Abstract—Emerging technologies and increased computing resources challenge the universities to establish learning environments, where the students will learn to efficiently utilize multicore or multi-chip multiprocessors. In this paper, we present a new pilot project with the following goals: (1) as an e-Learning platform for the course on Parallel and Distributed Computing, and (2) as a benchmarking platform to measure the algorithm performance indicators such as speed, speedup and efficiency of the parallel implementation.

Index Terms—Education; Multi-processor; OpenMP; Performance; Remote Engineering.

I. INTRODUCTION

CPU vendors have changed the research and development directions in the last decade, that is, from pursuing the CPU clock speed towards building CPUs with multiple cores per chip, since the CPU clock speed increases the power consumption and heat dissipation exponentially. Many of today's multiprocessors are not just multicore, but even multi-chip multiprocessors. These innovations of establishing multicore and multi-chip multiprocessors raise the need for more relevant and efficient parallel algorithms. Although the parallelism topics are listed as elective, the IEEE Computer Society and ACM believe that the role of parallelism throughout the computer science curriculum needs to be considered [1]. Also, the parallel multiprocessor architectures are proposed to be covered even in the basic course Computer Architecture and Organization, rather than digital circuits. Many universities implement parallelization issues in other traditional (core) courses, such as Computer Architecture and Organization, Operating Systems, Pervasive Computing, and other courses [2]. Organizing regular contests will disseminate the parallelism among computer science students [3].

Learning parallel and distributed computing based courses requires usage of appropriate equipment, especially when students work on lab exercises and projects. Although students' desktop computers or notebooks are multicore based, there are a lot of obstacles. For example, these personal computers are usually installed with operating systems and inappropriate to support a parallel computing platform, such as OpenMP or MPI. The alternative to develop a new pilot e-Learning and benchmarking platform in Section V.

Learning the PDC course. Zhi et al. [9] designed an execution-driven parallel simulator for predicting the performance of large-scale parallel computers, simulating the processors of each node, network components and disk I/O components. However, we observed that none of these
simulators could be used in the teaching and learning processes, due to their complexity.

Breuer and Bader [10] designed a software package that supports teaching and learning different parallel programming models (OpenMP, MPI and CUDA). However, this package is useful for more advanced courses, rather than basic PDC.

Rague [11] introduced the Parallel Analysis Tool (PAT), a pedagogical tool, which assists the undergraduate students to transform a specific code section from sequential to parallel. The students can examine the impact of parallelizing a block of code at a time on overall program performance. Atanasovski [12] and Ristov [22] developed the EDUCache simulator and exercises with examples for it to simulate cache hits and misses for a particular memory trace in randomly designed multiprocessor. However, none of these simulators allow the students to start the learning process and incrementally develop more complex parallel algorithms.

Many authors proposed methodologies to lighten the learning. Liu et al. [13] developed software and application lab modules, which give the students clear understanding of the principles and practice of parallel programming on multicore systems. Bunde [14] propose an introduction of a small unit for multi-threaded programming in core courses. Michailidis and Margaritis [15] proposed a simple and fast analytical model to predict the performance of matrix computations by taking memory access costs into account and data access schemes that appear in many scientific applications. Delistavrou and Margaritis [16] survey the software environments for parallel programming categorizing them according to their main function. Freitas [17] proposed OpenMP (shared-memory model) to be used in the computer science curriculum before MPI (message-passing model), but after consolidating sequential programming, operating systems, computer architecture and networks.

Li et al. [18] improved the PDC course including the information about multicore needs in the learning materials, put emphasis on OpenMP and MPI, and involved programming practice. However, although these issues were already implemented in our PDC course, the students faced many other challenges and obstacles described in Section III-B. Iparaguirre et al. [19] evaluated that practical work, lab exercises and projects using the same runtime environment according to the results of their colleagues.

The focus is set on practical work. Homework is assigned both from the theoretical lectures and exercises on a weekly basis. Students are supposed to work on two projects, the first one using OpenMP and the second, MPI. After finalization they have to present the programs and results. The students must attend the theoretical lecture part of the course, defend both projects and finish all homework to pass the course. The hands-on lab exercises are not graded since they are intended as individual preparatory work. The students should not only develop a parallel implementation of a certain algorithm, but they should perform benchmark tests, that is, to measure the algorithm speed, speedup, efficiency, utilization, and so on.

B. Challenges and Motivation

The students faced several obstacles and challenges. Their main reasonable obstacle is the impossibility to continue to work on lab assignments at home if they had not finished all required tasks during the lab. However, the access to the lab is forbidden outside the University network, and all those that are not present in the University Campus face real problems. The students also faced problems while they were working on the homework assignments. Their notebooks or homework stations usually run on Windows operating system, or Mac OS, rather than Linux. The students can solve homework assignments using a virtual machine instance, but they might expect performance degradation in this case. From teacher’s perspective we have faced other problems, for example, we could not verify the homework correctness due to existence of different hardware infrastructure per student that is each student achieved different speed, speedup, efficiency, CPU utilization and so on. Another problem was raised when some of students tried to fabricate the results, tailoring the output according to the results of their colleagues.

All these challenges motivated us to develop a new web based e-Learning and benchmarking platform for learning PDC course, trying to solve or reduce the previously mentioned obstacles as much as possible. Using this platform, all the students are now enabled to work on their homework, lab exercises and projects using the same runtime environment and the same hardware infrastructure, appropriate for the course. Also, all their results can be compared with the teacher’s nominal expectations.
IV. THE E-LEARNING AND BENCHMARKING PLATFORM

This section describes our new e-Learning platform for PDC course, its main features and user interface on the front-end application.

A. The Architecture

Figure 1 depicts the architecture of the e-Learning and benchmarking platform. It is a client-server based platform independent system developed in JAVA. The platform consists of two servers: the Application and Testing servers.

The application server is Ubuntu server 12.04 and hosts the web application which is the front-end application accessed by the user. The application offers the students several possibilities. They can execute several predefined linear algebra algorithms, such as matrix-matrix multiplication, matrix-vector multiplication, dot or inner product and so on. The students can select the sequential or parallel implementation with certain number of threads, the matrix and the vector sizes. The application sends the parameters to the web service hosted on the testing server, which executes the chosen algorithm with the parameters and returns the results to the students. The next Section IV-C describes the front-end application user interface in detail. This server works also as a database server to store the students’ credentials. The same database is also used for storing the results of previous executions.

The most important subsystem is the testing server. It is also an Ubuntu 12.04 sever installed with necessary OpenMP and MPI libraries. The testing server hosts the web service and benchmark program. The web service is a gateway between the front-end application and the benchmark program. It accepts the parameters (entered by the students) from the front-end application and calls the benchmark program using the selected algorithm. Finally, it continues with execution of multi-threads within the OpenMP environment, or multi-processes within the MPI environment, or with sequential execution of a single process (thread). The benchmark program returns the results to the application through the gateway web service.

B. The User Interface

The e-Learning and benchmarking platform user interface is a user-friendly visual environment and uses the Multiple Document Interface (MDI) paradigm. The previous section described the architecture of the e-Learning and benchmarking platform. In this section, we describe the user interface of the front-end application. Figure 2 depicts the main menu after student’s successful login.

The main window contains 3 main panels described in details in the following three subsections.

1) Left Panel - The Learning Panel

The panel on the left side offers the students a possibility to select which algorithm they would like to analyze. This pilot platform offers three algorithms: matrix-matrix multiplication, matrix-vector multiplications and inner (dot) product. The application depicts the algorithm execution. Figure 2 depicts how the matrix-matrix multiplication algorithm works.

2) Central Panel - The Benchmarking Panel

The central panel offers the student a possibility to benchmark the selected algorithm in the learning panel. They can select either a sequential or a parallel implementation. The parallel implementation further offers a selection between OpenMP and MPI. A according to the selected algorithm and its implementation, this panel automatically shows the necessary parameters: the matrices’ or vector’s sizes and the number of threads or processes for parallel implementation. We implemented an additional field NLOOPS in order to repeat the execution NLOOPS times to achieve more reliable results. After clicking “START”, the application checks the database if the selected algorithm was executed with the same parameters previously. If there are results stored in the database, the application presents them immediately. This is usually the case on the lab exercises since all the students work on the same exercise. However, this is not useful when they work on their homework. There is a “REFRESH” button to execute the algorithm again even though the results are stored in the database.

After achieving the results of a web service (or loading from the database), they are presented in the lower part of the benchmarking panel. The application shows the average execution time of the sequential implementation, and additionally the execution time, speedup and efficiency of the parallel implementation. For example, we observe almost linear speedup of 2.36 using 3 threads with OpenMP multiplying matrices 100x200 with 200x100.

3) Right Panel - Learning and Benchmarking Panel

The right panel offers students a source code of the selected algorithm and certain implementation in C++. All algorithms are organized in functions and the parameters are transferred as arguments for the already compiled C++ testing program. The built in example in the main program generates a random matrix of vector elements, measures the average execution time and checks the correctness of the result.

Inappropriate usage of OpenMP directives can lead to a possible execution slowdown or even wrong execution results of the parallel implementation, as presented by Li et al. [20]. Therefore we introduce this panel where students can get the code example in order to check the correctness of the results. Presenting the expected speedup also will guide the students if their directives are written well, that is, if they achieve the expected speedup.

C. Sequence Diagram for Test Execution

Figure 3 presents the sequence diagram of the e-Learning platform in terms of how the student executes a particular test. In order to login at the web page, the student must first insert login credentials. These credentials
are then passed to an external authentication system, which authenticates the user and authorizes user access for the web application.

After successful authentication the user can request tests by passing test information such as algorithm type, benchmarking algorithm type (sequential or parallel, OpenMP or MPI) and benchmarking parameters such as matrix or vector sizes, the number of threads or processes, and number of test repetition. Based on the test parameters the application queries the database for previous executions. If the test result data is found, it is passed back to the user; otherwise a special web service is invoked. This web service requests OpenMP test execution on one of the configured servers. After the OpenMP test finishes, the test results are sent back to the user through the web service and these results are written in the database.
V. Conclusion and Future Work

This paper presents the new pilot project for e-Learning and benchmarking platform for learning the PDC or other related courses. The contribution of this platform is twofold: 1) it allows the students a possibility to learn parallelization concepts and parallelize more complex algorithms; and 2) to benchmark the algorithm execution with a particular problem size and check if it achieves the expected performance indicators speed, speedup and efficiency for parallel implementation.

The students can work remotely on the same hardware infrastructure and runtime environment and compare their benchmarking results for a particular algorithm with a certain problem size. This is very important since the memory access time, cache misses and hits, and also cache associativity affect the algorithm overall performance.

Although this platform reduces the students’ problems, we faced a problem even at the beginning. This is a situation when two or more students execute the benchmark programs in the same time using higher number of threads or processes, greater than the server can stand for. Most of the results in this case become unreliable. Therefore, we will use the cloud based architecture of the e-Assessment system tool in order to upgrade this e-Learning and benchmarking platform prototype into a scalable and elastic platform, where the system will send the execution on a virtual machine instance hosted on the cloud node with available resources.

REFERENCES


AUTHORS

Marjan Gusev, Sasko Ristov, Goran Velkoski, and Bisera Ivanovska are with “Ss. Cyril and Methodius” University, Skopje (Faculty of Information Sciences and Computer Engineering, Rugjer Boshkovik 16, PO Box 393, 1000 Skopje, Macedonia). Email: {marjan.gusev, sasko.ristov}@finki.ukim.mk, velkoski.goran@gmail.com, ivanovskabisera@yahoo.com

This article is an extended and modified version of a paper presented at the 2013 IEEE International Conference on Teaching, Aessment and Learning for Engineering (TALE2013), held 26-29 August 2013, Bali Dynasty Resort, Kuta, Indonesia. Submitted 29 September 2013. Published as re-submitted by the authors 20 March 2014.