International Conference on Computational Science, ICCS 2010

Learning parallel programming: a challenge for university students

Ronal Muresano¹, Dolores Rexachs¹, Emilio Luque¹

Universitat Autonoma de Barcelona
Computer Architecture and Operating System Department, Barcelona, SPAIN, C.P 08293

Abstract

Currently, the need to learn parallel applications topics in students has become an important issue due to the rapid growth in the parallel computing field. In fact, this topic has been included in Computer Science curriculum, but students present difficulties to design MPI parallel applications efficiently. We present a novel methodology for teaching parallel programming centered on improving parallel applications written by students through their experiences obtained during classes. The methodology integrates theoretical and practical sections which are focused on teaching two parallel paradigms, master/Worker and SPMD. These paradigms were selected due to their different communication and computation behaviors, which generate challenges for students when they wish to improve performance application metrics. Our methodology allows students to discover their own errors and how to correct them. In addition, students analyze the issues and advantages in the application designed in order to enhance the performance metrics. Applying this methodology gave us a significant progress in parallel applications designed by students, where we have observed an improvement of around 47% in the students’ skill about parallel programming when they design parallel applications.

Keywords: Active Learning, Parallel Programming, Methodology, Performance Metrics.

1. Introduction

Nowadays, the scientific applications are designed with more complexity and precision, in order to obtain more accurate results. Nevertheless, when complex algorithms are designed, we need to include parallel environments to obtain a fast application execution. In fact, parallel application development has been included in different areas such as: biology, physics, economics, engineering, etc. [1], and this has meant that parallel programming courses have had to be included in the teaching curriculum of diverse knowledge fields in order to improve student skills and cover the need to learn this topic. [2].

Indeed, the need for students to learn parallel application topics is growing. However, students have some programming problems when they wish to achieve an improvement in parallel application performance metrics. One of the difficulties for students is to change their previous programming knowledge, which is focused on designing
sequential applications. This focus is totally different when parallel applications are programmed, since parallel application tasks have to be divided and distributed among the processes, and these processes are executed in a group of available computational resources.

Other issues for students designing parallel applications are related to: tasks distribution, mapping strategies and scheduling policies to execute the parallel applications. Students have to manage these issues in order to enhance the performance metrics in their applications.

Given the above, parallel programming courses have to incorporate teaching strategies that allow students to solve their programming issues. They have to design parallel applications with the objective of executing these efficiently. For this reason, we have designed a novel methodology for teaching parallel programming, which allows students to develop parallel applications through the experiences obtained during classes. The main goal of this methodology is to obtain a constructivism learning process of parallel computing, where students can design their applications considering performance metrics such as: execution time, speedup and efficiency.

This methodology includes theoretical and practical classes which are distributed among the course content. The theoretical classes define the concepts necessary to develop a parallel application. Also, these classes enable students to obtain a relationship between the parallel performance metrics defined and the influences of the parallel application over a parallel environment.

Moreover, the practical sections are addressed to show the way to parallelize an application efficiently. To design these classes, students have to evaluate parallel application performance and they have to develop two parallel applications using different parallel paradigms. We are focused on teaching the most widely used paradigms in computational science: one of them is master/Worker and the other is SPMD (Single Program Multiple Data). Both paradigms have different communication and computation behaviors and students have to consider these behaviors when they wish to achieve an improvement in parallel performance metrics.

Hence, the way to parallelize an application depends on how tasks are distributed among the resources. These distributions have to be clearly explained to the student in order to take advantage of the strategies of our methodology. For example, the master/Worker paradigm uses a master which is responsible for dividing the workload among workers. Furthermore, the workers have to execute the tasks assigned and return the results to the master once they are finished [3] (Fig.1(a)). On the contrary, the SPMD pattern is to execute the same program in all processing elements, but with a different set of data tasks [3]. SPMD applications are synchronized and they have high communication volumes in which several iterations are repeated (Fig. 1(b)).

Each of these parallel paradigms has a different approach and students have difficulties with diverse issues related to how these paradigms behaved. Two examples are shown in figure (Fig.1(a)),(Fig. 1(b)), where we can observe two of the most frequent problems with both paradigms. For example, in the master/Worker paradigm a computational imbalance factor is found among nodes. This is the most common error presented by students when they selected
this paradigm. However, the SPMD paradigm shows more frequent design problems related to the communications unbalance factors due to the great communications number between neighbors and the students have to manage when a heterogeneous communication environments as Multicore are used. The diversities of these parallel paradigms make their issues and advantages more interesting and challenging when these paradigms are taught to students.

Taking advantage of the problems presented by students when they are designing parallel applications, our methodology uses strategies which allow students to design an active learning process which integrates them into the teaching of parallel programming classes. This learning process enables students’ constant class participation, in which parallel application issues are discussed. Our teaching method is designed to change the traditional lecture method applied in computational classes. We mainly take into consideration the student interactions and their parallel designed contributions.

In order to achieve our objective, the methodology is composed of three phases in which students can identify their own parallel programming problems. The phases are defined as: parallel design problem identification, application performance evaluations and the parallel application improvement. These enable students to develop parallel applications efficiently.

The main function of the parallel design problem identification phase is to permit students to create a first version of their parallel application with the objective of discovering the most relevant issues about the application they have designed. Then, this phase allows us to reinforce performance topics and we can explain to students how to fix the trouble in order to obtain an improvement in the parallel execution.

Next, we start the second phase of our methodology which is defined as application performance evaluations. This phase permits students to evaluate the computational and communications parameters in both parallel paradigms. Then, we take in to consideration efficiency, speedup and execution time as performance evaluation metrics. Next, students analyze their results obtained and they propose the performance improvement which will be applied in the last version. Finally, the phase of parallel application improvement enables student to apply their proposed enhances, and then they have to expose their final results. Also, this phase includes solutions for students which permit them to solve and improve their last parallel applications version. A final comparison is made by students, when they finish the first and the last versions.

The main contribution of our methodology is to allow students to generate new points of view about how to design parallel applications with their own experiences obtained during classes. This methodology was applied for two years in computer engineering classes to students of the Autonomous University of Barcelona and also it was included in the High performance computers subject of the computational science and engineering Master’s Degree which is taught by the Computer Architecture and Operative System Department (CAOS) [4].

This paper is structured as follows: the parallel programming course content is described in section 2. A description of the methodology is presented in section 3. Section 4 describes performance evaluation, and finally conclusions are given in section 5.

2. Parallel Programming Course Content

In recent years parallel programming has increased significantly in different knowledge fields [2]. This increase has motivated universities to design parallel programming course content according to the necessity for students to be able to develop parallel applications in an efficient manner. Moreover, parallel programming is an applied technique which allows students to solve their computational challenges focusing on optimizing the computational resources. In addition, this type of programming is acquiring an ever increasing acceptance and it is applied to solve high computational performance scientific problems in diverse fields. The trend of using parallel programming is thanks to its achieving more accurate results, highest performance, lowest cost, and lowest execution time [5].

The objective of this parallel programming course is that students can learn how to distribute a serial application toward a parallel environment while considering the performance metrics. The complexity of dividing a serial program into a parallel one could generate confusion in students when they wish to enhance the application metrics. However this complex transformation has been obtained through experience acquired in the classroom, where the main problems related to parallel applications presented by students have been the workload distributions and communications managing.

Given the above, we designed a course content and methodology allowing students to interact and express their doubts to the instructor in order to improve their knowledge in the parallel computing fields. To achieve this, we
apply techniques to explain to students how to develop an application efficiently. These techniques are centered on teaching the parallel tools which allow students to design and monitor parallel applications. Also, these techniques enable students to consider the effects of the parallel architecture and how the application could be programmed in order to achieve an improvement in the performance of the parallel machine.

Table 1 shows the parallel programming course content which is organized according to the learning students needs in this topic. This course starts with a description of parallel computers. Unit 1 shows the importance of diverse parallel machines to students. We describe different parallel architectures in the top500 list. These descriptions are centered on explaining the components which could improve the performance metrics in a parallel application. An example of the descriptions are related to cache memory, interconnection network, memory hierarchy, etc.

Next, unit 2 is included to give details about interconnection networks and how the messages in parallel applications can travel through the links. This unit allows students to describe the different paths and links which are presented in the parallel machines, for example in a Multicore architecture, the communication process could be made through cache memory, main memory or through network links. Unit 3 explains the parallelism inside the nodes and how the parallel application should be designed in order to benefit from this topic. Once units 1, 2 and 3 are finished, the students have to select a parallel machine and they have to prepare an in depth analysis of the characteristics of the architecture selected. They have to explain the most relevant characteristics according to units 1, 2 and 3 and how these characteristics have to be managed in order to improve the parallel performance. This presentation gives students an initial approach to parallel architecture.

Then, students begin to learn the parallel programming language and the parallel paradigms (unit 4,5). In these two units, students have to learn about the message passing interface libraries and the parallel paradigms which they will use in their application design. These paradigms are SPMD, master/worker, pipeline, divide and conquer, etc. However, we mainly focus on SPMD and master/worker since these paradigms are the most widely used in the parallel computing field [6]. Next, students have to be able to understand the parallel issues and which of these issues have occurred in their application designed in practical sections. Problems relate to type of paradigm, communication patterns, communication overheads, mapping, scheduling, load balancing etc, which can worsen the parallel application performances.

After this, students can see some of these issues in the practical sections and they have to be able to correct their mistakes through the strategies and techniques explained in classes. Next, units 6 and 7 teach students how they have to apply the performance metrics concepts in order to improve their parallel application. Then, students have to evaluate and to enhance their performance results.

The goal of this course content is to explain to the students that parallel programming is not only to divide a serial program in a number of parallel processes. The objective is to permit students to obtain some criteria which allow them to evaluate the performance and identify the problems. The figure 2 shows the goal of this course content which is divided in three phases: design problem identification, application performance evaluations and parallel application improvement. These phases are the focus of our methodology and these allow students through active learning to design their parallel applications with the concepts taught in classes.

This course can be considered successful, when students propose and apply the strategies which allow them to enhance the parallel performance metrics. The evaluation process is made through programming assignments which students have to present at the end of the course. Two applications under master/worker and SPMD paradigm have to

---

1) Parallel Computer
2) Interconnection Network
3) Compute Node Parallelism
4) Parallel Programing Introduction and Paradigms
5) Languages, Tools and Fault Tolerance for parallel programing
6) Performance Metrics Evaluation
7) Parallel Efficient Application Management

Table 1: Parallel Programing Course Content Topics

---

2TOP500 is a list which provides a rank of the parallel machines used for high performance computing www.top500.org
be developed. These applications are given by the instructor with the aim of obtaining uniform groups.

3. Proposed Methodology

Our parallel programming teaching methodology is focused on creating an environment where students can achieve significant learning through their own experiences obtained during classes. This methodology is composed of three phases: parallel design problem identification, application performance evaluations and parallel application improvement (Fig. 3) and these phases allow us to have active interactions with students.

The methodology integrates theoretical and practical classes where students can learn and practice the concepts of parallel programming. In practical sections, the students can apply the concepts taught in theoretical classes through assignments. Also, these practical classes permit students to discover common errors in parallel programming design and how they can to fix these issues. Figure 3 illustrates the phases of our methodology and the processes related to how these phases could be applied in students’ learning processes. These phases are designed to permit students to develop different point of views and to create some criteria related to how to design parallel applications applying their own experiences.

3.1. Parallel Design problem identification phase

The main objective of this phase is to permit students to learn what can be the problematic issues in parallel applications and how these could be solved in order to enhance the performance metric. In this phase students have to design two parallel applications under different parallel paradigms (master/Worker and SPMD). To develop these applications, students have to apply the concepts learned in theoretical classes, next a first draft of both parallel applications are obtained. Then, students and instructor have interactions where issues and the program problems are discussed in order to improve the parallel applications. The instructor has to identify the students’ needs in their parallel application and he/she has to reinforce the concepts with the aim of the student being able to fix the problems.

From the errors detected in applications designed by students, the instructor has to discuss some strategies to analyze the applications correctly. This phase is finished when students can list the troubles included in their applications. In this part, the instructor has to determine the weak points of the students’ applications. Finally, the instructor has to use the most relevant issues and he/she has to propose the strategies necessary for each student so they can in turn improve their parallel applications performance.

3.2. Applications performance evaluations phase

The main objective of this phase is to permit students to carry out an evaluation of the performance metrics in each parallel application designed and in doing so they discover the influences of their parallel design issues in the performance metrics (Fig 3).

Students have to analyze four aspects related to the parallel applications and the computational environment where these applications are executed. This analysis allows students to determine the different communications links, the
computational and communication patterns and the behavior of the application inside the parallel environment. These evaluations are done through a characterization module which allow students to obtain a communication and computation parameters of the applications. These characterization module is explained to students using a methodology exposed in [7]. Once students finish this characterization, the performance metrics are calculated. The efficiency, speedup and execution time are studied and evaluated by students in both parallel applications. These evaluations allow students to determine if their parallel executions have to be modified to enhance the performance.

After these evaluations, the instructor and students have interactions in which the performance metrics are discussed and analyzed by them. Thus, in this phase the students have to explain their applications and their errors. The presentations are centered on evaluating the issues of all students and the instructor can determine if the problems for example are related to mapping, load balancing, imbalanced communications, etc. Finally, the instructor gives feedback about the presentations and recommends the changes that students have to carry out in order to improve the next parallel applications. Also, students through their own experiences can propose some changes which are discussed with the instructor. These interactions permit students to have an active learning process through their experiences obtained in labs. Students will apply their changes in next phase with the aim of obtaining an efficient parallel execution.

3.3. Parallel application improvement phase.

The aim of this phase is to analyze how to improve the performance metrics in the parallel application versions of the students. This improvement is achieved through the union of the considerations of the instructor and students, and also with the problems detected in the first phase of this methodology. Then, the proposed improvements are applied to the new versions of both applications which are tunned to the cluster architecture using mapping and scheduling strategies in order to improve the performance metrics [7]. The instructor follows a checking process in order to verify that students do not have troubles applying the proposed changes.

Next, a new performance evaluation has to be done by the students. This evaluation permits students to analyze the proposed changes and the influences of these changes in the new parallel application versions. Then, the students have to evaluate if they in fact have achieved an improvement in the parallel application metrics. The last activity is the oral presentation of their applications. This activity enables students to discuss their parallel applications results.
The objective of this activity is to realize an analysis of the methodology application and the result obtained with all students. The interactions allow students to have a feedback which is explained in a final writing work.

As mentioned before, our methodology attempts to develop in students the strategies to solve a parallel problem efficiently. An important element of our methodology is that it allows students to learn through their own mistakes and experiences and learn how to solve them efficiently.

4. Methodology Evaluation and Experience Obtained

The parallel programming course has been taught for some years in the Universitat Autnoma of Barcelona. This course is a course where an important number of students register every year. This methodology was applied during the last two years of the computer engineering students, and also it was included in the High performance computers course of the computational science and engineering (CSE) master. This master is an innovative course model allows students without a computational science background to learn parallel computing [4].

This methodology has been applied to fifty six (56) students during 2008-2009 periods. These students were divided in the following manner 46 from computer engineering and 10 from CSE master. Additionally, these students groups were evaluated with our methodology using two applications. One of them was the matrix multiplication which is designed under master/Worker paradigm. The other application was the heat transfer application under SPMD paradigm. Both applications have different behaviors which generate challenges and issues to students when they wish to obtain an improvement in the performance metrics.

4.1. Design problem identification phase evaluation

The objective of this evaluation is to show the students the most frequent errors that they included in their parallel application. Figure 4(a) shows an execution trace of the matrix multiplication application designed by a student group. This trace illustrates an example of the workload imbalanced, where some processes end their execution in different execution time. This imbalanced generates that performance metrics worsen. Other example is shown in figure 4(b). This is the execution trace of heat transfer application presented by other student group. In this case, we can observe the communication imbalanced problems. Both trace are obtained through a tool of performance evaluation called jumpshot. The most common troubles observed were workload division, locality, static and dynamic tasks division, few overlap strategies etc.

Besides finding these errors in parallel application designed by student, we obtained some error related to other parallel topics such as: mapping strategies used, scheduling, communication number among tasks, etc. These errors were detected through the interactions between instructor and students. After these interactions, the students start to evaluate the performance metrics in the first draft of their parallel applications in the next phase.
4.2. Applications performance evaluations phase evaluation.

Once the issues are detected, we have to teach the students the performance parameters and how they have to do in order to evaluate them. Then, students prepare an exposition of their first application design. Figure 5(b) shows an example of the efficiency obtained by student in the heat transfer application for a specific problem size. This figure illustrates that 20% of students obtained efficiency over a 60%, but the 80% of students obtained values under 60% of efficiency. These efficiency values allow us to give details to students about where they have presented their parallel application issues. Also, we can explain how to fix these issues in order to enhance the performance applications.

Another efficiency example is shown in figure 5(a), where we can observe the efficiency obtained in some student groups. We selected five of these groups and we evaluated the efficiency obtained. This figure gives us an idea about the heterogeneity of these student groups, where some of them have obtained an application efficiency value around 80% and others have got application efficiency around 25% for the same application. These differences allow us to justify the instructor interaction due to the group heterogeneity.

4.3. Parallel application improvement evaluation.

The last evaluation is related to the parallel application improvement designed by students. In this phase, the instructor has explained and has proposed some changes for the new parallel application. For example, mapping strategies are discussed with the aim of allocating the processes into each node and the scheduling which allows student to apply overlapping strategies. This strategies allows students to overlap the computations and the communications[7]. The instructor’s strategies are applied by student and they obtained an significant improvement in their parallel execution.

The figure 6(a) shows the efficiency obtained by students after the changes discussed by instructor through the issues found in their parallel applications. This figure allows us to demonstrate the improvement in the parallel programming class, where 67% of students obtained an efficiency over 60% and only a 6% are under 40% of efficiency. These result evidences the improvement in students’ skills when we analyses both students application versions.

Finally, the figure 6(b) illustrates an example of the improvement obtained in the class. This figure shows a comparison of the first and the last version of a student group, where the efficiency was improved around 80% over the original value.

4.4. Experience Obtained.

Once the methodology was applied, we obtained a significant improvement in students’ skills in designing parallel programs. This improvement was due to the active learning process in which students’ opinions are very important to evaluate the success of this methodology. Our methodology is focused on allowing students to learn through their experiences obtained during classes, and these experiences are achieved through the issues presented in their parallel applications.
Each class is totally different and the students have a mixed knowledge level. For these reasons, we have to consider the heterogeneities in the students, and these heterogeneities have to be managed by the instructor in order to obtain significant learning in the class. The improvement strategies are dealt with in two ways, firstly considering the needs of each student, and secondly the other strategies are taught to all the class. The success of these strategies mainly depends on the interaction between instructor and students. This methodology was successfully applied in university students, in which they acquired a significant improvement in their parallel application metrics as was demonstrated in our evaluation.

5. Conclusion

This paper has presented a methodology for teaching parallel programming in computational science students. The objective of this methodology is to allow students to develop efficient parallel applications through their experiences obtained during class. We demonstrated a considerable improvement in students’ ability to cope with performance applications metrics. The main difficulties that students had were communication patterns, communication overheads, mapping, scheduling, load balancing etc. These difficulties were successfully resolved by students through the active learning applied with our methodology. This methodology is actually used to teach the parallel programming course at the Universitat Autnoma de Barcelona.

Acknowledgment

This work has been financed by the MEC-Spain under contract TIN2007-64974.

References