NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTAMENTO DE
INFORMÁTICA

DEPARTAMENT OF
COMPUTER SCIENCE

Project Assignment: Storms of High-energy Particles

João Lourenço

April 29, 2021

Abstract

Please read this document carefully... be certain that it took me much longer to write it than it will take you to read it!

This is the project assignment for the course of Concurrency and Parallelism, edition 2020-21. This assignment is strongly based in the *EduHPC'18 Peachy Assignment* from the Trasgo Research Group of the Universidad de Valladolid.[1]

The deadline for the project submission (code and report) is June 1, 2021.

The project submission is done by: i) adding me as an "observer" (or a "developer") to your Git repository; and ii) fill a form with the commit ID of your code and project (the date/time od the commit ID must be before the deadline). I More info on the project submission will be provided later.

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.

The problem and the provided code are 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.

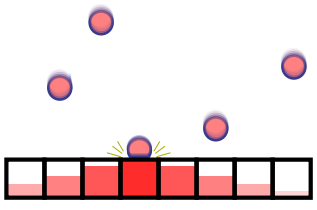

Figure 1: A collection of high-energy particles approaching the target surface. The impacting particle transfers energy to the impact point and to its neighborhood.

[1] Please remember that you are expected to develop your own project solution, and that any attempt of cheating implies immediate failure in the course!

Campus de Caparica
2829-516 CAPARICA | Portugal
T: +351 212 948 536
di.secretariado@fct.unl.pt
di.fct.unl.pt

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.

```
0.3996 |ooooooooooooooooooooooooooooooooooooo
0.4325 |oooooooooooooooooooooooooooooooooooooooo
0.4684 |ooooooooooooooooooooooooooooooooooooooooooo
0.5092 |ooooooooooooooooooooooooooooooooooooooooooooo
0.5308 |oooooooooooooooooooooooooooooooooooooooooooooo
0.5398 |ooooooooooooooooooooooooooooooooooooooooooooox
0.5371 |oooooooooooooooooooooooooooooooooooooooooooooo
0.5472 |oooooooooooooooooooooooooooooooooooooooooooooo
0.5646 |ooooooooooooooooooooooooooooooooooooooooooooooo
0.5914 |oooooooooooooooooooooooooooooooooooooooooooooooo
0.6009 |ooooooooooooooooooooooooooooooooooooooooooooooooox M2
0.6004 |ooooooooooooooooooooooooooooooooooooooooooooooooo
0.5878 |ooooooooooooooooooooooooooooooooooooooooooooooooo
0.5858 |ooooooooooooooooooooooooooooooooooooooooooooooooo
0.5829 |ooooooooooooooooooooooooooooooooooooooooooooooooo M0
0.5692 |ooooooooooooooooooooooooooooooooooooooooooooooooo
0.5464 |ooooooooooooooooooooooooooooooooooooooooooooooooo
0.5219 |ooooooooooooooooooooooooooooooooooooooooooooooooo
0.5078 |ooooooooooooooooooooooooooooooooooooooooooooooooo
0.5042 |ooooooooooooooooooooooooooooooooooooooooooooooooo
0.5079 |ooooooooooooooooooooooooooooooooooooooooooooooooo
0.5122 |ooooooooooooooooooooooooooooooooooooooooooooooooo
0.5157 |ooooooooooooooooooooooooooooooooooooooooooooooooo
0.5238 |ooooooooooooooooooooooooooooooooooooooooooooooooo
0.5363 |ooooooooooooooooooooooooooooooooooooooooooooooooo
0.5488 |ooooooooooooooooooooooooooooooooooooooooooooooooox
0.5475 |ooooooooooooooooooooooooooooooooooooooooooooooooo M1
0.5297 |ooooooooooooooooooooooooooooooooooooooooooooooooo
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.

2 Sequential Code Description

2.1 Program arguments

The program receives the following arguments:

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

2.2 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. You are allowed (and encouraged) to share your own wave files and simulation results with your colleagues, preferably publicly in the Piazza forum.

2.3 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 as listed below.

1. **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.

2. **The relaxation process.** 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.

3. **Maximum energy point location.** In this 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.

2.4 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 Development and Submission

3.1 Working rules

The following working rules apply. As this list is not exhaustive, in case of doubt do not hesitate in asking me.

- The project work will be done in grops of 3 students (unless explicitly authorized otherwise).
- Each group must fork the given *original repository* to create a new project in one the publicly available Git hosting sites (e.g., github, gitlab, bitbucket) which will become the group's remote master repository. Your new master repository name must be "`cp2020-2021_gNN_AAAAA_BBBBB_CCCCC`", where *gNN* is your group number (e.g., "*g07*"), *AAAAA*, *BBBBB*, and *CCCCC*, are the student numbers (IDs) of the three group members sorted ascending, i.e., lower(AAAAA) to higher (CCCCC).
- Remember that **your new repository must be private** and shared only among the group members and me (search for me by my email "`joao.lourenco@fct.unl.pt`" or use my ID "`joaomlourenco`" in GitHub and BitBucket and "`joaomlourenco1`" in GitLab). Add me to your repository as a *viewer* (read-only) if possible, or as a *developer* otherwise.
- The group must respect and use the "Git Workflow". Each member will have a local copy of the repository in his/her computer where individual work shall take place.
- The group members must make extensive use of branching (and merging). One branch for each work-task.

- Each student will commit his/her own individual work into the group shared repository.
- Commit messages must describe clearly what was changed/added in that commit (and why). Commit messages are very important. Take you time to write good commit messages!
- Remember to push your (local) commits to your group remote master repository.
- Each project must have a `README.md` file with instructions on how to compile and run the test programs.
- The project report must have the look and feel of a research article, using the template provided in the *original repository*, with a maximum of 4 pages (see Section 3.4 for more info on the Project's Report). You may (and should) use Goole to learn on how to write a research article.

3.2 Working methodology

You are given a reference sequential version of the code (`energy_storms.c`), and another identical source file (`energy_storms_omp.c`) to develop the parallel OpenMP version. The following strategy is recommended:

1. Download the source code by forking the following repository: https://github.com/joaomlourenco/cp2020-21_project_assignment.
2. Study the original source code and be sure you understand it. Please feel free to discuss the source code with your colleagues (and the teacher) in Piazza or, if necessary, directly with the teacher;
3. Identify the best candidate statements/functions to exploit parallelism using OpenMP;
4. Identify and remove the code dependences that may hinder the parallelization correctness and/or performance impact;
5. Use the OpenMP programming model to parallelize the program.
6. Run some preliminary tests to confirm the correctness of your parallelization approach;
7. Evaluate the impact of your parallelization;
8. Find potential performance bottlenecks (maybe using a profiler);
9. If any performance bottleneck is found, fix it and goto step 6;
10. If no performance bottleneck is found, look for additional (minor) blocks/functions that can be parallelized;
11. If some is found goto step 4;

12. Log into the cluster (instructions will be provided later) and run **selected** performance tests there (remember that your CPU time in the cluster is limited);

13. If your results in the cluster do not match your expectations, go back to your development machine and to step 4;

14. Write your report. See Section 3.4 for suggestions/recommendations/rules for the report.

3.3 Grading Criteria

Remember my talk about the Bloom's Pyramid (see Figure 3). I will certainly try to assess both ***your group*** and ***your individual competences*** in all the levels, but I will certainly value more your outcome on the top three levels: *Analyze*, *Evaluate*, and *Create*.

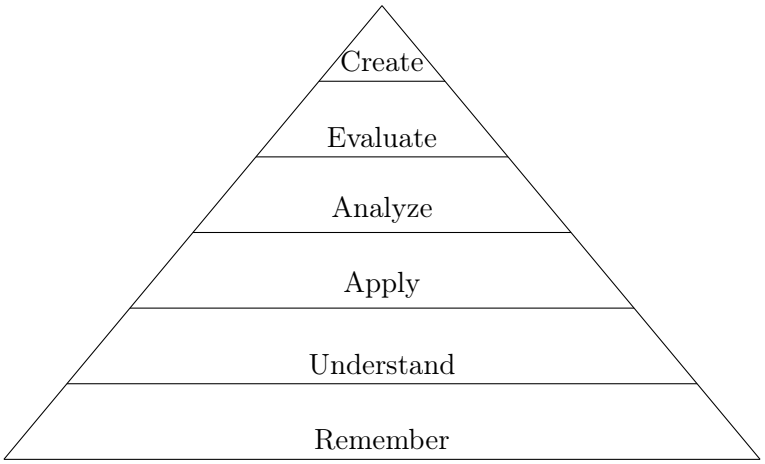

Figure 3: Bookm's Pyramid

This means that spending almost all of your project time programming and optimizing your and, and you do your final benchmarking in the last hours to quickly write your report, most certainly you will have a project grade much below your expectations. So, please...

> ...leave **at least** one week for benchmarking your solution, and for creating and writing your project report!

3.3.1 Group grading criteria ($\mathcal{G}$)

The group grade ($\mathcal{G}$) is the same to all the elements of the group and will strongly depend on my appreciation of the group's work (on all the levels of the Bloom's Pyramid) **as perceivable from the written report**. Other relevant criteria include: easy of compilation and execution, cleanness and readability of the source code, correctness of the results, execution time/performance, scalability, etc.

3.3.2 Individual grading criteria ($\mathcal{S}$)

The individual grade ($\mathcal{S}$) is possibly different for each group member and **will strongly depend** on both, the work division as reported by the group, and on my own appreciation of the individual contribution of each group member (on all the levels of the Bloom's Pyramid) to the joint group's work as perceivable from the written report. Other relevant criteria include: compliance to the "Git Workflow", size and number of commits, meaningfulness of commit messages, etc.

3.3.3 Individual lab grading ($\mathcal{L}$)

The individual lab grade ($\mathcal{L}$) will be calculated as a weighted averaged following the formula:

$$\mathcal{L} = 0.7 \times (\mathcal{G}) + 0.3 \times (\mathcal{S})$$

3.3.4 Grade discussion

Both the group members and the professor may ask for an individual or collective (group) presentation and discussion of the project. As a result of such a presentation and discussion, both the group grade ($\mathcal{G}$) and the individual grade ($\mathcal{S}$) may be revised up or downwards.

3.4 Project Report Format, Structure, and Important Dates

The report shall be written **strictly** following the format provided in the `Templates` folder in the source repository and have the *look&feel* of a research paper. The report length is **limited to four pages** of text and graphics, plus **one extra page** for:

Section	Description
`bibliography`	Remember to add **all the documents and web-sites** relevant to your work to the bibliography section... and remember that you must cite them in the text wherever they were relevant;
`acknowledgments`	Please identify (number and name) your colleagues that were somehow helpful to your group while developing the project... also explain why/how they helped you;
`individual contribution(s)`	Please present your group working methodology and list clearly: i) how the work was divided between the group members, and ii) the relative contribution of each member (in percentage). For example, *A did whatever (20%), B did something else (30%), and C did a lot of stuff (50%).* If the group members cannot agree in the terms/contents for this sectin, add one (identified) separate statement for each group member.

`comments, critics, and suggestions`	All your comments, critics and suggestions are very welcome. My goal is to provide you with an interesting and educational project (and course) and all your feedback is very welcome (of course this section **will have no impact on your grade**).

Version history:

Date	Version	Description
2020-04-29	`1.0`	Initial version.