Agent-based Simulation of Fire Extinguishing: An assignment for OpenMP, MPI, and CUDA/OpenCL

1st Arturo Gonzalez-Escribano
Dept. Informatica, Universidad de Valladolid
Valladolid, Spain
arturo@infor.uva.es

2nd Jorge Fernandez-Fabeiro
Dept. Informatica, Universidad de Valladolid
Valladolid, Spain
jorge@infor.uva.es

***Abstract*—We present a new assignment used in a Parallel Computing course to teach the approaches to the same problem in different parallel programming models. It targets concepts of shared-memory programming with OpenMP, distributed-memory programming with MPI, and/or GPU programming with CUDA or OpenCL. This assignment is based on a simplified agent-based simulation where teams of firefighters aim to extinguish a set of fire focal points in a dynamically evolving scenario. The program is designed to be simple, easy to understand by students, and to include specific parallelization and optimization opportunities. Although there is a quite direct parallel solution in the three programming models, the program has plenty of opportunities for further improvements. It extends the ideas of a previously presented assignment, in order to use more interesting data structures, load balancing techniques, and code optimizations. It has been successfully used in parallel programming contests during a real course, using the performance obtained by the students' code as a measure of success.**

I. Idea and context

Different programming models use different approaches for the parallelization of application structures. Understanding these differences is key for students to get into more advanced techniques, and to face parallel programming in current heterogeneous platforms. For several years, we have been teaching a course of Parallel Programming that introduces the basics of OpenMP, MPI, and CUDA or OpenCL. A peachy-assignment was previously presented [1], which was designed to be parallelized by the students using each one of these programming models during three one-week programming contests. The students compete to obtain the best performance on each contest. Although this kind of assignments can be perfectly used to teach a single programming model, they can also show which concepts and ideas can be reused across different models, and which can not, exposing the approach differences and the conceptual shift between them. For example, the students learn the differences between controlling race-conditions in shared-memory vs. using distributed data structures with explicit communications, or dealing with tiling and memory hierarchies in GPU coprocessors.

This new peachy-assignment maintains a clear focus on simple but effective code parallelizations and optimizations, while introducing more opportunities for the advanced students and more choices to teach a single programming model. This includes dealing with 2D data structures, a wider range of synchronization issues with different solutions in different models, and more interesting load-balancing and code optimization decisions.

Heat-propagation simulation is a classical example that can be used to teach concepts of parallel programming. For example, in [3] a peachy-assignment was presented that simulates the heat diffusion of a single campfire point, with the possibility of including isolation surfaces, and with an interesting online graphical representation that helps the students to visualize even the effect of typical parallelization bugs.

In this assignment we merge heat propagation with an agent-based simulation, where agents are fire-extinguishing teams. This combination is the key to allow the design of different load-balance situations. In our assignment, two-dimensional arrays store the heat values on a discretized representation of the simulation arena. Fire focal points arise at established places and simulation times, and their heat is propagated to neighbor cells step by step. Teams of firefighters are represented by agents that take simple decisions to advance, one cell at a time, in the direction of the next nearest focal point to extinguish it. While teams advance, they also reduce the heat of neighbor cells in a given radius. There are three different team classes with different rules to move and different heat-reduction radius. The size of the simulation surface grid, the number and start-frequency of focal points, and the number and class of the agents, are the main dials to design a new scenario with different load-balance properties.

As in the previous assignment, the provided material includes a sequential code, a test-bed of input files, and a handout explaining the assignment. The students can use common compilers and PC platforms to develop and test their codes. An automatic judge tool with an on-line public ranking is used to provide a fair arena, and to keep the students engaged during the contests. Other gamification tools are also included [2].

II. Concepts covered

The stages on each simulation step the program are: (1) Activate new focal points according to the simulation clock; (2) Propagate heat using a stencil operator implementing a Jacobi iterative solver for the Poisson's equation; (3) Reduction to obtain the maximal residual error; (4) Movement decisions for each team; (5) Execute team actions. To balance the speed of the team movements with the heat propagation, ten iterative

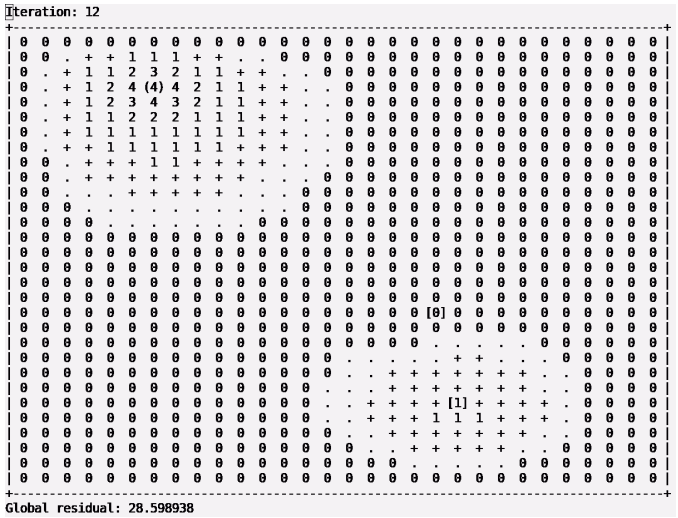

Fig. 1. Graphical representation of the output of fire-extinghuising simulation program at a given step. The numbers and symbols represent the heat at the different points of the 2D surface. Active focal points are represented with brackets. The position of firefighter teams are represented with square brackets.

propagation steps are computed sequentially. The simulation stops after a given amount of steps, or when the residual heat arrives at a stable point with a given threshold. The output of the program is the number of simulation steps and a list of final heat values at selected positions. In debug mode, the program also writes on each simulation step a text-mode graphical representation that can be used to visualize the simulation 1. It represents the heat on the surface, the active focal points, and the team positions.

The basic concepts covered in OpenMP are parallelization of loops, reductions, and avoiding race conditions with atomic operations. In MPI the students work with array partitions, halos and neighbor communications, and generic reductions. For GPU programming, the main ideas are embarrassing parallel kernels, minimizing host-device communication operations, thread-block sizes, non-trivial atomic operations, and simple reductions.

More advanced optimizations can be discovered and applied. Apart from those presented in [1], this new assignment introduces opportunities to test different partition policies for 2D arrays, precompute access patterns during the heat-reduction applied by the teams, use different load-balancing techniques in terms of input data features, using loop scheduling clauses in OpenMP, taking decisions about replicated vs. distributed computing in MPI, fusing kernels, new uses for the shared memory or non-trivial reductions on GPUs, etc.

III. Variants

The assignment can easily be adapted and modified by the teacher to include new simulation details, such as wind, obstacles, heat damage, etc. Instead of focusing only in parallelization and code optimization, the decision rules of the agents that simulate the firefighter teams can also be targeted to be optimized by the students. Better graphical and online interfaces can be devised to enrich the learning experience (see for example [3]).

References

[1] E. Ayguadé, L. Alvarez, F. Banchelli, M. Burtscher, A. Gonzalez-Escribano, J. Gutierrez, D.A. Joiner, D. Kaeli, F. Previlon, E. Rodriguez-Gutiez, and D.P. Bunde. Peachy Parallel Assignments (EduHPC 2018). In *IEEE/ACM Workshop on Education for High-Performance Computing (EduHPC 2018)*, Dallas (TX), USA, 2018. IEEE.

[2] J. Fresno, A. Ortega-Arranz, H. Ortega-Arranz, A. Gonzalez-Escribano, and D.R. Llanos. *Gamification-Based E-Learning Strategies for Computer Programming Education*, chapter 6. Applying Gamification in a Parallel Programming Course. IGI Global, 2017.

[3] O. Ozturk, B. Glick, J. Mache, and D.P. Bunde. Peachy Parallel Assignments (EduPar 2019). In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 342–346, Rio de Janeiro (Brasil), May 2019. IEEE.

Appendix: Reproducibility

The assignment has been used in the context of a Parallel Computing course, in the third year of the Computing Engineering grade at the University of Valladolid (Spain).

The material of the assignment, including a handout, the starting sequential code, and some input data sets to be used as examples will be made publicly available through the CDER courseware repository.

The on-line judge program used in the programming contests is named *Tablon*, and it was developed by the Trasgo research group at the University of Valladolid (https://trasgo.infor.uva.es/tablon/). The contest software uses the Slurm queue management software to interact with the machines in the cluster of our research group. During the course we used Slurm 18.08.3.

The machine of the cluster used for the OpenMP contest is named *heracles*. It is a server with four AMD Opteron 6376 @ 2.3Ghz CPUs, with a total of 64 cores, and 128 GB of RAM.

The machine used in the CUDA/OpenCL contests is named *hydra*. It is a server with two Intel Xeon E5-2609v3 @1.9 GHz CPUs, with 12 physical cores, and 64 GB of RAM. It is equipped with 4 NVIDIA's GPUs (CUDA 3.5), GTX Titan Black, 2880 cores @980 MHz, and 6 GB of RAM.

During the MPI contest we use *hydra* in combination with two other servers to create a heterogeneous cluster. The other two machines are: *thunderbird*, with an Intel i5-3330 @2.4 GHz CPU and 8 GB of RAM; and *phoenix*, with and Intel QCore @2.4 Ghz CPU with 6 GB of RAM.

All machines are managed by a CentOS 7 operating system. The compilers and system software used have been GCC v7.2, and CUDA v10.0.

The assignment provides the sequential code and the input files of the test-beds for the students. Other test-beds used by the on-line judge during the contest are also provided.

The results of the contests are publicly available until the start of the next semester at http://frontendv.infor.uva.es.