Peachy Parallel Assignments (EduPar 2019)

Ozcan Ozturk
Bilkent University
Ankara, Turkey
ozturk@cs.bilkent.edu.tr

Ben Glick
Lewis & Clark College
Portland, OR, USA
glick@lclark.edu

Jens Mache
Lewis & Clark College
Portland, OR, USA
jmache@lclark.edu

David P. Bunde
Knox College
Galesburg, IL, USA
dbunde@knox.edu

***Abstract*—Peachy Parallel Assignments are a resource for instructors teaching parallel and distributed programming. These are high-quality assignments, previously tested in class, that are readily adoptable. This collection of assignments includes face recognition, finding the electrical potential of a square wire, and heat diffusion. All of these come with sample assignment sheets and the necessary starter code.**

***Index Terms*—Parallel computing education, High-Performance Computing education, Parallel programming, OpenMP, Java Executor framework, Local binary patterns, Poisson's equation, Numerical differential equations**

I. Introduction

An important part of teaching a course on parallel and distributed computing is creating engaging programming assignments. Students generally spend more time working on homework than engaging with other aspects of the course, making assignments integral both to student learning and student perceptions of the subject matter. That said, creating great assignments is a large time commitment and is not guaranteed to succeed; sometimes even seemingly-great assignment ideas turn out to have flaws when implemented and given to students. To help overcome these issues, we are presenting Peachy Parallel Assignments at the Edu* series of workshops, most recently EduHPC 2018 [1].

Peachy Parallel Assignments all go through a competitive review process based on the following criteria:

- Tested — All Peachy Parallel Assignments have been successfully used in a class.
- Adoptable — Peachy Parallel Assignments are easy to adopt. This includes not only the provided materials, but also the content being covered. Ideally, the assignments cover widely-taught concepts using common parallel languages and widely-available hardware, have few prerequisites, and (with variations) are appropriate for different levels of students.
- Cool and Inspirational — Peachy Assignments are fun and inspiring for students. They encourage students to spend time with the relevant concepts. Ideal Peachy Assignments are those that students want to demonstrate to their roommate.

This effort is inspired by the SIGCSE conference's Nifty Assignment sessions, which focus on assignments for introductory computing courses. (See http://nifty.stanford.edu for more details.)

In this paper, we present the following Peachy Parallel Assignments:

- Face recognition using Local Binary Patterns (LBPs),
- Computing the electric potential of a square wire, and
- Animating heat diffusion

See the Peachy Parallel Assignments webpage (https://grid.cs.gsu.edu/~tcpp/curriculum/?q=peachy) for the materials (e.g. assignment handouts and starter code) for these assignments. The website also lists Peachy Parallel Assignments presented previously. We hope that you find these assignments useful and encourage you to consider submitting your own great assignments for future presentation!

II. Face Recognition — Ozturk

Our first assignment requires students to use OpenMP to implement a parallel face recognition algorithm using the idea of Local Binary Patterns (LBP) [2]. Local Binary Patterns is a simple but effective idea in texture analysis, which is also used for face recognition and gesture recognition applications. To apply this technique to an image in the dataset, it compares the brightness of each pixel in the image to its 8 neighbors. Through these comparisons, the pixel is converted into an 8-bit number, where each bit indicates the result of the comparison ("0" if greater, "1" otherwise). The algorithm then computes a histogram of these numbers, which is used as a 256-dimensional feature vector for comparing images.

We used this application to create an assignment based on OpenMP for students with no previous exposure to parallel programming topics. It has been used in multiple offerings of an elective course named "Parallel Computing" at Bilkent University, Turkey [3]. The students were expected to know general programming at least at the level of CS1/CS2 and those who are proficient in C/C++ programming benefit from this assignment the most. In order to understand the LBP implementation, it is also necessary for students to have some Calculus background. Based on these prerequisites, we believe this assignment could be used at the sophomore level or above.

A. Assignment

In class, students are first briefly introduced to shared memory architectures and then to the basic concepts of OpenMP programming with examples. For the assignment itself, the LBP-based algorithm is explained so students can understand its logic and they are provided with a detailed description of it. They are asked to write a serial implementation of it, profile this implementation to determine which parts do most of the work, and then parallelize their implementation using OpenMP.

Students are also provided with image data sets to test their implementations. Each of these is split into two parts based on a parameter k, with the first k images being the training set while the others are used for testing. Most students obtained the expected results. Figure 1 shows accuracy results collected by two separate students. As can be seen in this figure, the accuracy of the implementation varies with k, ranging from 0.96 (for $k = 1$) to 1 (for $k = 10$ and beyond). From both sets of results, it is clear that 100% accuracy is achieved with $k = 10$.

Students were referred to online sources of information for studying the topic, especially the tutorials and study guides provided by OpenMP Architecture Review Boards (ARB) [4].

Details of the assignment and necessary files can be found at http://cs.bilkent.edu.tr/~ozturk/OpenMP_assignment.html.

B. Strengths and Weaknesses

We believe that a key strength of this assignment is student enthusiasm for face recognition. This is a cool problem and, with its use in many cell phones, is also perceived as practical. Students provided anonymous feedback through the standard Bilkent University course evaluation surveys given at the end of every semester. Students' written responses reflected considerable enthusiasm for this project: "Provided real world context with the emerging parallel applications", "Helped me understand how OpenMP works", and "I discovered a new career path". In addition to the anonymous course evaluation surveys, the course was discussed with students in an informal environment after the semester had finished. This discussion showed that students are very keen to learn more about parallel architectures, programming, and OpenMP development. The majority of them suggested that it would be a good idea to introduce OpenMP earlier and integrate it as a part of the computer engineering undergraduate curriculum.

In addition, we believe that this assignment is suitable for integration into a variety of courses at the sophomore level or higher. This is mainly due to the fact that OpenMP pragmas can easily be understood and the given assignment is not too sophisticated despite being interesting.

The main weakness we see in the assignment is that it gives limited scope for student creativity. In most cases, students parallelize their application simply by adding a few pragmas to different loop nests. While students need to understand and test different configurations, available options are limited since students have only introductory knowledge of OpenMP. Therefore, if the student wants to learn deeper optimizations, she will be on her own.

C. Variations

In the assignment's current form, students are given clear step by step instructions. A possible variation is to let students develop their own implementations without such detailed guidelines. A different variation is to eliminate the profiling step. Specifically, students can figure out the complexity of the computation tasks in the application and select the loops for parallelization themselves. This may improve the assignment in terms of creativity. Finally, different sets of images could be selected to port the assignment to a different domain.

III. Finding the Electric potential of a Square Wire — Glick, Mache

Our second assignment explores the following situation: A wire is bent in the shape of a square of length 1 m and placed in the $X_1 - X_2$ plane, inside a conducting square of length $2m$. The conducting square is connected to the ground so that its potential is 0, as shown in Figure 2. The wire carries a constant linear charge density of $\rho = 1$ in some appropriate units. This means that there is a fixed amount of charge which is distributed along the square wire. Because the square wire is mounted onto a conducting plate, the charge will dissipate along the conducting plate over time. Eventually, the plate will reach an equilibrium state with some final distribution of charge over the plate. In order to find the steady-state potential at any point (x_1, x_2) in the plane, one needs to solve Poisson's equation:

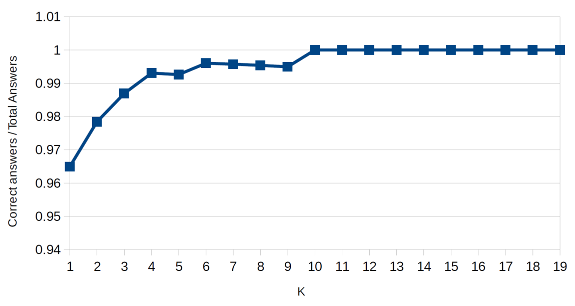
$$\nabla^2 V(x_1, x_2) = -\rho(x_1, x_2) \tag{1}$$

where $\nabla^2 = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2}$.

Poisson's equation is closely related to the heat equation, which is a second-order differential equation that describes the distribution of heat in a given region over time. The difference is that in Poisson's equation, the amount of "heat" (in this case, charge is an analogue for heat) is constant in time, where with the heat equation, the total amount of heat is often some decreasing function of time. Both Poisson's equation and the heat equation model situations in which some quantity (represented by a scalar field) becomes diffused over time. In this assignment, that scalar field is charge, which is initially distributed in the square wire on the conducting plate. In this assignment, we would like to model and visualize the steady-state solution to this differential equation.

When there are no boundary conditions, the solution of Poisson's equation can be found in a closed form. Steady state solutions to Poisson's equation with boundary conditions cannot be solved analytically, and simulations like this assignment are the best way to investigate them. In this case, the boundary conditions are the Dirichlet boundary conditions. This means that the potential (voltage) at the edge of the conducting plate is set to be 0 at all times. There are a number of methods which can be used to perform these simulations, but the one encouraged in this assignment is the Gauss-Seidel method. An example of a solution is shown in Figure 3.

The Gauss-Seidel method is an iterative method of solving this system of differential equations. We build a matrix representing the initial distribution of charge. Though the physical scenario is a continuous distribution of charge, we must discretize it so that we can compute on it. We can trade finer resolution for longer execution time. This is a balance that often has to be struck in the "real world".



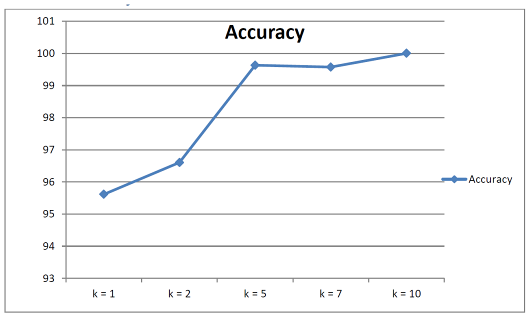


Fig. 1: Experimental results collected by two different students with the provided dataset.

Each element in the matrix represents the charge at one point in space. We distribute the charge initially according to the problem's setup. Because of the boundary conditions, we manually set the charge around the edge of the plate to be 0. After that, we iterate by building a new matrix based on the old one, with the value of each point in the new matrix being the average value of that point's four neighbors in the old matrix. We do this "double buffering" so that any iteration depends only on the iteration before it, and not itself. It can be shown that the simulation error is bound by how much the distribution changes from pass to pass. We can therefore choose a maximum acceptable error and simply keep iterating until the difference per pass is smaller than our specified error.

Each iteration can be computed in parallel, because the charge at each point depends only on what the charge at its neighbors was in the previous iteration, not on what it is in the current iteration. Because the solution involves building a matrix which represents the distribution of charge, the visualization is extremely easy. Students can simply use Python's MatPlotLib to create a density plot of the solution matrix and it should "just work", coloring in the more charge-dense areas brighter and the less charge-dense areas darker. They can even make an animation by just saving the state of the matrix at multiple time values.

Each copy of the assignment was run on a single worker node of Lewis & Clark's small HPC system. Each worker node has 48 CPU cores and 500 GB of main memory. To run the fully parallelized version on that system took between 30 seconds and a minute.

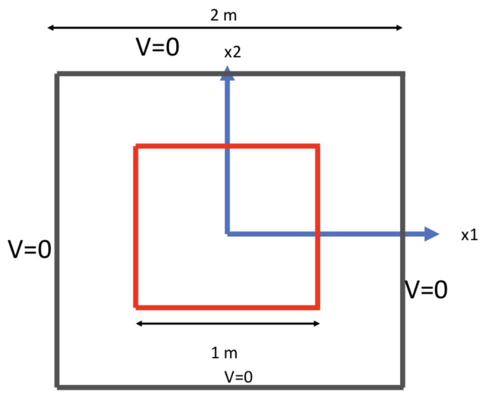


Fig. 2: Set-up. Origin is in the middle.

A. Concepts & Content

This assignment introduces the concept of physical simulations to students as an application of parallel and high-performance computing. It introduces numerical solutions to differential equations as well. The assignment encourages use of the Gauss-Seidel method of approximating a solution by building a matrix which represents charge at a discretized set of points on the conducting plate and iteratively solving the equation. The matrix has some internal dependence, requiring students to be clever about what they parallelize. Each point on the matrix is dependent on its four nearest neighbors, which means students need to think about how to divide up work before they can parallelize their code. It also forces students to think about the physical scenario in which their code is relevant in addition to teaching about concepts intrinsic to parallel programming. Concepts taught in this assignment include python multiprocessing, stencil operators, and MapReduce. It introduces the concept of needing to hit a specific error target and has realistic terminating conditions (the program keeps running until the simulation is accurate enough). The assignment also encourages students to think about the trade-offs which need to be made when carrying out HPC work, including the required level of accuracy. This is a common worry that many computational scientists face every day.

B. Educational Advantages & Disadvantages

The assignment has several advantages. First, many students are visual learners and this assignment produces images to help students develop an intuition about parallel programming. Second, whereas some parallel programming assignments are somewhat abstract, this assignment is particularly good at providing students with an understanding of how parallel programming is necessary for real-world applications. Third, it also does a good job of showing students the power of parallel programming. A serial implementation of the program, even with a very high error tolerance, will take many hours to

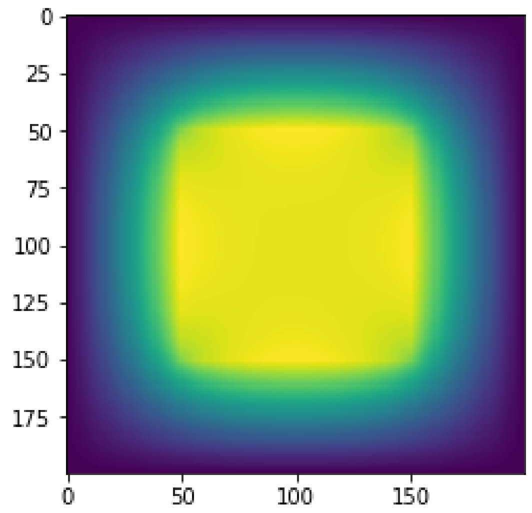


Fig. 3: Output of successful solution.

complete processing, but it parallelizes efficiently. The parallel version will run in a couple minutes.

A potential weakness of this assignment is that it requires some understanding of physics and math, which not all students in many CS courses may have. However, this example is relatively simple to understand, because it is about how charge distributes itself across a conducting material, which is likely something that most students will have some intuitive understanding of. Experience with partial differential equations is only required to understand the subtleties of the assignment.

C. Use in Courses

Two versions of this assignment have been used in two very different courses. The first course was a computational physics course for physics majors who had completed the introductory sequence and had studied partial differential equations. The second was a parallel computing course which required only the CS introductory sequence.

The physics version of the assignment focused more heavily on solving the math problem and setting up the physical situation. The parallelism component centered around using stencil operators to break up work. The computer science version of the assignment gave the expression of the mathematics involved to help set up, but required students to use MapReduce within each block of work generated by stencil operators. It focused more on parallelizing both spatially (solving at multiple points at once) as well as parallelizing the method of arriving at those solutions.

When this assignment was tested in the Parallel and High-Performance computing course for undergraduate students, at Lewis & Clark College in Spring of 2019, 17 students were asked for feedback. Feedback included "the assignment made sense", from one student. Another student added "It was nice to learn Python multiprocessing and use Jupyter notebooks". Some more responses included "This assignment demonstrates

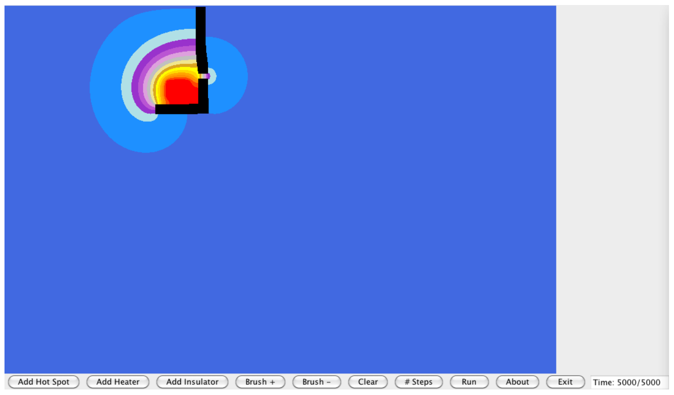


Fig. 4: Screenshot of heat diffusion program (from [7])

the power of parallelism very well", and "I like the example. It was more applicable to the real world". When asked "What are suggestions for improvement?", a student responded "It was interesting and useful. Just reduce the number of examples and move a bit more slowly".

D. Link to Full Assignment

Assignment files are available in a live Jupyter notebook [5], [6] format at jupyter.datasci.watzek.cloud. Please use the username 'edupar19' and the password 'reviewMe!' to log in when prompted.

IV. Graphical Heat Diffusion — Bunde

The application used in our third assignment is heat diffusion, which is similar to the second assignment, but the idea is packaged quite differently. Heat diffusion is presented as seeing how heat from a campfire distributes on a cold night. The region being simulated is represented by a 2D matrix, with each cell holding the temperature for a point. The simulation's initial conditions are set by the user with an interface similar to a drawing program, shown in Figure 4. The following three "colors" can be drawn:

- heater, which maintains a constant warm temperature (the fire in the campfire example),
- hot spot, which starts warm but does not generate heat to replace that which dissipates, and
- insulator, which neither absorbs nor generates heat.

Once the user has drawn these features as desired, they hit the "Run" button, which causes the simulation to advance by a fixed number of time steps, each of which updates the temperature of each cell with a weighted average of its temperature and the temperature of neighboring cells. Thus, although similar math occurs as in the electric potential problem, this version of the problem has more flexible initial conditions and inherently creates an animation, while not proceeding all the way to a steady-state solution or raising issues of accuracy.

The target audience for this assignment is CS 2 students. To make this possible, the students are given a working serial program and just asked to parallelize it. The given code is written in Java, the instruction language for our CS 2 course. In different iterations of the assignment in CS 2 and in higher-level courses, students have been asked to perform the parallelization using Java threads and/or a wrapper for Java's `Executor` framework. This wrapper, described with this assignment by Graf and Bunde [7], was designed to hide some details and simplify the framework for use by novices.

A. Evaluation

The assignment's graphical nature is a particular strength and one that extends beyond the fun of watching an animation rather than waiting for a program's elapsed time to print. A parallel version of the heat diffusion program runs noticeably faster than the original serial version. In addition, many of the bugs that students encounter cause visible artifacts in the animation. The most common bug has been having multiple threads update overlapping regions of the temperature matrix, which causes blooms of heat to appear rather than a smooth dissipation. Another issue is leaving a gap between the regions assigned to each thread, which causes an unchanging stripe to appear in the animation.

Other aspects of the assignment are more of a mixed bag. Being in Java allows for portable graphics and makes it possible to use the assignment early in the curriculum at institutions like ours which start in Java. At the same time, Java programs do not perform as well as those written in C, making the assignment somewhat awkward to justify while explaining the shift to parallelism on performance grounds. The assignment also works best at relatively low levels of parallelism since drawing the graphics is a serial task that will prevent the program from achieving high levels of speedup.

B. Variations

One extension of this assignment is to follow it up on exams with written (i.e. non-programming) questions about concepts related to parallelism and concurrency. These rely on student familiarity with the heat diffusion code. For example, we have asked students what is wrong with parallelizing the loop over time steps (assigning different time steps to each thread) rather than assigning each thread a disjoint region of the matrix. Another question was why it was bad to join each thread immediately after it was started.

References

[1] E. Ayguadé, L. Alvarez, F. Banchelli, M. Burtscher, A. Gonzalez-Escribano, J. Gutierrez, D. Joiner, D. Kaeli, F. Previlon, E. Rodriguez-Gutiez, and D. Bunde, "Peachy parallel assignments (EduHPC 2018)," in *Proc. Workshop on Education for High-Performance Computing (EduHPC)*, 2018.

[2] "Local binary patterns," https://en.wikipedia.org/wiki/Local_binary_patterns.

[3] "CS 426 - parallel computing course," http://www.cs.bilkent.edu.tr/~ozturk/cs426/.

[4] "Openmp architecture review board," https://www.openmp.org/about/about-us/.

[5] B. Glick and J. Mache, "Jupyter notebooks and user-friendly HPC access," in *Proc. 2018 IEEE/ACM Workshop on Education for High-Performance Computing (EduHPC)*, 2018.

[6] ——, "Using Jupyter notebooks to learn high-performance computing," *J. Comput. Sci. Coll.*, vol. 34, no. 1, pp. 180–188, Oct. 2018. [Online]. Available: http://dl.acm.org/citation.cfm?id=3280489.3280518

[7] M. Graf and D. Bunde, "Using wrappers to simplify task parallel programming," in *Proc. 21st Annual Consortium for Computing Science in Colleges Midwestern conference*, 2014, pp. 73–79.