Teaching PDC in the Time of COVID: Hands-on Materials for Remote Learning

Joel C. Adams
Calvin University
Grand Rapids, MI, USA
adams@calvin.edu

Richard Brown
St. Olaf College
Northfield, MN, USA
rab@stolaf.edu

Suzanne J. Matthews
U.S. Military Academy
West Point, NY, USA
suzanne.matthews@westpoint.edu

Elizabeth Shoop
Macalester College
St. Paul, MN, USA
shoop@macalester.edu

***Abstract*—In response to shifts in the hardware foundations of computing, parallel and distributed computing (PDC) is now a key piece of the core CS curriculum. For CS educators, the COVID-19 pandemic and the resulting switch to remote-learning add new challenges to the tasks of helping learners understand abstract PDC concepts and equipping them with hands-on practical skills. This paper presents several novel teaching materials for teaching PDC remotely, including: (i) using a Runestone Interactive "virtual" handout to learn how to run OpenMP multithreaded programs on a Raspberry Pi, and (ii) using Google Colab and Jupyter notebooks to run mpi4py instances on remote systems and thus learn about MPI distributed multiprocessing. The authors piloted these strategies during a multi-day faculty development workshop on teaching PDC. Assessment data indicates that the materials greatly aided professional development and preparedness to teach PDC.**

***Index Terms*—Raspberry Pi, Runestone Interactive, Google Colab, Remote Education, Parallel & Distributed Computing, MPI, OpenMP, Python**

I. Introduction

Prior to 2006, most computers were uniprocessors, meaning they had central processing units (CPUs) that could only execute one machine instruction at a time. Virtually all of today's computers are *parallel multiprocessors*, meaning their multicore CPUs can execute multiple machine instructions simultaneously. Over roughly the same time span, *cloud computing services* have become available. These allow a computation to be *distributed* between a local machine and one or more remote "in the cloud" machines.

To take advantage of these seismic shifts in modern computing's hardware foundations, computer science educators need to be teaching their students about parallel and distributed computing (PDC) to prepare them for careers in modern software development.

Accordingly, the CS curriculum recommendations from the IEEE Technical Committee on Parallel Computing (TCPP) recommend that every CS student learn about PDC [1], and the ACM/IEEE CS 2013 report added a new PD knowledge area to the CS core curriculum [2]. Likewise, the Accreditation Board for Engineering and Technology (ABET) now requires accredited CS programs to demonstrate that all of their students learn about PDC.

National Science Foundation DUE-1822480/1822486/1855761

One way to expose every CS major to PDC is to inject PDC topics into existing core CS courses [3]–[7]. As examples: a *Computer Organization* course should cover multicore architectures [8]; an *Algorithms* course could include parallel sorting algorithms; a *Programming Languages* course can include coverage of the distributed computing message-passing primitives in languages such as Scala or Erlang; and so on. Using this approach (i) students learn about PDC topics as they fit naturally within the context of existing courses; (ii) they acquire hands-on PDC skills throughout the CS curriculum; (iii) PDC topics can be introduced early and revisited later in greater depth, using a "spiral" pedagogy [9]–[12]; and (iv) no new courses need to be added to the crowded CS curriculum.

Teaching PDC is challenging under ordinary circumstances; the need to teach remotely increases the difficulty further by requiring that all learning activities occur online, potentially asynchronously. These issues were heightened during the COVID-19 pandemic of 2020-2021, when the need for free, high-quality, interactive, remotely accessible PDC materials became evident. The authors identified the following high-level goals for teaching PDC in a remote environment:

> **Goal 1**: To provide effective conceptual and hands-on learning about *multicore* parallel computing.
>
> **Goal 2**: To provide effective conceptual and hands-on learning about *distributed* parallel computing.
>
> **Goal 3**: To identify what types of educational PDC experiences are especially useful to learners.

To enable educators to achieve the above goals, we developed a series of novel teaching materials [13]–[16] that allow instructors to introduce multicore and distributed computing concepts in a remote environment. The materials are freely available, and enable educators to quickly inject multicore and distributed computing concepts into existing courses, without having to develop additional materials themselves. We piloted the use of these materials at a virtual summer workshop attended by 22 faculty interested in teach PDC. Preliminary assessment suggests that they are an engaging way of learning about PDC in a remote environment.

The rest of this paper presents related background work, an overview of the materials we developed, an initial assessment of these materials, lessons learned, and conclusions.

II. Background and Related Work

Software. Two popular libraries for PDC are *OpenMP* for shared memory on multicore machines, which is built into nearly all modern C/C++ compilers, and the *Message Passing Interface (MPI)* for distributed C/C++ computations on a multicore machine or Beowulf cluster. Using patternlets [17] and scaffolding code, OpenMP and MPI make it relatively easy to introduce PDC in courses that use C/C++. For Python-based courses, the *mpi4py* library [18] provides a Python-based MPI API, and for Java-based courses, the *OpenMPI* distribution [19] provides a Java-based MPI API for teaching message-based distributed computing.

Patternlets for PDC. Software design patterns are general, reusable solutions for commonly occurring programming problems, emerging from decades of experience of industry professionals. Design patterns for object oriented software were introduced in 1993 [20] and were popularized by the book by the same authors [21]. Subsequently, others identified design patterns for parallel programs [22], [23]. In particular, the OPL patterns project [24] led by Kreutzer (Berkeley) and Mattson (Intel) provides a hierarchically-structured, comprehensive organization of parallel patterns plus a problem-solving methodology. Patterns in PDC programming represent an opportunity to teach "parallel thinking" to students, i.e., time-tested PDC problem-solving practices based on decades of accumulated wisdom of industry professionals.

In 2015, Adams introduced *patternlets* as very short example PDC programs (e.g., OpenMP, MPI, or pthreads), each illustrating a specific parallel programming pattern [17]. The brevity of the code and the hands-on experience of running that code themselves gives even introductory students an accessible start at learning parallel thinking through patterns. The students gain a rapid initial understanding of key programming patterns, which have proven essential for effective PDC programming competence. The materials presented in this paper use patternlets to introduce key multicore and distributed computing concepts.

Single Board Computers as Manipulatives. Mathematics education researchers have produced extensive evidence about pedagogical tools known as *manipulatives* [25]–[28]. These are concrete objects that students can grasp in their hands, such as Base-10 Blocks, fraction strips, and interlocking cubes, which can lead them to effectively use mathematical symbols [29]. This idea dates back to Papert, who described manipulatives as "objects-to-think-with" [30]. Mathematics education research has shown this approach to be effective in helping students grasp abstract mathematical concepts, including algebra [31], fractions and ratios [27], and computation and problem solving [26], [28], [32]. Other researchers note that manipulatives make learning enjoyable and fun, increasing learner motivation [28], [33].

Inspired by this precedent, we view the Raspberry Pi [34] and similar single-board computers (SBCs) as manipulatives for CS students to learn about PDC. More precisely, a SBC allows students to see and touch system components such as the CPU, GPU, memory, network connections, and so on, and students can connect multiple SBCs to form their own Beowulf cluster [35]. As a tangible device, an SBC can help CS students progress from concrete experience to visual representations of abstract concepts (e.g., figures and diagrams), and finally arrive at a deeper understanding of PDC abstractions.

The Raspberry Pi SBC was introduced in 2012 as a learning platform for students and hobbyists. Raspberry Pi clusters soon followed [36]. The first multicore Raspberry Pi was introduced in 2015, although other multicore SBCs had been used previously in CS education, including the ODROID SBCs used by Toth [35], and the Parallella SBC used by Matthews [37]. With suitable software configuration, the Raspberry Pi serves as the basis for an inexpensive kit (described in Section III A) that utilizes a laptop for monitor, keyboard, and mouse.

Online interactive computational platforms. Originally used by researchers to share Python code and results of analyses, iPython and its successor Jupyter [38], [39] are now being used in education [40], [41]. These notebook systems can be used to provide online interactive learning documents, in which students can read expository text and then run computations, which they can then modify and run again. Recently, researchers have created environments for teaching PDC that use Jupyter notebooks as GUI interfaces to connect to back-end compute servers, improving on the traditional terminal plus command-line interface to a compute server [42].

A web-only variant of Jupyter notebooks called Google Colab [43] extends this concept to the browser, also adding Google-doc features such as sharing and commenting. Unlike Jupyter notebooks, the Colab system does not require back-end server setup or front-end notebook software setup. One of the contributions of this paper is our discovery that these notebooks can be used to demonstrate MPI programming using mpi4py, which we describe in section III-B.

Besides notebook systems such as Jupyter, online interactive textbook platforms have emerged for CS in recent years. One of these is Runestone, which debuted in 2014 with a CS1 Python textbook [44]. Runestone Interactive now provides numerous textbooks for CS courses at all undergraduate levels, with enhanced learning features such as interactive activities and exercises. Runestone also offers teaching support such as course and assignment management for students. Some notebooks perform a computation by converting the code written in one language (e.g., Python) to JavaScript, which then runs within the user's browser. By contrast, Runestone and Jupyter notebooks are front-end, browser-based interfaces for back-end computations that typically run on a web server. Usually these computations are written in scripting languages; however back-end extensions exist for building and running non-scripted computations (e.g., C programs in a Data Structures book or SQL computations in a Databases text).

III. Overview of Materials

To teach PDC concepts in a remote environment, we developed two modules of materials, one for teaching shared memory parallel concepts and one for distributed memory

parallel concepts. We designed the shared memory module as a hands-on activity using OpenMP examples running on a Raspberry Pi SBC, with the exercise and code delivered via an interactive module [13] developed using Runestone Interactive. We designed the distributed memory module to use MPI examples [15] running on any of several remote clusters, with the exercise and code delivered via a combination of Google Colab [14] and Jupyter Notebooks [16]. Both of these modules [13], [14] are freely available for any instructor to adapt to their courses. We designed these modules to be self-paced, so that learners could work through these activities asynchronously. To match the duration of a standard lab period at many institutions, we designed each of these two activities to take approximately 2 hours. We give a brief overview of these materials in Section III-A and Section III-B below.

A. Materials for teaching Multicore Computing Remotely

The first module focuses on learning shared memory parallel concepts using OpenMP on the Raspberry Pi. We designed this module to use $100 Raspberry Pi kits that could be sent to remote learners in the mail, if they did not already own one. These kits included a customized Raspberry Pi system image [45] that could be mailed to remote learners who already own a Raspberry Pi. Table I provides details of our kits.

TABLE I
APPROXIMATE COST BREAKDOWN OF MAILED RASPBERRY PI KIT

Part	**Cost**
CanaKit with 2G Raspberry Pi [46]	$62.99
Ethernet-USB A dongle	$15.95
USB A-C dongle	$3.99
Ethernet cable	$1.55
16G MicroSD	$5.41
Kit case	$10.77
Total Kit Cost	$100.66

Note that in addition to the Raspberry Pi and its power supply, these kits included an Ethernet cable and a Ethernet-to-USB dongle (if needed) for connecting the Pi unit to a personal laptop or desktop. Note also we could build these kits for approximately $100 because several of these materials can be bought in bulk. These kits represent a significant innovation over the Pimoroni-based kits described in [47], which were more expensive, bulkier, and whose image only worked for limited hardware configurations.

The MicroSD cards in these kits contain the system image for the Raspberry Pi plus the OpenMP code examples for our shared memory parallel computing module. This image was tested and confirmed to work on the all Raspberry Pi models from the 3B onward. To keep these custom images up to date, we use Ansible and other software maintenance tools. This image is freely available for instructors and learners to download and use in their courses [45].

We also used Runestone Interactive [44] to create a free, online interactive "virtual" stand-alone module [13]. This module includes instructional videos showing learners how to set up their Raspberry Pi devices and begin using our custom system image. The free availability of this Runestone module and Raspberry Pi image, plus the inexpensive kit detailed in Table I make it easy and seamless for instructors to incorporate these materials directly into their own classrooms.

To introduce learners to shared memory parallel computing concepts, the Runestone module uses OpenMP C/C++ patternlets [17]. Our module has learners perform the handout's activities on their Raspberry Pi devices, so we did not use the Runestone Interactive *Active Code* feature. However, we incorporated other Runestone Interactive features, including video explanations, visualizations, and interactive questions (e.g., multiple choice, fill in the blank, drag-and-drop) to quiz the reader on key concepts. Figure 1 shows a small component of the virtual module where we explain the concept of race conditions in a video, and then encourage participants to check their understanding by answering a multiple-choice question.

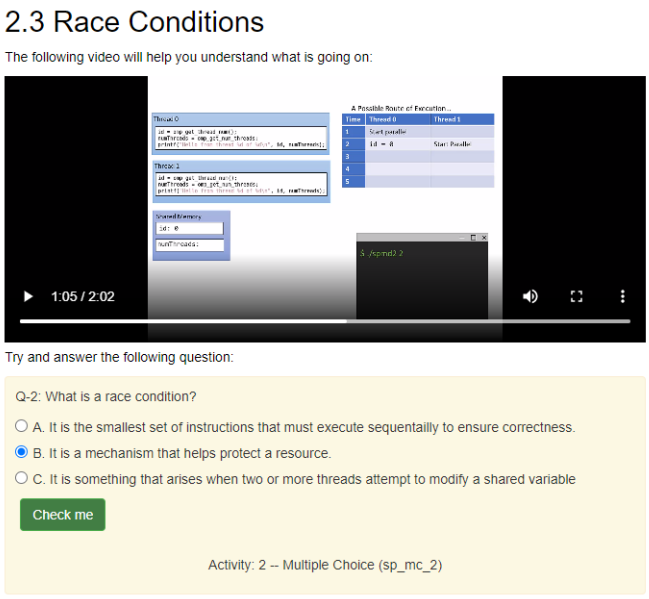

Fig. 1. View of small portion of Raspberry Pi virtual module.

This virtual module is designed to be completed in a self-paced 2-hour period. The first half hour presents an overview of processes, threads and multicore systems, and gives a short introduction to the OpenMP patternlets. During the next hour, learners work through a hands-on exercise in which they explore the patternlets at their own pace. The last half hour examines two OpenMP exemplars: numerical integration and drug design. In this manner, learners are exposed to general concepts and vocabulary, simple examples, more complex programs, and finally perform a small benchmarking study to reinforce the concepts introduced at the beginning of the module. The module is thus designed for use by students working remotely in a single lab period, either synchronously or asynchronously. It can be used either as a stand-alone introduction to parallelism or in conjunction with lectures developed by the instructor.

B. Materials for teaching Distributed Computing Remotely

Unlike the shared memory paradigm, MPI carries out parallel programming using independent *processes* that communicate with one another by sending and receiving messages. The mpi4py Python library provides a Python interface to the library of (C) MPI functions. To provide an activity in which remote learners could experience the message passing paradigm, we developed a module of mpi4py instructional materials for remote use on three different platforms:

1) Colab [43] is Google's free web-based variant of Jupyter notebooks, for authoring and executing Python code. The code runs on a single core Google Cloud VM, not a distributed system, but the key concepts of message passing can still be demonstrated. Our Colab material is freely available for educational use [14]; it requires a Google account to save in one's Google Drive space.
2) A Jupyter notebook connected to a cluster on the Chameleon Cloud [48], a cloud-based test-bed for experimental computer science research and teaching [49]. This cluster is a powerful high-performance distributed system, enabling its users to experience the speed and scalability of distributed computing. Chameleon users may contact us for a copy of this Jupyter notebook.
3) A VNC connection to a 64-core VM running on a large server at St. Olaf. Like Chameleon, this server provided good parallel speedup and scalability when running our mpi4py exemplar applications. Our material for using this VM is also freely available for educational use [15]. Faculty require access to their own server or an account on the St. Olaf system.

For the first hour of this module, learners use the Google Colab material to explore MPI patternlets (see Figure 2). After a brief introduction to Colab notebooks and running Python examples, learners work at their own pace through the patternlets material, providing them with hands-on experience applying MPI concepts and terminology.

In the second hour of this module, learners explore either of two exemplars that use the message passing patterns introduced during the first hour: (i) a Forest Fire Simulation, or (ii) a Drug Design example. Learners work through whichever of these examples most interested them; they also choose to use either the aforementioned Chameleon-backed Jupyter notebook or use a VNC client to connect to the St. Olaf VM where they complete these activities using a Linux terminal. After a brief explanation of these options, participants choose one and proceeded through the exercise at their own pace.

IV. Assessment & Results

To assess the effectiveness of these modules, we used them as teaching materials in a 2.5 day virtual PDC faculty development workshop in July 2020 [50]. The workshop participants worked through our first (shared memory) module during a 2-hour session on the first morning, and through the second (distributed memory) module the second morning.

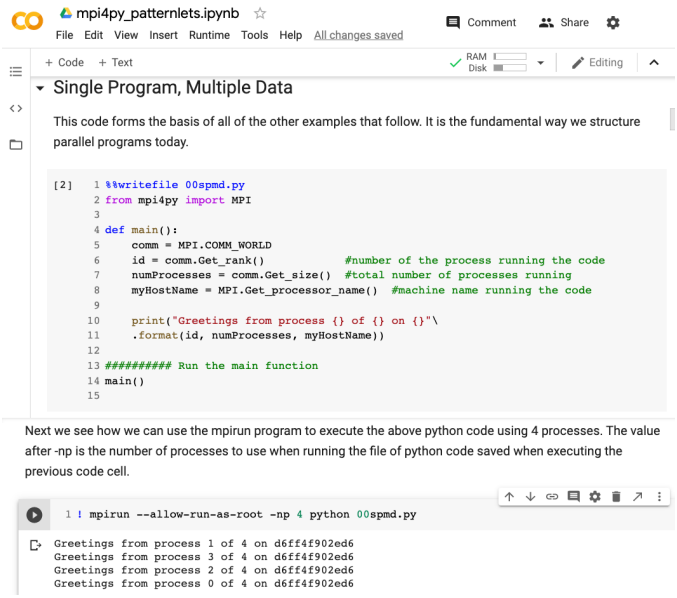

Fig. 2. View of small portion of colab notebook.

The workshop's afternoon sessions were devoted to other demonstrations and discussions related to PDC education.

The 22 participants in our workshop were a mix of faculty members (85%) and graduate students (15%). Of these, 19 were from institutions in the continental U.S., one was from Puerto Rico, and two were international. 77% of the attendees identified as male, 18% as female, and 5% as other. 46% of the attendees were tenured or tenure track at their institutions; 39% were non-tenure track; the 15% who were graduate students expected to graduate within the coming year and wanted to learn how to teach PDC at their future institutions.

To aid in gathering unbiased and independent feedback, we contracted with an independent evaluator, David Hiel & Associates (DHA), to survey the remote workshop participants. To evaluate the workshop, DHA incorporated both quantitative and qualitative methodologies in their survey. As part of survey process, DHA asked participants to indicate on a Likert Scale (1 is "not at all useful", 5 is "extremely useful") the perceived usefulness of each workshop session to helping them implement PDC topics in courses at their institutions and to their general professional development. Table II summarizes the participants' responses related to our modules.

TABLE II
How useful was each session for (A) implementing PDC in your courses; (B) your professional development?

Session	(A)	(B)
OpenMP on Raspberry Pi	4.55	4.45
MPI & Distr. Cluster Computing	4.38	4.29

As shown in Table II, the workshop participants rated our two modules very positively. While they rated each of the workshop's sessions at 4 or higher (out of 5), the highest and third-highest rated sessions where those in which they used

these two modules. (A session providing an overview of the *CSinParallel.org* project received their second-highest rating.)

DHA also surveyed participants about their plans for the fall 2020 semester, given the ongoing COVID-19 pandemic. As of July, 74% of participants anticipated that their institutions would be offering in-person and remote hybrid instruction. 39% anticipated teaching their courses fully remotely, with another 35% anticipating teaching their courses in a in-person+remote hybrid manner. Only 17% anticipated teaching their courses solely in-person. During session breaks, conversation commonly turned to teaching plans for the fall semester, reflecting the high anxiety that many of the participants were feeling, and underscoring the group's high interest in learning how teach PDC in remote settings.

A. Efficacy of Remote Multicore Computing Materials

By far, the most "high-risk" activity in the workshop was the use of the Raspberry Pi platform for teaching OpenMP concepts remotely. While prior work [47] had demonstrated the efficacy of using the Raspberry Pi to introduce multicore concepts and OpenMP in an *in-person* environment, it was unclear if a positive, hands-on learning environment could be provided for participants in a remote learning environment. Specifically, the in-person instructors had been crucial to the success of prior Raspberry Pi workshops [47], especially for aiding with setup and providing on-the-spot troubleshooting and assistance as participants worked through the hands-on activity. In addition, the Pi platform came with potential logistical challenges, such as ensuring that everyone had their Raspberry Pis set up prior to the start of the morning activity.

Despite these concerns, the remote, hands-on shared-memory session using the Raspberry Pi was extremely successful. As noted previously, the workshop participants rated our *OpenMP on the Raspberry Pi* session as the most useful for their professional development ($4.45/5$) and most useful for implementing PDC topics in their courses at their own institutions ($4.55/5$). None of the participants reported any technical difficulties during this session; overall, they appeared to greatly enjoy interacting with the Raspberry Pi, guided by the Runestone Interactive virtual handout.

We attribute the lack of technical issues to three factors:

1) The new Raspberry Pi image [45] developed for this activity works on a variety of Raspberry Pis and reduces the total number of steps required for setup.
2) The video walkthroughs available in the first chapter of the virtual module [13] provide step-by-step instructions for setting up the Raspberry Pi for initial use; they also included solutions to common issues participants might face during setup. Videos allow learners to pause and/or replay the instructions as needed.
3) The kits that we mailed to participants (a) were relatively inexpensive (see Table I) and (b) made it easy for learners to use their laptops as I/O devices for their Raspberry Pis, regardless of whether they were running Linux, MacOS, or Windows on their laptops.

We strongly believe that the improved Raspberry Pi image, videos, and kit that we created to support the shared-memory module were the key factors that combind to eliminate any technical issues when using the Raspberry Pis in the remote setting. All of these instructional materials are freely available via the workshop site [50].

In open-ended feedback, several participants gave high praise to the Raspberry Pi as a platform for teaching multicore concepts. *"We can see – using the Pi – several key concepts demonstrated. The level of difficulty was well in the range of our students. After this day – I immediately saw where we can show and use the exercises in our class!!"* exclaimed one participant. Participants also commented about the *"physically compelling"* nature of the Raspberry Pi, and how *"it brings concepts home in a way that nothing else seems to do"*.

Other participants remarked that they liked the idea of students using a Raspberry Pi to run parallel examples rather than their own laptops, due to the great diversity in laptops that students tend to have. *"Having a consistent system makes life so much easier and allows for a consistent experience"* said another participant. Another participant saw the Raspberry Pi as an asset for making it easier to learn PDC concepts in a remote environment because students do not need to connect to a remote server: *"Having students connect to Zoom and separately connect to a remote server can be hard on some wireless connections,"* they noted. These results clearly demonstrate that using a hands-on platform like the Raspberry Pi in conjunction with our virtual module supports effective instruction in a remote environment.

B. Efficacy of Remote Distributed Computing Materials

As noted previously, the participants also ranked the hands-on *MPI and Distributed/Cluster Computing* session as highly useful for both their professional development ($4.38/5$) and teaching PDC in a remote environment ($4.29/5$). This module used multiple strategies to introduce distributed computing in an engaging manner, focusing on how PDC concepts could be taught in early CS courses.

Several participants reacted very positively to the use of mpi4py to introduce message passing. *"It did show me that MPI can be used in Python;"* said one participant, *"this makes Python somewhat viable as a parallel teaching tool"*. Specifically, the use of mpi4py to introduce MPI patternlets in a Google Colab was extremely popular. While Colab VMs have just one core, they are still effective at illustrating fundamental message-passing concepts, thanks to the simplicity of the message-passing patternlets. *"Although they seem difficult,"* observed one participant, *"the parallel programming basics are not [difficult] when introduced correctly."*

Unfortunately, the Colab's single-core VMs prevent learners from experiencing parallel speedup, which is one of the goals of the second part of this session. To experience speedup, learners must run the exemplar programs on a cluster or multicore processor. To accomplish this, we gave participants two options: (i) using a Jupyter notebook whose backend was the Chameleon cluster, or (ii) using a 64-core VM at St. Olaf.

The goal in giving participants a choice was to demonstrate flexibility by showing them that PDC can be effectively taught on different hardware platforms.

While the Jupyter-Chameleon approach worked seamlessly, a minor issue was encountered in using the St. Olaf VM: Some "eager beaver" participants raced ahead of the instructions and tried to log in incorrectly, triggering a VNC-firewall issue that temporarily suspended their remote access via VNC. The participants could still *ssh* to the VM to complete the exercise, but the VNC-firewall issue caused some consternation amongst those participants: (*"The platform switches seem to be a little confusing."*) Despite this hiccup, participants found these hands-on exemplar activities to be very motivating. Some explicitly mentioned that they planned to incorporate the Forest Fire Simulation exemplar [15] into their courses.

C. Lessons Learned

DHC used pre- and post-workshop surveys with common questions to gauge the workshop's effects on the participants' confidence and preparation for implementing PDC topics in their courses. Figure 3 shows how these modules increased participant confidence. A paired Student's t-test indicates that participants experienced a significant increase in confidence ($pre_\mu = 2.82$, $post_\mu = 3.59$, p = 0.0004).

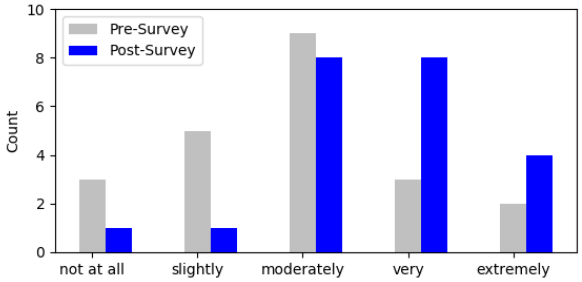

Fig. 3. Indicate your current level of confidence in implementing PDC topics in your courses.

Figure 4 shows how our modules increased participant preparedness. A paired Student's t-test showed this increase to be significant ($pre_\mu = 2.59$, $post_\mu = 3.77$, $p = 4.18e-08$).

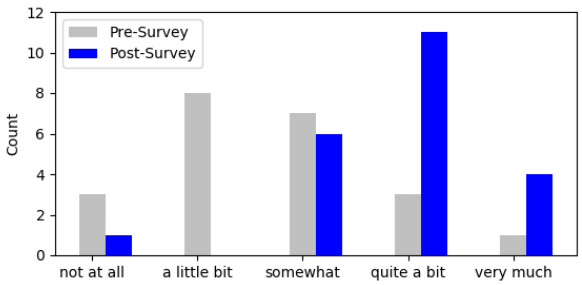

Fig. 4. How prepared do you feel to successfully implement PDC topics in your courses?

In their open-ended responses, participants rated the quality of the free teaching materials and their enjoyment of the interactive hands-on activities very highly. *"The level where the material was presented was perfect"* noted one person. *"I got a lot of material and I feel quite prepared to offer a course on parallel computing this coming Fall"* said another. The quantitative and qualitative feedback provide strong evidence that our modules were highly effective in providing conceptual and hands-on training for multicore and distributed computing.

Despite the need to mail Pi kits to participants, our multicore session was much easier to prepare than our distributed computing session, because it took significant effort to get the Jupyter notebooks working on Chameleon; we could not have done it without help from the Chameleon support staff. Once the notebooks were set up, the Chameleon environment worked seamlessly and participants enjoyed the experience.

By contrast, setting up the St. Olaf VM system was significantly easier for our team. However doing so requires significant Linux systems administration experience and the non-trivial hardware cost ($\approx$ \$5,000.00 for a 64-core multicore server). Furthermore, "eager beaver" students who neglect to follow directions may cause issues, which can be especially problematic when learners are working asynchronously.

Community building is an important part of any classroom experience. To that end, we asked our virtual learners to leave their webcams on during instructional sessions so that we could read facial expressions. (They could turn them off during self-paced exploration times.) However, it took a special effort to get some learners to actively participate in discussion sessions. One participant explained, *"I'm pretty quiet/shy in general and have telephone anxiety... I think I would have contributed more if we weren't trapped in the online format."* At the same time, other more extroverted participants had a tendency to dominate conversations, requiring special effort to curtail their contributions. We believe that our remote workshop experiences apply to the virtual classroom, where it takes extra effort (i) to keep extroverted students from dominating discussions and (ii) to draw out shy students, whose anxieties may be amplified in an online format. In a virtual environment, extra care is needed to include everyone.

V. Conclusions

The online environment adds new obstacles to the challenge of teaching PDC concepts and skills. Motivated by the need to teach PDC during the COVID-19 pandemic, we developed a set of high-quality, self-paced materials that are specifically designed for use in a remote environment. Our preliminary assessment data offers evidence that our materials can be used to effectively teach PDC in a virtual classroom environment. In closing, we describe the specific strategies we used to achieve the goals we listed in Section I:

Strategy 1: *Learners can learn multicore computing concepts effectively in a remote environment by using a Raspberry Pi and our standalone virtual module*. Despite the risks and potential challenges of having each student set up and

operating Raspberry Pis independently and without an in-person instructor's help, our results suggest that a personal SBC, a flexible SBC image, and detailed setup videos greatly streamline the learner's experience. To get the image onto to the Pi, learners just burn the image onto the provided microSD card. Instructors wishing to streamline the process further can mail or provide pre-flashed microSD cards to students. Once set up, the Pi provides a local, tactile platform for learning OpenMP multi-threading, achieving Goal 1.

Strategy 2: *Remote learners can learn distributed computing concepts by using Google Colab and the mpi4py version of the MPI patternlets*. Our assessment results indicate that running mpi4py patternlets within a Google Colab is an effective way to *introduce* message-passing patterns. The mpi4py library reduces the syntactic complexity of most MPI commands, making this approach accessible to even first-year students. While the Colab's unicore VM greatly limits the performance and scalability of distributed computing applications, it is perfectly adequate for introducing message passing concepts via the MPI patternlets. Furthermore, the Colab patternlets require no setup on the part of instructors; students simply need a (free) Google account in which to save the materials.

To let students experience speedup and scalability, it is preferable to run exemplar applications on Chameleon via a Jupyter notebook or on a private system like the St. Olaf VM. Our results suggest that the two-pronged approach of (i) introducing message-passing using a Colab to run the MPI patternlets, and (ii) delving deeper by using a remote cluster to run MPI exemplars is an effective strategy for teaching distributed computing concepts in a remote environment.

Strategy 3: *Remote learners will enjoy highly interactive materials that they can work through at their own pace*. Creating high-quality free materials that are effective in a remote environment takes considerable time; the authors devoted much time to developing the Raspberry Pi kit and image [45], the Runestone shared-memory module [13], the Google Colab notebook [14], the mpi4py exemplars [15], the Jupyter notebook for the Chameleon cluster [16], and setting up the remote VM system at St. Olaf. However, the benefits are clear: For those learning about PDC for the first time, our highly interactive materials make the learning engaging, even in a remote environment. For instructors (who are typically short on time and may be new to PDC), our free materials make it relatively easy to inject PDC into their classrooms in a single two-hour lab period. We anticipate that our materials will be extremely useful to educators even in non-pandemic circumstances, as classrooms increasingly attempt to provide learning experiences that are amenable to an online format.

We also believe our results have implications for anyone teaching computing in a remote environment. An SBC like the Raspberry Pi provides an engaging, tactile, motivating platform for learning about multicore computing, but such devices can also be used to teach other computing topics (e.g. [51], [52]). Likewise, we have shown that browser-based front-ends like Jupyter notebooks and Google Colabs provide user-friendly ways to introduce MPI distributed multiprocessing; it would be even easier to use these tools to teach other computing topics to remote learners.

We acknowledge that our assessment results are preliminary; a key limitation is that our assessment population were faculty learners, some with prior experience with PDC, and all with an interest in teaching PDC. We note however that our results seem promising, since our faculty learners: (i) expressed intentions to adopt our materials into their own classrooms, and (ii) rated these materials as holding high value for their professional development. We believe that if CS faculty find these materials to be useful for professional development, then those materials are also likely to be useful for the professional development of today's CS students.

In our virtual workshop, we found it took extra effort to provide every participant with a communal experience, compared to an in-person workshop. Anecdotally, this also appears to be true in remote classrooms. While mandating that remote-learners' cameras stay on can help somewhat, special efforts are still needed to engage with remote-learners and keep virtual discussions balanced.

Lastly, we hope this report will encourage other PDC educators to make use of our materials in their own classrooms, and build upon the approaches described here to create additional high-quality materials for use by the PDC educational community.

ACKNOWLEDGMENT

This work was supported by the National Science Foundation, DUE-1822480/1822486/1855761. The views expressed in this article are those of the author(s) and do not reflect the official policy or position of the Department of the Army, Department of Defense or the U.S. Government.

REFERENCES

[1] The NSF/IEEE-TCPP Curriculum Working Group, "NSF/IEEE-TCPP curriculum initiative on parallel and distributed computing - core topics for undergraduates," http://www.cs.gsu.edu/ tcpp/curriculum/, 2012.

[2] ACM/IEEE-CS Joint Task Force on Computing Curricula, "Computer science curricula 2013," ACM Press and IEEE Computer Society Press, Tech. Rep., December 2013. [Online]. Available: http://dx.doi.org/10.1145/2534860

[3] J. R. Graham, "Integrating parallel programming techniques into traditional computer science curricula," *SIGCSE Bull.*, vol. 39, no. 4, p. 75–78, Dec. 2007. [Online]. Available: https://doi.org/10.1145/1345375.1345419

[4] R. Brown and E. Shoop, "Modules in community: Injecting more parallelism into computer science curricula," in *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 447–452. [Online]. Available: https://doi.org/10.1145/1953163.1953293

[5] H. C. de Freitas, "Introducing parallel programming to traditional undergraduate courses," in *2012 Frontiers in Education Conference Proceedings*, 2012, pp. 1–6.

[6] M. Burtscher, W. Peng, A. Qasem, H. Shi, D. Tamir, and H. Thiry, "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*, ser. SIGCSE '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 36–41. [Online]. Available: https://doi.org/10.1145/2676723.2677270

[7] S. Ghafoor, D. W. Brown, and M. Rogers, "Integrating parallel computing in introductory programming classes: an experience and lessons learned," in *European Conference on Parallel Processing*. Springer, 2017, pp. 216–226.

[8] H. Wan, X. Gao, X. Long, and B. Jiang, "Introducing parallel computing concepts in computer system related courses," in *2017 IEEE Frontiers in Education Conference (FIE)*, 2017, pp. 1–7.

[9] J. Bruner, *The process of education*. Harvard University Press. Cambridge, Massachusetts, 1960.

[10] D. DiBiasio, W. M. Clark, A. G. Dixon, L. Comparini, and K. O'Connor, "Evaluation of a spiral curriculum for engineering," in *FIE'99 Frontiers in Education. 29th Annual Frontiers in Education Conference. Designing the Future of Science and Engineering Education. Conference Proceedings (IEEE Cat. No.99CH37011*, vol. 2, 1999, pp. 12D1/15–12D1/18 vol.2.

[11] R. M. Harden, "What is a spiral curriculum?" *Medical teacher*, vol. 21, no. 2, pp. 141–143, 1999.

[12] T. J. Dowding, "The application of a spiral curriculum model to technical training curricula," *Educational Technology*, vol. 33, no. 7, pp. 18–28, 1993. [Online]. Available: http://www.jstor.org/stable/44428015

[13] S. J. Matthews, E. Shoop, J. C. Adams, and R. Brown. (2020) Raspberry Pi - Virtual Handout. [Online]. Available: https://pdcbook.calvin.edu/pdcbook/RaspberryPiHandout/

[14] E. Shoop. (2020) Distributed parallel programming patterns using mpi4py. [Online]. Available: https://colab.research.google.com/drive/1yxusXcFQ9ea1bff4_q5iToMhGeQnmSwC

[15] E. Shoop and R. A. Brown. (2020) CSinParallel mpi4py examples. [Online]. Available: https://github.com/csinparallel/mpi4py-examples

[16] E. Shoop. (2020) Jupyter forest fire simulation. [Online]. Available: https://chi-dyn-192-5-87-59.uc.chameleoncloud.org/

[17] J. C. Adams, "Patternlets: A teaching tool for introducing students to parallel design patterns," in *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*. IEEE, 2015, pp. 752–759.

[18] L. Dalcín, R. Paz, and M. Storti, "MPI for python," *Journal of Parallel and Distributed Computing*, vol. 65, no. 9, pp. 1108–1115, 2005.

[19] S. in the Public Interest. (2020) Open MPI: Open source high performance computing. [Online]. Available: https://www.open-mpi.org/

[20] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design patterns: Abstraction and reuse of object-oriented design," in *European Conference on Object-Oriented Programming*. Springer, 1993, pp. 406–431.

[21] ——, *Design patterns: elements of reusable object-oriented software*. Addison-wesley Reading, MA, 1995.

[22] T. G. Mattson, B. A. Sanders, and B. Massingill, *Patterns for parallel programming*, 6th ed., ser. The software patterns series. Boston, Mass.: Addison-Wesley, 2010, oCLC: 688595488.

[23] K. Keutzer, B. L. Massingill, T. G. Mattson, and B. A. Sanders, "A design pattern language for engineering (parallel) software: Merging the PLPP and OPL projects," in *Proceedings of the 2010 Workshop on Parallel Programming Patterns*, ser. ParaPLoP '10. New York, NY, USA: Association for Computing Machinery, 2010. [Online]. Available: https://doi.org/10.1145/1953611.1953620

[24] O. W. Group, "A pattern language for parallel programming," Retrieved September 14, 2013, from tt http://parlab.eecs.b erkeley.edu/wiki/patterns/patterns.

[25] E. J. Sowell, "Effects of manipulative materials in mathematics instruction," *Journal for research in mathematics education*, pp. 498–505, 1989.

[26] W. M. Carroll and D. Porter, "Invented strategies can develop meaningful mathematical procedures," *Teaching Children Mathematics*, vol. 3, no. 7, pp. 370–375, 1997.

[27] L. Jordan, M. D. Miller, and C. D. Mercer, "The effects of concrete to semiconcrete to abstract instruction in the acquisition and retention of fraction concepts and skills." *Learning Disabilities: A Multidisciplinary Journal*, vol. 9, no. 3, pp. 115–22, 1999.

[28] R. Ross and R. Kurtz, "Making manipulatives work: A strategy for success," *Arithmetic teacher*, vol. 40, no. 5, pp. 254–258, 1993.

[29] E. H. Fennema, "The relative effectiveness of a symbolic and a concrete model in learning a selected mathematical principle," *Journal for Research in Mathematics Education*, pp. 233–238, 1972.

[30] S. Papert, "Mindstorms: Computers, children, and powerful ideas," *NY: Basic Books*, p. 255, 1980.

[31] M. F. Chappell and M. E. Strutchens, "Creating connections: Promoting algebraic thinking with concrete models," *Mathematics Teaching in the Middle School*, vol. 7, no. 1, p. 20, 2001.

[32] J. Sarama and D. H. Clements, ""concrete" computer manipulatives in mathematics education," *Child Development Perspectives*, vol. 3, no. 3, pp. 145–150, 2009.

[33] P. L. Moch, "Manipulatives work!" in *The Educational Forum*, vol. 66, no. 1. Taylor & Francis, 2002, pp. 81–87.

[34] M. Richardson and S. Wallace, *Getting started with Raspberry Pi*. " O'Reilly Media, Inc.", 2012.

[35] D. Toth, "A portable cluster for each student," in *2014 IEEE International Parallel & Distributed Processing Symposium Workshops*. IEEE, 2014, pp. 1130–1134.

[36] S. J. Cox, J. T. Cox, R. P. Boardman, S. J. Johnston, M. Scott, and N. S. O'brien, "Iridis-pi: a low-cost, compact demonstration cluster," *Cluster Computing*, vol. 17, no. 2, pp. 349–358, 2014.

[37] S. J. Matthews, "Teaching with Parallella: A first look in an undergraduate parallel computing course," *J. Comput. Sci. Coll.*, vol. 31, no. 3, p. 18–27, Jan. 2016.

[38] H. Shen, "Interactive notebooks: Sharing the code," *Nature*, vol. 515, no. 7525, pp. 151–152, 2014.

[39] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. B. Hamrick, J. Grout, S. Corlay *et al.*, "Jupyter notebooks-a publishing format for reproducible computational workflows." in *ELPUB*, 2016, pp. 87–90.

[40] J. B. Hamrick, "Creating and grading IPython/Jupyter notebook assignments with nbgrader," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, ser. SIGCSE '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 242. [Online]. Available: https://doi.org/10.1145/2839509.2850507

[41] H. Manzoor, A. Naik, C. A. Shaffer, C. North, and S. H. Edwards, "Auto-grading Jupyter notebooks," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1139–1144. [Online]. Available: https://doi.org/10.1145/3328778.3366947

[42] L. B. Ngo, A. T. Srinath, J. Denton, and M. Ziolkowski, "Unifying computing resources and access interface to support parallel and distributed computing education," *Journal of Parallel and Distributed Computing*, vol. 118, pp. 201–212, 2018.

[43] Google Research. (2020) Colaboratory. [Online]. Available: https://colab.research.google.com/

[44] B. Miller and D. Ranum, "Runestone interactive: tools for creating interactive course materials," in *Proceedings of the first ACM conference on Learning@ scale conference*, 2014, pp. 213–214.

[45] R. Brown. (2020) Raspberry Pi image. [Online]. Available: http://csinparallel.cs.stolaf.edu/2020-06-18-csip-image-3.0.2.zip

[46] C. Corp. (2020) CanaKit Raspberry Pi 4 2GB Basic Starter Kit. [Online]. Available: https://www.amazon.com/gp/product/B07VXBMWQK/ref=ppx_yo_dt_b_asin_title_o07_s01?ie=UTF8&psc=1

[47] S. J. Matthews, J. C. Adams, R. A. Brown, and E. Shoop, "Portable parallel computing with the Raspberry Pi," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 92–97. [Online]. Available: https://doi.org/10.1145/3159450.3159558

[48] K. Keahey, J. Anderson, Z. Zhen, P. Riteau, P. Ruth, D. Stanzione, M. Cevik, J. Colleran, H. S. Gunawi, C. Hammock, J. Mambretti, A. Barnes, F. Halbach, A. Rocha, and J. Stubbs. (2020) Lessons learned from the Chameleon testbed. [Online]. Available: https://www.chameleoncloud.org/media/filer_public/8e/4f/8e4f2dc7-35f3-4ca1-b886-4a196298be05/atc20-14.pdf

[49] ——, "Lessons learned from the Chameleon testbed," in *2020 USENIX Annual Technical Conference (USENIX ATC 20)*. USENIX Association, Jul. 2020, pp. 219–233. [Online]. Available: https://www.usenix.org/conference/atc20/presentation/keahey

[50] J. Adams, R. Brown, S. Matthews, and E. Shoop. (2020) CSinParallel summer 2020 virtual workshop. [Online]. Available: https://csinparallel.org/csinparallel/workshops/Virtual20/index.html

[51] H. ElAarag, "Deeper learning in computer science education using Raspberry Pi," *J. Comput. Sci. Coll.*, vol. 33, no. 2, p. 161–170, Dec. 2017.

[52] H. Guerra, A. Cardoso, V. Sousa, J. Leitão, V. Graveto, and L. M. Gomes, "Demonstration of programming in Python using a remote lab with Raspberry Pi," in *2015 3rd Experiment International Conference (exp.at'15)*, 2015, pp. 101–102.