PDC Course Development and Assessment Process for the betterment of Teaching-Learning Process

Neelima Bayyapu
Department of Computer Science and Engineering
Manipal Institute of Technology (MIT)
Manipal Academy of Higher Education (MAHE), Manipal
Karnataka, India. 576104
0000-0002-7201-2217

***Abstract*—Outcome-based education is being adapted in most of the engineering colleges across India. This process is being accredited by a national body called National Board of Accreditation (NBA) in India. This accreditation process involves a continuous process of defining Program Outcomes (POs) and Course Outcomes (COs) and these are evaluated to check the OBE outcome level. This course evaluation includes course content, evaluation methods used for students' evaluation, pedagogy of teaching, Bloom's taxonomy mapping of learning levels, feedback from stakeholders and results and outcome analysis. If the required level is not attained, the COs and POs are reviewed and accordingly curriculum is revised. There are a lot of challenges being faced by faculty in this process of creating and assessing the curriculum as per NBA standards. These challenges are unique and non-trivial for PDC/HPC (Parallel and Distributed Computing/High-Performance Computing) courses. The preparation of OBE-based PDC/HPC course involves a lot of stakeholders and brainstorming over multiple sessions. Many universities are adopting PDC/HPC courses more recently across the world. With an intention to create a pointer for developing, delivering and reviewing a PDC/HPC course, this paper presents the course development process for the benefit of various stakeholders and specially PDC/HPC educators and institutions. This research paper presents the undergraduate teaching experience of parallel computing (PC) course with a critical evaluation based on course Outcomes (COs) and Bloom's taxonomy mapping of learning levels. This paper compiles a list of challenges faced by PDC/HPC educators and stakeholders focusing on the Indian education scenario. It lists the activities and the suggestions that can be applied to address these challenges suitably. The research and teaching experience-based discussions and strategies proposed in this paper help other PDC/HPC educators and stakeholders in the CS (Computer Science) community.**

***Index Terms*—Undergraduate Education; Parallel and Distributed Computing (PDC) Education; High-Performance Computing (HPC) Education; Computer Science Community; Indian Education; Bloom's Taxonomy Learning Levels; Course Outcomes; Parallel Computing Course;**

I. Introduction

Computer science and engineering education have seen significant changes, especially over the past decade. This change applies to curricula, the expectations of the engineering education system stakeholders, evaluation of various student competencies, exposure of diverse knowledge through the internet medium, and newer demands in the industry. Alongside this, the 20th century's pandemic has added more unique needs in education. These changes have given new opportunities and challenges for an academician or practitioners of computer science education. This paper gives details of the experience of teaching a PDC course in an Indian university and summarizes the resources available for a smooth learning process, challenges faced, and strategies that helped address those challenges. We used outcome-based evaluation of the courses. Outcome-based education [1] focuses on learning and performance of the students rather than only on training efforts by a teacher.

The paper explains the need of documenting the course development process, the resources available and challenges that a PDC educators might come across along with strategies that might help address those challenges. The details given in this paper are an effort to help early adopters of PDC education in various institutions.

The PDC resources and their availability is increasing to grow, but there is a clear need to expand such efforts. With this need and motivation, this paper is an attempt to document the experience of teaching a PDC course along with an effort to document a list of challenges and resources for the benefit of other CS practitioners.

Specifically, the paper has the following objectives set:

- To share the experience of teaching a parallel computing course to undergraduate students; Course outcome (CO) [1] based assessment of the course; activities that helped to engage the students
- To discuss challenges in PDC education and present mitigation techniques for solidarity and advocacy of early PDC educators with a focus on Indian education perspective

The paper is organized as follows: Section 2 details the experience of teaching a parallel computing course in Fall-2019 along with the outcome-based course evaluation.

Section 3 details challenges that exist, especially for PDC educators; resources that are available to address PDC education challenges specifically from the Indian HPC education perspective. This section also refers to the efforts and work of the Indian HPC community. Section 4 gives general suggestions and proposals and concludes in Section 5.

II. Outcome-Based-Evaluation of PC Course

The PDC course, described in this section, was taught in a centrally funded national institute in India. The course named parallel computing (PC) (with course code IT-300) for the academic year 2019-2020 (Fall-2019) was offered as a program core to the 5th semester or 3rd-year students. This course broadly covers shared-memory, distributed-memory programming along with accelerator computing. There were 103 students enrolled for this class, with a distribution of 90 men and 13 women. This course is a three-credit program for the students, with three hours of lecture and 2 hours of laboratory per week. All the courses and course evaluations described here are designed as per NBA (National Board of Accreditation) [1] standards, that includes course objectives, course outcomes (COs) and mapped to Bloom's taxonomy of learning levels [2]. The course logistics included shared memory programming with a specific study of OpenMP (Open specification for Multiple Processing), distributed memory programming with a focus on MPI (Message Passing Interface), and accelerator programming using CUDA (Compute Unified Device Architecture). The course has detailed practical learning through laboratory exercises and a miniproject as mentioned in Table I.

Laboratory exercises include implementing matrix multiplication in OpenMP; application to demonstrate usage of reducing and other work-sharing directives such as for and sections; application to understand synchronization construct such as atomic, barrier; Programming in MPI includes parallel sort application using blocking send and receive; applications to understand the usage of non-blocking send and receive and collective operations; and CUDA programs on matrix multiplication to understand data parallelism; applications to understand dynamic parallelism, streaming, shared memory and constant memory usage, to mention a few. Mini project is introduced to the students so that a broader picture of parallel computing can be observed and learned by the students. It is carried out by a two-or three-member team. The process involves a paper selection by a student team or faculty, and then the student implements the given paper as it is and presents it to the faculty. If students realize a novel extension of the paper, they are encouraged to work on it during their Fall semester break; if the team is interested, guidance is provided further to publish the work.

The board of studies (BoS) of the department decides the course broadly and teachers are given the freedom to choose the sub-topics within those broader definitions of the syllabus. The course syllabus is given in Table I.

The students have studied computer organization and architecture (COA) courses in their previous semester that was set as a pre-requisite for the PC course. In COA, this batch of students studied pipelining and hazards but not the types of data hazards; learned memory but not about the cache coherence policies etc. After speaking to the students, we decided to add these concepts to the PC course and used the initial two weeks of classes for all the architecture concepts required for the PC course.

Topics	**Sub-Topics (52 hours/17 weeks)**
Study of Concurrent Parallel Systems (12 Hours)	Concurrency Vs. Parallelism; Overview of concurrent parallel systems; the evolution of multi-core processors, the concept of free lunch is over; Study of Moore's law; design principles and memory hierarchy of multiprocessor machines; classification of parallel computers; types of classification; Shared memory multiprocessors; cache coherence protocols; types of dependency; Bernstein conditions for detection of parallelism; dependence graphs and notations; studying towards vectorization as a limited granularity of parallelism; distributed memory systems; types of interconnections; study design principles of interconnections; principles of communication; analyzing parallel code; speed-up and performance metrics and laws for parallelization; basic optimization techniques for serial code; principles of parallel algorithm design
Shared-memory parallel programming (14 Hours)	Shared-memory programming need, motivation, and evolution; overview of OpenMP development; concepts and syntax of OpenMP; parallel execution; data scoping; work-sharing constructs; synchronization; reductions; environment variables; runtime library routines and miscellaneous; parallel application development and implementation; Introduction to TBB, TL2, Cilk++, etc. and software transactional memory techniques
Distributed-memory parallel programming (12 Hours)	Distributed-memory programming model evolution; message and point-to-point communication; collective communication; non-blocking point-to-point communication; virtual topologies; MPI programming constructs and application development and implementation.
Accelerator computing (14 Hours)	Introduction to many-core architectures; the evolution of GPGPUs; compute unified device architecture (CUDA) and programming; processing elements and memory hierarchy in many-core architecture; use of occupancy calculator; study and compare two versions of many-core architectures; Introduction to CUDA programming; CUDA program structure; memory transfer; kernel launch parameter design; kernel creation; steps of compilation; CUDA application development and implementation. Introduction to Heterogeneous programming using Xeon-Phi and OpenCL
Practical Laboratory: CO2, CO3, CO4	To implement various concepts of shared memory programming using OpenMP, distributed memory programming using MPI and accelerator programming using CUDA while learning to install various tools and APIs (Spanned over semester)
Mini project: CO4	To implement any research paper related to the syllabus (Spanned over semester)

TABLE I
Parallel Computing (IT-300) Topics and Sub-Topics

After detailing key points from the pre-requisites, the students were given principles of parallelization, laws of parallelism following with shared memory introduction. The students were introduced to OpenMP programming. Similarly, distributed memory programming techniques were given with an introduction to message-passing using MPI. The course covered MPI basics, point-to-point communications, collective communications, virtual topology, blocking, and non-blocking communication principles. The students were given details of accelerator computing architecture based on NVIDIA GPUs (Graphics Processing Units) and were taught CUDA programming basics. All the concepts covered were experimented during the hands-on laboratory sessions along with application development and implementation. The course objectives and course outcomes (COs) are given in Table II. Table II provides

Course Objectives	
1	Study principles of concurrency and parallelism
2	Learn parallel algorithm design and analysis
3	Implement High Performance Parallel Programs using programming models and libraries
Course Outcomes (COs)	
CO1 (L1, L2 and L3)	To recapitulate concurrent parallel systems and sources of concurrency
CO2 (L4)	To design and analyse parallel algorithms
CO3 (L5)	To explicate parallel algorithms using shared, distributed, and accelerator programming models and libraries
CO4 (L6)	To implement applications using OpenMP, MPI and CUDA

TABLE II
COURSE OBJECTIVES AND OUTCOMES FOR PARALLEL COMPUTING (IT-300) IN FALL-2019

Project	
Minor Project (PJ)	CO1, CO2, CO3, CO4
Laboratory (Lab)	CO1, CO2, CO3, CO4
Mid Sem Theory	
Mid Sem Question 1 (MQ1)	CO1, CO2
Mid Sem Question 2 (MQ2)	CO1
Mid Sem Question 3 (MQ3)	CO1, CO2
Mid Sem Question 4 (MQ4)	CO2, CO3
Mid Sem Question 5 (MQ5)	CO2, CO3
End Sem Theory	
End Sem question 1 (EQ1)	CO1
End Sem question 2 (EQ2)	CO1, CO2
End Sem question 3 (EQ3)	CO3, CO4
End Sem question 4 (EQ4)	CO2, CO3, CO4
End Sem question 5 (EQ5)	CO2, CO3

TABLE III
MAPPING OF COURSE OUTCOME ON EVALUATION CRITERIA FOR PARALLEL COMPUTING (IT-300)

COs	MAPPING OF QUESTIONS TO EACH COs
1	PJ, LAB, MQ1, MQ2, MQ3, EQ1, EQ2
2	PJ, LAB, MQ1, MQ3, MQ4, MQ5, EQ2, EQ4, EQ5
3	PJ, LAB, MQ4, MQ5, EQ3, EQ4, EQ5
4	PJ, LAB, EQ3, EQ4

TABLE IV
COURSE OUTCOME COVERAGE OVER THE GIVEN CRITERIA FOR PARALLEL COMPUTING (IT-300)

the mapping of COs with Bloom's learning levels. These mappings are used in evaluating the courses to understand the learning attainment towards the end of this section.

The PC course's student evaluation plan was given as follows: 60% weightage was given to theory evaluations that include continuous assessments, mid-semester exam, and end-semester exam; 40% weightage was given to laboratory evaluations that included continuous lab evaluations, mid-semester laboratory exams and a mini-project implementation.

The feedback and evaluations show that all these students have different backgrounds and exposure to programming skills. This batch of students were not introduced to Unix programming and basic instructions such as 'ssh' were not known to many of the students. The first two laboratories were dedicated to the programming basics required for the lab. Further, the lab systems were not equipped with GPUs for CUDA programming. The institute has a DGX server with GPU devices [3] that can be used over internal IP (Internet Protocol). Remote logins were created and through this access, students could experiment with the CUDA programming assignments. From the students' feedback, it was very clear that having a hands-on session helped the students understand the parallel programming concepts and makes the course interesting.

To understand the evaluation process, Table III gives the mapping of course outcomes with the evaluation criteria of the PC course. For example, a Minor project (PJ) covers all course outcomes defined, i.e. CO1, CO2, CO3, and CO4. Similarly, Mid-semester question number one covers CO1 and CO2 only and so on.

Table IV gives the details of course outcome coverage in the PC course evaluation. For example, CO1 was covered in mini-project (PJ), laboratory component (LAB) along with mid-semester exam questions 1, 2, and 3 (MQ1, MQ2, and MQ3) and end-semester examination questions 1 and 2 (EQ1 and EQ2). After all the evaluations, the number of students who scored more than 40% of marks in each of the evaluation criteria is computed and listed as in Table V.

The normalized evaluation of course outcome components is evaluated as per the formula given in column 2 of Table VI. From Table VI, it can be observed that the attainment of course outcome 1 (CO1) is 81.4%, attainment of CO2 is 72 %, attainment of CO3 is 71.9 % and attainment of CO4 is 81.3 %. From the definitions of COs from Table II and comparing with the attainment level, it is clearly understood that students performed well in the basic introduction part (i.e. CO1) and software implementation part (i.e. CO4). While parallel algorithm design and core concepts of parallel computation as defined in CO2 and CO3 were not performed well by students on an average in parallel computing class.

Further detailed analysis of COs is given in Figure 1. From the analysis of CO1 from Figure 1, 29 students attained 61-70% of CO1, 25 students attained 71-80% of CO1, 18 students attained 81-90% of CO1 and four students attained 91-100% of the CO1, with a total of 76 students with more than 60% attainment. Similarly, there were a total of 27 students (11+10+2+2+2+0) from CO1 attainment, who attained less

Component	Component Number	Number of Students who scored >40%
Project	PJ	102
Laboratory	LAB	99
5*Mid Sem Theory	MQ1	66
	MQ2	76
	MQ3	88
	MQ4	65
	MQ5	50
5*End Sem Theory	EQ1	79
	EQ2	77
	EQ3	81
	EQ4	53
	EQ5	68

TABLE V
NUMBER OF STUDENTS WHO SCORED >40% IN THE LISTED CRITERIA FOR PARALLEL COMPUTING (IT-300)

COs	NORMALIZED EVALUATION COMPONENTS	% OF ATTAINMENT
1	SUM(((PJ, LAB, MQ1, MQ2, MQ3, EQ1, EQ2) /103)*100)/7	81.4
2	SUM(((PJ, LAB, MQ1, MQ3,MQ4, MQ5, EQ2, EQ4, EQ5) /103)*100)/9	72.0
3	SUM(((PJ, LAB, MQ4, MQ5, EQ3, EQ4, EQ5) /103)*100)/7	71.9
4	SUM(((PJ, LAB, EQ3, EQ4)/103)*100)/4	81.3
	AVERAGE ATTAINMENT	76.65

TABLE VI
NORMALIZED EVALUATION OF THE COMPONENTS FOR THE COMPUTATION OF COURSE OUTCOME ATTAINMENT

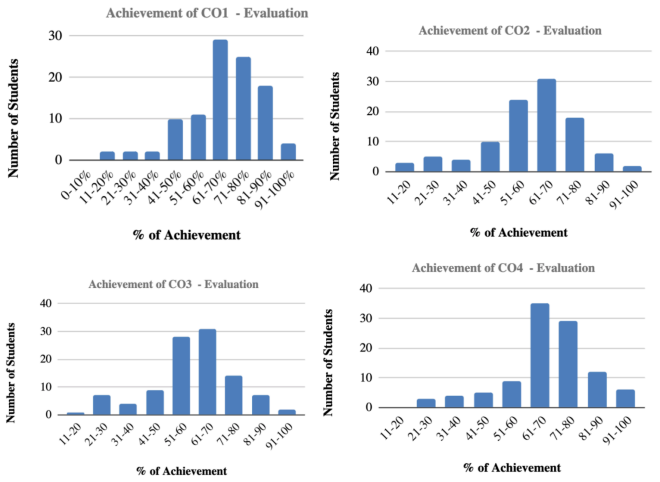

Fig. 1. Course Outcome Analysis (of CO1, CO2, CO3 and CO4) for the Parallel Computing Course (IT-300) for Fall-2019 with 103 Students

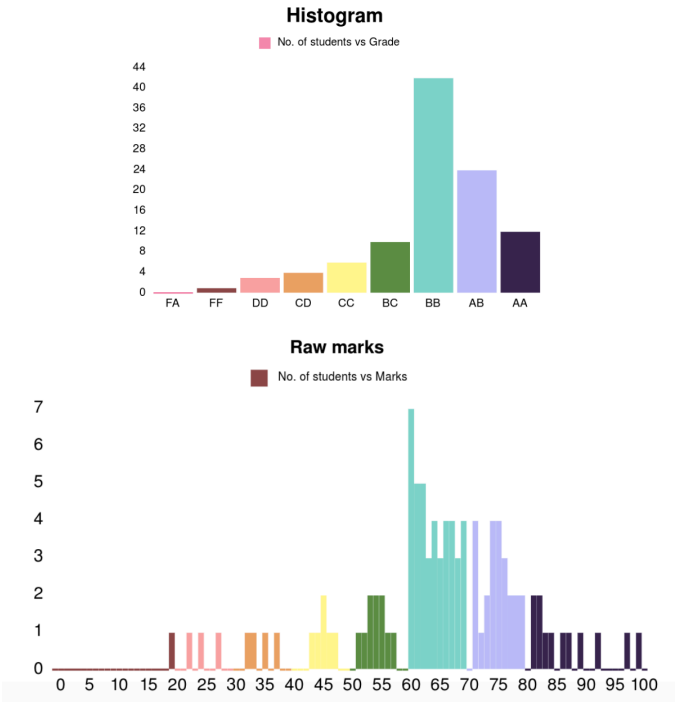

Fig. 2. Analysis of Overall Evaluation of the Students for the Parallel Computing Course

than or equal to 60% of CO1. Further, observations of CO2 gives a total of 57 students with more than 60% attainment of CO2 and a total of 46 students attained less than or equal to 60% of CO2 attainment; observations of CO3 gives a total of 54 students with greater than 60% attainment of CO3 and a total of 49 students obtained less than or equal to 60% attainment of CO3; observations of CO4 gives a total of 77 students with greater than 60 % attainment of CO4 and a total of 21 students obtained less than or equal 60% attainment of CO4. It is clear that the course was very successful in achieving the CO1 and CO4 attainment and needs improvements for CO2 and CO3. In summary, the overall course attainment is 76.65 as given in Table VI.

Figure 2 gives an analysis of the overall performance of the students from the parallel computing course. The bell curve indicates the best performance distribution of the class from the graph. The standard deviation of the overall performance is 64 marks. There is a lot of variation in students' performance. This attributes to different skill sets and need for a deeper understanding of the subject by individual students, it can also be attributed to the organization and evaluation process of the class that can be improved. The more general activities and approaches to PDC education are discussed in the later sections, while specific details of the PC course are given in this section.

Due to the inherent depth of PDC courses, it was tough to gain the confidence of all the students in the subject and a lot of handholding was required even during the laboratory sessions. Apart from extra efforts in teaching and mentoring, the following are some of the activities that helped achieve the class's goals. These challenges and solutions proposed here are not based on one-semester teaching experience but based on the observations gathered over various semesters of teaching PDC course:

- **Pre-requisites coverage:** As mentioned before, the course needed some additional inputs that are not part of the syllabus. These modules were added to the course plan and this helped the students to relate to this subject.
- **Mini-projects and peer-mentoring:** One initiative was a mini-project that involves an existing research paper implementation of the student's choice (accepted after a faculty review to assess the level of learning possible through the chosen statement). The list of mini-projects is given in Table VII. There were 36 groups with 103 students. There are a few batches with the successful implementation of the defined objectives and others modified

their objective based on their capabilities. But one issue noticed was in a group of three students, all students' involvement is not equal, and it was hard to evaluate the course's learning skills. The course was assigned with a few Ph. D. students as Teaching Assistants (TAs) in the laboratory, but they were not with HPC background. Peer-mentoring was added as a support for the successful implementation of mini-projects. Peer-mentoring involved discussion sessions among students about mini-projects during the laboratory hours, discussions on class group, and the presentations made by the students about the mini-project to the other students in the class.

- **Quizzes:** A small informal quiz from the previous class before starting the lecture seems to help them follow the class well based on the feedback from students.
- **Organized learning material:** An intermediate quick class feedback had full requests for reading and learning resources. An undergraduate student with no exposure to HPC indeed gets confused with a lot of information available online. For example, for OpenMP, students were asked to refer to www.openmp.org [4, 5] website. But most of them did not use this resource as the available information is enormous. So a tailor-made repository of the resources from the available online resources was created. This repository helped students to focus well.
- **Use of piazza platform:** Through-out the course, we have utilized peer-mentoring. Though the institute has internal access to the Moodle system, students were comfortable and excited using Piazza platform [6] where the course material, announcements, and interactions were managed. These kinds of engaging and connecting with the students beyond the classroom helped create interest in the subject, based on feedback from the students. Pre-Covid-19, online platforms were not widely used by our students and using Piazza for the course helped students who missed the classes as well.

S. No.	PC Mini-Project Title
1	Parallelization of Cooley-Tuckey Fast Fourier Transformation Algorithm using CUDA
2	Parallelization of PSO(particle swarm optimization)
3	Parallel Linear Equation Solver
4	Fast-tracking K-Means Clustering with Parallel Implementation and GPU computing
5	Graph Betweenness Centrality for Sparse Graphs using CUDA
6	Classification of Hyperspectral Images
7	Parallelization of Steganography using Genetic Algorithm with Visual Cryptography
8	A Fast and Efficient Parallel Method for Coloring Graphs
9	Parallel Romberg Integration
10	Parallelization of Deep Neural Networks
11	Parallelizing Collaborative Filtering
12	Parallelization of Decision Tree Classifier
13	Parallel Scanline Algorithm for Rapid Rasterization
14	Parallelizing of Genetic Algorithm using CUDA based on Island Models
15	Optimizing Sparse Matrix-Vector Multiplication on GPUs
16	Performance Evaluation of Long Integer Multiplication using OpenMP and MPI on Shared Memory Architecture
17	Parallelized Monte Carlo Method to Evaluate Inverse Matrix using OpenMP
18	Parallelized KMP Algorithm for String Matching
19	Performance Evaluation of CUDA based Bitonic Sorting in Virtual Machines and Docker Based Containers
20	Parallel Algorithms for Matrix Multiplication
21	Parallelising AES Encryption Algorithm using OpenMP and CUDA
22	Parallelising Branch and Bound Algorithm using OpenMP to Solve TSP
23	Comparative Study of Knuth-Morris-Pratt & Boyer Moore Algo. for String Matching
24	Parallelizing the Set-up of Deeply Joint Informed Neural Networks With Decision Trees
25	Parallel Dual Population Genetic Algorithm for solving 0/1 Knapsack Problem
26	Median Filtering Algorithm for Images using CUDA
27	Parallelisation of Convolutional Filter Calculation for Images using OpenMP and CUDA
28	Scalable Parallel Word Search in Multi-core Systems using OpenMP and PHP
29	Comparative Analysis of Parallelised Shortest Path Algorithms using OpenMP
30	CUDA-Based Parallelization of Power Iteration Clustering for Large Datasets
31	Parallelization of KMP String Matching Algorithm

TABLE VII
LIST OF MINI-PROJECTS (LISTED 31 OUT OF 36) CARRIED OUT BY STUDENTS AS PART OF PARALLEL COMPUTING COURSE

III. CHALLENGES AND MITIGATION STRATEGIES FOR A PDC COURSE: INDIAN PERSPECTIVE

From the information and observations in the previous sections and general observation over various previous semesters (i.e. teaching experience of PDC course), present existing challenges and resources along with the future steps required in PDC education is presented in this section. The community knows the PDC education efforts of a decade almost and still the reachability of these efforts is low. Hence, it is the time to spell-out the challenges in this process, try to mitigate them and increase the visibility and reachability of the available resources. One such recent effort is given in [7]. We are sure that many PDC educators will correlate with the following points and add further points in their future publications or experiment with some of the suitable suggestions. Though most of the points are interrelated, these are listed pointwise for a better appreciation of the observations as follows:

1) **Pre-requisites of the PDC course:** Many institutions offer PDC course at the undergraduate level as one program core or as an elective. There are common challenges such as: students take this course without having enough background to understand the concepts; students are not clear on how to use PDC course in their career; they apprehend to take this course due to the inherent difficulties in this course; lack of organized learning resources and availability of hardware infrastructure and lack of visibility. These issues are like the chicken-egg problem and there is no one way to say this is the solution. The faculty concerned need to analyze the situation and act accordingly. The following are some of the pointers for the same, based on the positive results experienced by the author: **Challenge-**

Mitigation 1: Due to the inherent requirement of knowledge depth required for a PDC course is more than most of the other undergraduate courses. The faculty should understand this need and provide room for the review of pre-requisites with either additional classes or handouts to help the students. **Challenge-Mitigation 2:** Because the undergraduate students are exposed to a tiny bit of PDC course, they do not see this course's usability in career development. Case-studies and mini-projects helped the students understand the big picture of the course in my experience. One can find such success story by Malakar at [8, 9]**Challenge-Mitigation 3:** Undergraduate students on an average need organized learning methods. It is always a good idea to provide the students clearly with each class's objectives and provide learning materials while giving references for advanced learning for the benefit of an aspired student. This is clearly emphasized in OBE-based curriculum development. **Challenge-Mitigation 4:** Undergraduate students have less visibility of a PDC course. One such reason is they do not see these course requirements in any interviews or competitive examinations. Industry expert lectures and invited talks from researchers helps them understand the need for this course on a given job profile.

2) **Addressing faculty needs:** There is a training period for most of the jobs but not for teaching. **a) Teacher training** A person is expected teach once he obtains his Ph. D. degree. While Ph. D. is a narrower program of study and gives exposure to teaching in teaching assistantship, a person entering into a teaching profession needs training, especially with computer science-related subjects where curricula changes are very huge. This training need not be towards the subject content, but to train on pedagogical methods and to provide pointers of help. For example, one might be an excellent parallel programmer, but one should be able to bring in parallel thinking into students. Centre for Development of Advanced Computing (CDAC) is one place to look for HPC-training updates in India. [10] **b)Mentoring and collaborations:** For any profession guidance is important. Mentoring is to be considered for faculty at all levels both from within and outside the university and Industry. This enables mutual interactions among the stakeholders and the perspective of things certainly gets changed. **c) Self-education and targeted training:** Universities should encourage for self-education of faculty. Further faculty needs bigger exposure and experience before handling a given course. There is a need to support PDC/HPC training of faculty and researchers in short duration through industry internships or workshops or tutorials. One such need of training programs is documented by Shenoi et. al at [11]. **d) Teaching material:** Teaching material for PDC/HPC education is another concern. It is very hard to create assignments, quizzes, and exam papers every semester for a huge undergraduate student batch. There are instances when I had to refer to just presentation slides or a research paper for a few topics as per the syllabus. There is a need to have a repository of basic concepts to be created with the HPC community's help. There are few efforts in this direction [12] and these efforts need to be reached to a larger community. **e) Visibility and motivation:** HPC education is inherently inter-disciplinary. One is expected to use a fast system to tune a domain-specific application and improve performance. Restricting to the content of the syllabus misses the big picture for the student. The faculty should make sure that the student feels the importance of these courses. We have seen a batch of students name their project as a machine learning-based application while they take an existing functionality and improve it on GPU. They did not realize the importance of adding HPC functionality to the title of the project. This is how the visibility of HPC is lost. We feel these minute things need to be worked out at the bottom-level such as student-faculty interactions at the undergraduate level to increase the visibility of PDC/HPC usage.

3) **Increasing/Retaining HPC workforce :** In my opinion, a faculty should not just be concerned to complete teaching course, but also there is a need to hold and retain the students when they wish to learn PDC beyond the curriculum needs. **a)Summer internship programs:** Most Indian universities have 2.5 to 3 months of summer breaks. We can use this time of students to gain more specialization in HPC. This can be with credit in the form of an audit course or without credit based on the students' interest. Faculty need to provide a low-hanging carrot for the students to attract and retain the students. One such success story is presented in [8]. **b) Undergraduate degree with specialization:** There is a need to build the strengths of the HPC community. We tried a quality circle system at an Indian university. This involves defining quality circles within a department where faculty are also teamed up based on their specialization. This faculty group offers different courses within that specialization and adopts students into the quality circle and trains the students through various activities. The department can build resources, laboratories along with a team of faculty, Ph. D. (Doctor of Philosophy), graduate and undergraduate students. This applies to any specialization. These efforts and the HPC infrastructure available can also be published on the university website for the benefit of other universities/institutes in need can approach or collaborate. This also increases PDC education visibility. If possible, at the university level, undergraduation with a specialization such as a degree in HPC or a degree in machine learning, etc. could be offered. Recently the Indian government has proposed a new education policy [13] that is inclined towards such specialized courses. It is one such opportunity for the HPC community to create PDC/HPC specializations to

attract students and retain them in this field.

4) **Improving career prospects and awareness:** A person joins in an engineering course with an expectation of a great job/career prospect. **a) Need of popularizing HPC:** Students opt for machine learning or data science or deep learning or cyber-security course than a PDC/HPC course, despite similar challenges such as resource availability and learning challenges. PDC/HPC courses being less attractive for students is certainly because of less awareness and less job prospects at the undergraduate level. HPC is inter-disciplinary and easy to be lost from sight. We need to make it a point to mention the use of HPC in job profiles or lists in competitive exams. Industry talks and expert lectures can be mandated in the syllabus for such an exposure. **b) HPC research and collaboration:** There is a small percentage of undergraduate students looking for a research career and higher education in HPC. While the average percentage of Ph. D.s obtained in India in HPC is very less when compared to other specializations. The HPC/PDC research perspective has changed over a decade, but it is not up to the mark yet. With the national super-computing mission of India [14], there is a clear scope of increased career opportunities, but awareness about this specialization from industry and research needs to be made available. Any appropriate statutory body or professional society or special interest group needs to be created. A quarterly newsletter to the community should help familiarize the people and infrastructures available in both academia and industry. The main platforms, to the best of the author's knowledge, that are available at present in India for good HPC exposures are HiPC [15] and Indosys [16] conference.

There are many international efforts and resources such as training programs and workshops and collaboration platforms. A few of them are listed below:

- Curriculum Development and Educational Resources (CDER) [17, 18]
- Peta-scale Computing Institute (PCI) [19]
- Argonne Training Program on Extreme-Scale Computing (ATPESC) [20].
- Workshop on Education for High-Performance Computing (EduHPC) [21] co-located with the supercomputing conference (SC), NSF/TCPP Workshop on Parallel and Distributed Computing Education (EduPar-20) [22] co-located with International Parallel and Distributed Processing Symposium (IPDPS) and a European Workshop on Parallel and Distributed Computing Education for Undergraduate Students (Euro-EDUPAR) [23] co-located with European Conference on Parallel and Distributed Computing (EuroPAR).

In summary, a lot of information compiled by PDC education workshops co-located with various conferences needs to be mobilized into smaller institutions and communities because the knowledge base's benefit is enormous. Listing such resources in this section is one such effort to make these existing efforts get exposed and familiarized in a larger community. It has to be made sure that our demographics help increase the reachability of this document and the resources listed in this document to a larger community.

IV. Discussions

In summary, there is no doubt that PDC education has become a mandate in many institutions at the undergraduate level recently. It is better to strengthen this eco-system while building only, rather than looking for refinements at a later stage. It is time to voluntarily publicize the efforts as these resources' reachability is very low at present. It is time to express challenges openly and create a support system for the same. This research-based analysis and observation is one such step forward to help increase the visibility and reachability of PDC educators' efforts across the world. PDC faculty need to introspect the need for resources and training and look for such open opportunities. With rapid changes in the computer science domain, it is time to learn things every day. At the same time, PDC concepts are not hard to convey as they seem to be. With appropriate tools and activities, these courses can be made exciting and pedagogical innovations that work for one should be published to help many directly and indirectly.

In summary, the following list is a take-away from this paper:

- PDC educators need additional efforts to bring in parallel thinking into the undergraduate students with various unplugged activities, by providing a review of pre-requisites, by involving them in mini-projects or summer internships to retain the students' interest and providing them with the required learning and hardware resources.
- PDC educators should take initiatives in collaborating with other members of the community from both academia and industry by seeking help in curriculum creation, for creating resources, and for mutual solidarity in PDC education to increase the visibility of these courses and improve workforce.
- At the same time, there is a need to understand the challenges of academicians handling PDC courses for undergraduate students. The community should come forward to create regional and international teaching and learning resources and increase the reachability of these efforts. The industry should be ready to train faculty and grants should be provided for creating learning resources.
- The professional societies or special interest groups should focus on creating and publicizing forums for PDC educators' training and collaborations.
- The community should create webinars or expert lectures on making the stakeholders realize the career prospects in this area of study.

By this, it is understood that there is a need for a collaborative effort from all the PDC/HPC community stakeholders. Any small contribution is worth to be shared and if it helps even a small percentage of this community, the goal of these

efforts is full-filled. There is a need for creating an eco-system to reach the community at large.

V. Conclusions and Future Work

This research paper has summarized complete 360 degrees view of PDC education. Firstly, the teaching experience of a parallel computing course is being shared and the course evaluation based on course outcomes mapped to Bloom's taxonomy learning levels. Based on these insights, a summary of the efforts required in PDC education is given. This paper presented challenges in PDC education in the Indian education system and provided a list of resources and efforts. The paper aims to provide a pointer to all these existing resources that are a need of many early and prospective educators.

The paper also proposes some future strategies such as the need of more trained PDC educators, the creation of an eco-system for training, collaboration and support for teaching PDC courses, efforts needed to increase the career prospects, and visibility of the course at various stages by the stakeholders of the community.

References

[1] Govt. of India Ministry of Education. *National Board of Accreditation*, 2020 (accessed 10, September, 2020).

[2] University of Utah. *Blooms Taxanomy for Learning Outcomes*, 2020 (accessed 10, September, 2020).

[3] NVIDIA. *NVIDIA Developer Website*, 2020 (accessed 10, September, 2020).

[4] OpenMP Consortium. *OpenMP Website*, 2020 (accessed 10, September, 2020).

[5] Gabriele Jost Barbara Chapman and Ruud van der Pas. *Using OpenMP:Portable Shared Memory Parallel Programming*. MIT Press, 12 2007.

[6] PC (IT-300). *Piazza Course Page*, 2020 (accessed 10, September, 2020).

[7] V. Ford D. W. Brown and S. K. Ghafoor. A framework for the evaluation of parallel and distributed computing educational resources. In *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), New Orleans, LA, USA, 2020, pp. 261-268*. IEEE, Aug 2020.

[8] B. Neelima. High performance computing education in an indian engineering institute. *J. Parallel Distributed Comput.*, 105:73–82, 2017.

[9] Preeti Malakar. Experiences of teaching parallel computing to undergraduates and post-graduates. In *2019 26th International Conference on High Performance Computing, Data and Analytics Workshop (HiPCW)*. IEEE, dec 2019.

[10] CDAC. *Centre for Development of Advanced Computing*, 2020 (accessed 10, September, 2020).

[11] V. Venkatesh Shenoi, Vaishali Shah, and Sandeep K. Joshi. HPC education for domain scientists: An indian experience and perspective. In *2019 26th International Conference on High Performance Computing, Data and Analytics Workshop (HiPCW)*. IEEE, dec 2019.

[12] CDER. *CDER Book Project on PDC*, 2020 (accessed 10, September, 2020).

[13] MHRD. *New Education Policy*, 2020 (accessed 10, September, 2020).

[14] NSM. *National Supercomputing Mission*, 2020 (accessed 10, September, 2020).

[15] HiPC Trust. *HiPC Conference*, 2020 (accessed 10, September, 2020).

[16] Indosys Committee. *Indosys Conference*, 2020 (accessed 10, September, 2020).

[17] CDER-2020. *PDC Curriculum Early Adopter Grant and Summer Training Program*, 2020 (accessed 10, September, 2020).

[18] NSF-Edu-Workshop. *NSF Workshop on Broadening Parallel and Distributed Computing Undergraduate Education*, 2020 (accessed 10, September, 2020).

[19] National Center for Super Computing Applications (NCSA). *Petascale Computing Institute*, 2020 (accessed 10, September, 2020).

[20] ATPESC. *Argonne Training Program on Extreme-Scale Computing*, 2020 (accessed 10, September, 2020).

[21] EduHPC-20. *Workshop on Education for High Performance Computing*, 2020 (accessed 10, September, 2020).

[22] EduPar-20. *NSF/TCPP Workshop on Parallel and Distributed Computing Education*, 2020 (accessed 10, September, 2020).

[23] EuroEduPar-20. *NSF/TCPP Workshop on Parallel and Distributed Computing Education*, 2020 (accessed 10, September, 2020).