Highlights

Hands-on Parallel & Distributed Computing with Raspberry Pi Devices and Clusters

Elizabeth Shoop, Suzanne J. Matthews, Richard Brown, Joel C. Adams

- Free online interactive modules for learning PDC with Raspberry Pis and Pi clusters.
- Self-organizing cluster: connects disparate Pis into a working cluster in minutes.
- Free disk image pre-loaded with all activities for painless classroom adoption.
- Our materials increase student confidence about PDC and motivation to learn more PDC.
- Our materials increase faculty confidence and preparedness to teach PDC.

Hands-on Parallel & Distributed Computing with Raspberry Pi Devices and Clusters

Elizabeth Shoop[a], Suzanne J. Matthews[b], Richard Brown[c], Joel C. Adams[d]

[a]*Department of Mathematics, Statistics, and Computer Science, Macalester College, Saint Paul, 55105, MN, USA*
[b]*Department of Electrical Engineering & Computer Science, United States Military Academy, West Point, NY, 10996, USA*
[c]*Department of Mathematics, Statistics, and Computer Science, St. Olaf College, Northfield, 55057, MN, USA*
[d]*Department of Computer Science, Calvin University, Grand Rapids, MI, 49546, USA*

Abstract

Parallel and distributed computing (PDC) concepts are now required topics for accredited undergraduate computer science programs. However, introducing PDC into the CS curriculum is challenging for several reasons, including an instructors' lack of PDC knowledge and difficulties in accessing PDC hardware. This paper addresses both of these challenges by presenting free, interactive, web-based PDC teaching modules using inexpensive Raspberry Pi single board computers (SBCs). Our materials include a free disk image that makes it possible for instructors to build Raspberry Pi clusters in minutes and use our software in a variety of curricular contexts. Our multi-year assessment of these materials on students and faculty members indicates that: (i) our materials increased students' confidence regarding

*Corresponding Author: Elizabeth Shoop
Email addresses: `shoop@macalester.edu` (Elizabeth Shoop), `suzanne.matthews@westpoint.edu` (Suzanne J. Matthews), `rab@stolaf.edu` (Richard Brown), `adams@calvin.edu` (Joel C. Adams)

important PDC concepts and motivated them to study PDC further; and (ii) our materials increased faculty members' confidence and preparedness in teaching key PDC concepts at their own institutions.
Keywords: Raspberry Pi, Raspberry Pi clusters, parallel & distributed computing, open-access materials

1. Introduction

It is critical that computer science (CS) undergraduates are exposed to parallel & distributed computing (PDC) topics. Interest in incorporating PDC in the undergraduate computer science curriculum heightened in recent years due to the Accreditation Board for Engineering and Technology (ABET) introducing a new requirement that accredited CS programs demonstrate that all their undergraduate students are exposed to PDC [1]. This new requirement for CS accreditation has forced faculty to take a closer look at how best to incorporate PDC into already crowded computing curricula.

1.1. PDC Pedagogical Challenges

Incorporating PDC into undergraduate computer science curricula is challenging for a number of reasons. First, there is a need to determine *how much* PDC material to cover and *where* in the curriculum to do it. The ACM/IEEE-CS/AAAI Joint Task Force on Computing Curricula's 2023 technical report (CS2023) recommends that all undergraduate CS programs require 9 hours of core topic coverage in PDC, and that institutions who focus on Systems development as a core competency area include up to 26 additional hours of topic coverage [2]. These recommendations are an expansion of CS2013, which recommended only about 15 hours of PDC education [3].

The recommendations of both reports are distilled from the earlier 2012 Core Topics report [4] by the NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing (now updated to version 2, or TCPP2020). Specifically, TCPP encourages faculty to “judicially sprinkle” PDC topics into existing courses, rather than radically redesign courses to include PDC [5].

If faculty are to “sprinkle” PDC into existing courses, they must first develop *confidence* to teach PDC and have access to *high-quality materials* with which to teach. NSF-funded initiatives such as the Center for Parallel and Distributed Computing Curriculum Development and Educational Resources (CDER) [6] and CSinParallel [7] regularly conducted faculty development workshops for faculty new to PDC over the last decade. CDER and CSinParallel both maintain repositories of educational materials related to PDC: CDER’s repository collects teaching materials submitted by external faculty, while CSinParallel’s repository takes a more curated approach, consisting of a large number of highly-detailed modules written by a few PDC educators. Both repositories assume that would-be educators have access to PDC hardware with which to demonstrate PDC concepts.

A lack of *access to hardware* capable of demonstrating PDC topics compounds the difficulty of teaching PDC to undergraduates. Colleges and universities with larger budgets may maintain larger “in-house” clusters that allow a large number of students to apply PDC knowledge. Smaller departments may deploy “virtual clusters” [8, 9] built on powerful multicore servers that enable smaller groups of students to demonstrate PDC knowledge. However, in-house systems are often costly, and usually require dedicated hired personnel to build, maintain and update the clusters. Building an in-house system is infeasible for faculty looking to quickly inject PDC into their courses. Cloud-based solutions such as Amazon's EC2 cluster offer an alternative, but some faculty have remarked on potential issues of the "pay-as-you-go" model in the classroom [10, 11]; without faculty oversight, students who are new to cloud and PDC computing may quickly exhaust available funds, or incur unintended expense. Another approach is to give students remote access to supercomputing resources (e.g., the NSF-sponsored XSEDE/ACCESS [12] project). However, account creation may take days to complete, delaying an otherwise "quick" injection of PDC concepts. Also, most supercomputing clusters utilize a batch-reservation system for job management. Without priority access (which is often costly), students may see long waits for the output of their PDC jobs, extending the time needed to cover PDC concepts.

1.2. Single Board Computers

In conducting this research, we aim to make it painless for faculty to use individual Raspberry Pis and Raspberry Pi clusters for teaching PDC topics. We focus on the Raspberry Pi for its extreme popularity, high level of community support, project maturity, and adoption in a variety of computing courses. There is a rich history [13] of using the Raspberry Pi for computer science and engineering education, and it is a well-established platform for such purposes. For example, educators have widely used Raspberry Pi devices in the context of CS0 [14], CS1 [15, 16] and computer systems courses [17, 18, 19, 20]. The popularity of using the Raspberry Pi to teach computer systems topics is an especially compelling reason to focus on the Raspberry Pi for teaching PDC, as it allows faculty to continue to use a familiar platform in a course that is a natural fit for introducing many PDC concepts. Furthermore, the wide support of the Raspberry Pi in other courses means that students can reuse and/or continue to learn about a platform they were previously introduced to in other courses.

Mathematics education researchers have produced extensive evidence showing that manipulative objects, or *maipulatives*, are effective for learning abstract mathematical topics [21, 22, 23, 24, 25, 26, 27]. Papert [28] was perhaps the first to apply the concepts of manipulatives to computer science, giving birth to the notion of "tangible computing" – the use of physical objects to manipulate and represent digital information [29]; and "physical computing" – using inexpensive hardware to prototype real systems to drive curiosity, imagination, and creativity [30]. Using specialized LEGO pieces combined with a language called Logo, Resnick, Ocko, and Papert demonstrated that manipulative robotic machines could be built and programmed by school children, enabling abstract ideas to become concrete and allowing children to "learn through their fingers" [31]. This use of robotic manipulatives was extended by Resnick with the introduction of the language MultiLogo, which enabled learners to program multiple LEGO robots to work concurrently, marking the first use of physical computing to enable understanding of parallel computing concepts, providing "new metaphors and constructs for controlling multiple processes at once" [32].

In recent years, educators have begun to look at single board computers (SBCs) and SBC clusters to teach students PDC [33, 34, 35, 36, 37, 38, 39, 40, 41]. Early SBCs had a single core, and thus in isolation were not suitable for teaching parallel computing. Consequently, educators started networking

SBCs together to form “microclusters”, in which communication between the SBC nodes was enabled through the use of the Message Passing Interface (MPI) (see [33, 34] for some early examples). However, modern SBCs have multiple cores and may include access to co-processor or GPU chips, making them suitable as a platform for teaching shared memory concepts [37, 39, 38], and as building blocks of educational clusters for teaching larger PDC concepts [35, 38, 40, 41]. Toth was perhaps the first [35, 40] to demonstrate the practicality (and affordability) of having students build *personal clusters* from SBCs to learn PDC topics.

SBCs and SBC clusters are attractive to teach PDC topics for a variety of reasons. First, the tactile nature of a single board computer helps students “touch” and “see” elements of a computer system (e.g., CPU, memory unit, secondary storage, networking) that are traditionally presented in an abstracted fashion. Assigning each student their own SBC provides students with a standardized architecture and prevents the issue of student workloads interfering with each other. Second, pre-configured disk images for SBCs can easily be downloaded from the Internet, ensuring that students start with a uniform work environment, and makes setup and maintenance relatively painless. For busy faculty attempting to quickly inject parallelism into their courses, a painless setup is vital. Lastly, compared to other parallel systems, SBCs are relatively inexpensive. For example, the Raspberry Pi (arguably the most popular SBC today and the focus of this paper) has a listed price of $35.00 for its model 4 board. To reduce the cost-burden further, departments can simply assign students SBCs (or SBC clusters) as lab equipment, and collect the devices back at the end of the lab period or course.

Despite their promise, SBC clusters are not widely used for several reasons. First, while SBC clusters and their use in the classroom are widely discussed in several papers (e.g., [35, 42, 38, 40]), cluster setup has been a significant barrier, usually requiring a non-trivial amount of technical knowledge to fully configure a cluster for use. Next, while prior works [35, 38] discussed how low-cost clusters can be combined with open-access educational materials, the onus is largely on the faculty member to figure out how to combine the two. Lastly, few studies provide assessment of student/faculty classroom experiences using these systems.

Our work is *novel* for several reasons. First, we design and make freely available interactive modules for learning PDC topics on Raspberry Pi devices and clusters. Our interactive modules are built using Runestone Interactive [43], a well-known and popular framework for building open-access interactive educational material. Second, we design and make freely available a Raspberry Pi disk image that makes it painless for faculty to set up Raspberry Pi devices and/or clusters for classroom use. A key component to our design is the concept of a *self-organizing cluster*, which automatically connects disparate Raspberry Pis into a functional cluster capable of running MPI jobs in a matter of minutes. By simultaneously offering materials to teach PDC concepts and simplify the use of the Raspberry Pi as a PDC platform, we provide the scaffolding instructors will need to reproduce our approach and ascertain its benefits at their own institutions. All our materials are free and accessible at [44]

2. Overview of Materials

Our contributions are two-fold. First, we design novel, online, interactive modules for learning PDC topics in conjunction with the Raspberry Pi SBCs and clusters. Second, we design a novel Raspberry Pi cluster concept known as a "self-organizing" cluster. To enable faculty to easily build their own self-organizing clusters, we provide a free Raspberry Pi disk image file that faculty simply burn to SD cards, insert those SD cards into a set of networked Pis, and then follow our online instructions to boot the cluster. This lets students have a personal Raspberry Pi cluster ready to use in a matter of minutes. The subsections that follow describe each of our sets of materials in detail.

2.1. On-Line Interactive Modules for Learning PDC using Raspberry Pis

We developed two distinct PDC modules for use with Raspberry Pis: one for teaching shared memory parallel concepts, and one for distributed memory parallel concepts. Both modules are built using Runestone Interactive [43], an NSF-sponsored, open-source platform for building free interactive textbooks and tutorials. We use many of Runestone's custom directives to build interactivity into our modules, such as video explanations and visualizations of difficult concepts, plus interactive questions to check readers' understanding of key PDC concepts. Runestone is heavily used in the CS Education literature; there is also extensive literature that demonstrates that interactive content improves students learning of computing concepts [45, 46, 47, 48], including parallel computing concepts [49].

The asynchrony of parallel algorithms used in both multicore shared-memory and cluster distributed-memory computations naturally produce non-deterministic behaviors, which is new and potentially confusing to less-experienced programmers. To ease students into parallel concepts, our materials first use PDC *patternlets* to introduce key multicore and distributed computing concepts. Introduced by Adams in 2015 [42], patternlets are very short example PDC programs that each illustrate a particular parallel programming pattern. The brevity of the code and the hands-on experience of tracing and running that code gives introductory students a rapid initial understanding of key PDC programming patterns. Prior work [42] has demonstrated the efficacy of patternlets for teaching students PDC. To illustrate how these patternlets lend themselves to larger, real-world programs, our materials then present one or more "exemplar" applications that students can explore and use in a hands-on benchmarking activity.

Shared Memory module. Our first module focuses on introducing students to parallel concepts on an individual multicore Raspberry Pi. The module is designed to be completed in a self-paced 90-minute period. The first fifteen minutes presents an overview of processes, threads, multicore systems, and provides a short introduction to OpenMP patternlets. The CPU and memory chips are visible on the learner's Pi; our module describes the four-core capability of that CPU, and helps learners form a mental model of relevant architectural concepts, which is made concrete by the physical Raspberry Pi. During the next 45 minutes, learners work through a hands-on exercise where they explore select OpenMP patternlets at their own pace. The final 30 minutes examines two OpenMP exemplars: numerical integration and drug design. Critically, we guide students through a scenario in which a race condition occurs when a code example is run. Students then work through how to properly avoid it. 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, such as speedup. The module is thus designed for use by students in a single lab period, working remotely or in-person, 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. The shared memory module is accessible at [50].

Distributed Memory module. Our second module explores parallel concepts in distributed memory systems (such as a Raspberry Pi cluster) using the Message Passing Interface (MPI), a standard for distributed memory parallel computing. In MPI, independent *processes* communicate with one another by sending and receiving messages. To make MPI accessible to students with different experience levels, we created two versions of our module: one using traditional MPI C programs, and one using mpi4py, a Python library that provides a Python interface to the MPI C functions. The distributed memory modules are designed to be completed over a 90-minute to 2-hour period. The first part of the module presents an overview of clusters and their components, along with instructions on how to set up the self-organizing cluster. This part takes at most 20 minutes to complete. During the next 45 to 60 minutes, students work through a series of MPI patternlets. In one, we guide the learners through a deadlock experience; the exercise helps them understand why deadlock is a problem, what causes it, and learn how to avoid it. During the last half hour or so of the module, students have the option of working through one of two message passing exemplars: a Monte Carlo forest fire simulation or a drug design example. The Distributed Memory module implemented using mpi4py is accessible at [51]. The C version of the aforementioned module is accessible at [52].

2.2. Hands-On Hardware: Self-Organizing Raspberry Pi Clusters

The 2015 release of multicore Raspberry Pi SBCs enabled shared memory concepts to be taught on the Raspberry Pi for the first time [39]. Early kits were limited by their bulkiness, cost, and level of required setup. For example, in 2016, Matthews et al. [39] described using the “Pimoroni” (a self-contained Raspberry Pi kit) to teach shared memory parallel concepts. Each “Pimoroni” unit cost approximately $150.00, and consisted of a Raspberry Pi attached to a Pimoroni LCD monitor, a small mouse and a keyboard. The units however were relatively bulky; a large Pelican case was needed to house just 16 units, making it difficult to scale to larger groups of students. The same paper describes a more bare-bones “headless-VNC” kit [39] that allowed users to remotely access a Raspberry Pi’s desktop via an Ethernet cable between the Pi and the user’s laptop. A separate microUSB cable allowed the Pi to be powered directly by the laptop, eliminating the Pi’s bulky power supply. This bare-bones kit was highly portable, and enabled a user to interact with the Pi via the laptop’s screen, mouse, and keyboard. The use of a VNC client like VNC Viewer also allowed students to view the Raspberry Pi’s desktop from their laptop, allowing them to experience the Pi as a “normal” computer. This bare-bones kit cost about $60.00, but required faculty to statically set IP addresses and install additional packages for classroom use. As successive iterations of the Raspberry Pi increased the power consumption of the device, it became impossible to power the Pi using a microUSB cable from the laptop, forcing the use of an external power supply, and increasing the form-factor of the kits.

The strong community support and large user base behind the Raspberry Pi resolved many of these issues in subsequent years. Successive software updates to the Raspberry Pi rendered some of the auxiliary setup unnecessary; as of this writing, the latest Raspberry Pi OS comes with “zeroconf mDNS” technology that greatly simplifies connecting a laptop and a Raspberry Pi.

Furthermore, USB-C ports have become standard on laptops in recent years. Matthews demonstrated in 2021 [53] that a USB-C cable can be used to power a Raspberry Pi from a user’s laptop (provided it has a USB-C port) and wrote a tutorial on how to set up a Raspberry Pi for classroom using the new “zeroconf” capabilities. The revised bare-bones kit costs about $65.00.

Spurred by the success of prior work that demonstrated the utility of Raspberry Pis for teaching shared memory parallel topics, the authors began exploring the creation of *self-organizing Raspberry Pi clusters* that individual students could use to explore distributed memory parallel topics. Clusters traditionally require significant technical know-how to properly configure; a challenge to deploying clusters to the classroom is to minimize (or eliminate) the configuration steps needed to teach distributed memory concepts within a single lesson or lab exercise. If students perform these configuration steps within the classroom, it can easily consume most of an entire lab period [37]. If faculty perform these steps prior to a class, the extra work may well be prohibitive. Relatedly, the traditional effort required to deploy Raspberry Pi clusters in the classroom may be worthwhile for courses that will use the clusters repeatedly over the course of a semester, it proves to be too much for courses looking to quickly inject PDC concepts into an existing course for a single two-hour period or less.

For this reason, Brown and his students developed the *self-organizing cluster (SOC)*, which, through the running of a few commands, allows a cluster to be configured quickly and automatically, with very little work on part of students (or professors). The SOC software requires additional hardware for physically forming the cluster, namely a network switch and Ethernet cables (and perhaps laptop adapters) for connecting Raspberry Pi SBCs via the switch. The SOC software lets a user configure a Pi cluster for MPI computing using just two commands:

1. `head-node`, performed on any one of the Raspberry Pi units connected to a switch, uses the DHCP protocol to (i) establish that Pi as the cluster's head node, and (ii) to organize the other Pi units as worker nodes; and
2. `soc-mpisetup`, prepares the resulting cluster for MPI computing (e.g., generating an MPI hostfile that lists Pi units available on that network).

A third command `worker-node` returns a head-node Raspberry Pi to its original settings and shuts down the cluster.

These commands require no prior knowledge of which physical Raspberry Pi systems or network switches might be connected together. This lets inexperienced students configure a cluster spontaneously in a few minutes during a class meeting (and to quickly shut down those clusters at the end of class). It also allows a non-specialist professor to easily configure a permanent physical cluster of dozens of Raspberry Pi units using just the SOC configuration commands.

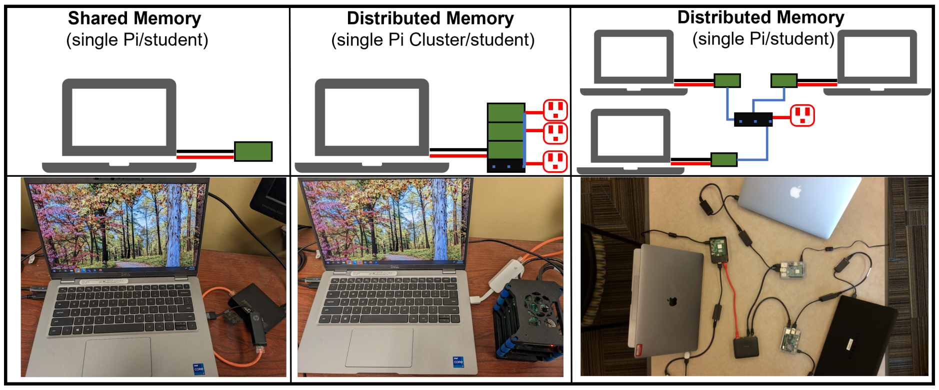

Figure 1: Raspberry Pi Modalities for Learning PDC

This flexible design allows students to explore Raspberry Pi devices and clusters in any of several modalities, while utilizing a single, common disk image:

- Students can explore shared memory parallel concepts using the disk image on a single Raspberry Pi (Figure 1a).
- Students can explore distributed memory parallel concepts by connecting to an existing Raspberry Pi cluster built by their instructor and using it to run MPI jobs (Figure 1b).
- A group of students can collaboratively create a Raspberry Pi cluster by each connecting their personal Pi to a switch, then running the above-mentioned two commands on any of those Pi units that cause the individual nodes to "self-organize" into a cluster (Figure 1c). The group of students can then use the resulting cluster to run MPI jobs.

In this manner, students interact with a touchable physical cluster of Raspberry Pi units, a network switch, cables, and adapters. In the classshares-one-cluster modality (see Figure 1b), each student interacts with a pre-built computer cluster, observing the cluster's Raspberry Pi nodes and the physical network that connects them. In the modality where a team of students creates a cluster (see Figure 1c), students connect their own Pi devices to a network switch to create that cluster. Likewise, in the one-cluster-per-student modality (Figure 1a), each student builds their own personal cluster from multiple Pi devices and a network switch. Each modality thus provides concrete experiences that help students develop accurate mental models of cluster operation and distributed computing.

Table 1: Approximate Cost Breakdown of 3-node Raspberry Pi Cluster Kit, assuming the single Pi cluster/student model. This is the most expensive of the three modalities presented in Figure 1.

Qty.	Item	Cost/unit	Total
3	Raspberry Pi	$35.00	$105.00
3	microSD card	$8.00	$24.00
2	Raspberry Pi power supply	$10.00	$20.00
1	Gigabit 5-port Ethernet Switch	$12.00	$12.00
4	1-ft Ethernet Cables	$2.00	$8.00
2	Ethernet to USB Adapters	$10.00	$20.00
1	USB-C to microUSB (or USB-C) charging cable	$10.00	$10.00
3	velcro dots to connect Pis to switch	$0.02	$0.06
1	Kit case	$20.00	$20.00
	Total Kit Cost		$219.06

The cost of a cluster of three Raspberry Pi units and a 1G network switch used for the last two modalities is approximately $220.00, or about the cost of a textbook (see Table 1). Beyond purchasing the required parts, faculty just need to burn our Raspberry Pi disk image onto each Pi's microSD card. The above-mentioned SOC commands handle everything else. The Raspberry Pi disk image can be found at [54]. It requires at least an 8 GB SD card.

3. Research Methods

To assess the efficacy of our materials, we began by formulating the following research questions, which were designed to assess the materials' effects on their users' confidence and motivation:

- **RQ1**: Do our materials for the Raspberry Pi improve students' confidence about understanding shared-memory parallel computing concepts?
- **RQ2**: Do our materials for the Raspberry Pi clusters improve students' confidence about understanding distributed-memory parallel computing concepts?
- **RQ3**: Do our materials for the Raspberry Pi motivate students to learn more about PDC topics?
- **RQ4**: Do our materials for the Raspberry Pi improve instructors' confidence about understanding PDC concepts?
- **RQ5**: Do our materials for the Raspberry Pi clusters improve instructors' confidence to *teach* PDC topics?
- **RQ6**: Do our materials for the Raspberry Pi motivate instructors to learn more about PDC topics?

To answer these questions, we conducted a multi-year assessment over several populations of students and faculty, both at our home institutions and from around the United States. Our focus in evaluating students was to assess their self-confidence about different concepts that are central to shared memory and distributed memory parallel programming. Our focus in evaluating faculty instructors was to measure their self-confidence and preparedness for delivering PDC concepts to students in their courses.

Prior to performing our assessments, the IRB for our study was reviewed and approved at St. Olaf, which was designated the IRB institution of record. The other institutions then ceded IRB authority to St. Olaf. The assessments shown in this paper include faculty and student populations, some attending in-person workshops at conferences, others in in-person classroom settings. We did not gather or store any personal or identifiable data on participants.

3.1. Assessment Design

The surveys presented in this work were conducted over a six-year period prior to the release of CS2023, so we mapped our assessments and materials against CS2013, the primary curricula report available at the time. In Section 5.2, we discuss the mapping of our materials to the CS2023 topics; for the remainder of this section however, we focus on CS2013.

Table 2 presents our analysis of how our materials map to CS2013's PD knowledge area, which outlines 30 PDC learning outcomes spread over two "core" tiers: core tier 1 (C1) and core tier 2 (C2). According to CS2013, learning outcomes classified as "core tier 1" should be required of every CS undergraduate, while "core tier 2" outcomes are deemed essential [3] coverage. CS2013 further recommends that all core tier 1 outcomes, at least 80% of core tier 2, and "most" elective content be covered in undergraduate CS programs. Our Raspberry Pi modules cover a significant percentage of the learning outcomes required by CS2013.

Table 2: Coverage of CS2013 PD Knowledge Area Outcomes By Our Materials

PD Knowledge Area (Coverage)	**Outcomes covered by our materials**
Parallel Fundamentals (100% C1)	1. Parallelism vs. concurrency [C1] 2. Types of synchronization. [C1] 3. Data Races vs. Other Races [C1]
Parallel Decomposition (100% C1, 75% C2)	1. Explain need for synchronization. [C1] 2. Identify parallelism opportunities [C1] 3. Write correct scalable parallel program [C2] 4. Use task-based decomposition. [C2] 5. Use data decomposition. [C2]
Communication & Coordination (100% C1, 66% C2)	1. Use Mutual exclusion [C1] 2. Give example of a data race. [C1] 3. Give example of deadlock. [C2] 4. Uses of multicasts/event-based messaging. [C2] 5. Write concurrent task program [C2] 6. Used shared buffer among activities [C2] 7. Explain need for atomicity [C2] 8. Write program that reveals a data race [C2]
Parallel Algorithms Analysis, & Programming (71.4% C2)	3. Define speedup, explain scalability. [C2] 4. Identify independent tasks. [C2] 5. Describe what can(not) be parallelized. [C2] 6. Implement parallel divide and conquer. [C2] 7. Decompose a problem via map+reduce operations [C2]
Parallel Architecture (100% C1, 66% C2)	1. Explain shared, distributed memory differences [C1] 2. Describe SMP architecture [C2]

After determining our shared memory materials' coverage of the CS2013 PD outcomes (Table 2), we developed four questions to assess our materials' achievement of specific outcomes. Table 3 shows the mapping of the CS2013 PD Knowledge Area outcomes for shared memory parallelism topics (left and middle columns) to our four assessment questions (right column).

Prior to each session in which we used our materials, we asked attendees

Table 3: Shared Memory Assessment Questions for CS2013 PD Outcomes

PD Knowledge Area	**Outcomes covered by our materials**	**Assessment Questions**
Parallel Fundamentals	3. Data Races vs. Other Races [C1]	How confident are you that you can describe what a race condition is and how to avoid it when writing parallel programs that use shared memory?
Parallel Decomposition	1. Explain need for synchronization. [C1]	How confident are you that you could describe the advantages and disadvantages of using parallel programming on shared memory multicore machines to someone familiar with programming?
	5. Use data decomposition. [C2]	How confident are you that you can describe how to decompose a problem using multiple threads and implement it using a parallel loop?
Parallel Algorithms	3. Define speedup, explain scalability. [C2]	How confident are you that you can define speedup and describe it to someone familiar with programming?

to complete a 4-question pre-survey, consisting of four questions shown in Table 3. The response to each question was a 5-point Likert scale with "1" indicating "not confident at all", "3" indicating "somewhat confident", and "5" indicating "very confident". At the end of a session, we asked attendees to complete a post-survey that was identical to the pre-survey, except that it included a fifth question asking participants to assess (on a scale of 1 to 5) the effect of the Raspberry Pi in motivating them to learn more about parallel computing. Here, "1" indicated "no increase", "3" indicated "some increase", while "5" indicated "a lot of increase". This question was followed by an open-ended response opportunity to explain their answer further.

We followed an identical process when designing our assessment for our distributed memory module. Table 4 shows a similar mapping for distributed memory parallel computing concepts from CS2013 and the associated questions we developed for our pre- and post-surveys. The 5-question pre-survey asked participants to rate their confidence on each of five questions shown in Table 4, using a 5-point Likert scale with “1” indicating “not confident at all”, “3” indicating “somewhat confident”, and “5” indicating “very confident”. After completing the module, attendees were asked to complete a post-survey containing the identical five questions, plus two additional Raspberry Pi-specific questions to assess their level of engagement and their motivation to learn more.

Table 4: Distributed Memory Assessment Questions for CS2013 PD Outcomes

PD Knowledge Area	**Outcomes covered by our materials**	**Assessment Questions**
Parallel Fundamentals	2. Types of synchronization [C1]	How confident are you that you can describe the use of point to point communication between processes to a person familiar with programming?
Parallel Fundamentals	3. Data Races vs. Other Races [C1]	How confident are you that you can describe what deadlock is and how to avoid it in message passing programs?
Parallel Decomposition	1. Explain need for synchronization [C1]	How confident are you that you can describe the advantages and disadvantages of using message passing on a computing cluster to someone familiar with programming?
	5. Use data decomposition [C2]	How confident are you that you can decompose a problem by having multiple processes perform a parallel loop?
Communication	Uses of multicasts/ event-based messaging [C2]	How confident are you that you can describe the use of collective communication between processes to a person familiar with programming?

In addition to classroom use, we used these modules in multi-day, annual professional development faculty workshops held over three years, two which were conducted remotely due to the COVID-19 pandemic. To aid in gathering unbiased and independent feedback at these workshops, we contracted with an independent evaluator, Anne Gurnee Consulting (AGC) to survey the remote workshop participants. To evaluate the workshop, AGC asked participants to indicate on a 5-point Likert Scale (1 is "not at all useful", 5 is "extremely useful") the perceived usefulness of each workshop session (i) in helping them implement PDC topics in courses at their institutions, and (ii) for professional development.

AGC also used pre- and post-workshop surveys with common Likert-scaled questions to gauge the workshop's effects on the participants' confidence and preparation for implementing PDC topics in their courses. The first question was "Indicate your current level of confidence in implementing PDC topics in your courses.", where 1 corresponded to "not at all confident", 2 corresponded to "slightly confident", 3 corresponded to "moderately confident", 4 corresponded to "very confident", and 5 corresponded to "extremely" confident. The second question was "How prepared do you feel to successfully implement PDC topics in your courses?" For this question, 1 corresponded to "not at all", 2 corresponded to "a little bit", 3 corresponded to "somewhat", 4 corresponded to "quite a lot" and 5 corresponded to "very much".

3.2. Characteristics of our Student Participants

We evaluated our materials on six groups of undergraduate computing students, with three groups evaluating our shared memory materials, and three groups evaluating our distributed materials. While our materials were designed for CS undergraduates with no prior exposure to parallelism, we had very diverse audiences that included students from a variety of majors.

For our shared memory parallelism module, each of our three groups of students covered the material over a 90-minute session:

- Our first group consisted of 33 primarily undergraduates who had never been exposed to parallel computing (B for "Beginners"). These students self-selected to attend a workshop we offered at the ACM Richard Tapia Conference, which celebrates diversity in computing. Most students had no prior exposure to Raspberry Pi SBCs.

- The second group were 16 undergraduates in a college classroom with the majority having prior exposure to parallel concepts, but not OpenMP (I for "Intermediate"). A significant number also had prior experience with Raspberry Pi SBCs.

- Our last shared memory group consisted of 16 primarily graduate students, some with prior parallel programming experience (A for "Advanced"). This group of students chose to attend a parallel computing workshop at the SIAM Conference on Computational Science and Engineering. Some students had prior exposure to Raspberry Pi SBCs.

We also tested our distributed memory MPI materials on groups of undergraduate students in laboratory sessions at three different undergraduate institutions:

- Forty-two West Point students (W) used our materials over a 2-hour lab period near the end of a computer systems course that included a 10-lesson unit on concurrency with threads.

- Thirty-two St. Olaf students (S) used our materials over two one-hour class sessions in a hardware design course that used Raspberry Pi SBCs throughout; the students learned how to assemble their clusters in the first session and worked through our interactive MPI exercises during the second session.

- Twenty-three Macalester (M) students used our materials in self-directed pairs over two 1.5 hour class periods of a parallel and distributed computing course. This group of students had some prior experience with OpenMP shared memory parallel computing, but this experience was their first time using Raspberry Pi clusters and MPI.

We were thus able to test our materials on a wide variety of students.

3.3. Characteristics of our Faculty Participants

In addition to running classroom sessions with students, we piloted our materials to faculty at two separate 3-hour in-person workshops at the ACM SIGCSE conference, and also at two separate multi-day virtual summer workshops. We introduced these materials slightly differently to the SIGCSE participants vs. the summer participants, owing to the different amounts of time and modalities available to us:

- To demonstrate to faculty at SIGCSE how they would teach with our materials, we conducted each 3-hour SIGCSE workshop in-person; each included a 90-minute lab period that we ran exactly the same as we did with students in-person.

- In contrast, we conducted our 2020 and 2021 summer workshops in a fully remote format, due to the COVID-19 pandemic. Prior to the workshop, we mailed each participant a Raspberry Pi kit (see Table

1) and emailed them links to videos to guide them through assembly and setup. The workshop sessions were conducted synchronously. In both years, the workshop participants worked through our first (shared memory) Raspberry Pi module during a 2-hour session on the first morning. In 2021, the participants worked through the second (distributed memory) module the second morning. (In 2020, the SOC software was not completed.) We added an extra half hour for both sessions as a precaution, in case our remote participants encountered technical issues.

As with our student groups, these workshops let us test our materials with a variety of faculty members and under a variety of circumstances.

Our remote faculty development workshops attracted diverse audiences: The 22 participants in our 2020 summer 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 a year and wanted to learn how to teach PDC at their future institutions. Likewise, our 2021 summer workshop attracted 19 participants, consisting of 95% faculty and 5% K-12 classroom teachers. In this workshop, 64% of the attendees identified as male, 16% as female, and three preferred not to answer. 84% of those 2021 attendees were tenured or tenure-track at their institutions, while 11% were non-tenure track.

4. Results

4.1. Student Feedback

Here we report survey results from sessions of student participants.

Table 5: OpenMP Workshop Student Pre- and Post-Survey Results.

	Pre-Survey Means			Post-Survey Means			p-values
	B	I	A	B	I	A	
Questions/ Number of Responses	$n = 33$	$n = 16$	$n = 16$	$n = 32$	$n = 16$	$n = 16$	
1. How confident are you that you can describe how to decompose a problem using multiple threads and implement it using a parallel loop?	2.15	2.00	2.38	3.66	3.56	4.18	B: 3.129×10^{-8} I: 1.118×10^{-4} A: 1.653×10^{-4}
2. How confident are you that you could describe the advantages and disadvantages of using parallel programming on shared memory multicore machines to someone familiar with programming?	2.55	2.56	3.06	3.94	3.75	4.38	B: 1.071×10^{-6} I: 9.046×10^{-4} A: 6.742×10^{-4}
3. How confident are you that you can define speedup and describe it to someone familiar with programming?	2.27	1.94	2.81	3.95	3.69	4.5	B: 5.912×10^{-7} I: 5.890×10^{-6} A: 1.965×10^{-4}
4. How confident are you that you can describe what a race condition is and how to avoid it when writing parallel programs that use shared memory?	2.48	2.38	2.81	3.76	3.94	4.5	B: 2.349×10^{-5} I: 7.779×10^{-4} A: 6.933×10^{-4}

4.1.1. Shared Memory (OpenMP) Results

Table 5 summarizes the results for the student sessions using our OpenMP (shared memory) materials. The third row reports the number of responses we received on the pre- and post-surveys for the three populations (B=Beginner, A = Advanced, and I=Intermediate). For example, 33 Beginner participants completed the pre-survey, while 32 completed the post-survey.

The mean response for each pre- and post-survey question is shown in columns two and three respectively. We conducted a two-sample unpaired t-test using the R statistics package [55] to test the significance of the difference between the mean scores. For each question, the null hypothesis of the t-test was that the means for the two groups would be equivalent, and we reject the null hypothesis when $p < 0.05$. The rightmost column of Table 5 provides the results of our t-test analyses.

There is a significant difference in the means of the pre- and post-surveys for questions 1 through 4 for each of our student workshops, despite the fact that we had different populations of students with varying exposures to parallelism. The Beginner population workshop (which had our lowest p-values) had the highest number of novice students; the Advanced population workshop had more experienced students; and the Intermediate workshop contained a large component of undergraduate students who had some prior exposure to some parallel concepts. Unsurprisingly, more advanced groups had higher p-values than the beginning group; however, all measured p-values are below our significance threshold of 0.05, so we reject the null hypothesis. Table 5 thus provides evidence that our materials are effective in helping students from a variety of backgrounds gain confidence regarding shared memory programming concepts.

Table 6: MPI Lab Student Pre- and Post-Survey Results.

	Pre-Survey Means			Post-Survey Means			p-values
	W	S	M	W	S	M	
Questions/ Number of Responses	40	32	23	42	28	23	
1. How confident are you that you decompose a problem by having multiple processes perform a parallel loop?	2.63	2.06	2.57	3.42	2.86	3.73	W: 2.945×10^{-5} S: 7.815×10^{-3} M: 4.395×10^{-4}
2. How confident are you that you can describe the advantages and disadvantages of using message passing on a computing cluster to someone familiar with programming?	2.15	1.88	1.73	3.57	2.86	3.78	W: 3.016×10^{-11} S: 3.595×10^{-4} M: 1.488×10^{-9}
3. How confident are you that you can describe the use of point to point communication between processes to a person familiar with programming?	2.02	1.81	1.48	3.49	2.61	3.26	W: 2.222×10^{-13} S: 3.43×10^{-3} M: 2.894×10^{-7}
4. How confident are you that you can describe the use of collective communication between processes to a person familiar with programming?	2.08	1.81	1.61	3.43	2.57	3.61	W: 3.628×10^{-11} S: 2.452×10^{-3} M: 7.102×10^{-10}
5. How confident are you that you can describe what deadlock is and how to avoid it in message passing programs?	2.05	1.78	1.61	3.74	2.68	4.17	W: 3.263×10^{-13} S: 1.858×10^{-3} M: 2.317×10^{-12}

4.1.2. Distributed Memory (MPI) Results

Table 6 summarizes the MPI results for our three groups of students. As with our OpenMP results, we conducted a two-sample unpaired t-test using the R statistics package [55] to test the significance of the difference between the mean scores. For each question, the null hypothesis of the t-test is that the means for the two groups are equivalent, and we reject the null hypothesis when $p < 0.05$.

The West Point students, having completed a 10-unit lesson on shared memory programming (though no lessons on distributed computing) had the highest pre-survey confidence across across all sections. The Macalester students also had some experience with OpenMP shared memory programming, but had lower confidence regarding distributed programming going into the MPI programming exercises. Despite the variations in the way the intervention was conducted, all populations of students experienced a statistically significant increase in confidence on core concepts, providing evidence that our materials successfully helped students gain confidence in their understanding of distributed memory parallel concepts.

4.1.3. Impact on Student Motivation and Engagement

The post-surveys given to the students taking both the shared memory and distributed memory interventions contained a question asking students how much using the Raspberry Pi (Cluster) motivated them to learn more about PDC topics. On the OpenMP post-surveys, the question was "To what extent did using an inexpensive multicore computer (e.g. the Raspberry Pi) to run parallel programs motivate you to learn more about parallel computing in the future?". On the MPI post-surveys, the question was "To what extent did using a cluster of inexpensive small computers motivate you to learn more about parallel and distributed computing?". For both surveys, students were presented with a 5-point Likert scale where 1 corresponded to "no increase in motivation" and 5 indicated "highly motivated".

Figure 2 depicts the distribution of responses on both sets of surveys. A total of 64 students answered this question for the OpenMP post-surveys, and 93 students answered this question on the MPI post-surveys. The green and blue bars depict the students' responses to the motivation question from the OpenMP workshop surveys and the MPI lab surveys, respectively. We normalized the y-axis to depict the percent of students that gave a particular response. The mean response for the question on the OpenMP and MPI surveys are 4.05 and 3.66, respectively.

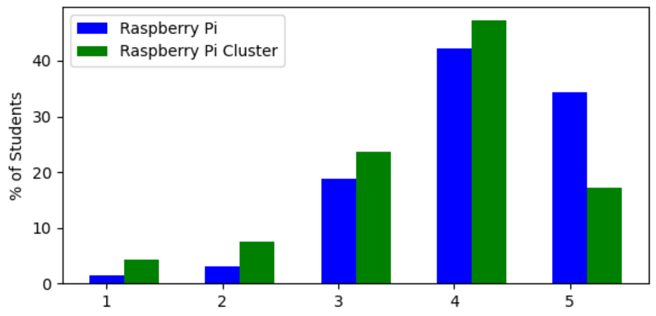

Figure 2: Students: "How much did using the The Raspberry Pi/Raspberry Pi Cluster motivate you to learn more?"

As can be seen in Figure 2, 76.6% of the students surveyed in the OpenMP groups answered with a '4' or '5', indicating that using the Raspberry Pi motivated them to learn more about parallel computing. A smaller majority (64.5%) of the students surveyed from the MPI group answered '4' or '5', again indicating that the Raspberry Pi clusters motivated them to learn more. We theorize that one of the reasons that motivation was lower for the MPI surveys had to do with the novelty factor of the Raspberry Pi devices: In our OpenMP sessions, most students had not seen (or used) a Raspberry Pi prior to the session. In contrast, the majority of the students in the MPI sessions (76.2%) had prior exposure to Raspberry Pis earlier in the semester at their respective institutions, reducing the novelty of the experience.

4.2. Faculty Feedback

In this section, we report the pre- and post-survey results of those using our materials at faculty development workshops.

4.2.1. In-person Faculty Survey Results

Table 7 summarizes the assessment results of the OpenMP and MPI faculty workshops at SIGCSE. The OpenMP workshop consisted of 17 participants, who all took the survey. The MPI workshop consisted of 13 participants, 11 of whom took the pre-survey and 8 which took the post-survey. We believe the reason for this difference in response rates had to do with the modality in which surveys were provided. At the OpenMP workshop, paper surveys were administered. In contrast, at the MPI workshop, electronic surveys were administered.

Table 7: Pre-Survey and Post-Survey Results of OpenMP and MPI workshops taken by Faculty at SIGCSE workshops. Note there were only 4 questions on the OpenMP survey.

	OpenMP Workshop Means			MPI Workshop Means		
Question	Pre-Survey (n=17)	Post-Survey (n=17)	p-value	Pre-Survey (n=11)	Post-Survey (n=8)	p-value
Q1	2.88	4.06	0.0046	2.72	4.00	0.0180
Q2	3.18	4.12	0.0275	2.82	4.00	0.0594
Q3	3.12	4.23	0.0208	2.63	3.75	0.1278
Q4	3.11	4.18	0.0077	2.36	3.75	0.0221
Q5	–	–	–	2.72	4.13	0.0264

At these workshops, faculty were given pre- and post-surveys that had identical questions to the student surveys discussed in Section 4.1. For the OpenMP workshop, faculty indicated a significant increase in confidence for all concepts. Interestingly, while faculty confidence did increase for the use of point-to-point communication (question 3), the increase did not meet the threshold for significance. We speculate this may be due to difference in population between the pre- and post-survey respondents for the MPI workshop; the small sample size is also likely a factor.

Figure 3 shows the the distribution of faculty responses to the post-survey question, "How much did using the Raspberry Pi (Cluster) motivate you to learn more about parallel (and distributed) computing?".

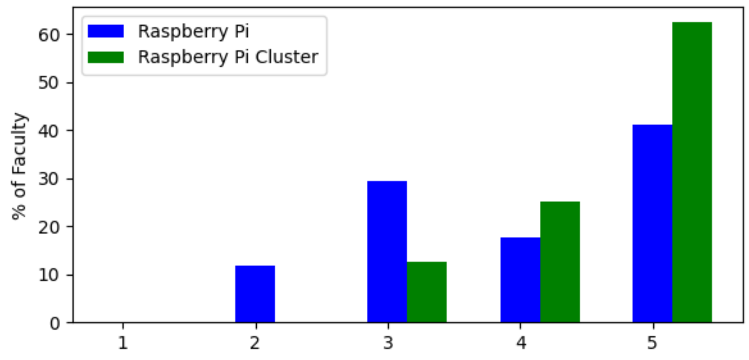

Figure 3: Faculty: "How much did using the The Raspberry Pi/Raspberry Pi Cluster motivate you to learn more?"

While the average score for the OpenMP workshop (where each participant was given a single Raspberry Pi) was 3.88, the average score for the MPI workshop (where each participant was given a Raspberry Pi *cluster*) was higher at 4.25. The MPI workshop survey added a question about *engagement*; faculty self-reported an average engagement of 4.5. We speculate that faculty found the clusters more motivating due to their novelty. All the concepts of the OpenMP workshop could be taught on any modern multi-core workstation or laptop. However, we suspect that the novelty, form-factor and additional processing elements provided by the Raspberry Pi clusters led faculty attendees to find them highly appealing as a learning platform.

Table 8: Top 3 Most Useful Sessions ranked by faculty for (A) implementing PDC in their courses; and (B) for their own professional development

2019 (n=19)		2020 (n=22)		2021 (n=19)	
(A)	(B)	(A)	(B)	(A)	(B)
OpenMP on Remote Multiprocessor	Building Raspberry Pi Clusters	OpenMP on Raspberry Pi	OpenMP on Raspberry Pi	OpenMP on Raspberry Pi	mpi4py on Raspberry Pi Cluster
MPI on Remote Clusters	TSGL; MPI on Remote Clusters (tied)	CSinParallel Modules; MPI on Remote Clusters; PDC Teaching Experiences (3-way tie)	CSinParallel Modules	PDC Resources "Show and Tell"; PDC Teaching Experiences; Curriculum Workshop (3-way tie)	PDC Resources "Show and Tell" / PDC Teaching Experiences (tie)
OpenMP on Raspberry Pi	Hybrid OpenMP/MPI programming	Strategies for teaching PDC Remotely	MPI on Remote Clusters/ PDC Teaching Experiences (tie)	CSinParallel Modules	OpenMP on Raspberry Pi / CSinParallel Modules (tie)

4.2.2. Remote Faculty Survey Results

Table 8 shows how useful faculty members found our remote workshop sessions in 2020 and 2021. For the sake of comparison, we also include data from our pre-pandemic summer workshop in 2019, which was conducted in-person. All multi-day summer workshops included other (non-Raspberry Pi) sessions that were devoted to other demonstrations of tools and modules, and discussions related to PDC education. We initially reported part of these results in 2021 [56].

In 2019, faculty ranked the "OpenMP on a Remote Multiprocessor" ($\mu = 4.47$) and "MPI on Remote Clusters" ($\mu = 4.41$) as the sessions that they found most useful to help them implement PDC topics at their own institutions. While the "OpenMP on the Raspberry Pi" ($\mu = 4.35$) session made it to the top 3, we believe it did not have as big of an impact as the other sessions due to the lack of ready-made online instructional material to accompany the hardware platforms. Faculty rated the "Building Raspberry Pi Clusters" session as the most useful ($\mu = 4.56$) in terms of professional development, while "MPI on Remote Clusters" ($\mu = 4.53$) and "Hands-On Hybrid MPI/OpenMP Programming" ($\mu = 4.40$) sessions were ranked second and third. We suspect that this is due to our participants generally being less familiar with distributed memory concepts coming into the workshop.

At the 2020 workshop, we debuted the OpenMP Runestone interactive tutorial to accompany the Raspberry Pi. At the 2021 workshop, we debuted the mpi4py Runestone interactive tutorial to accompany the (revamped) Raspberry Pi cluster kits. Our results clearly demonstrate the power of having compelling hardware accompanied by quality materials. Faculty ranked the "OpenMP on the Raspberry Pi" session the highest in 2020 and 2021 for PDC implementation, and the "mpi4py on Raspberry Pi Cluster" session the most highly for professional development in 2021. Furthermore, the "OpenMP for Raspberry Pi" was ranked the highest for professional development in 2020 and tied for third place for professional development in 2021. We believe these results reflect the quality and compelling nature of our interactive materials combined with hands-on hardware.

Figure 4 shows how our materials increased participant confidence. A Student's t-test indicates that participants experienced a significant increase in confidence ($pre_\mu = 2.82$, $post_\mu = 3.59$, $p = 7.624 \times 10^{-4}$). Figure 5

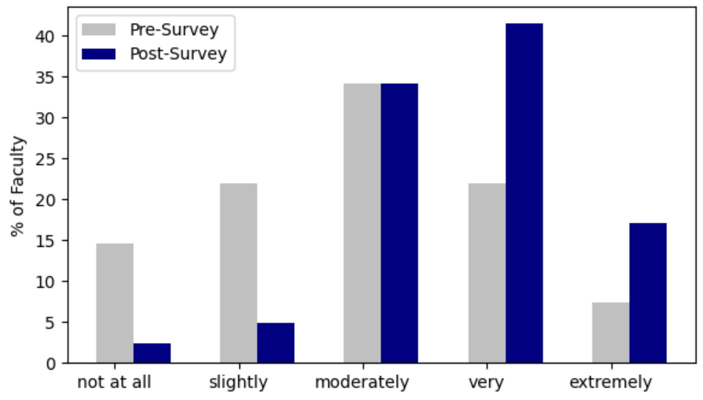

Figure 4: Faculty remote participants' confidence in implementing PDC in their courses

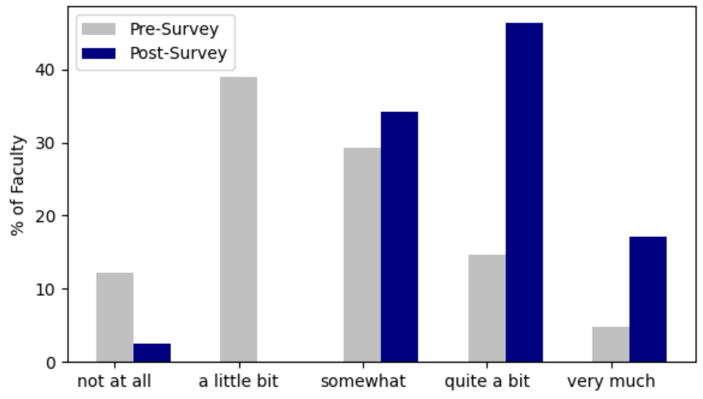

Figure 5: Faculty remote participants' preparedness in implementing PDC in their courses

shows how our modules increased participants' feelings of preparedness. A paired Student's t-test showed this increase to be significant ($pre_\mu = 2.85$, $post_\mu = 3.66$, $p = 4.947 \times 10^{-7}$).

5. Discussion

PDC content has become a necessary component in undergraduate computer science curricula, as expressed in reports such as CS2013 [3], TCPP2012 [4], and more recently in the ABET accreditation standards for CS [1] and the CS2023 report [2]. In order to address these imperatives, we have developed PDC learning materials for both shared-memory and distributed-memory parallelism that can be "judiciously sprinkled" [5] into existing courses at all undergraduate curricular levels, including the introductory sequence, without radically redesigning those courses for PDC.

Our work was inspired by the pioneering work in physical computing whose aim was to drive curiosity, imagination, and creativity in students learning computer science [30, 32]. We designed our materials for Raspberry Pi single-board computers, including individual Pi units attached to a user's laptop or desktop, or multiple Pi units connected to a network switch to form a Beowulf cluster via our self-organizing cluster software.

Our analysis in Section 4 (Results) provided evidence that the combination of our materials with Raspberry Pi SBCs and Raspberry Pi clusters helped both students and instructors gain confidence about learning PDC concepts, improved instructors' confidence about teaching PDC topics, and motivated both students and instructors to learn more about PDC. Note that our research questions **RQ1** to **RQ6** appear near the beginning of Section 3

(Methods).

5.1. Summary of Findings

1. Students gained confidence about their understanding of both shared-memory parallel and distributed-memory parallel computing concepts after using our materials for learning with Raspberry Pi technologies, as shown in Tables 5 (Section 4.1.1) and 6 (Section 4.1.2). This indicates an affirmative answer to research questions **RQ1** and **RQ2**.
2. Faculty (at in-person workshops) significantly gained confidence about understanding shared-memory parallel computing concepts after using our materials, as shown in Table 7 (Section 4.2.1). That section also suggests that faculty gained confidence about understanding distributed-memory parallel computing concepts, where a majority (but not all) of the survey questions reveal significant increases (at $p<=0.05$) in confidence. These comments provide a largely affirmative answer to research question **RQ4**.
3. Faculty (at remote workshops) reported a significantly higher mean level of confidence and preparedness to implement PDC concepts in their courses, as shown in Figure 4 (Section 4.2.2). This indicates an affirmative answer for research question **RQ5**.
4. Both faculty (at in-person workshops) and students reported increased motivation to learn more about shared-memory parallel and distributed-memory parallel computing after using our materials, as indicated by Figures 2 (Section 4.1.3) and 3 (Section 4.2.1). This indicates an affirmative answer to research questions **RQ3** and **RQ6**.

5.2. Mapping to CS2023

As previously mentioned, our materials were created and assessed prior to the recent release of the CS2023 Curricula Final Report [2]; it contains numerous CS knowledge areas, and breaks these down into knowledge units. The knowledge units are further divided into **CS core** topics and additional **knowledge area** (KA) topics. Each knowledge area also contains some example *Illustrative Learning Outcomes*, which don't necessarily match each topic (e.g., there are fewer learning outcomes than topics).

The revised Parallel and Distributed Computing (PDC) Knowledge Area in CS2023 has five knowledge units: Programs, Communication, Coordination, Evaluation, and Algorithms. Certain topics related to PDC are in other knowledge areas and are considered prerequisites for the PDC topics. These include System Fundamentals (SF), Architecture (AR), and Foundations of Programming Languages (FPL). According to CS2023 [2], the PDC CS Core topics "span approaches to parallel and distributed computing, but restrict coverage to those that apply to nearly all of them."

Table 9 shows the relevant knowledge area units (KA-Unit) in the left column and in the right column, their **CS2023 core** learning outcomes covered by our materials. Our analysis suggests that our materials, despite having been developed prior to the release of CS2023, are more relevant than ever, covering many of the important PDC topics in CS2023.

5.3. Additional Considerations

As noted in Figures 2 and 3, the majority of the people using our materials reported a desire to learn more about PDC. Some of this motivation may arise from the novelty of using the Pi devices as a PDC hardware platform.

Table 9: CS2023 Topic Coverage

KA-Unit	**Topics/Learning Outcomes Covered By Our Materials**
SF-Foundations	Write a parallel program from a sequential one. Evaluate performance of parallel programs via speedup. Explain how exploiting parallelism decreases elapsed time.
AR-Heterogeneity Heterogen. Arch.	MIMD architectures. Shared memory vs distributed memory.
FPL-Parallel Parallel and Distributed Computing	Explain lack of sequential consistency with data races. Implement correct concurrent programs. Use synchronization constructions. Model data dependency using simple programming constructs. Model control dependency using simple constructs.
PDC-Programs	Parallelism (declarative, defining order, ensuring ordering). Distribution (places: defining, naming, activities across). Starting activities (tools for actions, procedural, dependent.) Execution properties (nondeterministic execution, consistency, fault-tolerance).
PDC-Communication	Media (MPI, Shared Memory), hardware Channels (MPI). Memory (shared memory, memory hierarchy). Use task-based decomposition. Use data decomposition.
PDC-Coordination	Dependencies (why they exist, are needed) Control constructs (barriers, thread joins). Atomicity (mutual exclusion, deadlock avoidance).
PDC-Evaluation	Identifying errors. Performance metrics (scalability, communication costs). Performance of design choices (granularity, overhead). Scalability limitations (Amdahl's Law).
PDC-Algorithms	Implement a PDC algorithm (e.g., threads, MPI). Common application domains (multicore, data parallel, cluster).

Although our numbers of participants in our groups may seem modest by some measures, the p-values reported in our survey responses concerning student confidence in understanding of PDC concepts indicates a significant increase in that confidence, since each group met a criterion of $p < 0.005$ for each question, with one exception (which easily met a $p < 0.05$ standard). This provides evidence that using Raspberry Pi devices and clusters to learn about PDC concepts–including abstract notions such as race conditions and deadlock–is an effective pedagogical strategy.

It is worth mentioning that the global chip shortage that occurred during the COVID-19 pandemic temporarily made new Raspberry Pis harder to acquire. While many of the supply chain issues have eased (at time of writing) and Pis are once again easy to order, it is possible that disruptions are possible in the future. Importantly, the disk images that we created are fully compatible with (and tested on) older models of the Raspberry Pi. At West Point for example, all MPI clusters were built using Raspberry Pi hardware released nearly six years ago, and our image worked well on these clusters, as it has on clusters at St. Olaf and Macalester that use newer models. Our results indicate that older models of Raspberry Pis *can* serve as a successful platform for teaching and learning PDC. This in turn suggests that our materials can help breath "new life" into older models of Raspberry Pi, which are often neglected and underutilized as newer models are released.

If an institution prefers to use another type of single-board computer besides Raspberry Pi, our system setup would have to be ported to that hardware platform. This may involve porting of the SOC software, but aside from that, our learning materials should not require any changes.

We have found that it is more convenient to use our materials in a single longer laboratory session than in multiple shorter class meetings, as the longer sessions reduce the percentage of time spent in setup and teardown activities. Also, less experienced students will usually require more total time to work through our materials' exercises than more advanced learners.

Our learning materials (summarized in Table 10) use free open source software (Linux, the GNU software suite, MPI, and so on), and so have no cost to students and can be used on any Linux or Linux-related platform (e.g., MacOS, Windows Subsystem for Linux, etc.). However the Raspberry Pi SBCs and other hardware components in our kits do entail modest costs; in the cluster-per-student modality, these costs could constitute an entry barrier for some students. One way to reduce such barriers is for the department, university, or local industry partners to underwrite the cost of the kits. West Point, for example, built a set of Raspberry Pi cluster kits and have reused them in class over the last several years, incurring a one-time cost to the institution, and zero cost to students.

Table 10: Linked Materials

Module/Resource
• OpenMP Shared Memory Module
• mpi4py Distributed Memory Module
• C MPI Distributed Memory Module
• Disk Image

5.4. Future Directions

The findings in this paper encourage these future research initiatives:

Add more PDC concepts Our materials for the Raspberry Pi cover key outcomes for multiple units in the PD Knowledge Area (KA) of CS2013 and the PDC KA of CS2023. However some topics and outcomes remain uncovered; the feasibility of extending our materials to cover those topics is one possible direction.

Extend to other CS knowledge areas The learning effectiveness and motivational value of Raspberry Pi systems or other SBCs could be explored in other computing areas besides PDC, for example, computer organization/systems, operating systems, computer networks, or elementary programming courses.

Extend to other SBCs Given the plethora of other available SBC devices, Brown plans to create a version of the self-organizing cluster (SOC) software for Ubuntu Linux. This will make our materials deployable on other SBC devices.

6. ACKNOWLEDGMENTS

Special thanks to Max Narvaez '21 of St. Olaf College who played a large role in the production of the Raspberry Pi kits and their software image, including the Self-Organizing Cluster software. We are grateful to Anne Gurney Consulting for conducting faculty surveys for our multi-day remote workshops. This work was supported by the National Science Foundation grants DUE-1822480/1822486/1855761. The views expressed in this article are those of the author 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] ABET Computing Accreditation Commission, Criteria for accrediting computing programs, https://www.abet.org/wp-content/uploads/2018/02/C001-18-19-CAC-Criteria-Version-2.0-updated-02-12-18.pdf (2018).

[2] ACM/IEEE-CS/AAAI Joint Task Force on Computing Curricula, Computer science curricula 2013, Tech. rep., ACM Press and IEEE Computer Society Press (January 2024).
URL https://csed.acm.org/cs2023-report-with-feedback/

[3] ACM/IEEE-CS Joint Task Force on Computing Curricula, Computer science curricula 2013, Tech. rep., ACM Press and IEEE Computer Society Press (December 2013). doi:10.1145/2534860.
URL http://dx.doi.org/10.1145/2534860

[4] The NSF/IEEE-TCPP Curriculum Working Group, NSF/IEEE-TCPP curriculum initiative on parallel and distributed computing - core topics for undergraduates (version 2.0), https://tcpp.cs.gsu.edu/curriculum/ (2020).

[5] The NSF/IEEE-TCPP Curriculum Initiative, NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing (How to use these guidelines), https://tcpp.cs.gsu.edu/curriculum/?q=node/21742#3 (2020).

[6] The NSF/IEEE-TCPP Curriculum Initiative, The CDER center, https://tcpp.cs.gsu.edu/curriculum/?q=node/21183 (2012).

[7] J. C. Adams, R. Brown, S. J. Matthews, E. Shoop, CSinParallel: Parallel computing in the computer science curriculum, https://csinparallel.org/ (2010).

[8] R. Brown, E. Shoop, Teaching undergraduates using local virtual clusters, in: 2013 IEEE International Conference on Cluster Computing (CLUSTER), 2013, pp. 1–8. doi:10.1109/CLUSTER.2013.6702622.

[9] J. Eckroth, Teaching big data with a virtual cluster, in: Proceedings of the 47th ACM Technical Symposium on Computing Science Education, SIGCSE '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 175–180. doi:10.1145/2839509.2844651.
URL https://doi.org/10.1145/2839509.2844651

[10] J. Eickholt, S. Shrestha, Teaching big data and cloud computing with a physical cluster, in: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, SIGCSE '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 177–181. doi:10.1145/3017680.3017705.
URL https://doi.org/10.1145/3017680.3017705

[11] A. S. Rabkin, C. Reiss, R. Katz, D. Patterson, Experiences teaching mapreduce in the cloud, in: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, SIGCSE '12, Association for Computing Machinery, New York, NY, USA, 2012, p. 601–606. doi:10.1145/2157136.2157310.
URL https://doi.org/10.1145/2157136.2157310

[12] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. R. Scott, N. Wilkins-Diehr, Xsede: Accelerating scientific discovery, Computing in Science & Engineering 16 (5) (2014) 62–74. doi:10.1109/MCSE.2014.80.

[13] J. A. Ariza, H. Baez, Understanding the role of single-board computers in engineering and computer science education: A systematic literature review, Computer Applications in Engineering Education 30 (1) (2022) 304–329.

[14] B. Krupp, A. Watkins, Cs0: Introducing computing with raspberry pis, in: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, SIGCSE '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 832–838. doi:10.1145/3287324.3287488.
URL https://doi.org/10.1145/3287324.3287488

[15] M. Wirth, J. McCuaig, Making programs with the raspberry pi, in: Proceedings of the Western Canadian Conference on Computing Education, WCCCE '14, Association for Computing Machinery, New York, NY, USA, 2014. doi:10.1145/2597959.2597970.
URL https://doi.org/10.1145/2597959.2597970

[16] M. Kölling, Educational programming on the raspberry pi, Electronics 5 (3) (2016). doi:10.3390/electronics5030033.
URL https://www.mdpi.com/2079-9292/5/3/33

[17] D. Tarnoff, Integrating the arm-based raspberry pi into an architecture course, J. Comput. Sci. Coll. 30 (5) (2015) 67–73.

[18] J. Kawash, A. Kuipers, L. Manzara, R. Collier, Undergraduate assembly language instruction sweetened with the raspberry pi, in: Proceedings of the 47th ACM Technical Symposium on Computing Science Education, SIGCSE '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 498–503. doi:10.1145/2839509.2844552.
URL https://doi.org/10.1145/2839509.2844552

[19] W. Zhu, Teaching assembly programming for arm-based microcontrollers in a professional development kit, in: 2017 IEEE International Conference on Microelectronic Systems Education (MSE), 2017, pp. 23–26. doi:10.1109/MSE.2017.7945077.

[20] P. J. McGee, R. Latinovich, D. Brylow, Using embedded xinu and the raspberry pi 3 to teach operating systems, in: 2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2020, pp. 307–315. doi:10.1109/IPDPSW50202.2020.00063.

[21] E. Sowell, Effects of manipulative materials in mathematics instruction, Journal for Research in Mathematics Education 20 (5) (1989) 498–505, dOI=10.2307/749423.

[22] W. Carroll, D. Porter, Invented strategies can develop meaningful mathematical procedures, Teaching Children Mathematics 3 (7) (1997) 370–374.

[23] D. Clements, “concrete” manipulatives, concrete ideas, Contemporary Issues in Early 1 (1) (Childhood) 45–60.

[24] L. Jordan, M. Miller, C. Mercer, The effects of concrete to semi-concrete to abstract, instruction in the acquisition and retention of fraction concepts and skills 9 (1998) 115–122.

[25] P. Moch, Manipulatives work!, The Educational Forum (Fall 2001).

[26] R. Ross, R. Kurtz, Making manipulatives work: A strategy for success, The Arithmetic Teacher 40 (1993) 254–258.

[27] M. Chappell, M. Strutchens, Mathematics Teaching in the Middle School, National Council of Teachers of Mathematics, 2001, Ch. Creating connections: Promoting algebraic thinking with concrete models.

[28] S. A. Papert, Mindstorms: Children, computers, and powerful ideas, Basic books, 2020.

[29] M. Horn, M. Bers, et al., Tangible computing, The Cambridge handbook of computing education research 1 (2019) 663–678.

[30] M. Przybylla, R. Romeike, Physical computing and its scope–towards a constructionist computer science curriculum with physical computing., Informatics in Education 13 (2) (2014) 241–254.

[31] M. Resnick, S. Ocko, S. Papert, Lego, logo, and design, Children’s Environments Quarterly (1988) 14–18.

[32] M. Resnick, Multilogo: A study of children and concurrent programming, Interactive learning environments 1 (3) (1990) 153–170.

[33] S. J. Cox, J. T. Cox, R. P. Boardman, S. J. Johnston, M. Scott, N. S. O'brien, Iridis-pi: a low-cost, compact demonstration cluster, Cluster Computing 17 (2) (2014) 349–358.

[34] A. M. Pfalzgraf, J. A. Driscoll, A low-cost computer cluster for high-performance computing education, in: IEEE International Conference on Electro/Information Technology, 2014, pp. 362–366. doi:10.1109/EIT.2014.6871791.

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

[36] J. Wolfer, A model supercomputer for instructional support, in: 2015 3rd Experiment International Conference (exp.at'15), 2015, pp. 114–115. doi:10.1109/EXPAT.2015.7463231.

[37] S. J. Matthews, Teaching with parallella: A first look in an undergraduate parallel computing course, Journal of Computing Sciences in Colleges 31 (3) (2016) 18–27.

[38] J. C. Adams, S. J. Matthews, E. Shoop, D. Toth, J. Wolfer, Using inexpensive microclusters and accessible materials for cost-effective parallel and distributed computing education, Journal of Computational Science Education 8 (3) (2017) 2.

[39] S. J. Matthews, J. C. Adams, R. A. Brown, E. Shoop, Portable parallel computing with the Raspberry Pi, in: Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGCSE '18,

Association for Computing Machinery, New York, NY, USA, 2018, p. 92–97. doi:10.1145/3159450.3159558.
URL https://doi.org/10.1145/3159450.3159558

[40] S. Holt, A. Meaux, J. Roth, D. Toth, Making the one cluster per student method of teaching parallel computing financially practical, J. Comput. Sci. Coll. 33 (4) (2018) 106–113.

[41] D. Gooch, J. Rosewell, D. Leith, M. Richards, Passive or active learning: the challenges of teaching distributed computing using raspberry pi clusters to open distance university students, Open Learning: The Journal of Open, Distance and e-Learning (2022) 1–15.

[42] 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.

[43] B. Miller, 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.

[44] S. J. Matthews, E. Shoop, R. Brown, J. C. Adams, Learnpdc.org - free hands-on materials for learning pdc, https://www.learnpdc.org/ (2020).

[45] C. Kehoe, J. Stasko, A. Taylor, Rethinking the evaluation of algorithm animations as learning aids: an observational study, International Journal of Human-Computer Studies 54 (2) (2001) 265–284.

[46] T. L. Naps, G. Rößling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger, J. A. Velázquez-Iturbide, Exploring the role of visualization and engagement in computer science education, SIGCSE Bull. 35 (2) (2002) 131–152. doi:10.1145/782941.782998.
URL https://doi.org/10.1145/782941.782998

[47] C. A. Shaffer, M. L. Cooper, A. J. D. Alon, M. Akbar, M. Stewart, S. Ponce, S. H. Edwards, Algorithm visualization: The state of the field, ACM Transactions on Computing Education (TOCE) 10 (3) (2010) 1–22.

[48] E. Fouh, M. Akbar, C. A. Shaffer, The role of visualization in computer science education, Computers in the Schools 29 (1-2) (2012) 95–117.

[49] J. C. Adams, P. A. Crain, C. P. Dilley, S. M. Nelesen, J. B. Unger, M. B. Vander Stel, Seeing is believing: Helping students visualize multithreaded behavior, in: Proceedings of the 47th ACM Technical Symposium on Computing Science Education, SIGCSE '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 473–478. doi:10.1145/2839509.2844557.
URL https://doi.org/10.1145/2839509.2844557

[50] S. J. Matthews, E. Shoop, R. Brown, J. C. Adams, Raspberry pi - virtual handout, https://www.learnpdc.org/RaspberryPiHandout/ (2020).

[51] E. Shoop, S. J. Matthews, R. Brown, J. C. Adams, Distributed comput-

ing using python and the raspberry pi, https://www.learnpdc.org/RaspberryPi-mpi4py/ (2020).

[52] E. Shoop, S. J. Matthews, R. Brown, J. C. Adams, Distributed computing using mpi and the raspberry pi, https://www.learnpdc.org/RaspberryPi-mpi/ (2020).

[53] S. J. Matthews, Raspberry pi os - 64-bit headless with vnc, https://www.suzannejmatthews.com/post/2021-08-17-raspberrypi4-headless/ (2021).

[54] R. Brown, E. Shoop, S. J. Matthews, R. Brown, J. C. Adams, Csinparallel raspberry pi soc cluster image, https://www.learnpdc.org/images/shrunk_csip_mpi_010622.img.zip (2020).

[55] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria (2021). URL https://www.R-project.org

[56] J. C. Adams, R. Brown, S. J. Matthews, E. Shoop, Teaching pdc in the time of covid: Hands-on materials for remote learning, in: 2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2021, pp. 342–349. doi:10.1109/IPDPSW52791.2021.00061.