Again, see what impact this has on speedup.

1.6 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
Code – conforms to specification, code correctness, timing, style and comments	40
Documentation - all aspects required in the assignment brief are covered, e.g. Introduction, Methods, Results (with graphs and analysis), Conclusions.	50
Going the extra mile - excellence/thoroughness/extra work - e.g. additional investigations, depth of analysis, etc.	10
Total	100
Code does not execute / run	-50
Code executes but no GIT repository	-20
GIT repository but no regular updates (at least on 5 separate days)	-15

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**.