Storms of high-energy particles

EduHPC'18 Peachy Assignment

1 Introduction

You are provided a sequential code that simulates the effects of high-energy particles bombardment on an exposed surface, for example in the surface of space vessels in outer space.

It is conveniently simplified, considering only a cross section of the surface. The section is represented by a discrete number of control points equally distributed on the outermost layer of the material. An array is used to store the amount of energy accumulated on each point. The program computes the energy accumulated on each point after the impact of several waves of high-energy particles (see figure 1). The program calculates and reports, for each wave, the point with the highest accumulated energy, which presents the higher risk of being damaged.

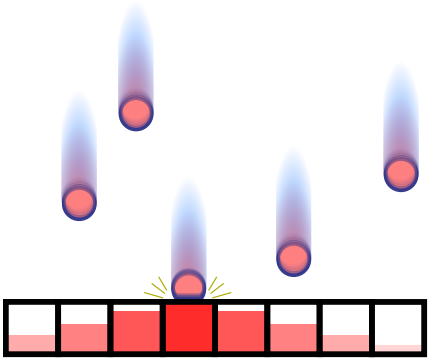

Figure 1: A collection of high-energy particles approaching the target surface. The impacting particle transfers energy to the impact point and to its neighborhood.

Figure 2 shows the output of the program for an array of 30 control points, after three waves with several random particles on each one. For each control point the output shows its final energy value, and a string of characters that graphically represents the value as a bar. The length of the bar is normalized with the maximum energy after the last wave. If the last character of the bar is an "x", it indicates that this point was identified as a local maximum after the last wave. An "M" character followed by a number at the end of the bar indicates that the value in the point was identified as the highest energy after the wave indicated on the associated number. This graphical representation is optionally shown when the program is compiled in debug mode. It is followed by the execution time of the main computation, excluding the times for start-up, reading data from files, and writing the output. The last line shows the list of positions and maximum energy values after each wave.

2 Sequential code description

Program arguments The program receives the following arguments:

1. size: Number of control points, or array positions to store energy values.
2. list of wave files: A sequence of filenames with the information of each wave. The same file name can appear several times.

```
    0.3996 |oooooooooooooooooooooooooooooooooooooo
    0.4325 |ooooooooooooooooooooooooooooooooooooooo
    0.4684 |oooooooooooooooooooooooooooooooooooooooo
    0.5092 |ooooooooooooooooooooooooooooooooooooooooo
    0.5308 |oooooooooooooooooooooooooooooooooooooooooo
    0.5398 |ooooooooooooooooooooooooooooooooooooooooooox
    0.5371 |ooooooooooooooooooooooooooooooooooooooooooo
    0.5472 |ooooooooooooooooooooooooooooooooooooooooooo
    0.5646 |oooooooooooooooooooooooooooooooooooooooooooo
    0.5914 |ooooooooooooooooooooooooooooooooooooooooooooo
    0.6009 |oooooooooooooooooooooooooooooooooooooooooooooooooooox M2
    0.6004 |ooooooooooooooooooooooooooooooooooooooooooooooooooooo
    0.5878 |ooooooooooooooooooooooooooooooooooooooooooooooooooooo
    0.5858 |ooooooooooooooooooooooooooooooooooooooooooooooooooooo
    0.5829 |ooooooooooooooooooooooooooooooooooooooooooooooooooooooo M0
    0.5692 |ooooooooooooooooooooooooooooooooooooooooooooooooooooooo
    0.5464 |ooooooooooooooooooooooooooooooooooooooooooooooooooooo
    0.5219 |ooooooooooooooooooooooooooooooooooooooooooooooooooooo
    0.5078 |ooooooooooooooooooooooooooooooooooooooooooooooooooo
    0.5042 |ooooooooooooooooooooooooooooooooooooooooooooooooooo
    0.5079 |ooooooooooooooooooooooooooooooooooooooooooooooooooo
    0.5122 |ooooooooooooooooooooooooooooooooooooooooooooooooooo
    0.5157 |ooooooooooooooooooooooooooooooooooooooooooooooooooo
    0.5238 |ooooooooooooooooooooooooooooooooooooooooooooooooooo
    0.5363 |ooooooooooooooooooooooooooooooooooooooooooooooooooo
    0.5488 |oooooooooooooooooooooooooooooooooooooooooooooooooox
    0.5475 |ooooooooooooooooooooooooooooooooooooooooooooooooooo M1
    0.5297 |ooooooooooooooooooooooooooooooooooooooooooooooooooo
    0.4957 |ooooooooooooooooooooooooooooooooooooooooooooooooo
    0.4537 |ooooooooooooooooooooooooooooooooooooooooooooooooo

Time: 0.000042
Result: 14 0.381967 26 0.499453 10 0.600869
```

Figure 2: Example of output of the program in debug mode for an array of 30 positions and three waves of particles.

Wave file format Each particle information is read from a text file and stored in an array. The base type of this array is a structure with fields for the details of each particle on the wave. The first line of a file contains the number of particles described in it. Each line afterwards contains the data of one particle in the wave. There are two integer data for each particle, separated by a white space: The index of the impact point, and the energy value in thousandths of a Joule.

Students are encouraged to manually create or automatically generate their own test files. Several examples are provided along with the code.

Functional description The program first reads the wave files and stores each one in an array of particles. Then, for each wave, it executes the same stages. The first one is the bombardment phase. For each particle in the wave, the program transforms the energy to a float value in Joules, and traverses the array positions computing how much energy should be accumulated at each point. When a particle impacts, its energy is transmitted to the contact point and to those in the neighborhood. The exact energy accumulated at a point is calculated taking into account an attenuation in terms of the distance to the impact point. The farthest the impact point, the lower the energy accumulated. However, there is a minimum energy threshold (which usually depends on the material) that can be accumulated due to the impact. No energy is accumulated at the points where the minimum threshold is not reached, due to its distance to the impact point.

After the bombardment stage of each wave, a relaxation process takes place. In this phase the material reacts by slightly distributing its charge. Each control point is updated with the average value of three points: the previous one, the point itself, and the next one. To avoid destroying the original values in the array while they are still needed to update other neighbors,

the array values are first copied to a second ancillary array before the relaxation. The old values are read from the ancillary array, while the new values are written to the original array.

Finally, in the last stage that process each wave, the point with the maximum energy is located and its value and position are stored. After all the waves have been processed, the maximums and positions for all waves are printed.

Debug mode If the source is compiled with the -DDEBUG flag, and the array size is not greater than 35 cells, a graphical representation of the results is presented, as discussed before. This output can be compared before and after program modifications. Consider executing a program several times with the same parameters to try to detect random changes in the output originated by race conditions.

3 Project goal

Use the proposed parallel programming model to parallelize the program, optimize the code, and obtain the best possible performance.

Code modifications allowed Students can modify the sequential code provided as long as they observe the following restrictions:

- Exploit parallelism using only the parallel programming model proposed.
- The argument processing section, array memory allocations, time measurement points, and output of results, should not be changed. The section of code that the students should target and modify is clearly identified in the main function. This section is found between the points where the time measurement is started and ended. Functions defined in the program that are called from the target section of the code can also be modified, substituted, or eliminated.
- Any change in the algorithm or data structures must be discussed with the teachers in advance, in order to avoid modifications that significantly alter the parallelism exploitation with the proposed parallel programming model, which is the purpose of the assignment.

To measure execution times, compile the program with the maximum optimization level of the compiler (for example gcc -O3). We want to focus on program changes that do not interfere, and even facilitate, compiler optimizations.