

Gamification–Based E–Learning Strategies for Computer Programming Education

Ricardo Alexandre Peixoto de Queirós
Polytechnic Institute of Porto, Portugal

Mário Teixeira Pinto
Polytechnic Institute of Porto, Portugal

A volume in the Advances in Game–Based
Learning (AGBL) Book Series



www.igi-global.com

Published in the United States of America by

IGI Global
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA, USA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: <http://www.igi-global.com>

Copyright © 2017 by IGI Global. All rights reserved. No part of this publication may be reproduced, stored or distributed in any form or by any means, electronic or mechanical, including photocopying, without written permission from the publisher. Product or company names used in this set are for identification purposes only. Inclusion of the names of the products or companies does not indicate a claim of ownership by IGI Global of the trademark or registered trademark.

Library of Congress Cataloging-in-Publication Data

Names: Queiros, Ricardo Alexandre Peixoto de, 1975- | Pinto, Mario Teixeira, 1967- author.

Title: Gamification-based e-learning strategies for computer programming education / Ricardo Alexandre Peixoto de Queiros and Mario Teixeira Pinto, editors.

Description: Hershey PA : Information Science Reference (an imprint of IGI Global), 2016. | Series: Advances in game-based learning | Includes bibliographical references and index.

Identifiers: LCCN 2016032941 | ISBN 9781522510345 (hardcover) | ISBN 9781522510352 (ebook)

Subjects: LCSH: Simulation games in education. | Computer programming--Study and teaching. | Computer games.

Classification: LCC LB1029.S53 Q45 2016 | DDC 371.397--dc23 LC record available at <https://lcn.loc.gov/2016032941>

This book is published in the IGI Global book series Advances in Game-Based Learning (AGBL) (ISSN: 2327-1825; eISSN: 2327-1833)

British Cataloguing in Publication Data

A Cataloguing in Publication record for this book is available from the British Library.

All work contributed to this book is new, previously-unpublished material. The views expressed in this book are those of the authors, but not necessarily of the publisher.

For electronic access to this publication, please contact: eresources@igi-global.com.

Chapter 6

Applying Gamification in a Parallel Programming Course

Javier Fresno

Universidad de Valladolid, Spain

Alejandro Ortega-Arranz

Universidad de Valladolid, Spain

Hector Ortega-Arranz

Universidad de Valladolid, Spain

Arturo Gonzalez-Escribano

Universidad de Valladolid, Spain

Diego R. Llanos

Universidad de Valladolid, Spain

ABSTRACT

Pursuing a college degree is a task that requires a great amount of time and effort. Universities are facing a big challenge to attract students and keep them motivated. The gamification of education is a practice that expects to increase the students' engagement, which in turn increases learning outcomes. Nevertheless, obtaining beneficial results from gamification requires educators to mold the teaching to include this new practice, usually involving a lot of effort. In this chapter, the authors present a new software tool developed to encourage gamification dynamics, and they describe their experience using this tool in a Parallel Programming course. The chapter describes the structure of the course, the different proposed activities, the organization of hardware resources, the design of the developed software tool, and an evaluation of the gamified course. The results show that the use of gamification techniques has been a great success. The authors have had a very positive response from their students, and there has been also a big percentage of passing students.

INTRODUCTION

The use of game design elements in non-game contexts is commonly known as gamification (Deterding, Dixon, Khaled, & Nacke, 2011). Education is a non-game context where gamification can affect the students' learning outcomes (Arnab, et al., 2014), behavior (Hakulinen, Auvinen, & Korhonen, 2013), motivation, and engagement (Muntean, 2011). The students are more likely to increase their willingness

DOI: 10.4018/978-1-5225-1034-5.ch006

Applying Gamification in a Parallel Programming Course

and desire to be successful when they are engaged in an active learning context (Dicheva, Irwin, Dichev, & Talasila, 2014). Teachers can take advantage of these potential benefits by including game design elements and common features of videogames into the learning contexts.

This chapter describes the required processes to carry out the gamification of a Parallel Programming course, together with some interesting results and situations. The main purpose of this work is to publish not only the positive results obtained, but also describe the tool used, the experience process, the feedback received, and our conclusions regarding the gamification usefulness.

Parallel programming consists in using two or more devices at the same time for carrying out the computations required to solve a problem. Usually the aim of applying parallel programming is to reduce the high temporal costs needed when using only a single device (Grama, Gupta, Karypis, & Kumar, 2003). Its use has been proven to be key to research high performance solutions (Navarro, Hitschfeld-Kahler, & Mateu, 2014). Due to this fact, the evolution of computing not only has focused on developing faster processors, but also on the creation of new computer architectures involving more processing units. Due to the evolution towards more complex architectures, it is a difficult task for the programmers to create optimal parallel solutions. Programming for these modern systems requires to solve some new practical issues, such as the data partitioning, data sharing, data transfers, the coordination and synchronization, or the migration to new computing paradigms.

The purpose of the Parallel Programming course, taught as part of the Computer Science degree at the Universidad de Valladolid, is to teach the students how to take advantage of the principles of parallel programming, focusing on three key examples of highly different programming models: OpenMP (OpenMP Architecture Board, 2013), MPI (Message Passing Forum, 2015), and CUDA (Sanders & Kandrot, 2010). The practical part involves a teaching methodology based on small projects development using the same sequential base program provided to the students. They have to create correct parallel solutions as they learn the foundation and techniques of each programming model. Due to the particular difficulties of learning new computing paradigms involving concurrency and data distribution, we have applied a gamification process to our course with the aim to attract the students' interest and to raise their motivation and engagement.

In order to introduce the gamification in the course, we have developed a software tool, called *Tablón*. It eases the submission procedure for the developed code to be executed on real chosen parallel machines. Moreover, it displays at real time the best performance codes submitted and tested so far. With this tool it is possible to implement some particular gamification dynamics, such as leaderboards.

The structure of the chapter is as follows. Section "Background" exposes the background related to the gamification process, the definition of the game design elements used, and some related examples found in the literature. Section "Gamifying a Parallel Programming Course" presents the designed course structure and the chosen gamification elements that have been introduced in our Parallel Programming course. Section "Hardware and Software Environment" describes the technologies and computing platforms needed to support the dynamics we want to encourage. The development of the software to ease the deployment of the codes into these computing platforms is presented in Section "Description of the Developed Software". Section "Experimental Validation" shows the results and experimental measures we have obtained during the enactment of the course. Section "Future Research Directions" describes the possible extensions and future work. Finally, Section "Conclusions" sums up the chapter contributions.

BACKGROUND

Games are interactive activities able to rise up player's feelings (e.g. excitement, fun, etc.) through challenges, competitions, and many other factors. Over the last years, videogames are considered as the main genre of human entertainment (Domínguez, Saenz-de-Navarrete, De-Marcos, Fernández-Sanz, Pagés, & Martínez-Herráiz, 2013). One of the reasons behind the popularity of videogames is that people are more likely to spend their time and effort in activities that produce such mentioned feelings (Fitz-Walter, 2015).

Some innovative teachers started to study the potential benefits of using games in learning environments. One approach that tries to use game features to enhance motivation and engagement is the so-called gamification. Gamification is considered as the use of game design elements in non-game contexts (Deterding, Dixon, Khaled, & Nacke, 2011). The aim of gamification is to identify the game elements and features that make videogames enjoyable and fun to play (i.e. game design elements), and adapt them to be included in non-game contexts (Simões, Redondo, & Vilas, 2013). Education is a non-game context where the use of game design elements can provide potential benefits to students while learning.

Game Design Elements

Game design elements are the elements and features that frequently appear in games and make them enjoyable and fun to play (Simões, Redondo, & Vilas, 2013). The most used and deployed game design elements in educational contexts are rewards (e.g. points and badges), and competition elements (e.g. leaderboards) (Dicheva, Dichev, Agre, & Angelova, 2015):

- **Badges** (also denominated as achievements, trophies, or ribbons): Optional rewards and goals whose fulfillment is outside the scope of the core activities (Domínguez, Saenz-de-Navarrete, De-Marcos, Fernández-Sanz, Pagés, & Martínez-Herráiz, 2013) (Hamari, Koivisto, & Sarsa, 2014). Badges are usually represented with graphical icons. Badges represent virtual rewards with no practical value in the physical world. However, students aim to obtain them in order to prove, to themselves and to the other students, that they have reached an important achievement. The description of the tasks which must be fulfilled to get these rewards is frequently public. Thus, badges can indirectly guide students to particular desired behaviors. Moreover, they can offer additional challenges without raising the requirements for the highest course grade as demonstrated by Hakulinen *et al.* (2013). Finally, some badges can be also designed as hidden and awarded by surprise when students meet some special conditions. The expectation to discover new rewards by extensive exploration of the solution space can increase the students' engagement and interest in the gamified activities.
- **Leaderboards**: Rankings sorted by one or more quantitative values (e.g. points, levels, or time to accomplish activities). Their main purpose is to make simple comparisons among the students (Zichermann & Cunningham, 2011). They can be also used to track and display desired actions, using competition to drive valuable behavior (Bunchball Inc., 2010).

Related Work

Many researches are gamifying their activities aiming to analyze the effects and potential benefits of gamification in ICT (Information and Communications Technology) learning contexts. The following paragraphs describe some of the proposals; Table 1 shows a comparison of their features.

Hakulinen *et al.* (2013) gamified an online environment for learning data structures and algorithms with public badges which did not alter the students' final grade. The results show a positive impact on students' behavior such as early tasks submission, avoid trial and error submissions, and better course grade even though badges have no impact of the final grade. However, not all the proposed badges seem to induce this effect.

Domínguez *et al.* (2013) developed a gamification plug-in for a virtual learning environment (VLE). They performed an experimental evaluation on the course "Qualification for users of ICT", which provides knowledge about how to effectively use common ICT tools. The gamification was carried out with medals and trophies (badges) and a public leaderboard. Experimental results showed higher initial motivation and better scores in both practical assignments and in overall score. However, they also reported that students performed poorly on written assignments and participated less on class activities.

Ibáñez *et al.* (2014) gamified a platform used for a C-programming learning course taken by undergraduate engineering students. To carry it out, they used rewards such as badges and a leaderboard. The results show an increment on the students' cognitive engagement with the gamified activities. Once they finished the compulsory tasks, they kept on doing the tasks because some students wanted all the badges, wanted to reach better positions on the leaderboard, or wanted to increase their knowledge. Moreover, students showed a moderate improvement in learning outcomes.

Aziz *et al.* (2015) describe the design process of a system to be used in a Parallel Programming course, to provide fast feedback and auto-grading to students through a leaderboard. However, this system has not already been implemented.

Previous works shows the current interest in the community for gamifying IT learning situations, including Parallel Programming courses. Thus, the aim of this chapter is to contribute with a system able to gamify parallel programming practices and providing first experimental results when gamifying such courses.

Table 1. Comparison of features between the related work and our proposal

Work	Area / Course	Badges	Leaderboard	Student Response
Hakulinen et al.	Data structures and algorithms	Yes		Positive
Domínguez et al.	Qualification for users of ICT	Yes	Yes	Mixed
Ibáñez et al.	C Programming		Yes	Positive
Aziz et al.	Parallel programming			Not implemented
Tablón	Parallel programming	Yes	Yes	Positive

GAMIFYING A PARALLEL PROGRAMMING COURSE

This section describes the general elements and features of the Parallel Programming course, together with our approach to improve the students' motivation using gamification mechanisms: Badges and leaderboards.

Course Description

The purpose of the Parallel Programming course is to teach the students how to develop applications that can take advantage of the new generation of parallel machines. After a theoretical introduction to parallelism and parallel computing, the course is composed of three different modules with theoretical and practical lessons. On each part, the students learn the fundamentals of parallel programming using a different parallel programming model: OpenMP (OpenMP Architecture Board, 2013) for shared-memory systems, MPI (Message Passing Forum, 2015) for distributed memory systems, and CUDA (Sanders & Kandrot, 2010) to program Graphics Processing Units (GPUs). For each technology, the students are required to develop the parallel version of a given sequential program to gain hands-on knowledge of the parallel programming task. We used the same sequential base program to focus on the similarities and differences between the chosen technologies. The exercises are done in two-people groups and they represent the 65% of the course grade. The other 35% is individually graded by using partial test exams about the theoretical concepts of each technology.

Notwithstanding the efforts done by the coordinators of the course to ease the learning of parallel programming models, their complex concepts are usually difficult to understand without the willing and motivation of the students. Thus, we have applied a gamification process to our course with the aim to attract their interest and to raise their motivation and engagement.

Leaderboard

One of the main goals of applying parallel programming mechanisms is to speed up its computing time. Depending on the proper usage of the resources, and correct tuning of the execution parameters, some of the proposed implementations would have lower execution times.

In the practical section of the course, each student group develops its own parallel solution to the problem, leading to different implementations. Then, the quality of the student implementations can be ranked according to the execution times. In order to carry out a fair comparison, the solutions are tested against several input sets chosen by the teachers which are not shown to the students, in a reference parallel machine. The features of this reference machine are public to the students.

Since we teach three different parallel programming models, we decided to create three leaderboards. We expected that publishing the students' results into a public leaderboard would be positive for their learning due to:

- **The Active Feedback:** The students would see when others have started to develop their work, and also which is the mean quality of the solutions of the rest of the class. Moreover, a small percentage (10-20%) of the total grade of the practical exercises is obtained based on the position of the leaderboard.

Applying Gamification in a Parallel Programming Course

- **The Competitive Environment:** The students could be motivated to learn more parallel techniques, or tune and improve their solutions, in order to climb up in the ranking.

Proposed Badges

The gamification goal consists on rewarding not only the good quality of the parallel solutions but also other particular behaviors related to the time invested, the effort done, or the engagement with the course, among others. In order to do so, we have created three different kind of badges in function of the difficulty to achieve them: Bronze, silver, and gold (see Figure 1). Badges are granted using a Moodle platform (Moodle Learning Management System webpage).

- **Bronze Badges:** They are created with the aim to motivate the students for getting involved in the course, by for example uploading a photo to their profiles, or being in the top position of the leaderboard (without caring about the quality of the obtained execution times). With this pioneer medal, we also want to encourage the students to start working in the exercise early. In this category, we have also included the so-called hidden medals, which are rewards for casual and curious events related to the use of the leaderboard. In this way, the use of this platform is not only a simply way to deploy their programs but also a possible source of fun.
- **Silver Badges:** They are created for specific situations that require more effort or that are more difficult. For example, keeping the first position for at least 3 days (“dominator”), obtaining a better time than a dominator (“assassin”), or getting one of the first positions in the first try of the leaderboard (“sniperwolf”). Additionally, we also want to encourage the students to get the top 5 by rewarding them with some funny medals related to famous groups of five people. See Figure 1 for more examples.
- **Gold Badges:** They are created to encourage and reward the most notorious students that complete the maximum goals proposed by the course. These badges are granted to the students that are in the first positions of the leaderboard or to the ones that obtain the maximum mark in the partial exams of theory.

HARDWARE AND SOFTWARE ENVIRONMENT

Due to the different programming models that are taught in our course, it is needed a wide variety of resources in order to properly carry out the laboratory exercises. Thus, several computing machines have been prepared to execute the programs of the students. The following sections describe these machines, the desired configuration to create the leaderboard element for each programming model, and the software responsible of coordination of programs submission, execution and bookkeeping. Table 2 summarizes some characteristics and details of these machines.

Target Platform Description

The practical exercises for the OpenMP programming model require handing different cores in a shared memory system. We use *geopar*, a 16 core machine with a clock speed of 1.68Ghz, and a total memory

Applying Gamification in a Parallel Programming Course

Figure 1. List of bronze, silver and gold badges, and the conditions to achieve them, as shown by the Moodle system. Badges marked as [H] are hidden.

	Badge Name	Criteria
	Booked!	Upload a photo to Moodle profile
	Pioneer	First 5 groups with a valid result in the Leaderboard
	Portal 1k	[H] Execution number 1000 in the Leaderboard
	All Lucky 7s	[H] Execution number 7777 in the Leaderboard
	We are up all night to get lucky	[H] Send a program after midnight
	Slow but safe	Last position of the first quartile of Leaderboard solutions
	The Wolf of the Leaderboard	Improve your ranking in the Leaderboard during 2 consecutive days
	Doominating!!!	Dominate the Leaderboard in first position during 3 consecutive days
	The Legend Assassin	Dethrone a dominator (dethrone a group that is maintaining the first position during 3 consecutive days)
	SniperWolf	Pass 90% of groups already in the Leaderboard in the 1st execution
	EcoHero	Lower computation quota consumed inside the first quartile of Leaderboard solutions
	The JacksOMP Five	Rank in the top 5 positions of the OpenMP Leaderboard
	MPI Rangers	Rank in the top 5 positions of the MPI Leaderboard
	Cudavoir Dogs	Rank in the top 5 positions of the CUDA Leaderboard
	I'm bOMP, James bOMP	Obtain a 10 grade in the theory exam of OpenMP
	This is a work for Superman	Obtain a 10 grade in the theory exam of MPI
	You're good... and you know it	Obtain a 10 grade in the theory exam of CUDA
	The King of POMP	Rank the 1st position in the OpenMP Leaderboard
	MegaZord	Rank the 1st position in the MPI Leaderboard
	Cudentin Tarantino	Rank the 1st position in the CUDA Leaderboard
	Something to prove	Be in the TOP 5 of all Leaderboards

RAM of 32 Gb, to be platform for this purpose. The students will learn that a proper parallelization of the algorithm, by coordinating these slow cores, can beat the sequential version, even if this sequential version is executed in a faster CPU.

As MPI is also able to handle different cores inside a shared-memory system, geopar is also a useful platform where to deploy the MPI programs. However, the most interesting part of the MPI programming model is to coordinate cores from different heterogeneous machines connected through a network. Thus, four different computers with different power capabilities have been configured: *thunderbird*, *titan01*, *titan02*, and *titan03*. All machines have 4 cores, representing a total of 16 cores in the complete cluster.

Applying Gamification in a Parallel Programming Course

Table 2. Description of the machines used in the academic heterogeneous cluster. The TB abbreviation stands for Titan Black.

Machine Name	Used for	#Cores	#Cores	Clock Speed	Memory	GPU Dev.
geopar	OMP - MPI	4x Intel Xeon	16	1.68 Ghz	32 Gb	-
thunderbird	MPI - CUDA	Intel i5	4	3.00 Ghz	8 Gb	Tesla K40
titan01	MPI - CUDA	QuadCore	4	2.33 Ghz	6 Gb	GTX TB
titan02	MPI - CUDA	QuadCore	4	2.33 Ghz	6 Gb	GTX TB
titan03	MPI - CUDA	QuadCore	4	2.50 Ghz	3 Gb	GTX TB
titanlb	CUDA	QuadCore	4	2.33 Ghz	3 Gb	GTX TB
portal	front-end	Core2Duo	2	1.86 Ghz	3 Gb	-

This is done with the aim of allowing the comparison between this distributed-memory environment and the shared-memory system of geopar. The students will learn that a proper distribution of the data, together with a good communication design, can lead to better performance, although the communications between the machines imply a penalty in the execution times. Additionally, as the machines have different running frequencies, they can apply a load-balancing policy, distributing more work to the more powerful cores. The first machine, thunderbird, is an Intel i5 with a clock speed of 3 Ghz, whereas the others, titan01-03 are QuadCores with clock speeds of 2.33 Ghz, 2.4 Ghz, and 2.5 Ghz, respectively.

The CUDA programming language takes advantage of the powerful capabilities of the modern Nvidia's Graphic Processor Units (GPUs). This company has provided several GPUs for teaching purposes in the context of the GPU Teaching Center programming a nomination awarded to University of Valladolid since 2014. Programming with this kind of hardware accelerators usually implies to re-design the strategy of parallelization in order to exploit the GPU architecture. Then, the students will learn that a proper handling of the GPU resources, together with a specific parallelization of the algorithm focused on these devices, can reach significant speedups compared with the sequential version. Furthermore, in some cases, the GPU solution can beat other parallel versions obtained using the previous programming models.

Configuration of the Leaderboards

All the machines described in the previous section can be used for trial executions to test the programs. Programs submitted to be considered in a leaderboard are executed in particular competitive configurations, and they have priority over the trial executions. The machines involved in the different leaderboards are the following:

- Leaderboard for OpenMP: geopar
- Leaderboard for MPI: thunderbird, titan01, titan02, and titan03
- Leaderboard for CUDA: titanlb

The OpenMP leaderboard is executed using the 16 cores of the geopar machine; the MPI leaderboard uses the four cores of the four machines (16 cores); and the CUDA leaderboard uses only one machine with a GPU NVIDIA GTX Titan Black. We have selected this GPU for the leaderboard because the majority of non-leaderboard machines have the same kind of device. Therefore, the trial executions, out

of the leaderboard, that the students perform to profile their programs, will have a similar behavior when competing in the leaderboard. Finally, the Tesla board is left out for those curious students that want to test their programs in a GPU with some different characteristics.

Coordination of the Computing Cluster

The performance of the programs created by the students is the key to rank them in the leaderboards. In order to obtain a clean measurement of the running time, the executions of their codes must not be perturbed with any other events, such as connections of other users, compilations, etc. Therefore, the machine or machines where a program is deployed should be exclusively reserved during the execution. To achieve this kind of isolation we have prepared another machine, named *portal*, which plays the role of the front-end of the cluster. The students should connect to this machine, and compile their programs there, instead of using straightly the cluster machines. We have installed a queue system in order to manage and coordinate the executions on the cluster resources from our portal front-end. We have used SLURM (Yoo, Jette, Grondona, & Springer, 2003), that is an open-source resource manager designed for Linux clusters. It provides three different functionalities:

1. It allocates exclusive and/or non-exclusive access to resources (computer nodes, or cores) to users for time slots so they can perform their executions;
2. it provides a framework for starting, executing, and monitoring the executions on a set of allocated nodes; and finally,
3. it arbitrates contention for resources by managing a queue of pending jobs.
 1. **Allocation of Resource Accesses:** The resources are gathered in partitions that are designed for the different roles they have to play depending on the current programming model. The following list shows the name of the created partitions, the machines that compounds each of them, and their description:
 - ***omp - geopar***: A partition for the OpenMP and MPI executions inside a shared-memory system
 - ***lbomp - geopar***: A partition with higher priority than its analogous partition, *omp*, with exclusive access for the OpenMP leaderboard.
 - ***mpi - thunderbird, titan01, titan02, titan03***: A partition for the MPI executions inside a distributed-memory environment.
 - ***lbmpi - thunderbird, titan01, titan02, titan03***: A partition with higher priority than its analogous partition, *mpi*, with exclusive access for the MPI leaderboard.
 - ***cuda - titan01, titan02, titan03***: A partition for the CUDA executions on the GTX TitanBlack boards.
 - ***tesla - thunderbird***: a partition for the CUDA executions on the Tesla K40c board.
 - ***lbcuda - titanlb***: a partition with exclusive access for the CUDA leaderboard.
 2. **Management Directives:** The SLURM queue system has several commands to monitor the partitions and their corresponding resources. A description of the available directives can be found in (Yoo, Jette, Grondona, & Springer, 2003).
 3. **Resource Contention:** SLURM automatically keeps the executions that are waiting for resources in a priority queue. Each partition is associated with a priority value. Thus, the order of assigning resources to the pending executions straightly depends on the partition to which

Applying Gamification in a Parallel Programming Course

the job has been submitted. With this resource contention system, we avoid that different executions interfere in the running times of each other, and we have an automatically dispatcher of resources for the requests that favors the leaderboard executions.

The partitions have a timeout that we have configured to a fixed amount, in order to avoid infinite running times of erroneous programs. The following code shows the configuration of the SLURM queue following all the descriptions shown before.

Code 1. SLURM's code configuration. All the partitions have also the parameter values Default=NO, MaxTime=2, State=UP, AllowGroups=tablon, that have been omitted in the figure for clarity.

```
## Machine description
NodeName=portal Sockets=1 CoresPerSocket=2 Feature=intel
NodeName=geopar Sockets=4 CoresPerSocket=4 Feature=intel
NodeName=titan01 Sockets=1 CoresPerSocket=4 Feature=intel
NodeName=titan01 Sockets=1 CoresPerSocket=4 Feature=intel
NodeName=titan02 Sockets=1 CoresPerSocket=4 Feature=intel
NodeName=titan03 Sockets=1 CoresPerSocket=4 Feature=intel
NodeName=titanlb Sockets=1 CoresPerSocket=4 Feature=intel
## Partition description
PartitionName=omp Nodes=geopar Priority=100
PartitionName=lbomp Nodes=geopar Priority=10
PartitionName=mpi Nodes=thunderbird,titan01,titan02,titan03 Priority=100
PartitionName=lbmpi Nodes=thunderbird,titan01,titan02,titan03 Priority=10
PartitionName=cuda Nodes=titan01,titan02,titan03 Priority=100
PartitionName=tesla Nodes=thunderbird Priority=100
PartitionName=lbcuda Nodes=titanlb Priority=10
```

DESCRIPTION OF THE DEVELOPED SOFTWARE

This section describes the system we have developed to control the execution of the students code. It runs on top of the SLURM queues described in the previous section, and gives us additional features to implement the gamification elements described in Section “Gamifying a Parallel Programming Course” The system is called *Tablón*, the Spanish translation of leaderboard, which is its main feature.

Tablón is composed of a client script and a server. The client is used by the students to send the code of their programs and to select the execution parameters (such as number of parallel threads or processes, or command line arguments). The server checks the user credentials, receives the source code, and validates the request. If the request is valid, the server compiles the source code and executes it in the appropriate SLURM queue. Tablón server has also a web interface, where the students can obtain the compilation or execution results of their program. It also keeps the accounting of the computing

resources spent by each group. Moreover, it can be also used by the teacher to check the uploaded codes. Figure 2 shows a diagram with Tablón architecture. The rest of the section describes with more detail the different features of the system and discusses their design.

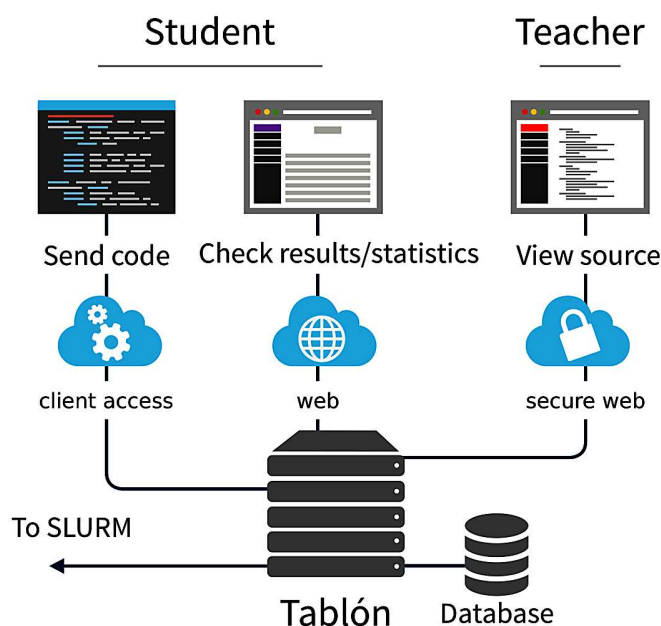
Queues and Leaderboards

The compilation, execution, and control of the programs in Tablón are managed by execution queues. The Tablón queues are associated to a SLURM queue and they extend their functionality to also take care of the compilation and deployment. The current version has queues to support programs implemented in C or C++. They can be sequential or parallel programs using OpenMP, MPI, and/or CUDA technologies. Several queues of each type can be defined in Tablón. A queue has an associated Makefile script that is used to define the compiler and its flags. The programs are compiled in the frontend machine and then they are deployed in the appropriate execution machines of the cluster. Tablón queues can be used to control the parameters of the SLURM queues. Thus, depending on the parallel technology, a queue can limit the number of maximum resources the students can use, such as number of OpenMP threads, MPI processes, or execution time.

In addition to the queues, Tablón allows us to define leaderboards queues. A leaderboard queue is a special case of queue that is used to validate a student solution to a proposed problem. For each exercise, the students must use a particular format for the input parameters. The output of the program is captured by Tablón from the standard output and therefore, it must also follow a specific format to be properly processed.

Tablón allows the teachers to define different tests that the student program should pass to be included in a leaderboard. Each test defines the execution parameters (such as the number of processes or threads),

Figure 2. Tablón architecture



Applying Gamification in a Parallel Programming Course

the input arguments (used to define the different input sets), and the expected output so it can be validated. If the solution is correct, the program is ranked according to its execution time compared with the other student solutions. The leaderboard list can be consulted by all the students using the web interface.

The Client Script

The client is a python script that is used by the students to send their codes. The client makes a previous process of the code checking the maximum size of the program. For the CUDA programs, it combines all the CUDA kernels in a single file. It authenticates the user and sends the codes using the Secure Remote Password protocol (SRP). SRP is a cryptographically strong authentication protocol for password-based, mutual authentication over an insecure network connection (Wu, 1998). With the client, it is possible to select the queue, number threads and/or processes, and the program arguments (see Code 2).

Code 2. Example of use of the Tablón client

```
$ ./client -h
Usage: client [options] program.c/.cu [program args]
Options:
-u USER, --user=USER
-q QUEUE, --queue=QUEUE
-n NPROCS, --nprocs=NPROCS
-t THREADS, --threads=THREADS
-w TIMEWALL, --timewall=TIMEWALL
$ ./client --user grupo00 --queue openmp --threads 5 program.c
Write your password:
Connecting ...
Connected to grupo00@tablon.infor.uva.es
Successful authentication
Sending request
Request sent successfully
Request id 2772
http://tablon.infor.uva.es/request?rid=2772
```

Quota Policy, Limits, and Security

Tablón implements a quota policy to limit the number of executions that a student can send and the frequency between requests. The objective behind this policy is to allow each user a fair amount of execution time. There are different parameters that are configurable. There is a time limit between requests in order to avoid continuous submissions from the same group. Moreover, there is a quota that limits the total time of the combined executions of each group. When a group consumes the entire quota, they are penalized with a bigger time limit between consecutive executions.

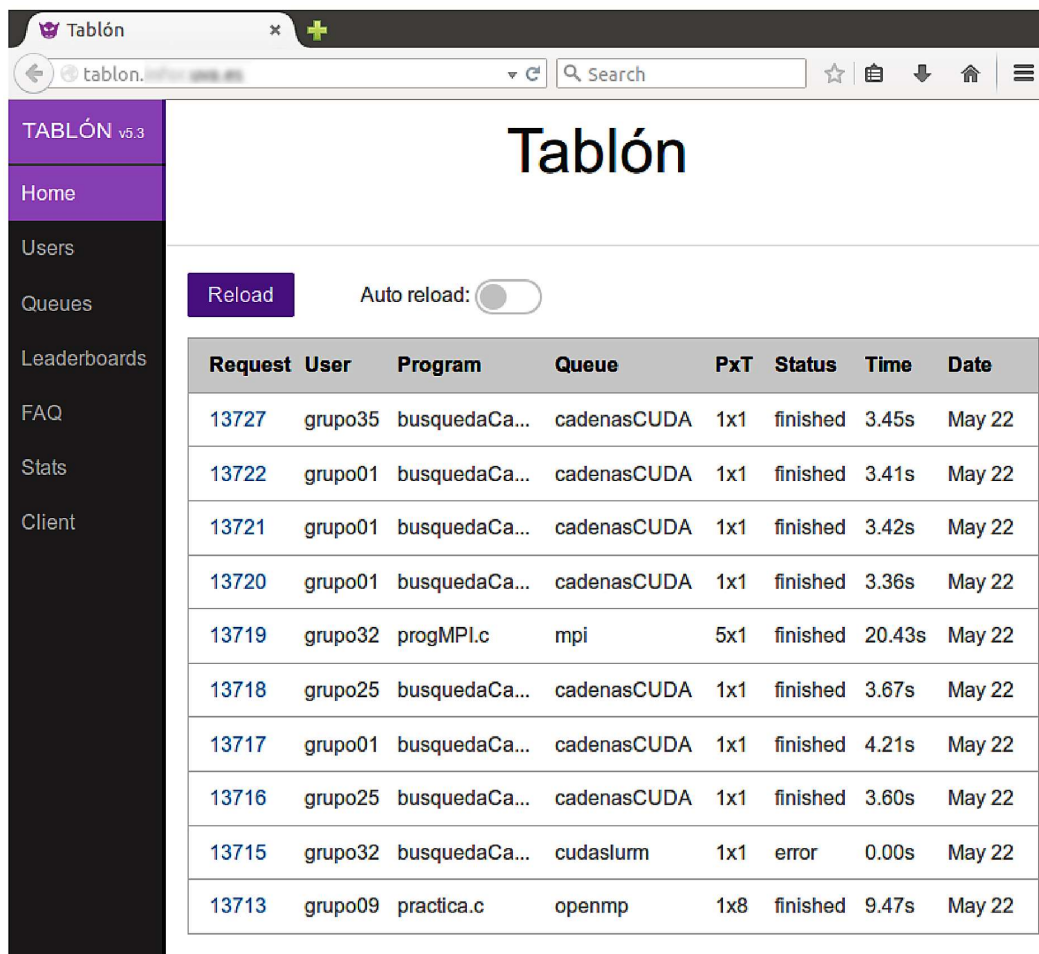
There are other configurable limits, such as a timewall that is defined for each queue. There is also a limit for the size of the standard output that the program can use. This output is shown in the web interface.

Finally, there are some security checks. Tablón analyzes the user program to detect unsafe code, such as input/output or system function calls. If the user sends some code that Tablón flags as unsafe, it will not be compiled nor executed. All the communications between Tablón and the program are done using the standard input and output streams. Moreover, we provide a custom library with limited file access to read the input sets of the proposed exercises, in order to prevent unauthorized accesses to the file system.

Web Interface

The Tablón web interface is used to allow the students to check the result of their executions. A screenshot can be seen in Figure 3. The main page has a list with a summary of the last executions. There is a page with more information for each execution request. The students can check the state of the execution (sent, compiling, queued, executing, or finished), the selected queue, the input parameters (program arguments, number of threads/processes), the output of the program, and the possible error messages.

Figure 3. Screenshot of the web interface of Tablón



The screenshot shows the Tablón web interface. The browser address bar displays 'tablón.' and the page title is 'Tablón'. A sidebar on the left contains navigation links: Home, Users, Queues, Leaderboards, FAQ, Stats, and Client. The main content area features a 'Reload' button and an 'Auto reload' toggle switch. Below this is a table with the following data:

Request	User	Program	Queue	PxT	Status	Time	Date
13727	grupo35	busquedaCa...	cadenasCUDA	1x1	finished	3.45s	May 22
13722	grupo01	busquedaCa...	cadenasCUDA	1x1	finished	3.41s	May 22
13721	grupo01	busquedaCa...	cadenasCUDA	1x1	finished	3.42s	May 22
13720	grupo01	busquedaCa...	cadenasCUDA	1x1	finished	3.36s	May 22
13719	grupo32	progMPI.c	mpi	5x1	finished	20.43s	May 22
13718	grupo25	busquedaCa...	cadenasCUDA	1x1	finished	3.67s	May 22
13717	grupo01	busquedaCa...	cadenasCUDA	1x1	finished	4.21s	May 22
13716	grupo25	busquedaCa...	cadenasCUDA	1x1	finished	3.60s	May 22
13715	grupo32	busquedaCa...	cudaSlurm	1x1	error	0.00s	May 22
13713	grupo09	practica.c	openmp	1x8	finished	9.47s	May 22

Applying Gamification in a Parallel Programming Course

All this information is stored for all the students' executions and can be looked up at any moment by using the link the client script provides when submitting a new program. There are other pages with the leaderboard ranks, frequently asked questions, and some stats.

EXPERIMENTAL VALIDATION

This section describes the results of the studies introduced previously to test the different desired attributes of the Tablón platform. The data has been obtained during the 2015 academic year course.

Experimental Studies

The Tablón platform stores all the data regarding not only the source code sent by the student groups, but also all the data and statistics of the student activity and their behavior patterns. With this information we are able to evaluate if the use of Tablón and the gamification techniques have been key to make the course more useful, attractive, and enjoyable for the students.

The following list describes the studies we will carry out:

- **Study I: The Usefulness of Tablón:** In the first study we will analyze the impact the use of Tablón has for the students, in order to determine its usefulness. We will look through the different measures this tool provides.
- **Study II: The Engagement of Tablón:** With this study we want to determine if the use of Tablón generates engagement and if the students are more motivated when using this kind of mechanics. We will study the students' submissions and their relationship with the proposed activities.
- **Study III: The Leaderboard Element for Competitiveness:** Another study consists on the evaluation of the leaderboard as element for competitiveness. The assignments the students have to complete are presented as a contest to obtain the best solution. Thus, using this tool they might challenge each other and we expect this could improve their results.
- **Study IV: Student's Satisfaction:** In order to test the students' satisfaction with the course, we have analyzed the results of the official student survey that is organized by the Academic and Educational Innovation Committee of our university. The survey gathers the opinion of the students on the quality of their courses, including the structure of the course, the used materials, the proposed activities, etc.
- **Study V: Student's Marks:** Finally, we will analyze the final marks obtained by the students of the course.

Study Results I: Checking Tablón's Usefulness

The use of Tablón has been a great success. In the considered academic year, the course had 37 student groups and there have been more than 20000 program executions. The average of executions per group was around 540. To do the proposed activities, the students were not required to use the Tablón platform, except for the leaderboard use. They could also use their own systems, the ones provided during the laboratory hours, or the general equipment of the university.

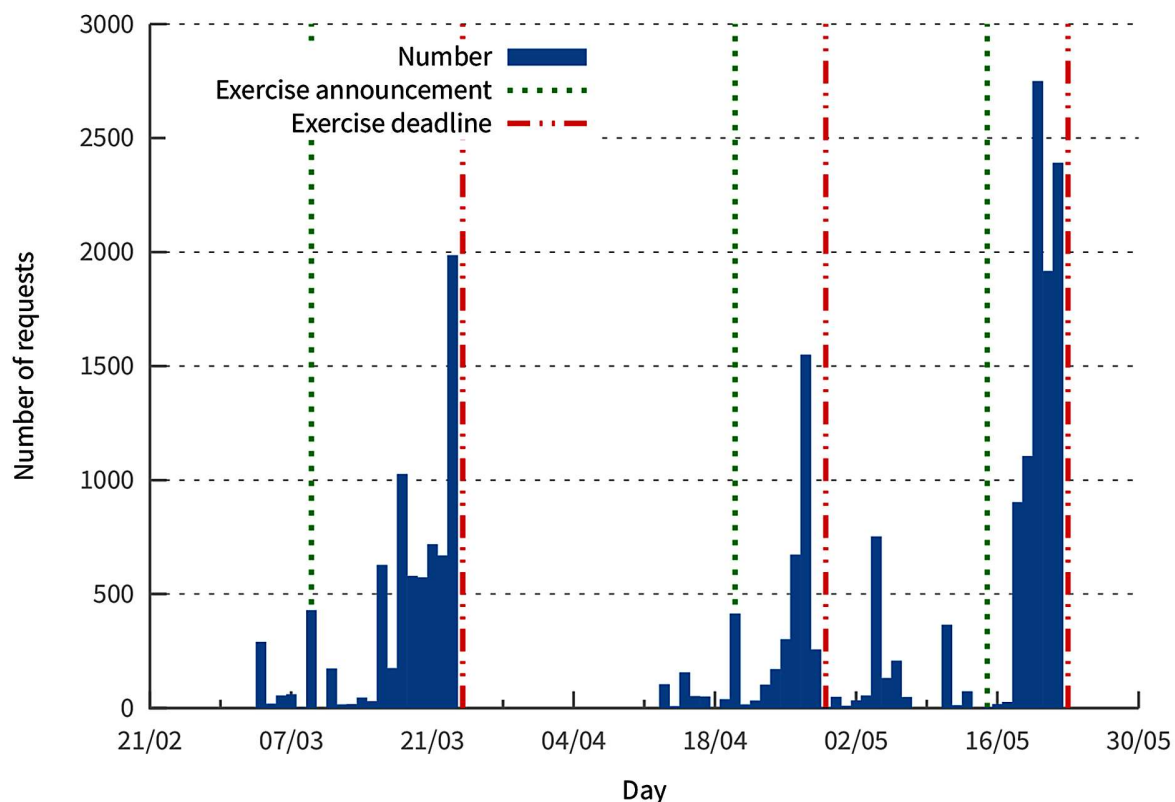
Interestingly, 53% of the requests that Tablón platform has received have been sent from outside of the university network. Without the developed system, the students would have had many difficulties to deploy and test their solutions, and probably the interest for the course would have faded. Thus, these results can be taken as an indicator that the use of the Tablón platform has been key for a successful development of the course.

Study Results II: Evaluating Tablón’s Engagement

Figure 4 shows the number of program execution requests by day along the semester the course was taught. There are three different periods of activity that match the three practical topics of the course (OpenMP, MPI, and CUDA). The number of execution increases the days before the final exercises’ deadlines. There are also some peaks of activity during the practical session days. The inactivity period after the first exercise deadline is due to Easter holidays, when Tablón machines were offline.

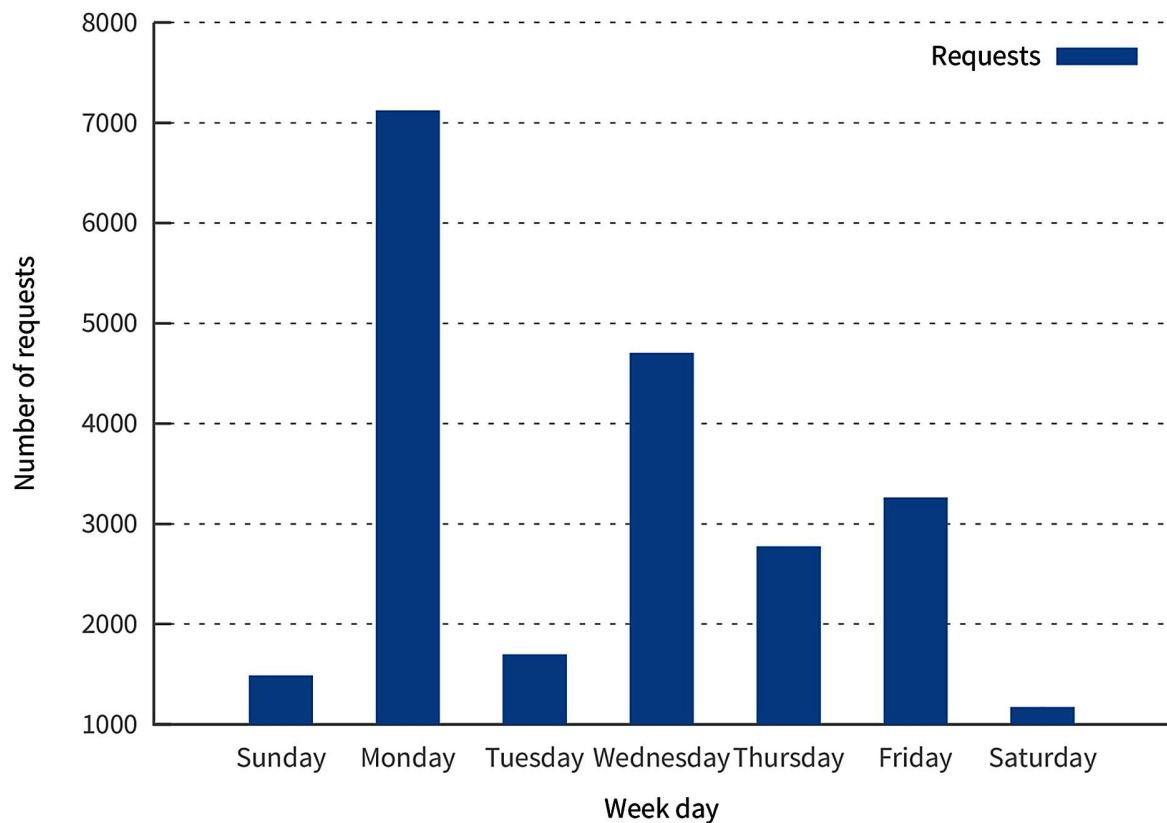
Figures 5 and 6 show the total number of requests by weekday and by hour respectively. The activity peaks match with the practical sessions of the course (Mondays and Wednesdays from 10 to 12) and with the deadlines of the exercises (Mondays and Fridays). The results show that in the afternoons the students have been working although there are not course sessions in that hours. The students use these

Figure 4. Total number of request by day. The vertical lines mark the announcements and deadlines of the exercises to be evaluated.



Applying Gamification in a Parallel Programming Course

Figure 5. Total number of request by weekday



time periods to finish the exercises (Figure 6 also shows that, contrary to common belief, computer science students do sleep at night).

Table 3 shows some statistical measures of the number of executions of different programs before and after the students obtained a valid execution in the leaderboard. Note that the table only shows the Tablón executions. However the students could also use the laboratory computers and their own to develop the exercise. The number of sends before having a valid result tells us about the complexity of the practical exercises. Once a student group obtained a valid result in the leaderboard, it was no mandatory to send more executions unless they wanted to improve the exercise mark (that derived in a 10-20% improvement depending on the exercise). The executions after the valid result indicate how committed and interested the students were with the course. This table shows a great variability. There are some students that only needed a few tries to obtain a valid solution while others needed much more work. However, an important fact is that the majority of student groups did not stop working after the first valid solution but rather they tried to improve their position on the leaderboard.

Study Results III: The Leaderboard Element for Competitiveness

Figures 7, 8, and 9 show the evolution of the leaderboard positions for the three different used programming models and technologies (OpenMP, MPI, and CUDA). We can see that the lines are crossing almost

Figure 6. Total number of request by hour

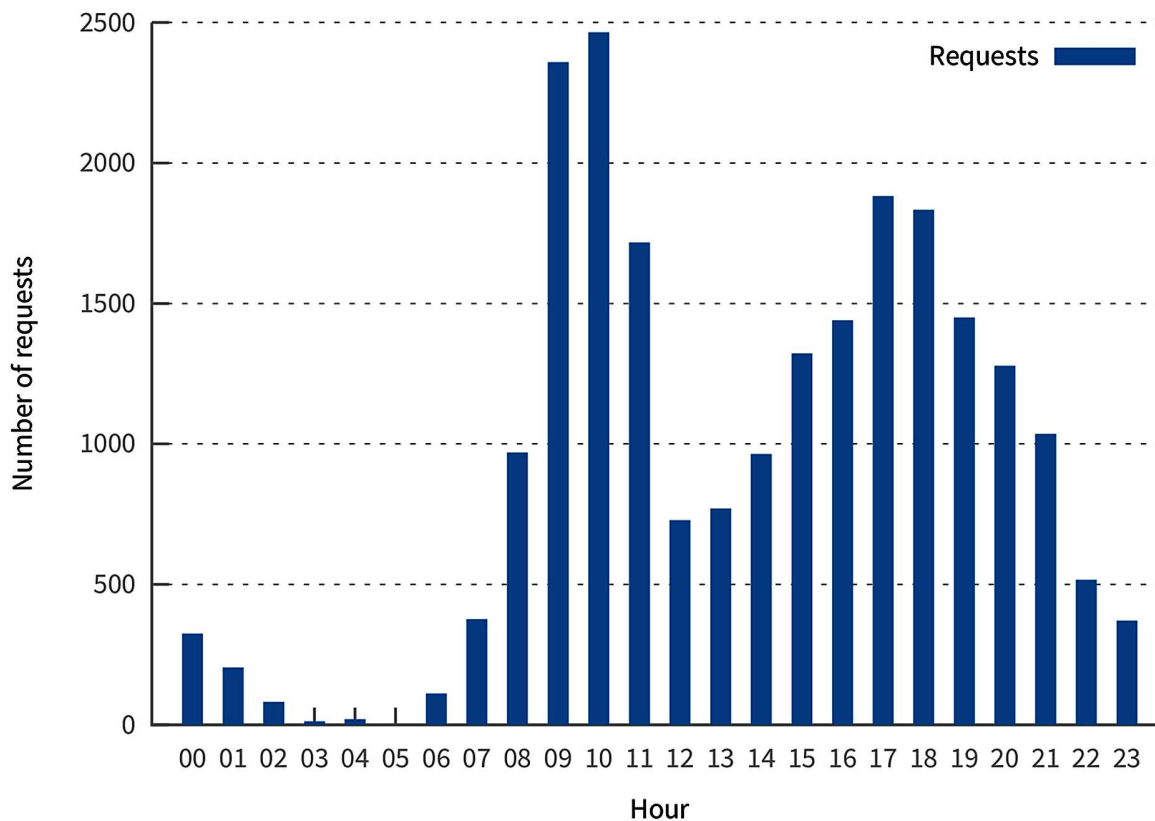


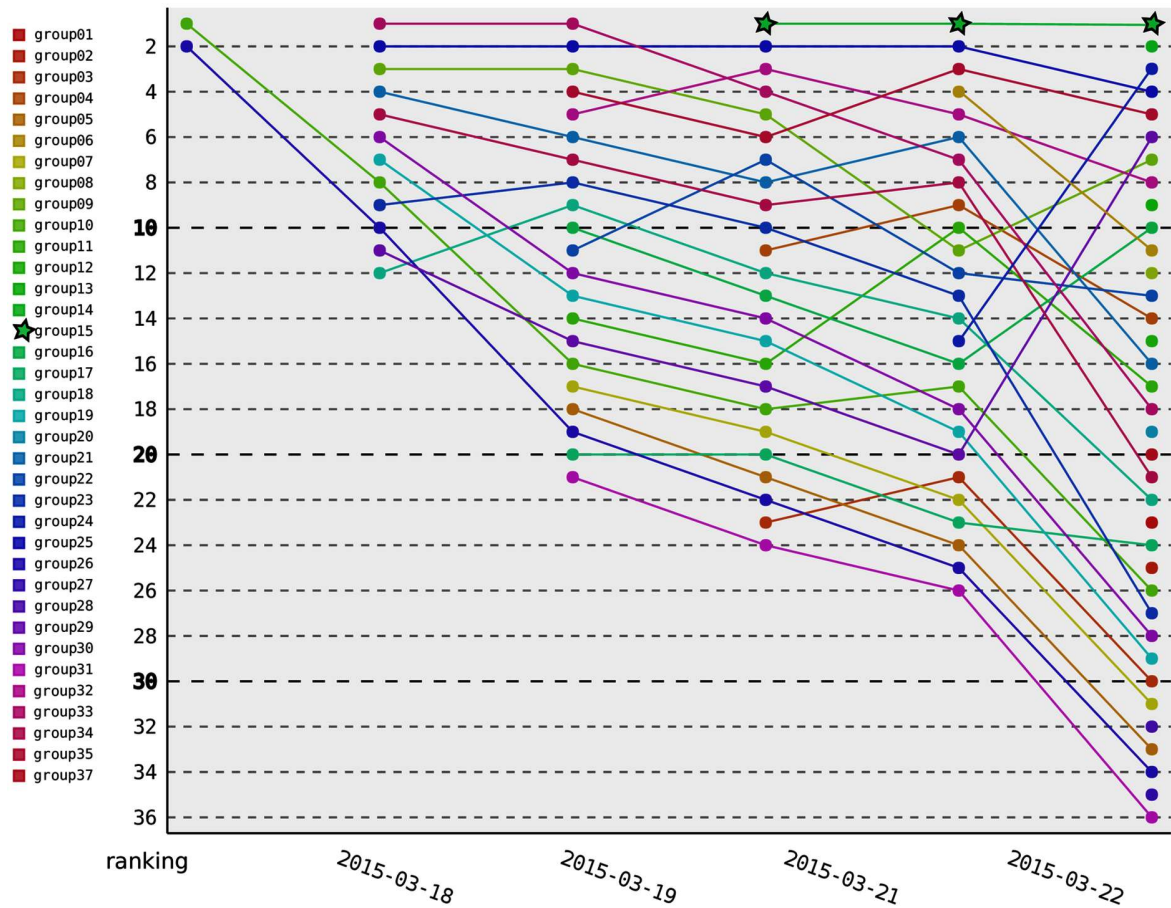
Table 3. Statistical measurements of the number of executions in Tablón before and after sending the first valid program to the leaderboard

	Leaderboard Accepted	min	max	average	std deviation
OpenMP	before	6	179	40.89	36.83
	after	0	205	34.17	50.94
MPI	before	6	233	63.13	51.00
	after	0	92	18.68	23.83
CUDA	before	1	530	159.33	107.86
	after	1	236	69.22	66.48

everyday, meaning that the students were constantly trying to improve their solutions, not only to get a better ranking but also to “defend” their earned position.

The particular case of the CUDA leaderboard can show us how the competitiveness has turned out to be an important element to keep the students motivated. The dates when the exercise was released were the previous weeks before finishing the course. Although its deadline also coincided with the work/practice deadlines of other courses, the students kept, and even raised, the work rhythm carried out in

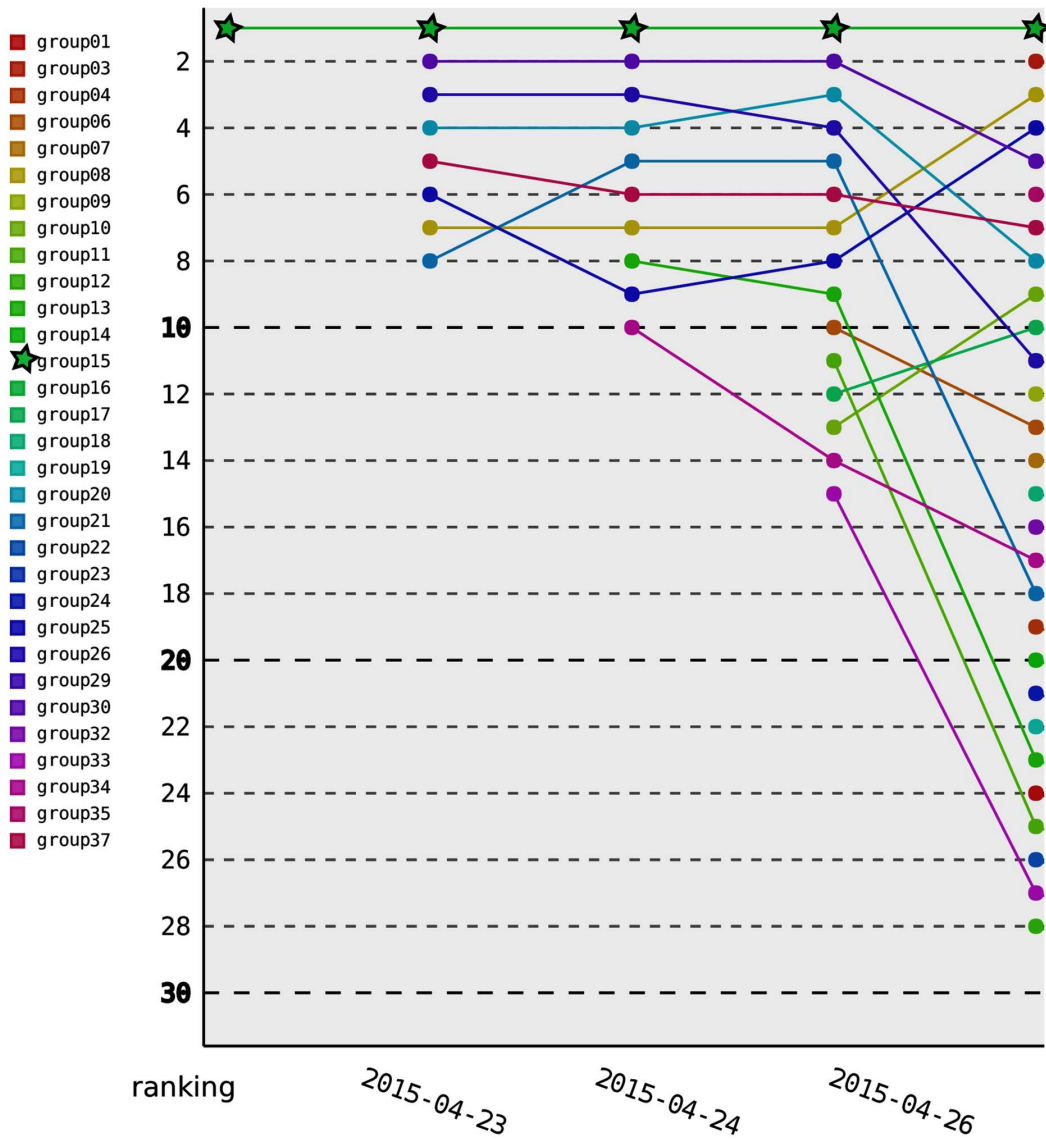
Figure 7. Leaderboard OpenMP



the rest of our practices (see Figure 4). Some subjective comments we have obtained from particular students were “we have so much work to do with the rest of courses, but we prefer to spend the time in something that we were really interested”.

We want also to highlight the particular case of the students of group 15. They were the group that ranked first position in the previous leaderboards (OpenMP and MPI). At the beginning they were also the first in CUDA leaderboard, with a great execution time. Once they achieved the first position these first days, they stopped improving and sending more versions. However, the day before the deadline, the group 37 achieved an even lower time, dethroning group 15 from the first position. Both groups had better times than the best reference provided by the teacher, and both also had close execution times between them. Thus, they knew their marks related to the leaderboard position and quality of the practice have already reached the maximum points. Nevertheless, group 15 started again to submit new solutions, hoping that in less than 24 hours they could take again the lead. The only motivation of being the first of the leaderboard supported them to program a more optimized version, which finally ranked them in the first position, even knowing that their grade was going to be the same.

Figure 8. Leaderboard MPI



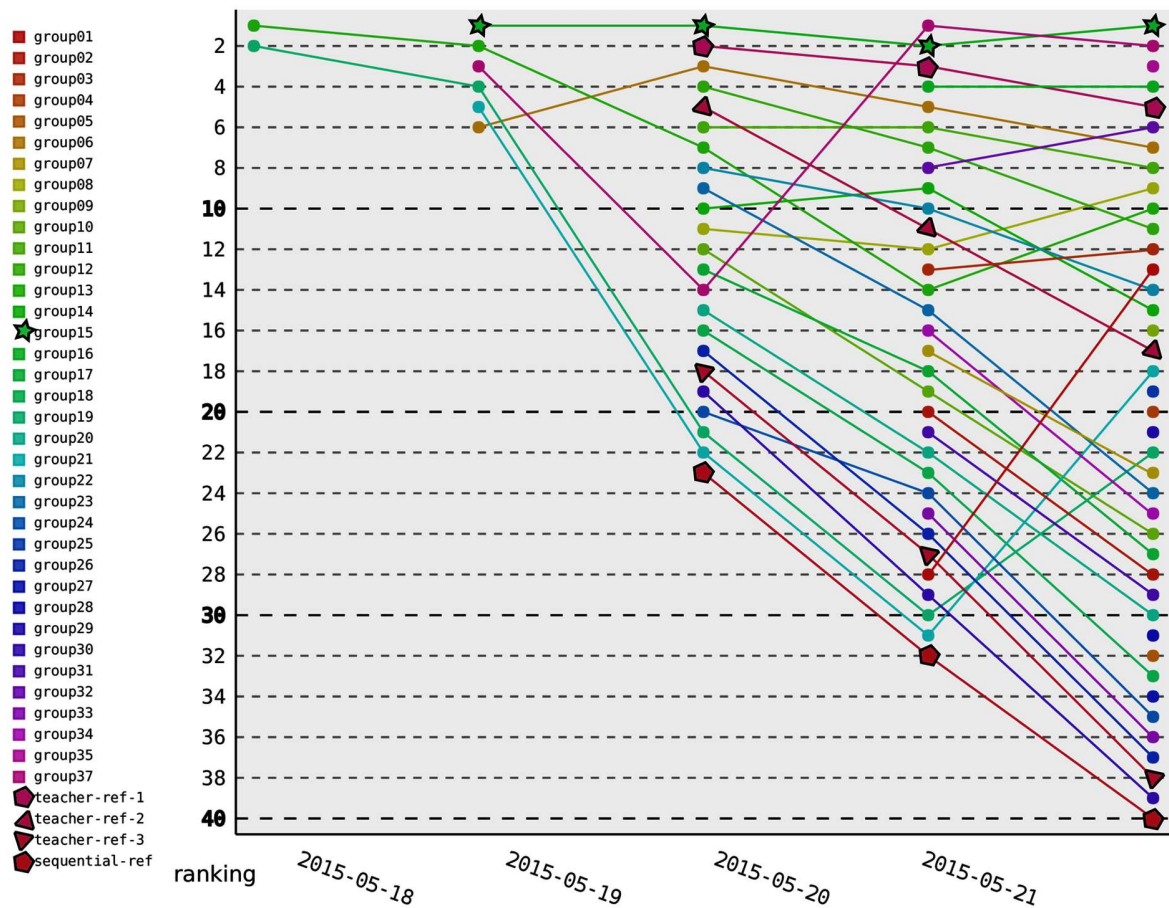
Study Results IV: Student's Satisfaction

The results of the official student survey done by our university were very satisfactory. For our Parallel Programming course, 31 of the total 75 of students completed this optional survey in the 2014-15 year. For all the questions available in the survey we analyze four of them which text is phrased as follows:

- The course was interesting
- I understood and learned the course material

Applying Gamification in a Parallel Programming Course

Figure 9. Leaderboard CUDA. The last four lines are the reference implementations developed by the teacher with different levels of optimization together with the sequential version.



- The teaching materials were well prepared and they have been explained carefully
- The workload is appropriate compared to other similar courses

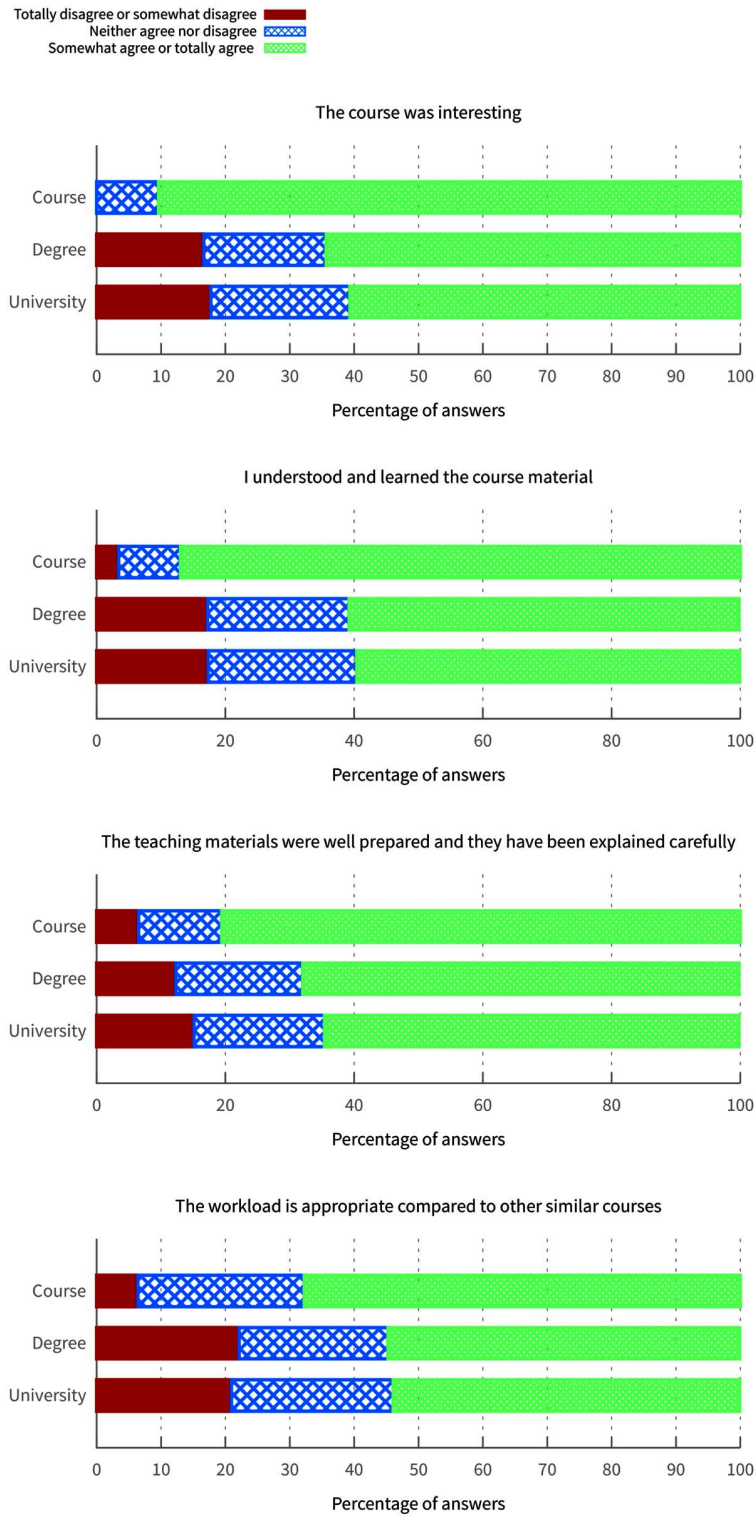
For each question the students have to answer to express their agreement with each question, using the following scale:

- Totally disagree
- Somewhat disagree
- Neither agree nor disagree
- Somewhat agree
- Totally agree

The survey allows us to compare the satisfaction of the students with our course compared to the average satisfaction of the degree, and with the satisfaction of the whole university.

Applying Gamification in a Parallel Programming Course

Figure 10. Students' survey results



Applying Gamification in a Parallel Programming Course

The results of the survey questions can be found in Figure 10. These plots show that our course had better results than the average of the degree and university. In particular, the students found the course very interesting, they understood the learning material provided, and they thought the materials were well prepared and well explained. Moreover, the workload was appropriate for this kind of course compared with others.

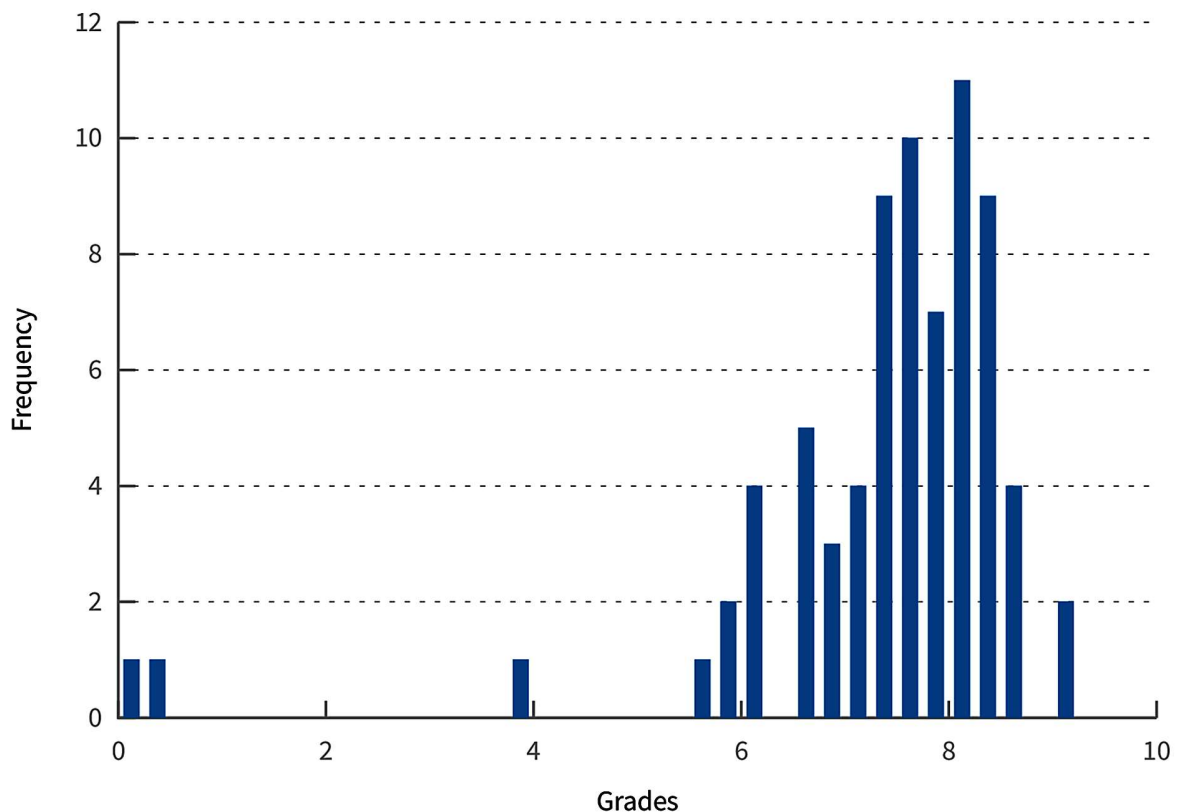
Study Results V: Student's Marks

Figure 11 contains the distribution of the final marks obtained by the students. With the exception of three of them who drop the course or did not take the final exam, all the students passed the course getting good grades. The average mark is 7.6 points. It is worthy to know that, although the groups are composed of two students, theoretical exams are carried out individually.

FUTURE RESEARCH DIRECTIONS

In this work we use different tools to support the gamification of the course: A Moodle system for the badges and Tablón for the leaderboards. An interesting future work would be to integrate both gamification

Figure 11. Final marks obtained by the students



techniques in Tablón. In this way, it would be possible to grant the badges automatically or interactively when the conditions are met.

Moreover, other possible extension of Tablón software is to add more information and statistics about the program executions. This information should be private for each group so it needs to be implemented using an account system.

Regarding the gamification techniques used, upcoming courses could be improved by adding other practices, such as quizzes, rating mechanisms, or appointment dynamics. It is also possible to enrich the current ones with more badges and different leaderboard behaviors.

CONCLUSION

In this chapter we have presented how we have gamified the Parallel Programming course we teach on Computer Science degree at Universidad de Valladolid. The main mechanisms we have included are the use of automatically-managed leaderboards together with several badges. The joint use of these gamification elements has delivered good results in terms of the student motivation, engagement, learning, and final grades.

The application of gamification mechanisms implies a higher effort done by the course instructors, compared to classical master lessons. This is due to the need of configuring the hardware resources, creating the corresponding software to handle them, and preparing exercises with enough elements that allow competitiveness. However, in light of the results of the studies we have presented, we believe that it is worthwhile to pay the cost of applying these techniques.

Our gamification framework for the Parallel Programming course provides more data and statistics not only of the final results of the students but also related to their behavior patterns. The study of these statistics gives also the possibility to monitor the activity of the students in real-time, to detect possible drawbacks or errors, and to promote the best and more profitable elements and activities for the next year.

Tablón software can be freely downloaded from <http://trasgo.infor.uva.es/>

REFERENCES

- Arnab, S., Lim, T., Carvalho, M. B., Bellotti, F., Freitas, S., & Louchart, S. et al. (2014). Mapping learning and game mechanics for serious games analysis. *British Journal of Educational Technology*, 46(2), 391–411. doi:10.1111/bjet.12113
- Aziz, M., Chi, H., Tibrewal, A., Grossman, M., & Sarkar, V. (2015). Auto-grading for parallel programs. *Workshop on Education for High-Performance Computing* (pp. 3:1-3:8). Austin, TX: ACM.
- Bunchball Inc. (2010). *Gamification 101: An Introduction to the Use of Game Dynamics to Influence Behavior*. White paper. Author.
- Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: defining gamification. *15th International Academic MindTrek Conference: Envisioning Future Media Environments* (pp. 9-15). ACM.

Applying Gamification in a Parallel Programming Course

- Dicheva, D., Dichev, C., Agre, G., & Angelova, G. (2015). Gamification in Education: A Systematic Mapping Study. *Journal of Educational Technology & Society*, 18(3), 75–88.
- Dicheva, D., Irwin, K., Dichev, C., & Talasila, S. (2014). A course gamification platform supporting student motivation and engagement. *International Conference on Web and Open Access to Learning (ICWOAL)* (pp. 1-4). IEEE. doi:10.1109/ICWOAL.2014.7009214
- Domínguez, A., Saenz-de-Navarrete, J., De-Marcos, L., Fernández-Sanz, L., Pagés, C., & Martínez-Herráiz, J.-J. (2013). Gamifying learning experiences: Practical implications and outcomes. *Computers & Education*, 63, 380–392. doi:10.1016/j.compedu.2012.12.020
- Fitz-Walter, Z. (2015). *Achievement unlocked: Investigating the design of effective gamification experiences for mobile applications and devices*. (PhD. Thesis). Queensland University of Technology.
- Grama, A., Gupta, A., Karypis, G., & Kumar, V. (2003). *Introduction to Parallel Computing* (2nd ed.). New York, NY: Pearson Education.
- Hakulinen, L., Auvinen, T., & Korhonen, A. (2013). Empirical study on the effect of achievement badges in TRAKLA2 online learning environment. In *Learning and Teaching in Computing and Engineering (LaTiCE)* (pp. 47–54). IEEE.
- Hamari, J., Koivisto, J., & Sarsa, H. (2014). Does gamification work?—a literature review of empirical studies on gamification. *47th Hawaii International Conference on System Sciences (HICSS)* (pp. 3025-3034). IEEE. doi:10.1109/HICSS.2014.377
- Ibanez, M., Di Serio, A., & Delgado Kloos, C. (2014). Gamification for Engaging Computer Science Students in Learning Activities: A Case Study. *IEEE Transactions on Learning Technologies*, 7(3), 291–301. doi:10.1109/TLT.2014.2329293
- Forum, M. P. (2015). *MPI: A Message-Passing Interface Standard Version 3.1*. Author.
- Moodle Learning Management System Webpage*. (n.d.). Retrieved 14, 2016, from <https://moodle.org/>
- Muntean, C. I. (2011). Raising engagement in e-learning through gamification. *6th International Conference on Virtual Learning ICVL*, (pp. 323-329).
- Navarro, C. A., Hitschfeld-Kahler, N., & Mateu, L. (2014). A Survey on Parallel Computing and its Applications in Data-Parallel Problems Using GPU Architectures. *Communications in Computational Physics*, 15(2), 285–329.
- OpenMP Architecture Board. (2013). *OpenMP Application Program Interface Version 4.0*. Author.
- Sanders, J., & Kandrot, E. (2010). *CUDA by Example: An Introduction to General-Purpose GPU Programming* (1st ed.). Addison-Wesley Professional.
- Simões, J., Redondo, R. D., & Vilas, A. F. (2013). A social gamification framework for a K-6 learning platform. *Computers in Human Behavior*, 29(2), 345–353. doi:10.1016/j.chb.2012.06.007
- Wu, T. (1998). The Secure Remote Password Protocol. *Internet Society Network and Distributed System Security Symposium*, 98, 97-111.