# Portable Parallel Computing with the Raspberry Pi

Suzanne J. Matthews[*]
Dept. of Electrical Engineering & Computer Science
suzanne.matthews@usma.edu

Joel C. Adams
Dept. of Computer Science
adams@calvin.edu

Richard A. Brown
Dept. of Mathematics, Statistics, & Computer Science
rab@stolaf.edu

Elizabeth Shoop
Dept. of Mathematics, Statistics, & Computer Science
shoop@macalester.edu

## ABSTRACT

With the requirement that parallel & distributed computing (PDC) topics be covered in the core computer science curriculum, educators are exploring new ways to engage students in this area of computing. In this paper, we discuss the use of the Raspberry Pi single-board computer (SBC) to provide students with hands-on multicore learning experiences. We discuss how the authors use the Raspberry Pi to teach parallel computing, and present assessment results that indicate such devices are effective at achieving CS2013 PDC learning outcomes, as well as motivating further study of parallelism. We believe our results are of significant interest to CS educators looking to integrate parallelism in their classrooms, and support the use of other SBCs for teaching parallel computing.

## CCS CONCEPTS

**• Computing methodologies → Parallel computing methodologies**; **• Applied computing → Education**;

## KEYWORDS

Parallel Computing, Raspberry Pi, Education

**ACM Reference Format:**
Suzanne J. Matthews, Joel C. Adams, Richard A. Brown, and Elizabeth Shoop. 2018. Portable Parallel Computing with the Raspberry Pi. In *SIGCSE '18: The 49th ACM Technical Symposium on Computer Science Education, Feb. 21–24, 2018, Baltimore, MD, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3159450.3159558

## 1 INTRODUCTION

Prior to 2006, parallel & distributed computing (PDC) topics were rarely covered in the computer science (CS) curriculum, let alone at the undergraduate level. The expense of parallel hardware (among other reasons) made PDC challenging to teach. This has largely changed over the last decade, due to the advent of multicore and manycore (GPU) architectures and the emergence of cloud computing. These innovations have greatly decreased the cost and increased the access to parallel architectures.

[*]Corresponding Author

Given the ubiquity of these technologies, the *ACM/IEEE CS 2013 Curriculum Recommendations* [16] moved PDC topics into the core CS curriculum. Unsurprisingly, many of the PDC core curriculum recommendations are focused on topics and concepts specific to shared memory (as opposed to distributed memory) parallel systems. This is due to the fact that all modern phones, tablets, laptops, and desktop systems contain multicore CPUs, making them shared memory parallel multiprocessors. It follows that students who create software need to be educated in the concepts and techniques required to write efficient software for multicore systems. CS 2013 thus requires every CS major to learn about PDC.

This raises many questions, including:

> *What hardware platforms should we use to teach students about parallelism (esp. shared memory)?*

In this paper, we argue that inexpensive single-board computers (SBCs) such as the Raspberry Pi provide a new option for teaching parallel computing. While we focus on the Raspberry Pi in this paper, we believe the same argument applies to other affordable SBCs. The advantages single board computers provide for teaching students about parallel computing include:

*Standardization.* Pre-configured disk images may be freely downloaded from the Internet (e.g., [7]), ensuring a uniform work environment for each student with the same operating system, compilers, debuggers, integrated development environments (IDEs), and other software development tools. Images can be pre-loaded with pedagogical code examples and instructions.

*Ease of setup and maintenance.* To start programming, it suffices for students to insert a pre-configured microSD card into their SBC and boot the system. If the disk image gets corrupted, students can simply download a new image and start over. For busy faculty who are new to parallelism and are trying to quickly inject concepts into a course, this ease of setup cannot be overstated.

*Affordability.* Compared to other parallel platforms, SBCs like the Raspberry Pi are very inexpensive. Some cost less than a typical textbook, potentially allowing each student to purchase her own device. Students can connect their SBCs to university-owned (or previously purchased) peripherals such as monitors, mice and keyboards. To minimimze costs further, students can purchase pre-loaded microSD cards and boot a university-owned SBC.

*Immediacy.* On a typical SBC, the CPU, memory unit, storage device, and other components are clearly visible, making them readily available for visual and tactile learners. Since their software is running on actual hardware, students can perform timing and scalability studies without fearing resource contention with other students or performance penalties from virtualization.

## 2 WHY NOT LAPTOPS, PHONES OR VMS?

Given the ubiquity of multicore CPUs, just about any modern computer has hardware to teach students about parallel computing. However several platforms have drawbacks, including:

**Mobile devices** (i.e., smartphones and tablets) have small screens, limited ports, and lack IDEs, making them poor platforms for *writing* software.

**Student laptops** offer larger screens, better ports, and IDEs for writing software, compared to mobile devices. However:

- Students may have different operating systems (e.g., Chrome, Linux, MacOS, Windows) on their laptops. This diversity can make it a challenge for instructors to create instructional materials that work for all students.
- Most laptop operating systems do not come with compilers, debuggers, IDEs, and other software development tools pre-installed. Instead, the student must install and configure these tools, creating a potential barrier to student learning.
- A laptop's expense is often correlated with its number of cores. A course in which students are expected to develop parallel software on their own laptops may thus discriminate against students from disadvantaged backgrounds, who may not be able to afford laptops with four or more cores. This in turn limits the amount of parallelism that can be observed on their system.

**Virtual machines** (VMs) running on laptops, desktops, or remote servers can be pre-configured in a way that eliminates many of the downsides of student laptops. However in our experience, running parallel software on a VM (especially free ones) incurs a non-trivial performance penalty.

**Remote multicore servers** can eliminate the preceding drawbacks. However, using remote hardware for parallel computing:

- Requires account management for students in courses, extending instructor or department overhead.
- Requires a mechanism to prevent student programs from interfering with one another. To accomplish this, students may need to use a reservation system, or submit their programs via a batch queuing system.
- Requires students to view the computer as an abstract entity. For students with particular learning styles (e.g., who learn best by seeing and/or by tactile manipulation), a local system they can touch and whose operation they can observe would be preferable.

In short, SBCs eliminate the drawbacks associated with mobile devices, student laptops, virtual machines, and remote servers, making them a viable platform for parallel computing education. They are also fun devices that many students find highly motivating.

## 3 BACKGROUND AND RELATED WORK

Perhaps the most well-known SBC is the Raspberry Pi, a $35.00 credit-card sized computer. The Raspberry Pi Model 3 features a 1.2 GHz quad-core ARM processor and 1 GB of RAM. Its low cost and small form factor inspires hobbyists to use the Raspberry Pi for various projects, including robots, drones and webservers.

Rasbian, the Raspberry Pi operating system, comes pre-configured with Scratch and Python. A goal of the Raspberry Pi project is to provide an inexpensive, accessible platform that enables children and teachers to learn computing. Notable early efforts include the Glyndŵr/BCS Turing Project [8], and the Bridge21 CPD effort [4]. Both projects aimed to expose high school students and teachers to computing concepts using the Raspberry Pi (the former used the Pi in conjunction with other devices). In both studies, the hands-on approach was well received, with educators in the Bridge21 CPD effort keen to incorporate the Raspberry Pi into their courses [4].

Educators at the Universities of Guelph [19] and Calgary [9, 10] were among the first to use the Raspberry Pi to teach programming topics in a college setting. Guelph students used the Raspberry Pi to learn beginning C programming, while second-year Calgary students used it to learn embedded programming (ARM assembly). At both universities, a dedicated Raspberry Pi lab was created for students to plug in their devices. At Guelph, students bought a custom Raspberry Pi "kit" as part of their standard lab equipment in the course [19]. Calgary students, however, were required to come to the Raspberry Pi lab in order to use their devices. While the Guelph students were markedly enthusiastic for the Pis, the reaction of the Calgary students was more neutral [9] – students were enthusiastic about working with the Pis, but they were frustrated by the "awkwardness" of the lab set up and scarcity of the shared resources [9]. This suggests the "one SBC per student" approach is preferable for CS education. We also note that Raspberry Pis were used to teach assembly at East Tennessee State University [15], though no assessment data was provided.

Early SBCs (including the earlier Raspberry Pis) had one core, so a single SBC did not lend itself well to teaching parallel computing. Instead, researchers and 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). IridisPi [5] is an early example using Raspberry Pis. Pfalzgraf and Driscoll proposed using single-core Raspberry Pis in conjunction with MPI to teach about HPC concepts [13]. Toth used dual-core Cubie board SBCs to enable each student to have their own 2-node Beowulf cluster [18] in a parallel computing course.

However, modern SBCs have multiple cores and may include access to co-processor or GPU chips. This enables the teaching of shared memory parallelism on SBCs for the first time (see [12] for an example). In this paper, we discuss efforts to use the Raspberry Pi to teach multicore computing. To the best of our knowledge, we are the first to suggest the use of the Raspberry Pi for this purpose. In the sections that follow, we discuss strategies and materials we have used to teach shared memory programming to students at our respective universities and integrate these concepts into our curricula. Motivated by the literature, we present workshop assessment data evaluating the efficacy and enthusiasm SBCs generate when teaching multicore computing concepts.

## 4 PI TEACHING PLATFORMS & ASSESSMENT

The small form factor of SBCs make them highly portable and amenable for introducing parallel concepts at workshops, without needing to coordinate access to remote architectures. Matthews et al. [7] recently organized a series of workshops (described in Section 4.2) centered on using the Raspberry Pi 3 to learn and teach parallel computing concepts. The workshops were extremely popular, with 15 to 30 people attending each.

In the following subsections, we introduce available teaching materials and present assessment results of using them in the workshop settings. Our goals included observing: (i) how well parallel computing could be taught using an SBC, and (ii) whether or not using an SBC would motivate participants to learn about parallel computing. In the assessments that follow, we focus on the Raspberry Pi 3. However, we stress that almost any SBC offers the same advantages listed in Section 1 – affordability, low power consumption, ease of setup/maintenance, and immediacy. We hypothesize that showing the effectiveness of one SBC for teaching parallelism indicates the effectiveness of others for teaching PDC.

## 4.1 Available PDC Teaching Modules

We have developed teaching modules, available at *csinparallel.org*, that are designed to be used in a variety of courses in a day or two, enabling instructors to include PDC topics in existing courses. The *csinparallel.org* patternlet modules are especially useful for teaching concepts to students in first and second-year undergraduate courses [2].

Patternlets [1] are small code examples that use established parallel design patterns to demonstrate particular PDC topics. There are currently 17 OpenMP patternlets for teaching shared memory parallel topics, as well as a number of patternlets for POSIX multithreading, MPI, and heterogeneous (MPI+OpenMP) computing.

```
#include <stdio.h>
#include <omp.h>
int main(int argc, char** argv) {
    printf("\nBefore...\n");
//    #pragma omp parallel
    printf("\nDuring...");
    printf("\n\nAfter...\n\n");
    return 0;
}
```

**Figure 1: Sample OpenMP fork-join patternlet.**

The patternlet examples can be used to achieve several of the CS2013 PD outcomes by enabling students to explore a concept using a simple program (typically one screen of code or less) that, when changed slightly, illustrates the desired learning outcome. Figure 1 shows an OpenMP patternlet that illustrates the fork-join pattern. Students explore this pattern by first running the program shown in Figure 1, observing the results, then uncommenting the pragma and re-running the program to observe changes in output.

The teaching modules accompanying each patternlet use diagrams to further illustrate concepts and provide questions for students to consider while trying out examples [6].

There are additional teaching materials at *csinparallel.org* that can be used in the classroom with the Raspberry Pi. Table 1 depicts the PD Core Tier 1/2 topic areas and learning outcomes from ACM 2013 that can be taught with Raspberry Pi SBCs. Our previous work [3] assessed several of these modules and showed that they met learning objectives similar to those in Table 1 in the Parallel Decomposition, Communication and Coordination, and Parallel Algorithms, Analysis and Programming areas. For example, a module on multicore programming with OpenMP enables students to use data decomposition, and to observe and fix a race condition on a shared variable.

**Table 1: PD Core Topic Areas [16] Teachable on the Pi.**

| PD Knowledge Area | Outcomes |
|---|---|
| Parallel Fundamentals | 1. Shared resource access<br>2. Types of synchronization |
| Parallel Decomposition | 1. Explain need for synchronization<br>2. Identify parallelism opportunities<br>3. Write correct scalable PD program<br>4. Use tasked-based decomposition<br>5. Use data decomposition |
| Communication & Coordination | 1. Use mutual exclusion<br>2. Give an example of data race<br>5. Write concurrent task program<br>6. Use synchronized task queue<br>7. Explain need for atomicity<br>8. Write a program that reveals a data race |
| Parallel Algorithms, Analysis, and Programming | 3. Define speedup and scalability<br>4. Identify independent tasks<br>5. Describe what can(not) be parallelized<br>6. Implement parallel divide and conquer |
| Parallel Architecture | 1. Explain shared vs. distributed memory<br>2. Describe SMP architecture<br>3. Describe tasks that match SIMD |

## 4.2 Raspberry Pi Workshops

The authors led parallel computing workshops at the 2016 ACM Richard Tapia Conference, the 2017 SIAM Computational Science & Engineering (CSE) Broader Engagement program, and the 2017 ACM SIGCSE Conference. Each workshop lasted about 90 minutes, and used the quad-core Raspberry Pi 3 to introduce the participants to OpenMP programming. Each workshop followed roughly the same format, though were presented to radically different audiences. The workshops at Tapia and CSE were attended primarily by students, while the workshop at SIGCSE was attended by faculty.

**Figure 2: Views of the Pimoroni unit used in workshops.**

*4.2.1 Workshop Overview.* Each pair of participants shared a "Pimoroni" [11] unit: a Raspberry Pi 3 mounted to a Pimoroni 7" display, plus a USB keyboard and mouse (see Figure 2). Each unit cost approximately $150.00. Since the quality of wireless at the workshops was unknown beforehand, paper handouts were prepared for each participant, including a "Raspberry Pi Basics" handout, adapted from the CSinParallel OpenMP patternlets module. We publicly share our workshop materials at this link [7].

**Table 2: Results of Pre-Survey and Post-Survey of OpenMP workshops.**

| | Pre-Survey | | | Post-Survey | | | $p$-values |
|---|---|---|---|---|---|---|---|
| | T | C | S | T | C | S | |
| Questions/ Number of Responses | 33 | 16 | 17 | 32 | 16 | 17 | |
| 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 | 2.38 | 2.88 | 3.66 | 4.18 | 4.06 | T: $3.129 \times 10^{-8}$ C: $1.653 \times 10^{-4}$ S: $4.556 \times 10^{-3}$ |
| 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 | 3.06 | 3.17 | 3.94 | 4.38 | 4.18 | T: $1.071 \times 10^{-6}$ C: $6.742 \times 10^{-4}$ S: $2.753 \times 10^{-2}$ |
| 3. How confident are you that you can define speedup and describe it to someone familiar with programming? | 2.27 | 2.81 | 3.12 | 4.03 | 4.5 | 4.23 | T: $5.912 \times 10^{-7}$ C: $1.965 \times 10^{-4}$ S: $2.081 \times 10^{-2}$ |
| 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 | 2.81 | 3.12 | 3.83 | 4.5 | 4.17 | T: $2.349 \times 10^{-5}$ C: $6.933 \times 10^{-4}$ S: $7.663 \times 10^{-3}$ |
| 5. To what extent did using an inexpensive multi-core computer (e.g. the Raspberry Pi) to run parallel programs motivate you to learn more about parallel computing in the future? | n/a | n/a | n/a | 4.22 | 4.13 | 3.88 | n/a |

Each workshop began with a short presentation on the Raspberry Pi 3, multicore architectures, and shared memory parallelism. We discussed the differences between processes and threads, and how threads can be executed concurrently or in parallel. The participants spent the next 50 minutes working in pairs, interactively completing the OpenMP exercises in the "Raspberry Pi Basics" handout. In the final 20 minutes of the workshop, the participants used the Pi to run a drug-design CSinParallel module and complete an accompanying worksheet. This module contains three versions of a program simulating the design of pharmaceutical drugs: two using OpenMP (with different thread-scheduling options) and one using C++11 threads. Students ran all three programs, varying the numbers of cores, measured the run times, and calculated speedup. Students were then asked to think about why the three programs ran differently. In the last few minutes of the workshop, we discussed the results, and why the programs behaved differently.

Each workshop had between 15 and 30 participants. While the workshop was designed with computer science undergraduates with no prior exposure to parallelism in mind, we had very diverse audiences. Some attendees were indeed undergraduates who had never been exposed to parallel computing. Others were graduates students with some prior parallel programming experience. Still others were faculty members seeking ideas on how to teach parallelism at their own institutions.

Prior to each workshop, we gave a 4-question pre-survey, asking the participants to rate their confidence on each of four learning objectives, using a scale of 1 to 5 with "1" indicating "not confident at all", "3" indicating "somewhat confident", and "5" indicating "very confident". The survey questions are listed in Table 2.

A post-survey was administered at the conclusion of the workshop that was identical to the pre-survey, except for a fifth question that asked participants to assess (on a scale of 1 to 5) the effect of the Raspberry Pi (or a device like it) 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 with an invitation to explain their answer further through an open-ended response.

*4.2.2 Results of Assessment.* Table 2 shows the results of our surveys. The third row reports the number of responses we received on the pre- and post-surveys for the Tapia (T), SIAM CSE (C), and SIGCSE (S) workshops. (For example, 33 Tapia 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 [17] 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 last column of Table 2 shows 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 workshops, despite the fact that each had different mixtures of faculty and students. The Tapia workshop (which had our lowest $p$-value) had the highest number of novice students; the CSE workshop had more experienced students; and the SIGCSE workshop was made up of almost entirely faculty. Unsurprisingly, more advanced groups had higher $p$-values than the novice group; however, all measured $p$-values are below 0.05. This strongly suggests that our approach can be used to cover many of the PDC learning outcomes of Table 1.

The fifth question in Table 2 only appeared on the post-survey. The mean for the fifth question was 4.22 for the Tapia workshop, 4.13 for CSE, and 3.88 for SIGCSE; the more students in a workshop, the greater the positive effect. A histogram depicting the combined set of responses to question 5 is shown in Figure 3.

The overall mean for the fifth question responses was 4.19, and 40 of the 48 total respondents answered either a "4" or "5". This suggests to us that the vast majority of the participants felt the

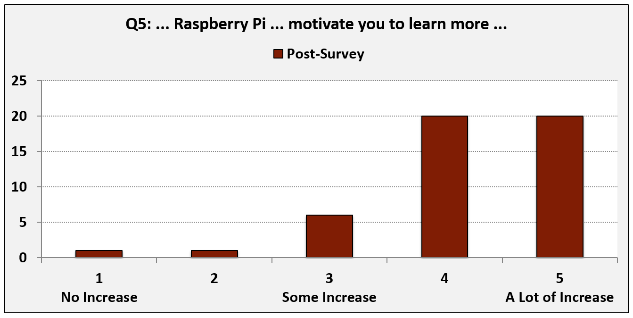

**Figure 3: Question 5 histogram of post-workshop survey.**

Raspberry Pi 3 SBC was a very positive factor in motivating them to learn more about parallel computing.

More information can be gleaned from the open-ended responses; 40 (83%) of the respondents to the post quiz provided an open-ended response. Their feedback was largely very positive and enthusiastic.

Several participants remarked at the perceived impact of the Raspberry Pi for learning about parallel computing. Said one respondent: *"The impact of it is incredible, would love to learn more."* One of our SIGCSE participants remarked, *"I have experience with parallel processing/programming, but not so much pis. That's why I took the workshop. I love using the pis! Wonderfully motivating! Gets students closer to the hardware and powerful enough to motivate studying parallelism. Great workshop!"*

Still other participants noted how effectively the Raspberry Pi could be used to teach OpenMP. *"Awesome seeing how easy this was to teach"*, said one of our Tapia participants. *"You can get right to the point quickly"*, said another. One of our SIGCSE participants remarked, *"I am impressed with how fast and easy it was to demonstrate this[sic] concepts. This is something I can easily see doing with students. The ease of the examples and the simplicity of the OpenMP inspires me to do this in class!!"*

Perhaps the biggest impact of our workshop was the fact that attendees could explore parallel computing on an inexpensive, personal device. *"Not having to have an expensive computer to try this stuff on is really motivating"*, noted one Tapia participant. *"Accessibility Helps!"* wrote another. *"I'll probably try to get my own Raspberry Pi to practice more and write my own code for it"* remarked one of our CSE attendees. *"Simple system and inexpensive"* stated a SIGCSE participant.

The newness of the Raspberry Pi added an extra dimension of novelty and relevancy to the material for others. *"It's fun to see it work on a small machine and get to play with the tech I keep hearing about (raspberry pi)"*, said a Tapia attendee. *"Amazing parallel processing system with multicore processor!"* remarked a SIGCSE participant. The perceived value was succinctly stated by another Tapia participant: *"Simple, cheap, not too intimidating! Very awesome =)"*.

We acknowledge the concepts taught on the Raspberry Pi can be taught on a workstation or supercomputer node. *"I think it's easier on a workstation"*, grumbled one SIGCSE participant, perhaps due to the Pimoroni's small screen. However, the portability and inexpensiveness of the Raspberry Pi makes it possible to teach parallel computing concepts while maintaining the transferability of knowledge to larger systems. *"It can help run lab experiments that can later be implemented at a larger scale"* observed one attendee. Another noted that *"The somewhat limited resources on the Pi (as compared to KVL or something) really makes you focus on efficiency and proper programming"*.

The true value of the Pi is having a personal machine that is relateable and inexpensive. *"I am already very motivated because I plan to teach the course ..."* said a SIGCSE participant, *"but my expectation is using an inexpensive system will motivate the STUDENTS to do so and I am really interested in how much that is true"*.

*4.2.3 Lessons Learned.* Teaching OpenMP concepts with the Raspberry Pi in a workshop setting was clearly effective. Furthermore, within a 90-minute period, we were able to touch upon several outcomes listed in the PD knowledge area, including outcomes 1 and 2 under parallel fundamentals; outcomes 1,2, and 4 under parallel decomposition, outcomes 1 and 2 under communication & coordination; outcome 3 under parallel algorithms, analysis, & programming; and outcome 2 under parallel architecture.

The feedback we received from the workshop indicates that people enjoyed using the Raspberry Pi to learn parallel programming. Put differently, our workshop participants found the use of SBCs to be highly motivating, and we believe that students will find the study of PDC using SBCs in the classroom equally motivating.

After the workshops, we held a series of "debriefing" sessions. During these sessions, several of us reflected on how much easier it was (logistically) to conduct this workshop using SBCs, compared to previous workshops in which we used remote HPC systems. In several prior workshops, automated security systems, poor wireless connectivity, electrical storms, misconfigured accounts, and other remote system issues caused problems. Such issues caused a great deal of stress, since we could not fix the systems ourselves. As a result, we always needed to have a "Plan B" system in place for our workshops, just in case something went wrong.

To prepare for our workshop using SBCs, we simply flashed all the devices' disks prior to the workshop, and then re-flashed them with fresh images afterwards. All the units fit into a single pelican case that was mailed to the workshop location prior to the event, and returned afterwards. For the SIGCSE workshop, we were able to check the Pis in their pelican case as a separate piece of luggage.

## 4.3 Alternative Teaching Platforms

The Pimoroni-based setup for teaching PDC material using Raspberry Pis has proven to be portable and effective, as demonstrated in the workshop experiences described above. This section describes an alternative setup that costs $50.00 and enables students to interact directly with the Pi's desktop environment via a network connection from a laptop or lab workstation. The system image [7] works as-is on the Pimoroni, or through the configuration described below, and contains the PDC materials used in our workshops.

Each kit (shown in Figure 4) consists of a Raspberry Pi, a Micro-USB card with the system image installed, cables, and a USB-to-Ethernet adapter (for laptops lacking an Ethernet port). The image enables students running Windows, OS X, or Linux to access the Pi desktop environment using open-source virtual network computing (VNC) [14] software. Students connect to the Pi using a direct wired

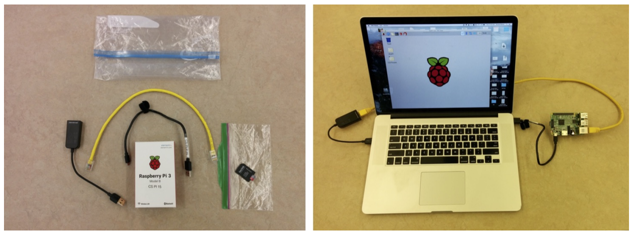

**Figure 4: Laptop connection kit.**

Ethernet connection between their laptops and the Pi, or through the use of a USB-to-Ethernet adapter.

The right side of Figure 4 depicts a VNC client displaying the Raspberry Pi desktop for user interaction. Users can also interact with their Pi on a command line via SSH (the VNC client requires an underlying SSH session). With printed or video instructions, we found that first-time users can assemble the system and begin computing in 10-15 minutes, even with no prior Raspberry Pi experience (except when hardware issues arise; e.g., a faulty USB socket). After the first time, setup and tear-down time totals about five minutes, making this approach feasible for in-class exercises.

## 5 CONCLUSIONS AND FUTURE WORK

In this paper, we describe how single board computers like the Raspberry Pi can be effectively used to introduce undergraduate students to parallel computing in a workshop setting. SBCs embody the "hands-on experiential" learning advocated by CS educators, and are a fun way to introduce students to parallel computing.

SBCs can also be networked together to create "microclusters" to introduce students to distributed and heterogeneous computing. For example, our current system image supports building Beowulf clusters through the use of a switch; we plan to extend this image to enable Pis to self-assemble into a cluster.

We have also discussed strategies for teaching parallel computing with the Raspberry Pi. In particular, the Raspberry Pi costs less than many course textbooks. With the availability of free, high-quality teaching materials, Raspberry Pis can be individually purchased by students like a textbook. This makes the Pi useful in a variety of different contexts, including required courses, elective courses, undergraduate research, and outreach experiences.

Our collective experiences strongly suggest that the Raspberry Pi is an inexpensive, accessible, cost-effective, and highly motivating way to introduce undergraduate students to parallel and distributed computing. We hope that our experiences will inspire others to explore the use of the Raspberry Pi and other SBCs to engage their students and help them achieve PDC learning outcomes.

As the price/performance ratios of system on a chip (SoC) architectures continue to improve, we will continue to see SBCs with updated processors. For example, the ODROID XU4 offers an octa-core processor for $59.00. We look forward to future work that explores the use of such systems to increase student engagement through teaching, research, and outreach activities.

## ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers for their feedback on this work. Funding for this project was partially provided by the National Science Foundation (NSF) and the DOD High Performance Computing Modernization Program (HPCMP). Adams, Brown, and Shoop are funded under NSF DUE Grants 1225739/1225796/1226172. Matthews is funded under HPCMP MIPRs 61550445/62464243. The opinions expressed in this work are solely of the authors and do not necessarily reflect those of the U.S. Military Academy, the U.S. Army, or the Department of Defense.

## REFERENCES

[1] Joel C. Adams. 2014. Injecting Parallel Computing into CS2. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. ACM, New York, NY, USA, 277–282. https://doi.org/10.1145/2538862.2538883

[2] Joel C. Adams. 2015. Patternlets: A Teaching Tool for Introducing Students to Parallel Design Patterns. In *Proceedings of the 2015 IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW '15)*. 752–759. https://doi.org/10.1109/IPDPSW.2015.18

[3] Richard Brown and Elizabeth Shoop. 2011. Modules in community: injecting more parallelism into computer science curricula *(SIGCSE '11)*. ACM, New York, NY, USA, 447–452. https://doi.org/10.1145/1953163.1953293

[4] J. R. Byrne, L. Fisher, and B. Tangney. 2015. Computer science teacher reactions towards raspberry Pi Continuing Professional Development (CPD) workshops using the Bridge21 model. In *2015 10th International Conference on Computer Science Education (ICCSE)*. 267–272. https://doi.org/10.1109/ICCSE.2015.7250254

[5] Simon J. Cox, James T. Cox, Richard P. Boardman, Steven J. Johnston, Mark Scott, and Neil S. O'Brien. 2014. Iridis-pi: a low-cost, compact demonstration cluster. *Cluster Computing* 17, 2 (01 Jun 2014), 349–358. https://doi.org/10.1007/s10586-013-0282-7

[6] CSinParallel. 2013. Patternlets in Parallel Programming. (2013). https://csinparallel.org/csinparallel/modules/patternlets.html

[7] CSinParallel. 2017. Raspberry Pi Workshop Materials. (2017). https://csinparallel.org/csinparallel/raspberry_pi.html

[8] Vic Grout and Nigel Houlden. 2014. Taking Computer Science and Programming into Schools: The Glyndŵr/BCS Turing Project. *Procedia - Social and Behavioral Sciences* 141, Supplement C (2014), 680 – 685. https://doi.org/10.1016/j.sbspro.2014.05.119 4th World Conference on Learning Teaching and Educational Leadership (WCLTA-2013).

[9] Jalal Kawash, Andrew Kuipers, Leonard Manzara, and Robert Collier. 2016. Undergraduate Assembly Language Instruction Sweetened with the Raspberry Pi. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 498–503. https://doi.org/10.1145/2839509.2844552

[10] M Lowey. 2013. Raspberry pi sweetens learning for computer science students. (2013). https://www.ucalgary.ca/utoday/issue/2013-11-13/raspberry-pi-sweetens-learning-computer-science-students

[11] Pimoroni Ltd. 2015. Raspberry Pi 7" Touchscreen Display with Stand. (2015). https://shop.pimoroni.com/

[12] Suzanne J. Matthews. 2016. Teaching with Parallella: A First Look in an Undergraduate Parallel Computing Course. *J. Comput. Sci. Coll.* 31, 3 (Jan. 2016), 18–27. http://dl.acm.org/citation.cfm?id=2835377.2835381

[13] A. M. Pfalzgraf and J. A. Driscoll. 2014. A low-cost computer cluster for high-performance computing education. In *IEEE International Conference on Electro/Information Technology*. 362–366. https://doi.org/10.1109/EIT.2014.6871791

[14] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. 1998. Virtual network computing. *IEEE Internet Computing* 2, 1 (Jan 1998), 33–38. https://doi.org/10.1109/4236.656066

[15] David Tarnoff. 2015. Integrating the Arm-based Raspberry Pi into an Architecture Course. *J. Comput. Sci. Coll.* 30, 5 (May 2015), 67–73. http://dl.acm.org/citation.cfm?id=2752981.2752998

[16] The ACM/IEEE Joint Task Force on Computing Curricula. 2013. Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. (December 2013).

[17] The R Foundation. 2009. The R Project for Statistical Computing. Internet Website, last accessed 09-20-16. (2009). https://www.r-project.org/.

[18] D. Toth. 2014. A Portable Cluster for Each Student. In *2014 IEEE International Parallel Distributed Processing Symposium Workshops*. 1130–1134. https://doi.org/10.1109/IPDPSW.2014.126

[19] Michael Wirth and Judi McCuaig. 2014. Making Programs With The Raspberry Pi. In *Proceedings of the Western Canadian Conference on Computing Education (WCCCE '14)*. Article 17, 5 pages. https://doi.org/10.1145/2597959.2597970