Activity Based Approach for Teaching Parallel Computing: An Indian Experience

P.Chitra [1]
[1]*Department of Computer Science and Engineering*
Thiagarajar College of Engineering,
Madurai, India
pccse@tce.edu

Sheikh. K Ghafoor [2]
[2]*Department of Computer Science*
Tennessee Tech University
USA
sghafoor@tntech.edu

***Abstract*—Due to the rapid growth in the multicore and GPU based computing devices, the need to teach parallel computing in CS/CE curriculum has become almost mandatory nowadays. A course on Parallel Computing Systems (PCS) has been designed to provide an understanding of the fundamental principles and engineering trade-offs involved in designing modern parallel computing systems as well as to teach parallel programming techniques necessary to effectively utilize these machines. An activity based learning approach was adopted for teaching the course and several parallel programming paradigms and technologies such OpenMP, MPI, and CUDA have been covered. This course was offered as a required course to graduate students. This paper describes the implementation of the course at Thiagarajar College of Engineering. Evaluation of the implementation of the course reveals that for students who have not been exposed to parallel and distributed computing, i) activity based learning results in better knowledge gain compared to the traditional approach, ii) learning OpenMP was much easier than MPI or CUDA, iii) some Parallel and Distributed Computing (PDC) concepts such as false sharing were harder to grasp compared to basic concepts, and iv) it is essential to introduce parallel computing in the undergraduate curriculum.**

I. Introduction

Nowadays, every computing device starting from tablets to high-end-servers and even mobile phones contain multiple cores and in some cases GPU capabilities. It is common to see processors with 4 cores, and soon it is expected to have up to 10s and 100s of cores [1]. Scientific and engineering applications are designed with more complexity and precision to take advantage of these new computing devices or to obtain accurate results faster. Apart from science and engineering applications, parallel and distributed computing is making its way more and more into commercial and business applications too.

Microprocessor manufacturers and hence the computer hardware is going in the direction of increasing multiplicity of components executing in parallel rather than just increasing the speed of the individual devices [2]. Also, the heterogeneity of the components in this hardware adds complexity to parallel computing. This heterogeneity requires an extra effort by the programmers to harvest the high performance out of the emerging machines. The advancements in recent topics such as big data processing and General-Purpose Graphic Processing Unit (GPGPU) computing highlight the use of high-performance computing platforms [3]. This changing computing landscape dictates that every computer science and computer engineering student should be trained in parallel and distributed computing to some extent. This necessitates that the existing CS/CE curriculum must include the topics of parallel and distributed computing to enable the future generation of computing workforce to take advantage of the new computing landscape.

To introduce parallel computing to students, we have designed a course on parallel computing systems to offer to the first semester graduate students of the Computer Science Engineering Department at Thiagarajar College of Engineering (TCE), in the state of Tamilnadu, India. This course covered both parallel hardware and software design, and key machine performance characteristics. Up until 2018 the course was conducted following traditional lecture and homework based approach. In 2018, the course delivery was redesigned using an activity based learning approach. The course content and targeted course outcome remained unchanged. Adopting active learning based methodology gave us significant progress in parallel applications designed by students, where we have observed that the course outcomes are significantly met at the end of the course.

The rest of the paper is organized as follows: Section II provides a brief overview of Indian higher education system and Thiagarajar College of Engineering. The related works are discussed in Section III. The course design, delivery, and evaluation methodology of the Parallel Computing course offered at TCE are described in Section IV and V. Conclusion and future works are presented in Section VI.

II. INSTITUTIONS DESCRIPTION: INDIAN UNIVERSITIES

Public universities and colleges in India are funded and governed by both central and state governments. Few institutions such IITs (Indian Institution of Technology) and NITs (National Institutions of Technology) are funded by central government. They are autonomous and set their own curriculums. There are two types of state institutions funded by state governments: i) big autonomous universities that set their own curriculums, as well as set and control the curriculums of affiliated private four year colleges; ii) smaller autonomous

public colleges that control their own curriculum. Some of the state public institutions are also funded by the central government in addition to the state governments. The admission to all the state institutions is controlled by a very competitive common entrance exam defined and administered by the state governments. The curriculum change in state institutions is a very time consuming, complicated, and bureaucratic process. An even minor change requires a long time and has a multi-step approval process.

Thiagarajar College of Engineering (TCE) is a 61 years old autonomous college, funded by both the central government and state government of Tamilnadu. The College offers 8 undergraduate and 13 Masters of Engineering degree as well as doctoral degrees in Engineering, Science, and Architecture. There are total 4300 students and 280 faculty members in the college. It is a highly reputed institution in southern India and is ranked in the 39th place in National Institute Ranking Framework (NIRF) during 2018. The Computer Science and Engineering department of TCE offers undergraduate, Masters of Engineering, and doctoral degrees. The department offers the Masters of Engineering degree under three tracks namely: Systems, Technology, and Theoretical Computer Science. The total number of students in in the department is approximately 700 for undergraduate and 50 for graduate.

III. RELATED WORKS

Different strategies to introduce parallel and distributed computing in the CS curriculum have been reported in the literature. In [4], the authors present their effort to implement parallelism in first and second year CS courses. The authors have reported that the students learned the material and had a good experience. Adams [5] suggests that CS2 is the natural place to introduce parallelism and the author uses patterns, called patternlets, to teach the students in CS2. Geist et. al [6] has suggested teaching the PDC as a stand-alone course during senior year.

Active learning methods have been adapted in many disciplines of education and the field of computer science and engineering is no exception. Active learning is considered as anything other than passively listening to lecture, where students match the learnings to real life situations [7]. One other type of active learning strategy suggests that the students must be allowed to talk about what they are learning, relate it to their real life experiences, and write about it [8].

Giacaman [9] points out that in a rapidly changing field as computer science, it would be beneficial if the students are provided with an interactive learning environment. Active learning includes many of the instructional methods, such as mini projects, role play, cooperative learning, etc. In this era, where smart phones have taken an important place in our day to day life, Yamaguchi et at [10] demonstrates how mobile phones can be used to broaden and enhance the use of active learning in large classrooms. Yamamoto and Wakahara demonstrated [11] the use of digital video assignments as an active learning tool.

IV. PARALLEL COMPUTING SYSTEM COURSE AT TCE

The course Parallel Computing Systems (PCS) is offered to the graduate students during their first semester at TCE. The PCS course is offered under the systems track as a required course and is an elective for the other tracks. The course is a semester long course with 3 credits. The class size is about 25. In 2018, the course has been redesigned to adopt active learning strategy. The students admitted to TCE are from both rural and urban backgrounds and form a heterogeneous group. The course presumes that the students have prior knowledge of computer architecture and operating systems at the undergraduate level. The curriculum is supplemented through programming assignments and mini projects to reinforce the learning. The course has a strong emphasis on hands on programming, such as lab exercises, homework, programming assignments, and mini-projects. The major course outcomes of the course are defines as: After completion of the course the students should be

- CO1: Able to select the appropriate parallel programming model for the given application.
- CO2: Able to apply the constructs of parallel programming technology to convert a sequential program to a parallel program.
- CO3: Able to design protocols for ensuring cache coherence using the directory based and snooping class of protocols.
- CO4: Able to develop parallel programs using OpenMP and MPI constructs.
- CO5: Able to characterize the benefits of using a GPU versus CPU for a typical parallel application.

A. Course Details

The topics for this course have been selected to give a breadth of knowledge to the students. At the beginning of the course, topics such as why we need parallelism, parallel programming models, parallelization process, and role of multi-core in thinking parallel are discussed. The communication and synchronization under different parallel programming models, the shared memory architectures and the associated coherence issues, protocols, global and point to point synchronization approaches are discussed. This is followed by introducing students to different parallel programming paradigms such as OpenMP, MPI, and GPU programming using CUDA. In addition, each paradigm is taught with a focus on practical applicability and matching recommended PDC topics. Table I lists the weekly plan of the course along with the alignment of the topics with the course outcomes.

B. Course Delivery Methdology

The main philosophy of the class was to engage the students through active participation rather than traditional unidirectional lectures and home works.

The redesigned classes adopted the following delivery mechanisms:

TABLE I: WEEKLY PLAN FOR THE PCS COURSES

Week	Topics	Alignment with Course outcomes
Week 1	Need for parallel architecture, multithreading and parallel programming models, shared address space, message passing	CO1
Week 2	Multicore architectures, data parallel processing	CO1, CO2
Week 3	Parallelizing computation versus data, parallelization of an example program, partitioning and mapping, synchronization	CO1, CO2
Week 4	Orchestration under data parallel model, orchestration under shared address space model, orchestration under Message-passing model	CO2
Week 5	Cache coherence problem, bus snooping, protocols-MSI and MESI	CO3
Week 6	Synchronization event, local and global event synchronization	CO2
Week 7	Project idea submission-selecting the problem for parallelization- review	CO1, CO2
Week 8	Open MP programming constructs parallel, work sharing and synchronization constructs	CO4
Week 9	MPI programming -MPI data types and tags, environment management routines	CO4
Week 10	GPU architectures	CO5
Week 11	Programming the GPU using CUDA	CO5
Week 12	Review of the mini project	CO1, CO2, CO4, CO5

1) Traditional lecture, where the instructor talked and students listen and take notes, this component of the course remained unchanged from previous offerings.
2) Discussions, where instructor and students both participate in interactive discussion guided by the instructor. Often flipped classroom approach was adopted for these discussion classes. The students could think and bring out an interesting way for demonstrating the cache coherence problem. One of the students developed a mobile application to illustrate the change in state of the cache blocks using the MESI protocol. Figure I show the screenshot of the application developed by the student.
3) Unplugged activity, where students participate in activities directed by the instructor to illustrate some PDC concept without the use of computer.
4) Short in-class programming, quiz, and assessment activities. For example while teaching the shared memory programming using OpenMP, an interactive quiz was conducted to test the understanding of the constructs. The students were divided into teams of 2 members each, randomly. The time to answer was 2 minutes for a team.
5) Lab based programming assignments. Students are given programming assignments to complete outside the class using the lab or their own computer. The programming assignments must be solved independently by the students at home and submitted through the canvas LMS, before the deadline. They can also use the laboratories during their leisure hours. Typically, the students preferred using laboratories for completing the simple MPI applications. For programming using OpenMP and CUDA applications they used their laptops.
6) Group mini projects. During week 3 of the lecture, a list of topics such as parameter sweep application, ocean current simulation, filtering the noise form an image, applications of Gauss Seidel algorithm for linear systems, odd-even sorting applications, etc. are given to the students. The students have the choice of selecting the topic and their team members for the mini Project. The team size is limited to maximum of 3 members. During week 7, the titles and their implementation ideas are reviewed. The review has a weightage of 40% of project grade. The rest 60% of the grade is for the implementation and submission of the report at the end of the semester.

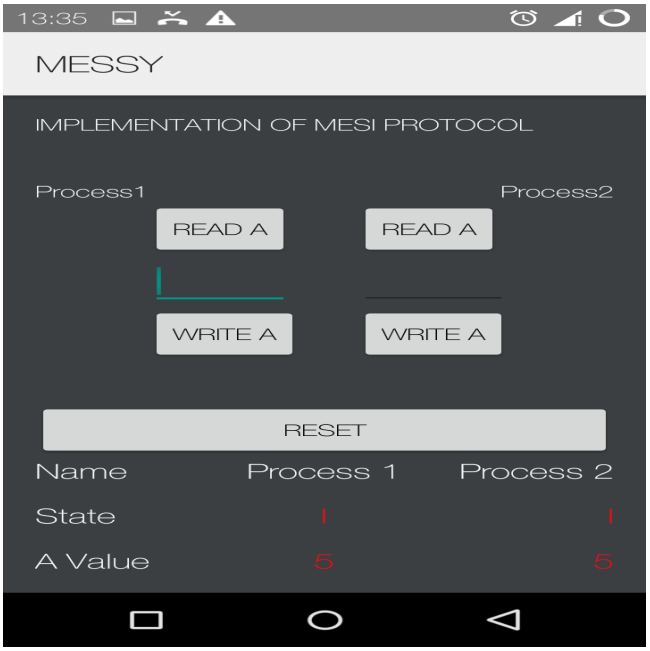

Figure I. Screenshot of the mobile application for MESI

Table II briefly describes some of the activities conducted during the course to achieve the course outcomes. These activities dont require a book or a material to make the concept clear. They are implemented by the students by discussion and involving themselves in quiz, home programming assignments and projects.

TABLE II: ACTIVITIES CONDUCTED DURING THE COURSE

Description of the topic	Activity Conducted	Methodology
To make the students understand the concept of thread and parallelism	A question to find the youngest student of the entire class was posed. The students followed different strategies. The concept of threading was illustrated by grouping them according to the desk wise (row) seating arrangement and asking them to find the youngest among them. The concept of computation and communication between the rows to arrive at the result was illustrated. The importance of the performance metric Communication to Computation ratio was demonstrated [12].	Unplugged activity
To make the students understand the significance of shared resources and synchronization	The students were made to realize that, the class room is a shared resource. The class room has the students enrolled for the PCS course. The entire faculty, who will handle the classes for the other course to these students, are processes which can use this resource.	Lecture with example
To make the students understand the importance of cache memory in improving the performance	Make the student visualize the kitchen in their home. Identify the items to be placed on the kitchen top near the stove. Illustrate the principle of locality of reference and mapping by placing the identified items on the kitchen top.	Discussion
To make the students understand the cache coherence	A role play was organized with the students bearing the pluck cards titled M, E, S, I and illustrating the corresponding changing states of cache.	Unplugged activity
To make the student understand the concept of data parallel model and role of GPUs	A flipped class room activity was given through a video lecture from NPTEL course on Parallel computing. The course instructor, takes an evening walk with few of the challenging students to discuss the concepts of slides and relate it to the state of art in GPU computing. These students prepared an interactive quiz and made the concepts clear.	Flipped class room
To address the students with vernacular problem	The student video shoots a five-minute talk about the multicore era and submitted as a video assignment. The student re-shoots it again and again, till he is convinced for submission, this improves the presentation skills.	Homework activity

TABLE III. SAMPLE QUESTIONS FOR QUIZ ON OPENMP

QZ1:

```
main(..)
{
int a;
a = omp_get_thread_num() ;
#pragma omp parallel
{
int x ;
x = omp_get_thread_num() ;
#pragma omp master
a += x ;
}
printf( "%d", a ) ;
}
```

Qz2:

```
main(..)
{
int n ;
n = omp_get_num_threads() ;
#pragma omp parallel
{
int x,y ;
x = omp_get_num_threads() ;
y = omp_get_thread_num() ;
#pragma omp single
n += x ;
}
printf( "%d", t ) ;
}
```

V. EVALUATION OF THE COURSE

To assess the performance of the students and the level of meeting the course outcomes, a comprehensive assessment and evaluation is needed. The survey was conducted using four point Likert scale (Strongly agree-1, Agree-2, Disagree-3, Neutral-4). Also we have assessed students understanding through short in class quizzes (non-graded). These questions for the quiz were framed to check the understanding with respect to the topics being taught. For example, during shared memory programing using OpenMP, a quiz was conducted to evaluate whether students are learning OpenMP constructs and its applications. The sample questions asked during OpenMP lesson are shown in Table III. The students were given a code snippet and were asked to provide the output of the code. They were allowed few minutes to think and answer. The answers were discussed in class afterwards and the instructor provided the correct answer with explanation. About 60% of the students provided correct answers, 20 % student provided somewhat correct answer, and the rest could not answer it correctly.

Also, a quick feedback about the quiz was collected from the students. The feedback included questions like:

- The quiz was useful in understanding the constructs.
- The material covered in the class enough for answering.
- The questions were easy.

The feedback revealed that 85% of the students felt that the content covered in the class was enough for answering the OpenMP quiz. And 90% of them felt that it was very useful the topics amidst the class.

Objective evaluations are done through graded programming assignment, project, and exams. The score break downs for the course grade was as follows:

Final Exam	40%
Mini Project	20%
Interim assessment tests	20%
Programming assignment	10%

The final examination was conducted for three hours. There are two interim assessment tests conducted for assessing the progress of the students understanding and implementation skills. Figure II shows the grades achieved by the students after using the active learning strategies compared to grades achieved by the students in 2017, offering that uses the traditional teaching method. From the result it can be seen that more student obtained S and A compared to the number of students in previous offering that uses traditional teaching method.

The items assessing the individual outcomes are evaluated separately and shown in the Figure III. This was done at the end of the term using an on-line survey, containing a set of short questions, inquiring about the level of achievement of the course outcomes. The instructor assessment was made by mapping the questions for final exam, interim assessment tests and programming assignments to the prescribed course outcomes. The attainment level of each of the course outcomes are measured by calculating the number of students who have answered correctly the corresponding questions. This enables us to see whether the student perceptions are consistent with the instructors actual measured results. Of the 20 students enrolled in the course, 18 students completed the survey.

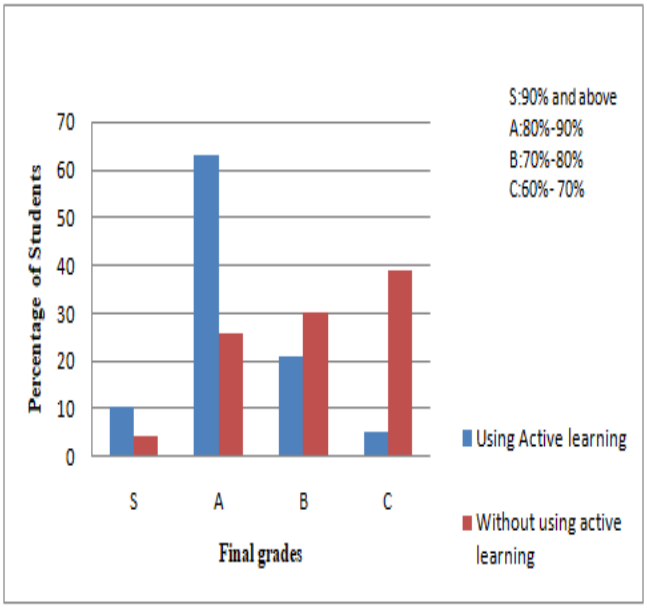

Figure II. Grade distribution of the students

A. Lesson learned

One of the features of the course was the use of real-life analogies and examples to supplement the theoretical discussions and easily introduced the fundamental concepts such as threading, deadlocks and synchronizations, importance of cache, etc.

After evaluation of the students assignment, we realized that the students struggled slightly in deciding upon the appropriate parallel programming model during the initial assessments through worksheets. Their understanding improved when they started to work for the mini projects. They learnt through seminar presentation of the mini project topics.

From the final exam result analysis and the mini project demo, we found that the MPI programming was relatively difficult for them compared to OpenMP. The students also struggled in relating the recent papers on false sharing problem with cache coherence protocol they have studied in the course. But, the passion towards making a mobile application to make other students understand the protocol was appreciable.

Since the students have not been exposed to parallel and distributed computing at the undergraduate level, very basic PDC topics had to be covered in this course. The course is very dense in number of topics and we feel that the course could be better organized in term of topics. After offering this course at the graduate level, we have realized the PDC need to be incorporated in the undergraduate curriculum.

It was challenging to make the students understand the complexity of big data processing and the need for parallel

programming. Initially, they were not comfortable with GPU programming.

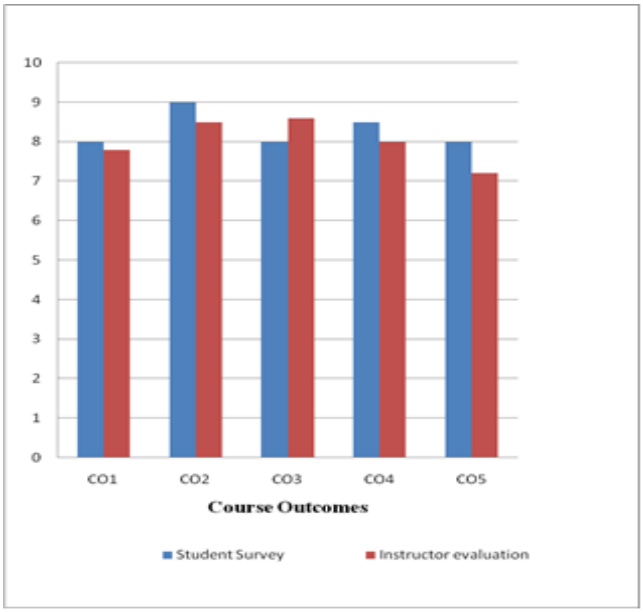

Figure III. Perception of students and instructor

We have realized that the PDC needs to be included in undergraduate curriculum. We have made a decision to include a required undergraduate parallel and distributed computed course. At present, we dont have a course which covers both the parallel architecture and programming at undergraduate level. We have realized that before introducing the undergraduate PDC course we need to slightly modify the course contents of Data Structures, Algorithms, Operating Systems and Computer Architecture to incorporate some PDC topics.

VI. CONCLUSION AND FUTURE WORK

This paper presents an effort to teach the parallel computing course at Thiagarajar College of Engineering, India. The active learning based approach combined several ingredients to improve the effectiveness of student learning. The students were engaged with in-class unplugged activity, visualizations, role play, and practical exercises. Moreover, we provided the students with informative notes and source code. We have evaluated the impact of the active learning strategies on students understanding of parallel programming. Our evaluation shows a significant improvement in understanding the theoretical and practical topics of parallel computing. Further, the active learning approach seems to give students an advantage in terms of allowing them to engage in the material and learn it more effectively. To complement our efforts, we allowed the students to practice parallelization topics, by asking them to use parallelism in their mini projects. Overall, the result was positive with most students successfully implementing parallel techniques in their own code. We plan to re-organize the course in terms topics covered and their relative emphasis and depth in the next offering to the graduate students, by exposing them to hybrid MPI and OpenMP programming, programming the heterogeneous cluster. This will make the students understand the strength of parallelism on different platforms. We have also decided to incorporate PDC in our undergraduate curriculum as described in the previous section.

REFERENCES

[1] G. Falcao, How fast can parallel programming be taught to undergraduate students?, IEEE Potentials, 32(4), 2013, pp. 2829.

[2] C. Ferner, B. Wilkinson and B. Heath, Toward using higherlevel abstractions to teach parallel computing, Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International, Cambridge, MA, 2024 May, 2013, pp. 12911296.

[3] M. Arroyo, Teaching parallel and distributed computing topics for the undergraduate computer science student, Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International, Cambridge, MA, 2024 May, 2013, pp. 12971303.

[4] Ko, Y., Burgstaller, B., Scholz, B.: Parallel from the beginning: the case for multicore programming in the computer science undergraduate curriculum. In: Proceeding of the 44th ACM Technical Symposium on Computer Science Education. ACM (2013)

[5] Adams, J.C.: Injecting parallel computing into CS2. In: Proceedings of the 45th ACM technical symposium on Computer science education. ACM (2014)

[6] Geist, R., Levine, J.A., Westall, J.: A problem-based learning approach to GPU computing. In: Proceedings of the Workshop on Education for High-Performance Computing. ACM (2015)

[7] T. Jenkins, Teaching programming a journey from teacher to motivator, in The 2nd Annual Conference of the LSTN Center for Information and Computer Science, 2001.

[8] A. Mouratidis and G. D. Sideridis, On social achievement goals: Their relations with peer acceptance, classroom belongingness, and perceptions of loneliness, The Journal of Experimental Education, vol. 77, no. 3, pp. 285308, 2009.

[9] Nasser Giacaman, The Active classroom: Students and Instructors Parallel Programming in Parallel, IEEE International Parallel and Distributed Processing Symposium Workshop, 2015,PP.739-745 .

[10] S. Yamaguchi, Y. Ohnichi, and K. Nichino, An Efficient High Resolution Video Distribution System for the Lecture Using Blackboard Description, Technical Report of IEICE, 112 (190), pp. 115119, 2013.

[11] N. Yamamoto, T. Wakahara, An Interactive Learning System Using smartphone for Improving Students Learning Motivation, Information Technology Convergence, Lecture Notes in Electrical Engineering Volume 253, pp. 305-310,2013

[12] Chitra, P., Sheikh Ghafoor, Unplugged Activity: Finding Youngest Student in Class, https://www.csc.tntech.edu/pdcincs/resources/modules/unplugged/Find _Youngest/Find _Youngest.pdf