Faculty Development Workshops for Integrating PDC in Early Undergraduate Curricula: An Experience Report

Sheikh Ghafoor
sghafoor@tntech.edu
Tennessee Technological University
Cookeville, Tennessee, USA

David W. Brown
Elmhurst University
Elmhurst, Illinois, USA
david.brown@elmhurst.edu

Mike Rogers
Tennessee Technological University
Cookeville, Tennessee, USA
mrogers@tntech.edu

Ada Haynes
Tennessee Technological University
Cookeville, Tennessee, USA
ahaynes@tntech.edu

Abstract

Parallel and Distributed Computing (PDC) has become pervasive and is now exercised on a variety of platforms. Therefore, understanding how parallelism and distributed computing affect problem solving is important for every computing and engineering professional. However, most students in computer science (CS) and computer engineering (CE) programs are still introduced to computational problem solving using an old model, in which all processing is serial and synchronous, with input and output via text using a terminal interface or a local file system.

Teaching a range of PDC knowledge and skills at multiple levels in Computer Science (CS) and related Computing and Engineering curricula is essential. The challenges are significant and numerous. Although some progress has been made in terms of curriculum recommendations and educational resources in computer science, trained faculty, motivation, and inertia are still some of the major impediments to introducing PDC early in computing curricula. The authors of this paper conducted a series of week-long faculty training workshops on the integration of PDC topics in CS1 and CS2 classes, and this paper provides an experience report on the impact and effectiveness of these workshops. Our survey results indicate such faculty development workshops can be effective in gradual inclusion of PDC in early computing curricula.

CCS Concepts

• **Social and professional topics** → **Computational thinking**; **Computational science and engineering education**; • **Computing methodologies** → **Parallel computing methodologies**.

Keywords

computer science education, parallel programming, faculty development

ACM Reference Format:
Sheikh Ghafoor, David W. Brown, Mike Rogers, and Ada Haynes. 2023. Faculty Development Workshops for Integrating PDC in Early Undergraduate Curricula: An Experience Report. In *Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W 2023), November 12–17, 2023, Denver, CO, USA.* ACM, New

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SC-W 2023, November 12–17, 2023, Denver, CO, USA
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0785-8/23/11…$15.00
https://doi.org/10.1145/3624062.3624097

York, NY, USA, 10 pages. https://doi.org/10.1145/3624062.3624097

1 Introduction

The emergence of multicore and GPU-based computing systems in recent years has brought about significant changes in the computing landscape. Every computing device from supercomputers and server farms containing multicore CPUs and GPUs to individual PCs, laptops, mobile devices, and embedded computing devices now relies on of parallelism and distribution. Parallel and Distributed Computing (PDC) has now become an integral aspect of nearly all computing activities. PDC concepts such as asynchrony, concurrency, decomposition, distribution, and parallel processing of big data objects has become fundamental aspects of learning to program computer applications. This shift in the computing landscape implies that programmers, regardless of their level of expertise, must possess at least some understanding of how parallelism and distributed programming influence problem-solving. Merely relying on traditional sequential programming skills is no longer sufficient, particularly for entry-level programmers. Consequently, there exists a clear demand for incorporating a comprehensive skill set in PDC technology throughout the educational programs offered by Computer Science (CS) and Computer Engineering (CE) programs, as well as in related computational disciplines. However, the rapid changes in hardware platforms, devices, languages, supporting programming environments continues to challenge educators in ascertaining appropriate content for curriculum and how to teach them effectively.

In many programs, foundational PDC concepts/topics are not introduced until upper level courses, many of which are electives. By this time, sequential programming patterns have been established in the student, and they think, and code, sequentially. Because this pattern has been reinforced for most of their academic career, many students who are exposed to PDC in this manner backslide and return to sequential programming once the course has been completed. Introducing PDC concepts in lower level classes can begin the task of allowing students to think in parallel, a notion that is extremely important in today's multicore world. Challenges are many, some significant challenges of integrating PDC in undergraduate curricula are:

- Lack of awareness among instructors about the need to integrate PDC concepts in their undergraduate programs
- Lack of trained instructors in PDC concepts
- Lack of instructor motivation to augment existing course with PDC concept
- Lack of teaching resources, such as text book, lecture slides, handouts, lab examples etc.
- Lack of sizable community
- Introductory courses are already overloaded with course material

We contend that organizing short training workshops for instructors whose graduate training and research areas are not PDC related and who typically teach early computing courses is one approach to alleviate some above-mentioned challenges. The authors of this paper had developed some curriculum materials for PDC integration in CS1 and CS2 (ref iPDC Modules). These curriculum materials cover some foundational concepts in PDC. We had also organized several faculty training workshops between 2018 and 2022. In this paper we report our experiences and lessons learned from the faculty development workshops as well as our evaluation of the impact of the workshop on faculty participants in integrating PDC in early computing classes. The evaluation results indicates that faculty training workshops are effective and impactful for integration of PDC in undergraduate computing programs.

2 Related Works

Faculty development workshops are a recognized form of imparting new proficiencies and ideas across a number of disciplines, and have become a common practice in higher education to enhance the knowledge and skills of their faculty members. [10], [19]. The benefits for the faculty are numerous, according to [3], participating in these workshops can result in increased job satisfaction and motivation, as well as improved teaching effectiveness. In addition, participation can lead to increased collaboration and networking opportunities among faculty members, which can lead to the development of new research projects and teaching strategies. Several studies have investigated the effectiveness of faculty development workshops in different contexts. For instance, [7] conducted a systematic review of the literature on faculty development workshops in engineering education. They found that workshops that focused

on active learning strategies, such as problem-based learning and inquiry-based learning, were effective in promoting student engagement, critical thinking, and problem-solving skills. Similarly, workshops that provided opportunities for faculty to collaborate with peers, reflect on their teaching practices, and receive feedback on their teaching were also found to be effective in improving teaching and learning outcomes.

Major forces encouraging integration of PDC topics into Computer Science curriculums are the ACM Curriculum Recommendations [1] and the NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing [13, 14, 16]. The ACM curriculum recommendations are the ACM's response to the need for Computer Science curricula that can keep pace with rapidly changing technologies and educational needs. The curriculum recommendations attempt to have a broad coverage and include Parallel and Distributed Computing (PD). The recommendations have 5 Core-Tier1 hours and 10 Core-Tier2 hours of PD that span a range of fundamental topics to advanced. The TCPP curriculum initiative is a curriculum guideline that intends to incorporate Parallel and Distributed topics so that graduates have basic skills in parallel and distributed computing. Additionally, it has topic recommendations for advanced and elective courses. The initiative also supports early adopter programs, publishes a book [15], and hosts other resources for those adopting PDC into their curriculum.

Research is ongoing on course materials and curriculum development. In [2], Adams proposes that CS2 is the natural place to introduce PDC concepts, and the author teaches the students using minimalistic parallel programming patterns, called *patternlets*. However, Brown, Shoop and Adams [1] assert that PDC concepts should be taught at all undergraduate levels. Researchers that have attempted to integrate PDC throughout the curriculum include Burtscher et al. [5] who taught PDC in several lower division courses and a senior capstone course. The authors show encouraging empirical results that measure student outcomes, engagement, and interest. Mullen et al. [11] created a self-paced online course focussed on parallelizing common High Performance Computing (HPC) use cases. Hundt et al. created Sauce which is a web application for the automatic evaluation of source code that can be incorporated as an interactive component in HPC computing lectures. Trejo-Sánchez et al. [17] presents a case study over four public universities in Mexico that shows some strategies that have been developed to incorporate the HPC concepts. These strategies attempt to overcome significant challenge in universities in developing countries like Mexico. Dewan et al. [6] created a module for hands-on training in Java Fork-Join abstractions for use in an interactive workshop for the "training of trainers" model. Newhall et al. [12] designed an introductory systems course that incorporates parallel computing, whose only prerequisite is CS1. The authors found that introducing parallel programming early helps the student gain confidence in their PDC abilities throughout their CS education.

3 Faculty Development Workshop

We have conducted five faculty development workshops from 2018 to 2022, and have adapted the content and structure of the workshops over time, especially in response to the pandemic situation in 2020 and 2021. We conducted three week-long, in-person workshops in 2018 and 2019. Each day of the workshops started at 8:30 a.m. and continued until 5:00 p.m. with two thirty-minute coffee breaks and a lunch break that was forty-five minutes long. We cancelled our 2020 in-person workshop by necessity because of the Pandemic. In 2021, we conducted the workshop over a two-week period, with virtual sessions lasting two hours each day via Zoom®. Additionally, we held one virtual office-hour for the participants. In 2022, we conducted the workshop in a hybrid format. This format consisted of one week of two-hour virtual sessions over Zoom®, followed by three days in person. Each workshop had fifteen participants on average. Most of them received a stipend to defray the cost of attendance. The virtual participants also received a stipend as an incentive. We selected participants whose graduate research topics and primary area of research is not PDC/HPC and do not teach PDC/HPC related courses. The contents of all three workshops were similar, with minor adjustments for later workshops based on feedback and lessons learned from previous workshops.

The objectives of the faculty training workshop were the following.

- Train faculty who teach introductory computing courses to write parallel programs in a shared memory environment.

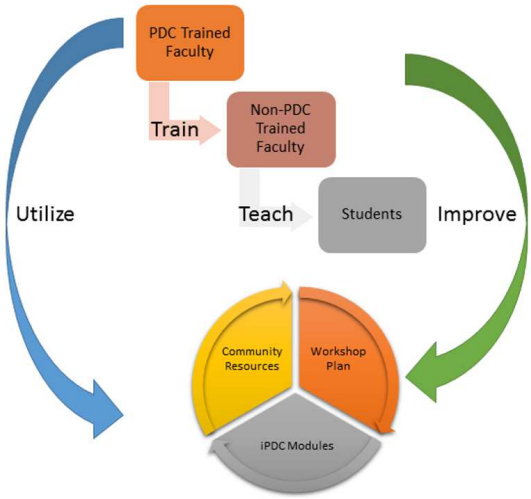


Figure 1: Training workshop Model

- Train non-PDC faculty to use the iPDC instructional modules support system
- Disseminate findings/results/resources of successful PDC education programs
- Build a community among instructors teaching introductory computing courses
- Share PDC integration strategies, research-based practices and field-tested PDC resources

The approach of the workshop is to treat the participants as advanced graduate students and have them complete a short intense course on parallel computing in a shared-memory environment. We used a hands-on method that involved writing many programs using different technologies. These programming activities were then followed by a participatory exercise and discussion on how to integrate PDC concepts and exposure of technologies to students in early computing courses. We encouraged the free-flow of ideas through collaborative discussion, showing practical examples, and sharing curriculum materials developed by the authors and others in the PDC education committee. Primarily, the workshops had three types of activities: 1) introduction of foundational concepts of parallel computing through lecture and unplugged activities to illustrate foundational concepts, 2) parallel program writing for shared memory environment using technologies such as OpenMP and Java threads, and 3) discussion and presentations on how to integrate PDC in early computing classes and community building. The model of our approach is shown in Figure 1.

3.1 Foundational Concepts

The objective of this activity was to explain core foundational concepts of parallel computing to the participants. One of our challenges was to determine the minimum set of topics to which students should be exposed in early computing classes. We decided to cover the four foundational concepts recommended by NSF/IEEE curriculum recommendations [16]. The recommended foundational concepts are: *Asynchrony*, *Concurrency*, *Locality*, and *Parallel Metrics*. We included these concepts in the minimal set and, in addition, we included *task and data decomposition*, *when to parallelize a problem (Amdahl's law)*, and *memory hierarchy* in the minimal set. We believe a student that understands these topics to some extent early in their curriculum would be better prepared to grasp PDC concepts and technologies when exposed to them in advance classes. We also cover some additional topics, such as parallel architecture, cache coherency, false sharing etc. We emphasise during the training that the instructor should expose the students to the foundational concepts at the Bloom's *comprehension* level. We have used lectures, discussions, unplugged activities [8, 9], and small in-class assessment quizzes to explain these topics to the participants.

3.2 Plugged Hands-On Activities

The objective of this activity was to teach the participants thread-based shared-memory programming technologies. We strongly believe that if the faculty participants become comfortable writing shared memory parallel programs using languages and technologies that are typically used in early computing classes, then they will be more inclined to introduce students to simple parallel program development. The most common languages that are used in early programming classes are C/C++, Java, and Python.

OpenMP has been the de facto standard thread library for parallel programming in shared-memory environments since the late nineties. OpenMP is a thread library that hides complexities of managing thread creation, scheduling, and synchronization from the users, and lets them focus on the algorithms. OpenMP provides compiler directives with associated clauses, a set of

functions, and environment variables, and is very easy to use. OpenMP comes standard with C and C++ compilers. Java and Python are the two of the most used languages in the early programming classes. However, no OpenMP thread libraries for JAVA or Python are robust or widely used. We have extended a Java-based OpenMP thread library [18], and developed a Python OpenMP library. These Java and Python OpenMP libraries implement a core set of directives, clauses and functions. We covered the following directives, clauses, and functions in depth during the workshop:

Directives: `parallel`, `critical`, `atomic`, `for`
Clauses: `default`, `private`, `shared`, `schedule`, `reduce`
functions: `Omp_get_numthreds`, `Omp_get_threadnum`, `Omp_set_numthreds`, and `Omp_get_wtime`

Apart from these directives and functions, we also briefly introduced other OMP directives and functions, and directed the participants to external resources. We also covered writing parallel programs using Java threads. This part of the workshop was hands-on. We spent a small amount of time lecturing to cover parallel program design methodologies and an introduction to parallel program technologies and constructs. The participants developed and executed multiple parallel programs using OpenMP in C/C++, Java, and Python, as well as using Java threads. The participants were given a series of problems, and they developed parallel solutions to those problems in groups. In some cases, participants were supplied with a serial solution of the problems. Participants were given a certain amount of time to solve the problem. After that allotted time, the instructors discussed their solutions with participants. These activities were repeated for every exercise problem. This hands-on thread-based parallel program writing covered 60% of the workshop. Some example programming assignments are available in [8].

3.3 PDC Integration and Community Building

The objectives of this activity are to 1) help develop each participant a plan to integrate PDC in their classes, 2)connect the faculty participants to the greater PDC education community, and 3) form a close bond among the participants so that they can exchange ideas, and help each other when they try to integrate PDC topics in their respective courses. To achieve the first objective from the beginning of the workshop, the participants were asked to develop a plan to integrate PDC into their respective classes. This planning was done incrementally as the participants progressed through the workshop. Each workshop day, time was set aside to discuss their plan. This discussion was done collaboratively, and allowed other participants and the organizer to provide feedback and suggestions. On the last day of the workshop, each participant submitted a written implementation plan on how they would integrate PDC topics in their class, and they presented their plan. The organizer and other participants provided feedback to the presenter. We provided the participants resources and examples of successful PDC integration in early computing classes.

For the second and third objectives, we adapted several strategies. Before the workshop we formed groups of three to four participants, and each group worked together throughout the workshop. During the in-person part of the workshop, the participants attended lunch and coffee breaks together. In addition, generally one afternoon an excursion was arranged (usually a short hiking trip or sightseeing) followed by a working dinner at a local restaurant. Most participants also stayed at a workshop designated hotel. Such close association helped in bonding and community building. We also provided participants information about PDC education conferences (cite edu * workshops), invited them to participate in these conferences in different capacities (author, volunteer, attendee, program committee member), and encouraged them to get involved with CDER activities, including CDER weekly meetings.

4 Results

To aid in determining the impact of our faculty development workshops, we conducted a survey among the workshop participants. About 10% of the participant's contact information had changed, and unfortunately, no updated contact information was available. Therefore, we sent the summative survey link to the remaining 62 participants that participated during the five years of the study.

Figure 2 shows both a breakdown of faculty participation and continued efforts by year and cumulatively. In particular, Figure 2a shows how many participants responded to the survey, how many implemented what they had learned in the workshop, and how many still continue to introduce the topics in their courses. Of these respondents, 31 (50%) completed the survey. This completion rate can be considered a good response rate for an email survey. The highest number of responses was from the participants of the most recent workshop in 2022, with about 78% percent of those participants responding. The response rate from the 2020 virtual workshop was lowest, which was about 26%. The response rate from the 2018–2019 workshops were 33%. Figure 2b, shows the same data in Figure 2a but in a year-by-year cumulative manner. From figure 2b we can see 87% (27 out of 31) have incorporated some PDC topics in their respective classes. We can also see that 74% (20 out of 27) participants are still continuing integrating PDC in their course. For a variety of reasons 26% (7 out 27) participants discontinued to integrate PDC in their coursework.

Figure 3 shows the reasons participants gave for discontinuing integration of PDC topics. The most common reasons are: they lacked the time needed to modify existing courses, they were not scheduled to teach an introductory course, and their classes were disrupted due to switching from an in-person to virtual format during the pandemic.

During the survey we asked the faculty what pedagogical methodologies they used for introduction of PDC topics. Figure 4 show the responses. Lectures (96%) and unplugged activities (78%) were the most commonly used methodologies, followed by demonstration (70%) and program writing (59%). From the responses it appears that the faculty thought that lectures, unplugged activities, and demonstrations are the most effective and perhaps easiest to adopt for introducing PDC concepts. This indicates developing a variety of effective curriculum materials for these methodologies would likely broaden the further adoption of PDC.

Many core PDC concepts and topics were covered during the workshop. In the survey we asked the faculty participants what topics they have introduced in their courses in what depth. Figure 5 shows the responses. We can see that *concurrency*, *synchronization*, *parallel overhead*, *speed up*, and *serial versus parallel* are the most common topics that faculty covered topics.

In addition, we asked the participants what would further help them continue their PDC integration efforts. Figure 6 shows their responses. The respondents indicated that focus on further development of exemplars, additional training, and creation of a cohort for continuing research and publishing on PDC topics would help the trained faculty continue their implementation efforts.

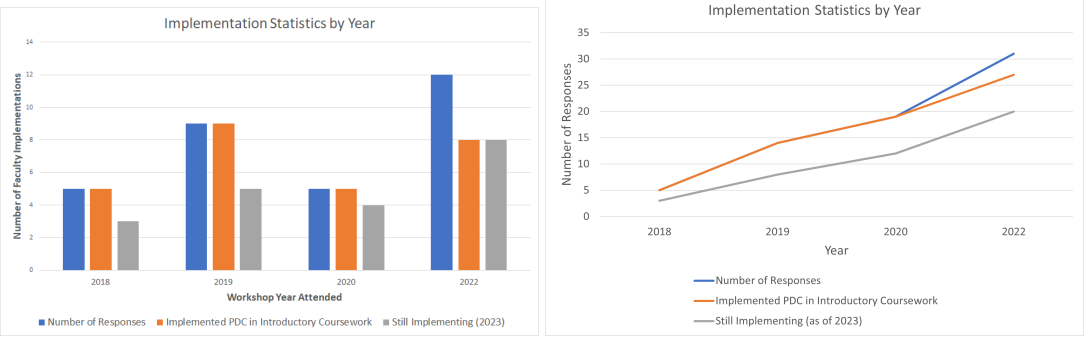
Perhaps the most important result from the survey is about student impact. We wanted to know how many students were introduced to PDC topics. We can see from Figure 7 that as the number of trained faculty and their PDC integration efforts continues the number of student exposed to PDC concepts and topics increased each year and reached about 4,500 students.

5 Lessons Learned

We identified what worked and what did not work in our workshops. We believe that this information can improve the success of workshops by using it to improve our future workshops, and also improve similar workshops that will be hosted by others. Furthermore, we learned important lessons in *practical community building*, *the longevity of PDC topic integration*, and *the density and content of workshop activities*.

5.1 Practical community building

Over a period of 5 years we have conducted the training workshops in three manners: 1) week-long in-person workshop, 2) two-week-long completely virtual workshop, and 3) hybrid workshop, consisting of two week-long virtual workshop meetings and three days of in-person workshop meetings. Based on the participant feedback, hybrid workshops seem to be the most effective. In-person workshops enhance community building. The added interactivity and collegiality helps the



(a) Implementation and Continuation by Year Attended **(b) Cumulative Implementation and Continuation**

Figure 2: Breakdown of faculty participation and continuation by year and cumulative.

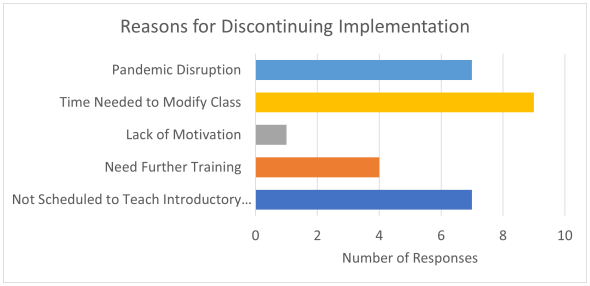


Figure 3: Reasons for Discontinuing PDC Integration

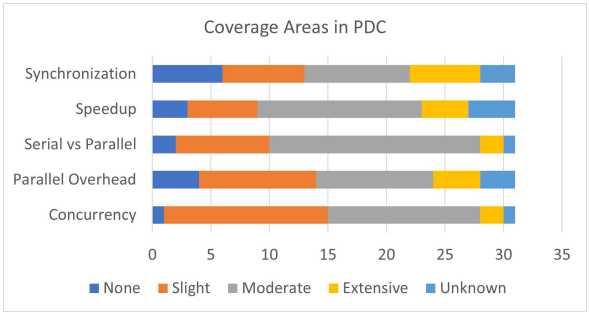


Figure 5: Extent of PDC Topics Coverage

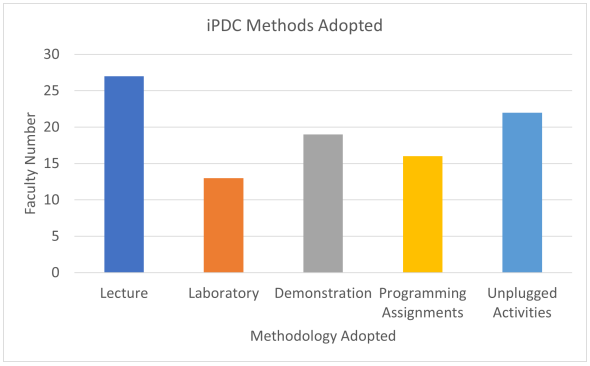


Figure 4: Pedagogical Methodology Used

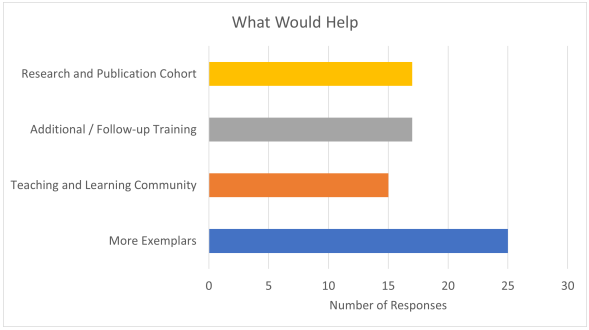


Figure 6: What Would Help Continuation

participants determine the best approaches to implement PDC in early classes. In-person workshops provide opportunities to discuss ideas with fellow participants and obtain feedback. However, week-long in-person workshops can be impractical for participants because of the time commitment and cost required. Virtual workshops provide flexibility of participation. Participants can join from anywhere and thus save both travel time and cost. However, day-long virtual sessions have their own issue: the participants get *Zoom fatigue* [4]. A hybrid approach provides training in the PDC technologies (OpenMP, Java threads, etc.) via virtual training sessions, and provides the benefit of the

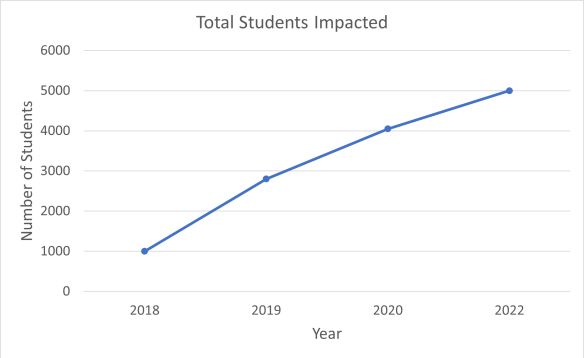


Figure 7: Students Introduced to PDC Topics

community building aspects in a shorter in-person session. It also shortens the time commitment away from family and reduces travel costs. A virtual component also helps by incorporating environment set up (compiler, runtime libraries, etc.) so that participants can spend more time problem-solving and interacting during the in-person component. In our opinion such workshops should be organized in a hybrid manner, taking advantage of both virtual and in-person meetings. The virtual meetings can cover environment setup, learning parallel program writing etc., and the in-person part can focus on motivating, community building, and discussing pedagogical methodologies of integrating PDC. The virtual part can be augmented with video lectures and virtual office hours too.

5.2 Longevity of PDC topic integration:

Unfortunately, the continuation of PDC integration efforts by workshop participants diminished over time. We believe that a few modifications to the workshops and some additions would improve the longevity of integration.

In particular, a formal way of community building is needed. Typically, immediately after the workshop, some participants keep in touch with one another and also with the organizers. Unfortunately, as time continues, this communication gradually subsides, and the decrease of communication was not affected by the type of workshop (in-person, virtual, or hybrid). Many participants mentioned additional follow-up training would help to continue integration of PDC. Therefore, an effort by the workshop organizers and members of the PDC community to keep the participants engaged through a follow-up training and discussion forum or conference participation would be helpful.

5.3 Density and content of workshop activities

Another important lesson that we learned from our experience and from participant feedback is to limit the number and variety of activities. If the participants are overloaded with too much, then the workshop becomes confusing, which affects participant morale. For example, in the 2018 workshop, we included an exercise in which each participant set up an account in our university cluster and submitted jobs. At first, the activity seemed to be a simple one, but difficulties in account setup, job script creation, compiling in an unfamiliar environment, etc. required significant time. The activity did not add to the participants' understanding, but only led to confusion and discouragement. Instead, we found that a focus on more hands-on programming activities in familiar environments (their own laptops) and less lecture led to better understanding and engagement.

As indicated in Figure 5, the workshop participants provided important feedback about the content of the workshops that will help us plan future workshops. In particular, the topics *serial versus parallel*, *speedup*, and *concurrency*, followed by *sychronization* and then *parallel overhead* were the topics the participants covered most thoroughly in their own classes. The reason that these topics were favored for coverage could be because either the topics were more easily integrated into their courses, the faculty determined that these topics were the most important, or both. Regardless, giving more focus to materials, exemplars, and instruction for these topics should increase the efficacy of future workshops.

6 Conclusion and Future Work

We have conducted a series of faculty training workshop on integrating parallel and distributed computing concepts and topics in early computing classes. Our approach was training of the trainers with the hope that will have a multiplicative effect. Our survey result indicate that our workshops were mostly successful. Most of the participants (77%) included some aspects of PDC in their courses, and the majority of those continued to integrate PDC as one of the topics in their

classroom. There PDC integration efforts by workshop participants diminished over time, this is somewhat expected, time and effort required is one of the major factor. Informal conversation with the past participants at conferences indicate the PDC topics for early computing classes has not yet become mainstream and part of the accreditation process. As a result, there is an inertia that needs to be overcome. Lack of curriculum material is another major factor. Our survey also indicates that more PDC curriculum materials and exemplars (lectures slides, handout, labs, text book etc.) for early computing classes would help in broader continuing adoption of PDC for early computing classes. Even though ACM and ABET has recommended PDC as a required core knowledge area, PDC has not yet been adopted in most of the computing programs. Growing the community who will adopt, champion adoption of PDC concepts/topics in early computing classes, develop new exemplars is needed to make PDC concepts/topics as one of the core topics for early computing classes. Faculty development workshops is one of the mechanism of growing the community.

In future we plan to continue the faculty training workshops. We plan modify our workshops based on the feedback and lesson learned. We also plan to develop a complete online version of the training workshop that and faculty can go through asynchronously and have it freely available for the community. Developing more curriculum materials is also part of our future plan.

References

[1] ACM. 2013. Computer Science 2013: Curriculum Guidelines for Undergraduate Programs in Computer Science.

[2] Joel C Adams. 2014. Injecting Parallel Computing into CS2. In *SIGCSE '14 Proceedings of the 45th ACM technical symposium on Computer science education*. ACM, Atlanta, Georgia, USA, 277–282. https://doi.org/10.1145/2538862.2538883

[3] Catherine E Brawner, Richard M Felder, Rodney Allen, and Rebecca Brent. 2002. A survey of faculty teaching practices and involvement in faculty development activities. *Journal of Engineering education* 91, 4 (2002), 393–396.

[4] Arthur C. Brooks. 2023. The Trouble with Zooming Forever. https://www.theatlantic.com/family/archive/2022/07/how-to-fight-zoom-fatigue/670513/ Publisher: The Atlantic.

[5] Martin Burtscher, Wuxu Peng, Apan Qasem, Hongchi Shi, Dan Tamir, and Heather Thiry. 2015. A Module-based Approach to Adopting the 2013 ACM Curricular Recommendations on Parallel Computing. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. ACM, 36–41.

[6] Prasun Dewan, Andrew Wortas, Andrew Worley, James Juschuk, Samuel George, Mike Rogers, Felipe Yanaga, and Sheikh Ghafoor. 2022. Hands-On, Instructor-Light, Checked and Tracked Training of Trainers in Java Fork-Join Abstractions. In *EduHiPC 2022: 4th Workshop on Education for High Performance Computing*. Bangalore, India.

[7] Richard M Felder and Rebecca Brent. 2010. The National Effective Teaching Institute: Assessment of impact and implications for faculty development. *Journal of Engineering Education* 99, 2 (2010), 121–134.

[8] Sheikh Ghafoor and Mike Rogers. 2016. iPDC Workshop. http://www.csc.tntech.edu/pdcincs/

[9] Sheikh K. Ghafoor, David W. Brown, Mike Rogers, and Thomas Hines. 2019. Unplugged Activities to Introduce Parallel Computing in Introductory Programming Classes: An Experience Report. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education* (Aberdeen, Scotland Uk) *(ITiCSE '19)*. Association for Computing Machinery, New York, NY, USA, 309. https://doi.org/10.1145/3304221.3325573

[10] Alexandra Milliken, Christa Cody, Veronica Catete, and Tiffany Barnes. 2019. Effective computer science teacher professional development: Beauty and joy of computing 2018. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. 271–277.

[11] Julia Mullen, Chansup Byun, Vijay Gadepally, Siddharth Samsi, Albert Reuther, and Jeremy Kepner. 2017. Learning by doing, High Performance Computing education in the MOOC era. *J. Parallel and Distrib. Comput.* 105 (2017), 105–115.

[12] Tia Newhall, Kevin C. Webb, Vasanta Chaganti, and Andrew Danner. 2022. Introducing Parallel Computing in a Second CS Course. In *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 321–329. https://doi.org/10.1109/IPDPSW55747.2022.00064

[13] Sushil K Prasad, Anshul Gupta, Krishna Kant, Andrew Lumsdaine, David Padua, Yves Robert, Arnold Rosenberg, Alan Sussman, and Charles Weems. 2012. Literacy for All in Parallel and Distributed Computing: Guidelines for an Undergraduate Core Curriculum. *CSI Journal of Computing* 1, 2 (2012).

[14] Sushil K. Prasad, Anshul Gupta, Arnold Rosenberg, Alan Sussman, and Charles Weems. 2023. CDER Center | NSF/IEEE-TCPP Curriculum Initiative. https://grid.cs.gsu.edu/~tcpp/curriculum/?q=node/21183

[15] Sushil K. Prasad, Anshul Gupta, Arnold L. Rosenberg, Alan Sussman, and Charles C. Weems. 2015. *Topics in Parallel and Distributed Computing: Introducing Concurrency in Undergraduate Courses* (1st ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[16] Sushil K Prasad and Anita La Salle. 2012. NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing -Core Topics for Undergraduates. http://www.cs.gsu.edu/~tcpp/curriculum/index.php

[17] Joel Antonio Trejo-Sánchez, Francisco Javier Hernández-López, Miguel Ángel Uh Zapata, José Luis López-Martínez, Daniel Fajardo-Delgado, and Julio César Ramírez-Pacheco.

2022. Teaching High-Performance Computing in Developing Countries: A Case Study in Mexican Universities. In *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 338–345. https://doi.org/10.1109/IPDPSW55747.2022.00066

[18] Vikas, Nasser Giacaman, and Oliver Sinnen. 2013. Pyjama: OpenMP-like Implementation for Java, with GUI Extensions. In *Proceedings of the 2013 International Workshop on Programming Models and Applications for Multicores and Manycores* (Shenzhen, Guangdong, China) *(PMAM '13)*. Association for Computing Machinery, New York, NY, USA, 43–52. https://doi.org/10.1145/2442992.2442997

[19] George Watson. 2006. Technology professional development: Long-term effects on teacher self-efficacy. *Journal of Technology and Teacher Education* 14, 1 (2006), 151–166.