An Open edX Extension for Parallel Programming Assignments with Automatic Configurable Grading

https://doi.org/10.3991/ijep.v11i4.20097

Luis German Garcia, Emanuel Montoya, Sebastian Isaza,
Ricardo A. Velasquez (✉)
Universidad de Antioquia, Medellin, Colombia
randres.velasquez@udea.edu.co

Abstract—Computing devices of all types have almost converged to using central processing units featuring multiple processing cores. In order to develop efficient software for such devices, programmers need to learn how to write parallel programs. We present an infrastructure to support parallel programming assignments for online courses. We developed an extension to the Open edX platform with a backend that handles the execution of student codes on a cluster lab. The web user interface offers instructors a wide range of configuration options for the programming assignments as well as a flexible definition of criteria for automatic grading. We have successfully integrated the software with Open edX and tested it with a real parallel programming cluster lab.

Keywords—Parallel programming, automatic grading, Open edX

1 Introduction

The need for software engineers around the globe is ever increasing, with most countries being unable to meet the demands of the digital transformation that is rapidly happening in all areas of the economy [1][2]. While many educational institutions and governments are rushing to increase their capacity for training software developers, there's a need to include some form of parallel programming training in order to keep up with technology changes [3]. Today, most computers are equipped with multiple processing cores that are capable of running, not only multiple applications simultaneously, but also a single parallelized program faster. While parallel programming is not required or at least not worth the effort for many types of programs, a significant and critical amount of applications can greatly benefit from it. Unfortunately, learning parallel programming requires a significant effort from learners and instructors, and any tools that help tackle such a challenge are welcomed.

Massive open online courses (MOOCs) are playing an ever more important role in increasing the capacity of training software engineers. Platforms such as Coursera, edX and Udacity have millions of students, many of whom are following computer science courses. At the same time, universities and colleges are using learning management systems such as Moodle or Blackboard to make teaching resources more readily available to students and to virtualize to some extend their courses. In any case, programming courses require a significant amount of practice and instructors need ways of evaluating the student's level of accomplishment.

Code autograders are software that help automate the evaluation of a program, typically by comparing its outputs against previously computed correct outputs. By using such a tool, the instructor can easily monitor, even with large groups, which students are being able to provide a correct solution for a programming problem formulation and thus, reaching the learning goals. In the case of parallel programming assignments, instructors are also interested in evaluating the speedups achieved by the student's parallel codes compared to the sequential versions, or in measuring how performance scales with an increased number of cores, among others, which brings further challenges to an autograder for parallel code.

In this work we set to find out the benefits and drawbacks of implementing a flexible plugin for teaching parallel programing on Open edX, one the most popular LMS platforms. Thus, we present here a software that integrates and extends Open edX to specifically manage parallel programming assignments with automatic grading. From the user viewpoint, it provides one module for the instructor to configure an assignment and one module for the student to submit an assignment solution. We have carefully designed the assignment configuration module to let the instructor specify different kinds of tests, use various performance metrics and compute the grade according to the importance he/she wants to give to every test. Our autograder has the ability to run performance tests that allow the instructor to go further than only correctness, yet with the advantage of an automatic tool that allows him to manage larger groups or use the time for other course activities.

We have implemented the software, integrated it with an Open edX instance and linked it to a computing cluster lab to run the student assignments. The cluster lab has been built after a study that allowed us to select efficient single-board computers (SBCs) to be used as compute nodes, rather than using standard computers.

In Section 2, we present other works with a similar aim and in Section 3 we tell the steps we followed to develop and test the platform. In Section 4 we give an overview of the platform and its details are in Section 5 and 6. The prototype computing cluster lab is presented in Section 7. Finally, Section 8 and 9 discuss the platform developed and draw some conclusions.

2 Related Works

Before the dominance of today's MOOC platforms, there were early attempts at building virtual platforms for teaching programing [4], some of them specifically for parallel programming [5]. Such works represented a bigger development effort as they had to build the whole platform from scratch. The availability of MOOCs in parallel programming is relatively new. By 2017, only four courses in the subject where available [6]. Professor Wen-Mei Hwu of the University of Illinois Urbana-Champaign offered the first online parallel programming course in 2012 on the Coursera platform [7]. The course focused on teaching the essential parallel programming concepts for programming multi-core CPUs and GPUs using OpenCL or CUDA. A CUDA course released in 2013 at the Udacity platform was very similar but focused specifically on teaching how to program with CUDA without much discussion of the fundamentals of parallelism [8]. The specialization "Clouds, Distributed Systems, and Networking" on the Coursera platform was a suite of courses focused on cloud computing [9]. Another course titled "Parallel programming" also at the Coursera platform teaches shared-memory parallelism in the Scala programming language [10]. A common characteristic of the first online courses that appeared was to be highly focused on the language rather than teaching the concepts of parallel algorithms.

Recently, new courses have been released, aimed at achieving a broader learning of parallel programming and high-performance computing (HPC). Sarkar et al. [11] described the process and challenges faced while they constructed a Coursera Specialization on Parallel, Concurrent, and Distributed Computing in Java. Besides the challenges associated with curriculum development, authors highlight technical challenges such as mini-project development, a custom parallel programming library for supporting the course, and automatic grading of student submissions. In [6], Sarkar et al. reported that, by far, the most demanding task for maintaining an online course was answering questions in the forum. Additionally, the authors reported that the performance tests are often the primary cause for a bad learner experience. Authors recommended to account for the volatility of results and keep speedup expectations low. The course uses the auto-grading platform provided by Coursera, where student submissions are executed within containers and limited to four cores.

Mullen et al. [12] tackled the need for training and education in HPC. They reported an experience in building a course with the Open edX platform but they found it challenging for supporting hands-on HPC experience.

Solutions for integrating MOOC platforms with hands-on training have been reported [13][14]. Dakkak et al. [13] proposed an online GPU development platform that provides students with a user-friendly scalable GPU computing platform. In this work, authors presented the original, revised, and upcoming WebGPU designs that address the requirements and challenges of offering sophisticated computing resources to many students. They developed an autograder for GPU codes using Open edX. The work presented here has a similar intention, but it is not for GPU programming. Instead, we aim at multicore CPU parallel programming using models such as OpenMP and Pthreads.

Staubitz et al. [14] introduced CodeOcean, a web platform to provide practical programming exercises for MOOCs. CodeOcean provides online courses with automated feedback and assessment of programming tasks. A web application provides the development environment for learners as well as an administration backend for instructors. Docker is used for code execution and assessment.

Carbunescu et.al. presented in [15] a comprehensive analysis of the different issues that arise when trying to implement an autograder for parallel codes. They warned about the challenges posed by, for example, the fact that many parallel programs may produce non identical outputs even if executed on the same hardware and software environment. In the same paper they also present an autograder developed as an extension of Moodle and their experience of using it for a bachelor course. They claimed that autograders often require customizations to meet the particularities of every course, particularly for the case of parallel programming.

While numerous authors [16][17][18] have built and studied parallel computing clusters based on SBCs, our paper presents for the first time (to the best of our knowledge), one such state of the art cluster connected to an online autograder for teaching parallel programming. Here we show which tools are required as well as their benefits and drawbacks.

This paper is a continuation of our preliminary results published in [19], where the platform was presented with less detail and the focus was on the study performed to determine the best SBC to build the cluster. In this work, we present the whole platform in detail, discussing the strengths and limitations of the most relevant components.

3 Methodology

To develop the infrastructure for an online learning platform, that allowed us to find out the benefits and drawbacks of implementing a flexible plugin for teaching parallel programing, we went through a three-stage process:

1. We have defined all use cases to provide web services to support configurable parallel programming assignments on the remote cluster lab. The use cases were defined in a few working sessions among students and professors who teach parallel programming. The project team devised then the platform architecture as presented in Section 4 using the web services paradigm to allow for a clear separation and concurrent development of the frontend (Web Interface) and the backend (Lab Controller). We selected Open edX as the learning platform based on the positive experiences we had as students in the past with edX. We set out to study the technical options for developing an extension and decided to use Xblocks [20] as well as the Javascript language, Node.js [21] and Bootstrap [22][22] to achieve the more complex functionality of the Assignment Configuration module and the Lab Controller.
2. We built a cluster lab with SBC nodes in order to incorporate that technology trend as part of the parallel programming learning experience. A specialized software infrastructure was set up in order to provide a queuing system for orderly running student assignments. This part of the platform is managed by the Lab Controller.
3. Through multiple iterations in which typical parallel programming assignments were used, the frontend and the backend were refined until we were ready to integrate them. During the integration, common bugs were corrected and, in the end, we were able to validate the platform through careful functional testing. Once the platform was ready, a few controlled tests were carried out with teachers and students.

4 Platform Overview

The overall goal of the system presented here is to offer an online platform for teaching parallel programming. We developed two modules for Open edX to allow for the configuration, submission and automatic grading of parallel programming assignments. To achieve this, the platform is composed of three main blocks as shown in Fig 1The Web Interface, the Lab Controller and the Cluster Lab.

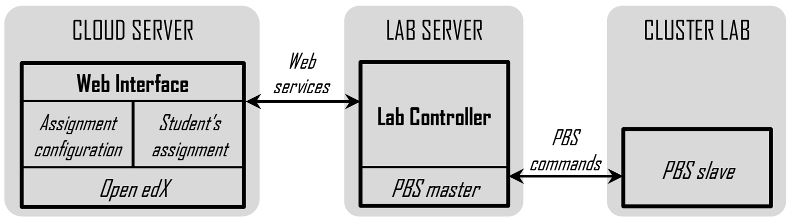

Fig. 1. General platform architecture.

The Web Interface is the platform's frontend and is, in general, the one provided by the Open edX platform with the configurations that the course creator chooses. All the resources and the organization of a parallel programming course are not discussed in this paper. Instead, we will explain one specific but very useful resource (see Section 5 and 6) we developed for any such a course.

Our contribution lies in the design and implementation from scratch of the two modules necessary for having configurable parallel programming assignments with automatic grading within Open edX. These two modules are: the one for the instructor to configure the assignment (Assignment Configuration inFig 1) and the one for the student to submit an assignment (Student's Assignment inFig 1). In our testing setup, the Open edX instance as well as the extension modules we developed, run on a cloud server but they could naturally run on any server provided the capacity is appropriate for the amount of users.

The Lab Controller is the platform's backend, receiving web service requests from the Web Interface, for example to run an assignment sent by a student, and issues the proper commands to the job scheduler, the Portable Batch System (PBS) [23] in our implementation. Thus, the PBS master receives execution requests that are then issued to the job scheduling agents (PBS slave) on the corresponding nodes to execute the code. Parameters, input and output data are handled by the Lab Controller as well as execution results to compute the grade. The Lab Controller should ideally run on an independent machine on the same network as the nodes where the student codes are run. However, it could also share the same machine with the assignment jobs as long as the hardware capacity and user traffic permits.

The so-called Cluster Lab is the infrastructure where the parallel programming assignments run. Although we have built a cluster based on SBC nodes (see Section 7), the other platform components (Web Interface and Lab Controller) are not restricted to working with such hardware and they only require one or more machines with PBS software installed to act as job scheduler.

5 Web Interface

Open edX is the learning management system (LMS) software used by edX, one of the largest e-learning providers today with millions of students around the world. The Open edX software is open source and is available on GitHub. We have installed Ironwood codenamed version (2019) of Open edX on a Linux server and developed an extension that provides two modules: one for the configuration of assignments on the instructor's view and one for the submission of assignments on the student's view. The Open edX installation plus the two modules make up the frontend from the point of view of the parallel programming assignment application.

5.1 Assignment configuration

When an instructor wants to set up a parallel programming assignment, he/she must access Open edX through the Content Management System (CMS) and add an instance of the parallel programming assignment Xblock. It will then offer the instructor's view with a wizard to configure an assignment in four steps. Although so far, our system only supports C/C++ as programming languages for the assignments, it could be easily extended to other languages. We hope to do so in the coming versions.

Step 1: Templates and input/output: The first step allows the instructor to specify the source and header files as well as a *makefile* to configure any compilation or linking parameters, if needed. At least one source file must be marked as editable as only editable files can be modified by the student. Such files can be blank or have segments of code for the student to complete. The Student Assignment module will not allow to modify the *makefile* so that it is not manipulated to influence performance metrics.

In the first step, the instructor should also specify the inputs and outputs of the program as shown in the example of Table 1. Inputs and outputs can be *variable* arguments or *files*, and can be marked as *execution parameters* or not (see Step 2). Outputs can be also marked as *performance* type, which allows them to be used for grading (see Step 3).

Table 1. Input / output configuration example.

Inputs		
Name	***Type***	***Execution parameter***
input_1	Variable	False
input_2	Variable	False
input_3	File	False
param_1	Variable	True
param_2	Variable	True
Outputs		
Name	***Type***	***Performance metric***
out_1	File	False
k_time	Variable	True

Finally, the first step asks the instructor to specify the following limits for any execution resulting from a student's submission of the assignment being configured:

- Core count: Maximum number of cores that may be used.
- Memory: Maximum system memory that may be consumed.
- Wall time: Maximum elapsed execution time allowed.

These limits need to be set, on the one hand, for a test execution requested by a student before the actual submission and are meant to prevent the Cluster Lab to easily saturate by buggy codes. On the other hand, the limits should also be configured for all the executions defined by the instructor and are applied only when the student submits his / her assignment for evaluation and grading. They are meant to block attempts at cheating performance metrics.

Step 2: Test vectors and execution parameters: Next, test vectors for the evaluation of student's assignments are configured. Each test vector is a set of values and files to match the program inputs defined in Step 1. An execution is a set of values for all the execution parameters in Step 1. The idea is to allow for example a scalability evaluation of the program, having a fixed test vector but multiple executions with varying number of threads. Each test vector can have one or more executions and at least one test vector and one execution must be set up for an assignment. Table 2 shows an example with two test vectors of three inputs. The first test vector has only one execution associated whereas test vector 2 has three executions. Notice that all test vectors must have the same amount of inputs and all executions must have the same number of parameters. Both must match the ones defined in Step 1. Notice the structure of Table 2 matches Table 1.

Table 2. Example structure of test vectors and execution parameters.

Test vector 1					
Input1: value		***Input 2: value***		***Input 3: file***	
Execution 1					
Parameter 1: value			Parameter 2: value		
Test vector 2					
Input1: value		***Input 2: value***		***Input 3: file***	
Execution 1		Execution 2		Execution 3	
Parameter 1: value	Parameter 2: value	Parameter 1: value	Parameter 2: value	Parameter 1: value	Parameter 2: value

Step 3: Grading: In this step a weight is given as a percentage to every execution and to every test vector. Then, the instructor should specify a function to compute the grade of the assignment, using the performance results of the student and the reference as variables, as shown in Table 3's example. Notice all percentages of test vectors must total 100%, as well as all executions of a test vector.

Every execution may have a cost function to compute a grade that will be weighted with the other grades of other executions and tests in order to produce the assignment grade. The cost function may be built based on the performance outputs defined on Step 1 and the standard performance metrics the platform always measures: *wall time*, *memory used* and *cpu utilization*. The cost function may compare the student's performance values obtained with those of the instructor's solution. The expression to represent the cost function must use Javascript syntax.

For every execution, the platform will verify its correctness by comparing its outputs to the ones produced by the reference (instructor's) solution. If they match, the cost function will determine the grade, if they do not, that execution will get a zero grade.

Table 3. Example of grading weights and cost functions.

Test vector 1			40%
Execution 1			
100%			

if (usr_time < ref_time){ 5 }
else if (usr_mem < ref_mem){ 4 }
else { 3 }			

Test vector 2			60%
Execution 1	***Execution 2***	***Execution 3***	
10%	20%	70%	

5	if(k_time<20)
	{5}
	else{3}
		if(k_time<30)
		{5}
		else{3}	

For the example of Table 3, assuming that only Execution 2 of Test vector 2 for a student's submission produced incorrect outputs, and that the values of the performance metrics were:

ref_time: 150
ref_mem: 7
usr_time: 160
usr_mem: 6
k_time: 15 (Test vector 2, Execution 2)
k_time: 42 (Test vector 2, Execution 3)

Then, the assignment grade would be computed by our platform as:

$$grade = (40*(4)+60*(10*5+20*0+70*3)/100)/100 \quad = 3.16$$

This particular evaluation configuration might seem a little harsh, but it is obviously only an example. By adding more executions such as Execution 1 of Test vector 2, in which only by submitting a code that produces correct outputs the student can get a 5.0, the final assignment grade will likely be higher in a real scenario. The flexibility of the grading configuration allows the instructor to carefully design the assignment evaluation by tuning not only the weights of every execution and test vector but also the cost function of every execution.

Step 4: Reference solution: Finally, the instructor must submit his/her solution code of the assignment. This code is run once for every execution that has been configured and the performance metrics are stored to be evaluated against the student's as specified in the cost function defined in Step 3.

After the reference solution is submitted, all tests will be applied so the process will naturally take some time. Once they finish, the status of the assignment configuration will switch to finished and only then it can be enabled for the students to start working on it.

5.2 Student's assignment

The student interface is meant to let him/her submit an assignment solution and it is composed of the following parts:

- Templates: To let the student download a set of source files, some of which are meant for him / her to complete in order to obtain a program that satisfies the problem defined in the assignment.
- Student solution: Let the student download his/her latest submitted code version of the assignment solution.
- Code editor: A large text field with syntax highlighting for the student to write the code and see the code of any of the assignment source files.
- Compiling: To compile the student's code and see the compiler's output. When the student issues a compiling action, the code is sent to the Lab Controller where it is compiled in the same tools' environment as the cluster nodes where the code is meant to run.
- Test: To let the student set input values for the program and execute the binary produced in the latest successful compilation. When the student issues a test, the inputs are sent to the Lab Controller and it executes the latest compiled binary of that student on the Cluster Lab. The program's outputs as well as the performance metrics are sent back to the frontend and shown to the student. These test executions have resource limitations as defined by the instructor and are handled in a low priority queue by PBS master, possibly sharing cluster nodes with other executions.
- Submit: When the student is satisfied with the test results, he/she can choose to submit his/her assignment solution. Then, the Lab Controller will run it with all test vectors and execution parameters defined by the instructor and unknown to the student. It may naturally take a while and in the end the student will be able to see his/her grade. Submit executions have different resource limitations meant to prevent cheating performance metrics and are sent to a high priority queue where they enjoy an isolated execution in order to minimize performance metrics noise. The assignment solution can be updated as long as the deadline has not passed.

6 Lab Controller

The Lab Controller is a custom software we have developed as a collection of web services to attend assignment compilation and execution requests from the online learning platform. It is a backend software developed using Node.js [21], the PBSJS [24] module to interact with the PBS master and a MongoDB database to store assignment data (IDs, inputs, outputs, etc). The job scheduling software requires the Lab Controller to run on a machine located on the same network as the cluster nodes.

6.1 Assignment submission use case

Fig 2 shows the basic flow of steps that are followed when a student develops an assignment through the application presented in Section 5.2. After writing the code, the student needs to compile it. If successful, he/she can proceed to run the generated binary (with the inputs of his/her choice). If the student is satisfied with the results, he/she can submit the assignment for evaluation. In this case, the Lab Controller will run the student's binary multiple times using the various test vectors and execution parameters provided by the instructor when he/she configured the assignment. The Lab Controller will also compare the student's outputs with the reference outputs from the instructor and report back a grade based on passed and non-passed tests, and on the cost functions defined.

When the Lab Controller needs to launch an assignment execution, it does so through PBS (see Section 6.2) in order to guarantee the code will run exclusively on the allocated cluster nodes so that performance metrics are not distorted by competing executions.

For every assignment that has been configured by the instructor, the Lab Controller keeps an organization of files as follows:

Instructor: Single folder containing the reference solution

- src: reference source codes
- build: reference object files
- bin: reference binary
- test vectors: input vectors for assignment evaluation
- outputs: reference outputs

Templates: Single folder containing the source code starting state

- src: source code files (to be copied to the student's folder)
- build: possible precompiled libraries
- makefile: default compilation script file

Student: One folder per student enrolled containing his/her assignment solution

- src: student source codes
- build: student object files
- bin: student binary
- debug vectors: input vectors for assignment execution
- outputs: student outputs

Notice that the above lists refer only to the assignment information that is stored as files. Naturally, other assignment data such as user IDs, performance metrics, grades, etc, are stored in a MongoDB database.

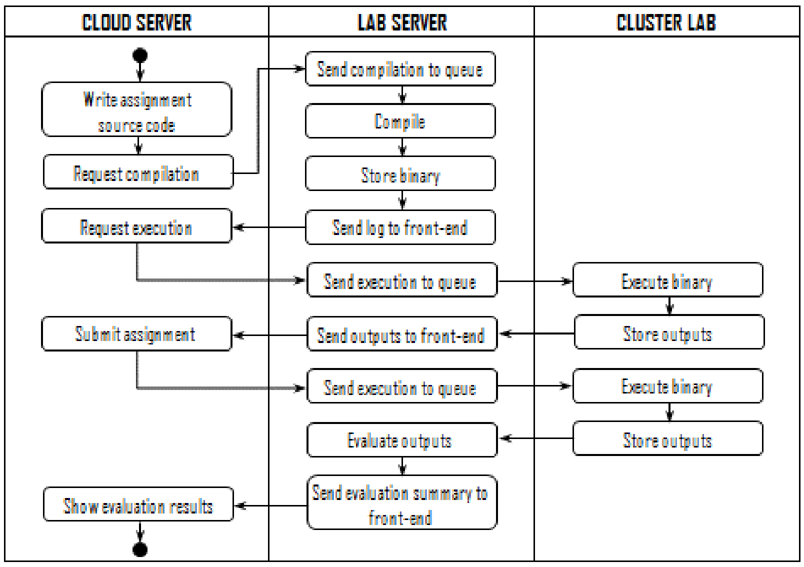

Fig. 2. Platform components interaction for a student assignment submission.

6.2 Job scheduler

The Portable Batch System (PBS) [23] is a software for managing job scheduling in HPC environments. It was created 20 years ago for NASA, and it has evolved into a sophisticated tool with ample documentation and support. Today, PBS has joined the OpenHPC [25] initiative for an open-source HPC software stack with the Linux Foundation. In this project, we are using PBS Professional Open Source Project version 19.0.0.

PBS has four main components: Server, Scheduler, Communication and MoM. In a cluster platform with a server acting as a PBS frontend and multiple execution nodes, like the one we are using in this work, the Server, Scheduler, and Communication components are typically installed on the PBS frontend (PBS master). In contrast, the MoM component or Job Executor is installed at each execution node (PBS slave). The Server is in charge of receiving job execution requests and, along with the Scheduler, sends the jobs to the corresponding execution nodes through the Communication component. Dedicated routing and execution queues are defined to serve as communication channels to dispatch jobs to the corresponding nodes. At each execution node, the MoM component is in charge of placing the job into execution.

We have used PBS to achieve the following:

- Queues to manage different priorities (e.g. instructors' executions have higher priority)
- Isolated execution of assignment source codes in order to evaluate performance.
- Limit the resources that assignment executions can use such as walltime, memory and number of cpus.
- Monitor the resources consumed by assignment executions.

7 Cluster Lab

In the current prototype, our online learning platform is connected through the Lab Controller to a cluster of SBCs. We have chosen such devices, instead of traditional desktop- or server-like clusters, given their advantages in terms of low cost, low power consumption and recent utilization for edge computing. In fact, we acquired seven different cards (Vim3Pro, Orange Pi4B, Odroid N2, Hikey 970, Jetson Nano, Raspberry Pi4B, and Latte Panda) and evaluated them in terms of processing latency (by running standard benchmarks), price, and software compatibility [19]. Such devices are based on ARM processors typically found on embedded electronic devices and cellphones. In the study we also compared the price and computing power of SBCs against high-performance workstation processors and showed that SBC cores were on average six times cheaper while only two times slower than Intel Xeon cores. In the end, we selected the Vim3Pro and the Orange Pi4B as the best overall SBCs of the ones we studied and built a cluster with 20 and 32 such cards respectively.

On every SBC node of the Cluster Lab runs the PBS slave. We installed only the MoM component on it, as a daemon that listens to commands from the PBS master to run the assignment jobs isolatedly. The MoM reports back to the master when a job is completed so that the Lab Controller can collect the results from the network storage and send them back to the platform frontend. Finally, the MoM is able to kill any execution job that has exceeded the resources granted to the process.

Although we have taken the time to setup this exotic computing cluster, all the software developed will run smoothly on a standard cluster setup based on commodity computers.

8 Discussion

This paper presents the first version of our platform and as such, it is undeniably not mature yet. However, the first set of tests done by professors that teach parallel programming and a few students, allows us to analyze some of the advantages they found.

8.1 LMS base platform

Most of the major LMS platforms such as Udemy, Coursera, Blackboard or Udacity, among others, provide some form of interface for developing extensions. However, they are paid platforms which limit their adoption for educational institutions in developing countries, like ours. On the other hand, there are open source LMSs such as Open edX, Google Open Online Education, Open HPI, Open OLAT, Forma.lms, and notably Moodle, with wide adoption across the world. At our institution, the platform of choice is Moodle but we chose Open edX because of its simpler user interface and wide adoption by large institutions and reputed enterprises. With this project we have got to know Open edX much better from the technical point of view and are now able to make more informed decisions when picking an LMS for a specific need.

8.2 Assignment templates

One feature that has been greatly appreciated is the possibility for the instructor to submit templates for the student assignments. Instructors have the option of submitting incomplete source files for the students to edit and submit entire source files that can be made non-editable by students while still part of the solution. By doing so, students can focus on developing the key parts of the parallel algorithms that the instructor wants to evaluate, without spending too much time on program accessories that are not within the course scope. It is often the case that assignments are either too simple when trying to consider reasonable developing efforts or too long when trying to evaluate problems with appropriate complexity.

This templates' feature also allows the instructor to insert instrumentation code in order to obtain any specific performance metrics for a more in-depth analysis of the student's submissions.

8.3 Assignment evaluation

Beyond the basic metrics of correctness and execution time, parallel codes require more metrics: memory increase, strong scaling, weak scaling, kernel times, among others. An important part of the learning process is to be able to observe the impact of different algorithmic and language decisions on the various performance dimensions. Our platform not only computes a grade but also allows the instructor and the student to get an insight into the challenging task of performance tuning. The feedback obtained through these more complete metrics allow the student to better comprehend what he/she is doing and learning.

Naturally, a more elaborated evaluation requires the instructor to spend more time in configuring the assignment. Furthermore, the more metrics he/she wants to evaluate, the more executions he/she needs to configure and the longer it will take for student submissions to execute on the cluster. Nonetheless, it should be noticed that our platform saves the instructor the time he/she would need to spend in creating scripts to perform the tests or even worse, to do them manually.

8.4 Management of executions

Job schedulers are mature tools from the HPC community that efficiently manage shared computing clusters. They use priority settings in order to assign resources according the predefined policies. In our case, PBS allows the platform to give priority to student submissions for which performance metrics are critical over student self-tests that are mostly used for verifying correctness. It can also handle specific resource boundaries for each execution to prevent cheating and enable exclusive executions to reduce noise in time measurements. Finally, the job scheduler acts as an abstraction layer between the platform backend (Lab Controller) and the cluster hardware, so no changes in the backend are needed when, for instance, the number of nodes in the cluster changes.

9 Conclusion

We have developed an Open edx extension for parallel programming assignments. In this paper, we presented its features and the way it is used. Moreover, we argued how its flexibility has a positive impact on the parallel programming learning process.

We found that Open edx has good documentation and community support, and provides an easy-to-use interface for students. However, the installation process was rather annoying, requiring more hacks than we expected. When adding an Xblock one needs to recompile the whole Open edx which is also annoying.

The Xblocks API turned out to be a good way of developing the student view but too limited for the assignment configuration module that required more support from the frontend to make the interface friendly. We eventually developed it with Javascript tools and embedded it easily into Open edx. When compared to Moodle, for example, Open edx has far less configuration options and resources, but perhaps because of this, it presents the student a very easy and linear interface to navigate.

Although we have performed initial tests of the platform with a limited set of instructors and students, a larger scale pilot is required to obtain a more realistic assessment.

10 References

[1] S. Fayer, A. Lacey, and A. Watson, “STEM occupations: Past, present, and future,” *Spotlight on Statistics*, vol. 1, pp. 1-35, 2017.
[2] D. Deming and K. Noray, “STEM careers and technological change,” National Bureau of Economic Research, 2018.
[3] The Joint Task Force on Computing Curricula, Curriculum Guidelines for Undergraduate Degree Programs in Computer Science, IEEE/ACM, 2013. https://doi.org/10.1145/2534860
[4] J. Prieto-Blazquez, J. Herrera-Joancomarti, A.E. Guerrero-Roldán, “A Virtual Laboratory Structure for Developing Programming Labs,” *International Journal of Emerging Technologies in Learning*, vol. 4, 2009. https://doi.org/10.3991/ijet.v4s1.789

[5] M. Gusev, S. Ristov, G. Velkoski, B. Ivanovska, "E-learning and Benchmarking Platform for Parallel and Distributed Computing," *International Journal of Emerging Technologies in Learning*, vol. 9, no. 2, 2014. https://doi.org/10.3991/ijet.v9i2.3215
[6] V. Sarkar, M. Grossman, Z. Budimlic, and S. Imam, "A one year retrospective on a mooc in parallel, concurrent, and distributed programming in java," in 2018 IEEE/ACM Workshop on Education for High-Performance Computing (EduHPC), 2018, pp. 61–68. https://doi.org/10.1109/eduhpc.2018.00010
[7] H. Wen-mei. (2012) Heterogeneous Parallel Programming. http://academictorrents.com/details/8903d0871c652b96c7b29db738cea76902d65888
[8] D. Luebke, J. Owens, M. Roberts, and C.H. Lee, (2013) Intro to Parallel Programming. https://www.udacity.com/course/intro-to-parallel-programming--cs344 .
[9] R. Farivar, A. Singla, I. Gupta, P.B. Godfrey, R.H. Campbell, Clouds, Distributed Systems, and Networking. https://www.coursera.org/specializations/cloud-computing .
[10] V. Kuncak, A. Prokopec, Parallel Programming. https://www.coursera.org/learn/parprog1 .
[11] V. Sarkar, M. Grossman, Z. Budimlic, and S. Imam, "Preparing an online java parallel computing course," in 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). IEEE, 2017, pp. 360–366. https://doi.org/10.1109/ipdpsw.2017.162
[12] J. Mullen, W. Filinger, L. Milechin, and D. Henty, "The impact of mooc methodology on the scalability, accessibility and development of hpc education and training," *Journal of Computational Science*, vol. 10, no. 1, 2019. https://doi.org/10.22369/issn.2153-4136/10/1/11
[13] A. Dakkak, C. Pearson, and W.-m. Hwu, "Webgpu: A scalable online development platform for gpu programming courses," in 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2016, pp. 942–949. https://doi.org/10.1109/ipdpsw.2016.63
[14] T. Staubitz, H. Klement, R. Teusner, J. Renz, and C. Meinel, "Codeocean-a versatile platform for practical programming exercises in online environments," in 2016 IEEE Global Engineering Education Conference (EDUCON), 2016, pp. 314–323. https://doi.org/10.1109/educon.2016.7474573
[15] Carbunescu, R., Devarakonda, A., Demmel, J., Gordon, S., Alameda, J., & Mehringer, S. (2014). Architecting an autograder for parallel code. Proc. of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment - XSEDE '14. https://doi.org/10.1145/2616498.2616571
[16] P. J. Basford, S. J. Johnston, C. S. Perkins, T. Garnock-Jones, F. P. Tso, D. Pezaros, R. D. Mullins, E. Yoneki, J. Singer, and S. J. Cox, "Performance analysis of single board computer clusters," *Future Generation Computer Systems*, vol. 102, pp. 278–291, 2020. https://doi.org/10.1016/j.future.2019.07.040
[17] S. Cox, J. Cox, R. Boardman, S. Johnston, M. Scott, and N. O'brien, "Iridis-pi: a low-cost, compact demonstration cluster," *Cluster Computing*, vol. 17, no. 2, pp. 349–358, 2014. https://doi.org/10.1007/s10586-013-0282-7
[18] L. Alvarez, E. Ayguade, and F. Mantovani, "Teaching hpc systems and parallel programming with small-scale clusters," in 2018 IEEE/ACM Workshop on Education for High-Performance Computing (EduHPC). IEEE, 2018, pp. 1–10. https://doi.org/10.1109/eduhpc.2018.00004
[19] R. A. Velasquez, S. Isaza, E. Montoya, L. G. Garcia, "Embedded cluster platform for a remote parallel programming lab," IEEE Global Engineering Education Conference (EDUCON), IEEE, 2020, pp. 763-772. https://doi.org/10.1109/educon45650.2020.9125270
[20] Open edx. Edx xblock. Accessed: 2020-11-25. https://openedx.atlassian.net/wiki/spaces/PLAT/pages/33358554/XBlocks .
[21] OpenJS Foundation. Node.js. https://nodejs.org .Accessed: 2020-11-25.

[22] Bootstrap team. Bootstrap. https://getbootstrap.com /. Accessed: 2020-11-25
[23] Altair Engineering. (2019) PBS Professional open-source project. https://www.pbspro.org . Accessed: 2019-12-20.
[24] Quantum HPC. (2019) PBSJS. https://github.com/quantumhpc/pbsjs . Accessed: 2020-11-25 .
[25] The Linux Foundation. (2019) OpenHPC Collaborative Project. https://openhpc.community . Accessed: 2020-11-25.

11 Authors

We thank the financial support from the Colombia Scientific Program within the framework of the call Ecosistema Científico (Contract No. FP44842-218-2018).

Luis Germán García is an assistant professor of the Electronics and Telecommunications Department at the Faculty of Engineering, University of Antioquia, Colombia. He obtained the Electronics Engineering degree and the PhD. in Electronics and Computing Engineering from the same university and the MSc. in Embedded Systems Design from the University of Lugano (CH). His research interests are embedded systems and computer architecture. Luis Germán assembled the cluster prototype, run the experiments and helped write the paper. german.garcia@udea.edu.co

Emanuel Montoya is freelance software developer and adjunct professor at Universidad de Antioquia. He obtained the Electronics Engineering degree and the MSc. in Telecommunications from the same university. His professional interests revolve around web application development and software defined networks. Emanuel developed the Lab Controller in this project. emanuel.montoya@udea.edu.co

Sebastián Isaza is an associate professor of the Electronics and Telecommunications Department at the Faculty of Engineering, University of Antioquia, Colombia. He obtained the Electronics Engineering degree from the same university, the MSc. in Embedded Systems Design from the University of Lugano (CH) and the PhD. in Computer Engineering from Delft University of Technology (NL). His research interests revolve around high-performance computing. Sebastián co-lead the project development and wrote the paper. sebastian.isaza@udea.edu.co

Ricardo A. Velásquez is an associate professor of the Electronics and Telecommunications Department at the Faculty of Engineering, University of Antioquia, Colombia. He obtained the Electronics Engineering degree from the same university, the MSc. in Embedded Systems Design from the University of Lugano (CH), and the PhD. in Informatics from the University of Rennes (FR). His research interests are embedded systems, computer architecture, machine learning, neuromorphic computing and robotics. Ricardo conceived the idea of this project, secured its funding, co-lead its development and helped write the paper.

This article is a revised version of a paper presented at the EDUCON2021 conference held virtually in Vienna, Austria, 21-23 April 2021. Article submitted 2020-11-27. Resubmitted 2021-04-22. Final acceptance 2021-04-23. Final version published as submitted by the authors.