Using Inexpensive Microclusters and Accessible Materials for Cost-Effective Parallel and Distributed Computing Education

Joel C. Adams
Department of Computer Science
Calvin College
adams@calvin.edu

Suzanne J. Matthews
Department of EE & CS
United States Military Academy
suzanne.matthews@usma.edu

Elizabeth Shoop
Department of Mathematics, Statistics, & Computer Science
Macalester College
shoop@macalester.edu

David Toth
Department of Computer Science
Centre College
david.toth@centre.edu

James Wolfer
Department of Computer & Information Sciences
Indiana University, South Bend
wolfer@jwolfer.net

ABSTRACT

With parallel and distributed computing (PDC) now in the core CS curriculum, CS educators are building new pedagogical tools to teach their students about this cutting-edge area of computing. In this paper, we present an innovative approach we call *microclusters* – personal, portable Beowulf clusters – that provide students with hands-on PDC learning experiences. We present several different microclusters, each built using a different combination of single board computers (SBCs) as its compute nodes, including various ODROID models, Nvidia's Jetson TK1, Adapteva's Parallella, and the Raspberry Pi. We explore different ways that CS educators are using these systems in their teaching, and describe specific courses in which CS educators have used microclusters. Finally, we present an overview of sources of free PDC pedagogical materials that can be used with microclusters.

Keywords

Beowulf clusters, microcluster, Computer Science, Distributed, Education, Parallel, Teaching

1. INTRODUCTION

Prior to 2005, parallel and distributed computing (PDC) were elective topics in the computer science (CS) curriculum, optional, and rarely covered at the undergraduate level. While parallel programming libraries such as OpenMP [21] and MPI [24, 23] existed, the expense of parallel hardware and the difficulty in accessing high performance resources made these concepts difficult to teach. Over the last decade, this has changed largely as a result of two watershed events: the birth of multicore architecture and cloud computing. Both of these innovations have greatly decreased the cost and increased the accessibility of programming parallel architectures.

Intel and AMD released the first of many commercial multicore CPUs in 2006. This changed the foundation of commodity hardware, because each core in a multicore CPU could run a program simultaneously (i.e., in parallel). Dual-core CPUs were followed by quad-core, then hexa-core, then octa-core, and so on. Today Intel offers 22-core Xeon and 72-core Xeon Phi CPUs [26]. Traditional sequential programs use only one of a CPU's cores. Software must be intentionally designed and written as parallel software, if it is to fully leverage multicore CPUs. In addition, Nvidia released the CUDA [35] library in 2007, spurring the birth of GPGPU and manycore computing.

Amazon introduced its *Elastic Compute Cloud* (EC2) in 2006, allowing users of its service to run their applications on rented virtual machines (VMs) on Amazon's infrastructure. Users can vary the number of VMs they rent (the "elastic" aspect of Amazon's service), allowing them to write distributed applications that run at varying scales. This ability to access scalable computing facilities without having to maintain the hardware infrastructure has proven highly attractive to the business community. This popularity has necessitated a shift in program design: Since a traditional sequential program runs on just one machine, software must be designed and written as parallel and distributed software to run across multiple VMs, if it is to take advantage of the scalability of a cloud service like EC2.

Recognizing the implications of these events – that all CS students now need to learn about PDC – both the *IEEE TCPP Curriculum Recommendations* [40] and the *ACM/IEEE CS 2013 Curriculum Recommendations* [44] moved PDC topics into the core CS curriculum.

These changes raise many questions, including:

What hardware, software, and teaching resources should we use to teach students about PDC?

DOI: https://doi.org/10.22369/issn.2153-4136/8/3/1

Given the ubiquity of the multicore CPU, just about any computer has hardware on which students might learn about parallel computing, but not distributed computing. Distributed computing by definition involves a computation being distributed across multiple machines. Hardware platforms that might be used for teaching distributed computing include a network of workstations (NoW), a Beowulf cluster [42], or a cloud system such as Amazon's EC2. A computer lab can be configured as a NoW, and so can be used to teach PDC. However, when students launch distributed computations in a heavily-used lab, each student's computational performance will suffer as the computations compete for the lab's limited CPU resources. A cloud service such as Amazon's EC2 can be used to teach PDC, but the processing all takes place at a distance from the student, resulting in a loss of immediacy in the student's learning experience. High performance Beowulf clusters can be used, but they are relatively expensive to build and maintain.

The widespread availability of inexpensive single-board computers (SBCs) provides a different option for teaching PDC. Just as inexpensive integrated circuits made the *microcomputer* possible, inexpensive SBCs make it possible to build a **microcluster** – a personal, portable, Beowulf cluster – on which students learn about PDC.

In this paper, we present several different microclusters that CS educators have built for teaching their students about PDC, and explore the different ways these educators are using them in their classes. In the next section, we present some background information on microclusters. We examine the above-mentioned microclusters in detail in Section 3. In Section 4, we describe our experiences using these microclusters for teaching, research, and outreach. In Section 5, we discuss some of the software and teaching resources that can be used on these systems to help students learn about PDC. We conclude with some observations in Section 6.

2. BACKGROUND

Wiglaf, the first Beowulf cluster, was built by Donald Becker and Thomas Sterling at NASA in 1994 [42]. Wiglaf was a small cabinet containing sixteen motherboards with 80486 CPUs, which communicated through 10Mbps Ethernet. Unlike most of its successors, Wiglaf might be considered a microcluster by today's standards, since the entire cluster fit into one cabinet with its monitor and keyboard on top. In contrast, its immediate successor *Hrothgar* consisted of three shelves containing sixteen Pentium PCs connected with 100Mbps Ethernet, initiating the "lots of boxes on shelves" model used by many subsequent Beowulf clusters.

Almost immediately after Wiglaf's creation, people began building microclusters. In the rest of this section, we describe a few of these early examples of microclusters, as the context for the systems described in Section 3.

2.1 TTL_Papers

The term "microcluster" was coined by Hank Dietz and his students at Purdue University. They debuted their *TTL_Papers* microcluster at the 1994 Supercomputing conference [22]. It consisted of four nodes, each with a 25-MHz 80486 processor, that communicated through their parallel ports via a custom-built interconnect unit. The entire cluster weighed 30 pounds and fit within a 1 foot (30.48 cm) cube.

2.2 SETI Stacks

In 1999, the SETI@Home project [29] released a distributed computing client that, running on a personal computer, would: (1) download a batch of signals received by a radio telescope; (2) analyze those signals, looking amid the noise for regular patterns that might be evidence of intelligent communication; and then (3) report its findings back to the project.

In the hope of being the first to find evidence of extraterrestrial intelligence, many enthusiasts built clusters dedicated to running the SETI@Home client. At least 26 of these were microclusters that were dubbed "SETI Stacks", consisting of stacked motherboards with Pentium-era CPUs, communicating via original Ethernet. Many of these systems had colorful names like "*Crunchenstein Stack*", "*Stomp Monster*", "*SetiCruncher*", and so on [33].

2.3 Ultimate Linux Lunchbox

In the early 2000s, Ron Minnich and Mitch Williams built a series of microclusters at Sandia and Los Alamos National Labs. These may have been the first clusters built using SBCs, and their efforts culminated in 2005 with the *Ultimate Linux Lunchbox*, a lunchbox-sized microcluster consisting of sixteen Technologic Systems TS-7200 SBC nodes, connected using 100Mbps Ethernet [34]. These SBCs had StrongARM CPUs, a precursor to the ARM processors used by the microclusters we present in Section 3.

2.4 LittleFe

In 2005, Paul Grey, Tom Murphy, and Charlie Peck built *LittleFe*, a six-node cluster-in-a-suitcase small enough to take on the road to conferences, workshops, and high school outreach events [14]. The name is a pun, as "LittleFe" is the opposite of "Big Iron".

The initial LittleFe was a wooden frame housing six microATX motherboards with 1-GHz x86 CPUs, 100Mbps Ethernet, a shared hard drive, and a customized Linux distribution. Subsequent versions replaced the wooden frame with a machined metal frame, upgraded the network to Gigabit Ethernet, and upgraded the CPUs to multicore CPUs with integrated GPUs. Each version cost less than $2800, which includes a Pelican case rugged enough to let its LittleFe survive an airline's baggage-handling system.

In 2010, Charlie Peck obtained a grant from Intel to fund the first *LittleFe Buildout*, a conference event at which attendees (generally faculty) were given the raw materials for a LittleFe and spent a day assembling them into a working cluster, aided by Charlie and his students from Earlham College. At the end of the day, participants took the system they had built back to their home institutions, at no cost to them or their institutions. The beauty of this approach is that in building their own cluster, the participants acquired much of the knowledge needed to maintain it. Subsequent support from Intel, the National Science Foundation, and XSEDE has funded a total of 7 LittleFe Buildouts, resulting in 106 LittleFe clusters being built and used to teach parallel computing in colleges and universities throughout the U.S. Most of the systems in this paper were inspired by LittleFe.

2.5 Microwulf

After seeing the first version of LittleFe at a 2006 conference, Joel Adams and his student Tim Brom built a "personal, portable, Beowulf cluster" at Calvin College. Their goal was to generate as much computational performance as

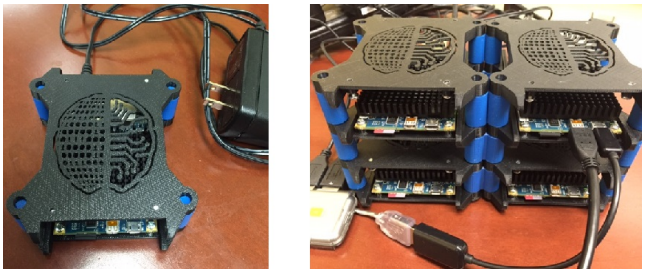

Figure 1: Parallella in case (left). 4-node Parallella Beowulf Cluster (right).

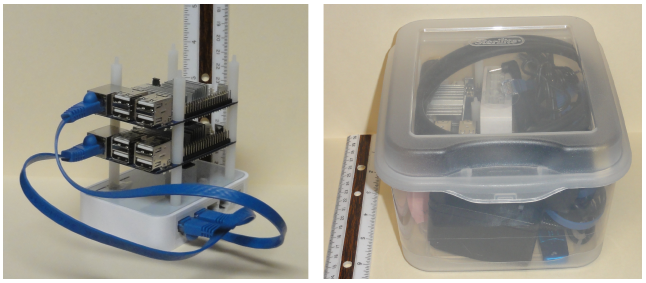

Figure 2: The 2-node HSC 6 Cluster (left). "Half-Shoebox" case (right).

possible within a $2500 budget. To accomplish this, Adams designed this cluster in keeping with "Amdahl's Other Law" [13]; balancing its CPU resources, main memory, and network bandwidth, to keep its computations from being CPU-, memory-, or network-bound.

The result was *Microwulf* [9], a microcluster of four nodes, each with a dual-core AMD Athlon64 CPU, 2GB of RAM, and two Gigabit Ethernet adaptors. For less than $2500 in early 2007, Microwulf achieved 26.25 Gflops on the HPLinpack supercomputing benchmark [38]. This gave Microwulf a price/performance ratio of $94/Gflop, making it the first Beowulf cluster to break the $100/Gflop barrier. By August 2007, the price of its components had dropped to $1256, improving its price/performance ratio to $48/Gflop.

Adams used Microwulf in his Fall 2007 *High Performance Computing* (CS 374) course at Calvin College, to provide students with hands-on experience developing and running MPI, OpenMP, and MPI+OpenMP programs.

3. MICROCLUSTERS FOR TEACHING PDC

In 2012, the Raspberry Pi Foundation released the Raspberry Pi, a $35 credit-card sized SBC. Its low cost catalyzed many CS educators and hobbyists to build Raspberry Pi microclusters. Notable early efforts with the Raspberry Pi include IridisPi [20], PiCloud [47], and efforts at Boise State [28] and FUB [8]. Despite its low cost, the Raspberry Pi's relatively weak CPU (700 Mhz ARM 11) and small onboard memory (512 MB) encouraged educators to turn to other SBCs for creating microclusters to teach their students about PDC[1]. In this section, we briefly give an overview of four recent parallel efforts at the authors' institutions.

3.1 StudentParallella

West Point was perhaps the first to incorporate the Parallella in the classroom [31]. Introduced by Adapteva via Kickstarter in 2013, the Parallella is a credit-card sized SBC with a dual-core Zynq ARM A9 CPU, 1 GB of RAM, gigabit ethernet, and a 16-core Epiphany co-processor. The board is extremely power efficient, requiring only a 5V/2A power supply, similar to a Raspberry Pi. The desktop edition of the board retails for $145.00, while the microserver edition retails for $99.00.

At West Point, Suzanne Matthews introduced undergraduates to the Parallella in a parallel computing elective course in the Spring 2015 semester. Each student had their own desktop edition of Parallella board. Epiphany programming was one of five programming modules covered in the course, along with C, Pthreads, OpenMP, and MPI. Matthews open-sourced [31, 30] her Parallella teaching materials, disk images, and setup guides in 2015, facilitating others to use the Parallella in the classroom. A custom case for the Parallella that enables it to be used on its own or configured into a *StudentParallella* microcluster was designed and open-sourced by Matthews and Blackmon [32]. In Figure 1, we show the Parallella in case assigned to each student, as well as a 4-node microcluster configuration.

Pros and Cons of Parallella-based Clusters.

Students were initially very excited by the Parallella. The small form factor and powerful 16-core co-processor were extremely motivating to students early in the course. However, their enthusiasm waned as the course went on, due to difficulties with the Epiphany Software Development Kit (eSDK) [31]. Students were largely dependent on Matthews' guide to the eSDK [30] to complete their co-processor assignment due to their lack of experience reading user manuals. The dual-core ARM CPU also limited students' speedup analyses. It is worth noting the Zynq SoC also has an FPGA which can be used in a parallel computing course, though Matthews did not do so.

Matthews concluded that while using a credit-card sized computer was very motivating for her students, it was unclear if the Parallella was the best option. Programming the co-processor is not significantly easier than programming a GPU, and some students at the end of the course expressed a desire to have learned CUDA instead [31]. While the Parallella system shows much promise, existing software packages and APIs could use more maturity before the Parallella is really ready for integration into an academic course.

3.2 Half Shoebox Clusters

David Toth of Centre College built the first *Half Shoebox Cluster* (HSC) in 2014 [45]. The project was motivated from a desire to build a low-cost, portable cluster for students to learn about PThreads, OpenMP, and MPI, necessitating a dual-core SBC.

The first HSC consisted of two dual-core Cubieboard2 [4] SBCs. After discovering the ODROID [3] SBC, six subsequent HSCs were built with various ODROID SBCs as they were released, beginning with the ODROID U3, and continuing with the C1, XU3-Lite, C1+, XU4, and most recently, the C2. The U3, C1, C1+, and C2 all have quad

[1]The release of the Raspberry Pi 2 in 2015 updated the board to a 900 MHz quad-core ARM Cortex A7 CPU, making it possible to use the board as a standalone unit for teaching PDC concepts

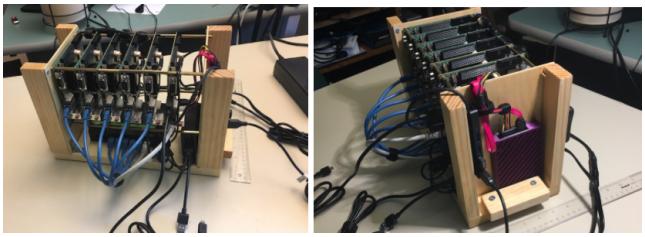

Figure 3: Rosie, a 6-node Nvidia Jetson TK1 cluster

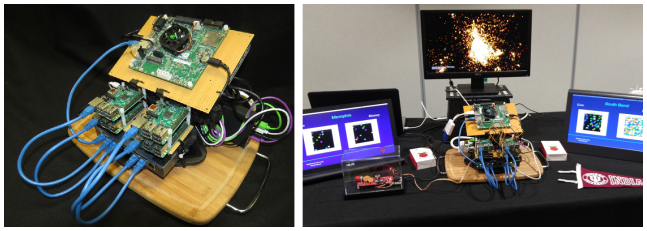

Figure 4: Cu-T-Pi

core CPUs, while the XU3-Lite and XU4 have 8-core ARM CPUs. The HSCs with the XU3-Lite and XU4 nodes also support OpenCL programming. The least expensive HSC costs about $150. Figure 2 shows the most recent HSC.

Toth maintains a web site [46] with disk images for the HSCs and a parts list so people can build their own HSCs and start using them immediately. HSC 6 is shown in Figure 2. The cluster is 2.75" wide by 4.25" long by 4.25" high. The box is 6.75" wide x 7.5" deep x 4.5" high. All other HSCs fit in the same box, but some HSC boards are slightly bigger than HSC 6's boards.

Pros and Cons of Half Shoebox Clusters.

Students loved having their own clusters. Students not enrolled in the course were very curious about the clusters and in some cases, envious of the students in the course. The ODROID nodes also have GPIO pins like a Raspberry Pi, allowing the nodes to be used in an Internet of Things course. The forum on the web site of the manufacturer of the ODROID nodes have a vast assortment of questions and contributions by users. In addition, questions posted to the forums get answered quickly, and there is a free ODROID magazine with lots of projects and ideas. In short, much like there is a large Raspberry Pi presence and community on the Internet, such a presence and community exist for the ODROID systems, too.

Some disadvantages of the HSCs are that only some of them (XU3-Lite and XU4 nodes) support GPU programming with OpenCL, and none support CUDA.

3.3 Rosie

In April 2014, Nvidia released a development board called the Jetson TK1, featuring their Tegra TK1 processor. The board consists of a quad-core ARM Cortex-A15 processor and an integrated Kepler GPU with 192 cores. A single Jetson board costs $192.00, or one dollar per core. This low cost, together with its capability of easily being flashed with an Ubuntu Linux distribution from Nvidia, makes these Jetson boards good candidates for building an inexpensive cluster with a great deal of computing power; this six node cluster provides (6 x 4) = 24 CPU cores plus (6 x 192) = 1152 CUDA-capable GPU cores.

Libby Shoop and a Macalester College undergraduate connected six of these boards via a Gigabit Ethernet switch to create a six-node microcluster named *Rosie* (see Figure 3). Rosie features an NFS-mounted disk that all nodes share, and each node is CUDA-capable, thanks to its Kepler GPU and Nvidia Linux distribution. Because each board has multicore and GPU capability, this cluster has been used to teach heterogeneous computing techniques with MPI and either OpenMP or CUDA.

Pros and Cons of Jetson-based Clusters.

Rosie is placed on a cart and rolled into class to demonstrate how this small cluster models larger supercomputers. Students gather around the hardware and are excited to see its pieces. Instructors can explain sources of overhead by pointing out the distance over the network that data must travel when using MPI for distributed programs. This visceral interaction with the hardware is valuable.

This value has a price: at $192, the Jetson is the most expensive of the SBCs. Setting up such a cluster also requires some effort. NVidia provides software for flashing the operating system onto these boards (including CUDA and MPI), but you need another Ubuntu Linux machine to do so. Afterward, the cluster network, network file system (NFS), passwordless ssh, and accounts must be set up for student users of the system to use MPI. For six nodes, this process took an undergraduate student under Shoop's supervision a few days. To make this easier, complete instructions, including a parts list, are available on [6].

3.4 Cu-T-Pi

Cu-T-Pi, shown in Figure 4, is a microcluster named for some of the technologies it aggregates (CUDA, Nvidia Tk-1, and Raspberry Pi). Created by James Wolfer [48] at Indiana University, South Bend (IUSB), this cluster is designed to be highly visible, portable, and have hardware and software architecture consistent with contemporary heterogeneous systems. The cluster consists of four Model B+ Raspberry Pi worker nodes and one Nvidia Jetson Tk-1 head node, connected through a gigabit Ethernet switch, all mounted in a terraced arrangement for instructional visibility. The resulting system provides eight ARM cores and 192 CUDA-capable GPU cores, supporting demonstrations and development using the OpenMP, MPI, and CUDA platforms.

In addition to its use for introducing parallel concepts, Wolfer has used CU-T-Pi to demonstrate benchmarking concepts in his *Parallel Computing* course. By using the HPL benchmark adapted for the Raspberry Pi [38, 15], we can observe and quantify the impact of asymmetric communication speeds. Details can be found in [48].

Pros and Cons of CU-T-Pi.

Positive aspects of this heterogeneous system include modeling current HPC architecture, supporting heterogeneous software development. Asymmetric speeds between component nodes offer unique opportunities for MPI benchmarking. Each node includes GPIO capabilities allowing interfacing with custom hardware.

Limitations include the relatively slow speed of the Raspberry Pi, and incompatible GPIO voltage/current requirements between the Raspberry Pi and TK-1 computers.

4. TEACHING, RESEARCH, & OUTREACH EXPERIENCES

Microclusters are useful for introducing PDC concepts inside and outside of the classroom. In this section, we describe the various strategies we have used to introduce students to parallel computing using microclusters. We describe best practice strategies when assigning each student their own cluster compared to having a single cluster for the entire classroom. We also discuss our experiences in parallel computing electives and core computer science courses. Additionally, we discuss our efforts using microclusters to engage undergraduates in research and outreach events.

4.1 Strategies for Introducing Microclusters

The primary way in which the authors have used microclusters is to introduce PDC concepts to undergraduates. CS Education research advocates the use of "hands-on experiential learning" [19] for teaching PDC concepts, while recommending a high degree of interactivity in the classroom [19, 17]. In all cases, microclusters had a definite "cool" factor that encouraged student engagement and enthusiasm for PDC concepts. There were two primary ways students engaged with microclusters in the classroom. In the first, students shared access to a common microcluster. In the second, each student had their own microcluster. Each strategy has benefits and draw-backs.

4.1.1 One Cluster per Class

The "One Cluster per Class" can take a variety of forms ranging from units designed for student access to classroom demonstration machines. Rosie was designed for both roles, being used at Macalester College in an upper-level *Parallel Computing* course first as a demonstration machine, and then for individual student access for in-class activities, homework problems, and course projects. Students are given accounts on the head node, which is connected to Macalester's network and given an internal IP address. They can then log in remotely for class activities and work on graded assignments. Cu-T-Pi was designed as a classroom demonstration machine; to provide a portable, preconfigured, "ready-to-rock" device that can move from class to class and provide an introduction to parallel computing.

Both Cu-Ti-Pi and Rosie enable instructors to expose their students to OpenMP, MPI, and CUDA programming. CUDA concepts can be taught via demonstration code provided by Nvidia CUDA SDK, or through open-source teaching materials such as those available through CSinParallel or CDER (see Section 5). For example, Macalester uses the modules available through CSinParallel to teach students CUDA, OpenMP, and MPI programming.

Pros and Cons of "One Cluster Per Class".

Students have a very positive reaction to seeing the cluster hardware, and its physical nature generates student curiosity and motivates discussion of system components. For example, students can observe communication "overhead" by monitoring the Ethernet lights on both the SBCs and the network switch. Rosie's NFS-shared hard disk allows for discussion of Network File Systems, and lets students see the storage device. By contrast, traditional HPC or cloud systems hide such details from students, which can hinder their ability to fully understand such mechanisms.

The portability of the "One Cluster Per Class" units encourages their use in classes across the spectrum, from first-year to graduate levels, inviting comments like *"...brought in a mini super-computer to demonstrate the power of parallel programming"* in a freshman class.

The gains in portability and instructional flexibility are balanced by its limitations. Specifically, except when connected online, these systems are not directly available to students, limiting first-hand experience–only the students involved in building Rosie, for example, gained system-level experience. When used outside class, demand is high at certain times and students compete for cluster cycles, perhaps limiting this approach to smaller institutions.

4.1.2 One Cluster per Student

West Point and Centre College both adopted the "one cluster per student" approach. The decreasing cost of SBC hardware enables each student to purchase SBCs of their own, or sign a cluster out as individual lab equipment. There are some immediate benefits to this approach, including eliminating resource contention between students and maximizing the individual learning experience. A microSD card pre-loaded with necessary course materials can be shared with students at the beginning of the course. Updates may be downloaded by students from a class website. Students may connect to their clusters using SSH or a more familiar desktop environment.

Both Centre College and West Point use this approach to teach students about the MPI, OpenMP, and Pthreads libraries. West Point students also learned about accelerator programming using the Parallella's Epiphany co-processor. Centre College maximizes the one cluster per student approach by assigning each student their own two-node cluster. In contrast, West Point assigns each student a single Parallella; as an in-class lab, students network their boards together to form a cluster [31].

Pros and Cons of "One Cluster Per Student".

Students at both institutions were very excited to have their own clusters. At West Point, students opted to primarily connect to their clusters via SSH, owing to the time and number of peripherals (e.g. monitor, keyboard, mouse) required for access. At Centre College, students typically brought their clusters to the dedicated computer science lab and hooked them up to large monitors, keyboards, and mice there. At both places, students were unable to connect their clusters to the Internet, due to institutional IT policies. This is consistent with experience of others experimenting with SBCs at their own institutions [43]. Instead, students transferred files between their laptops and their clusters via SCP/SFTP or USB flash drives.

A significant benefit of the one cluster per student model is that it removes the system administration work from the faculty member - a node crashing only affects a student (who can reboot it themself). Additionally, when disk images are provided, a faculty member new to teaching PDC can be up and running quickly, without having to learn about configuring the hardware. The one cluster per student model also enables students to operate in isolation from each other, thus allowing them to measure the performance of their sequential and parallel versions of their programs without other students' tests influencing their results.

4.2 Courses Using Microclusters

A natural place to use microclusters in the classroom is in a *Parallel Computing* elective course. Such courses introduce students to PDC concepts in a self-contained package. The drawback to electives is that they cannot guarantee that every student gets exposure to PDC concepts. The *ACM CS 2013* [44] report introduced the PDC knowledge area and recommended that 15 core tier-one and tier-two hours be included in the core undergraduate computer science curriculum. Likewise, the *NSF/IEEE TCPP Curriculum Initiative on Parallel and Distributed Computing* (NSF/IEEE TCPP) [40] recommends that PDC concepts be introduced in required computer science "systems" courses. The authors have used their microclusters to introduce students to PDC in required courses such as *Computer Organization* and *Computer Architecture*, and to delve more deeply in *Parallel Computing* electives.

4.2.1 Uses in Parallel Computing Electives

At West Point, the students used the Parallella to complete homework assignments. Epiphany co-processor programming was covered along with OpenMP, MPI, and Pthreads. For each course assignment, students wrote a C program and a parallel equivalent in one or two other libraries and performed timing studies to quantify their performance gains. Students were also required to complete a final project and summarize their results in a research paper. In addition, they were assigned 10 topics papers in which the summarized and reflected upon CACM articles covering PDC topics.

At Centre College, students used their HSC systems to learn about Pthreads, OpenMP, and MPI. Students chose a problem and over the course of the semester, they created sequential, OpenMP, MPI, and MPI+OpenMP programs to solve their problem. The programs were then tested and their performance was compared. The students then created posters about their projects, wrote a paper summarizing the project and the results, and gave presentations about their programs and what they had learned.

At Macalester College, students in the elective course used Rosie for an MPI homework assignment and for heterogeneous computing activities with MPI+CUDA and MPI+ OpenMP. A Monte Carlo solution for modeling a pandemic was used as a class activity. As the final project for the course, students chose a project for a particular architecture; those who chose a distributed computing solution used Rosie as their platform to explore and test for scalability.

Cu-Ti-Pi has also supported classroom demonstrations in the senior/graduate level *Parallel Computing* course at IUSB.

4.2.2 Uses in "Systems" Courses

At West Point, the Parallella architecture is briefly discussed as part of a larger lesson on hardware acceleration in the *Computer Organization* course. Due to the importance of the x86 architecture in West Point's CS curriculum, the custom Parallella architecture was eschewed in favor of traditional x86 multicore servers for the hardware base of the course.

At IUSB, Cu-Ti-Pi has supported the *Operating Systems* class by providing remote access to an interfaced Geiger counter, as the basis for a random number serving file system project [27]. It has also been used for classroom demonstrations in the *Computer Organization* course.

Rosie has served as a demonstration platform in the *Computer Systems* course at Macalester College, where it is wheeled into class and and used to demonstrate examples that illustrate scalability and speedup in a distributed system. The students work with Pthreads and OpenMP on shared-memory multicore machines first, so Rosie is shown as a contrasting architecture for distributed parallel computing. Plans in progress at Macalester are to include hands-on activities with new Raspberry Pi- based clusters by shortening some units earlier in the course to free up time for distributed programming at the end of the course.

4.3 Research & Outreach Experiences

Microclusters have been used at each institution to inspire students and faculty about parallel computing. For example, Cu-Ti-Pi has been used for demonstrations in a general education computer literacy course.

At West Point, a student project used a cluster of Raspberry Pi B+s to simulate a remotely operated "smart" mortar system [41] in 2014. The same cluster was used by another West Point student in 2015 to study password cracking for their final project in their parallel computing course. In 2017, students ascertained a Raspberry Pi 2 cluster's ability to detect power grid anomalies [18].

At Macalester College, the building of Rosie itself was a very beneficial summer research experience for undergraduate students. The experience of students has been passed on so that current students have not only rebuilt Rosie but also built new clusters based on newer Jetson boards and on the relatively new Raspberry Pi 3 boards.

The microclusters mentioned in this paper have also been demonstrated at the SIGCSE Technical Symposium [11, 12].

5. TEACHING MATERIALS

As described previously, traditional parallel libraries such as Pthreads, OpenMP, and MPI can all be taught using microclusters. For example, the Pacheco parallel programming textbook [37] and his earlier book for MPI programming [36] have been used at Calvin, Macalester, and West Point as a main text for their parallel computing electives.

Given the advances in computer architecture in the last ten years, a chief complaint is the scarcity of textbooks for teaching modern parallel computing concepts to undergraduates. In this section, we describe two NSF funded initiatives, CSinParallel and CDER, that aim to alleviate this shortfall. CSinParallel offers a series of parallel "modules" that can be used in the context of a course in place of a textbook. CDER has an initiative to build a modern textbook. We also discuss other sources of PDC educational materials. All the authors have used various combinations of these materials to supplement their coverage of PDC concepts in their courses.

5.1 CSinParallel

The NSF-funded *CSinParallel* project [16] maintains a site, CSinParallel.org, which contains a comprehensive set of course modules for PDC education. Each module is designed to be used over a short period of time in any of several courses, depending on the curriculum and course structure at an instructor's institution. Modules exist for various levels of experience, ranging from novices in introductory courses to experienced seniors in advanced electives. Modules also exist for various types of hardware and software. As

an example, for MPI, there are modules for distributed computing fundamentals, heterogeneous computing, a pandemic modeling exemplar using a Monte Carlo approach, and others. Modules can be used to briefly introduce students to parallel computing concepts, or provide them with in-depth exposure to programming examples and practices. CSinParallel modules have been used successfully at Macalester, St. Olaf, West Point, and other places, with students rating the material highly.

The patternlets module at CSinParallel.org is especially useful for teaching with microclusters. Patternlets are small code examples that demonstrate particular PDC topics, using well-known, tried-and-true parallel design patterns [10]. There are currently 21 MPI patternlets available for teaching distributed-memory parallel topics and 16 OpenMP patternlets for teaching shared-memory parallel topics, as well as patternlets for POSIX pthread multithreading and heterogeneous (MPI+OpenMP) computing.

5.2 CDER

The NSF-funded *Center for Parallel and Distributed Computing Curriculum Development and Educational Resources* (CDER) was created with mission of developing core PDC curricula that can be adopted at a wide array of institutions across industry and academia. CDER maintains a collection of instructor-produced PDC materials [25], and has synthesized several introductory topics into a book [39]. A second volume of the book will target students in upper-level computer science courses.

In addition to collecting PDC educational materials, CDER has also sponsored an Early Adopter Program to provide seed funding for faculty to integrate PDC concepts into their courses, and the EduPar workshop series to provide a venue for dissemination of these Early Adopters' results. The Early Adopter Program has funded over 100 institutions to date. CDER also provides access to parallel hardware platforms for teaching PDC topics.

5.3 Other Sources of Materials

There are many high quality PDC teaching materials available from other sources including HPCUniversity.org [2], the Computational Science Education Reference Desk [1], and Lawrence Livermore National Laboratory (LLNL) [7].

HPCUniversity.org has tutorials on programming languages such as C, C++, and Python, as well as UNIX and shell programming to help students acquire necessary background skills. There are also tutorials there on using OpenMP, MPI, MPI+OpenMP, the Intel Xeon Phi, GPGPU, and CUDA.

CSERD has a number of modules that show how to implement solutions to problems with different parallel computing libraries. These include classic examples like Conway's Game of Life and calculating the area under a curve to interdisciplinary applications of parallel computing such as biofilms and solving the party problem in mathematics.

LLNL is another source of useful tutorials, focused on Pthreads, MPI, and OpenMP.

6. CONCLUSIONS

We have described how microclusters can be used to introduce undergraduate students to PDC. The microclusters at Calvin College, Centre College, IUSB, Macalester College, and West Point have excited students about parallel computing concepts and applications. Microclusters embody the "hands-on" learning of PDC clusters advocated by CS educators, are a fun way to introduce students of all levels to parallel and distributed computing, and provide small-scale models of larger HPC systems.

We also described different strategies for engaging students with microclusters, including different courses in which students can use microclusters, and microcluster-based research and outreach experiences. We noted free teaching materials that can be used in conjunction with microclusters. Some of our SBCs cost less than many course textbooks; with the availability of free, high-quality teaching materials, SBCs and SBC clusters can be individually purchased by students in lieu of a textbook.

While we highlighted the one cluster per student model and the one cluster per class model, Macalester College has recently begun moving towards an intermediate "several clusters per class" model. In this approach, students work in groups, with each group using their own microcluster to explore hands-on CSinParallel.org activities. These clusters, built in summer 2017 using Raspberry Pi 3 boards, will be used in the 2017-2018 academic year in the sophomore-level systems course. This model should provide better scalability than the "one cluster per class" approach.

The price/performance ratios of SBC architectures continue to improve, as new processors and boards are released. For example, Nvidia introduced the Jetson TX1 in 2015, based on the Tegra TX1 processor, containing an updated ARM CPU and 256 GPU cores, and now offers an updated Jetson TX2 developer kit board for an educational discount price of $299.00. Macalester has recently built a new cluster using 4 TX2 boards. More affordable multicore SBCs, such as the 4-core Raspberry Pi 3 board ($35.00 without a power adapter) are also now excellent candidates for affordable small clusters. The future of Adapteva and its Parallella board is in some doubt [5]; however, it is the only SBC that supports teaching about FPGAs and co-proccessors.

For each SBC, Table 1 summarizes the software libraries that can be covered, the hardware features, and the cost per node. Additional items that may be needed include network router and cables, power supply, monitor, and keyboard. For demonstrations and student research projects, more expensive and powerful clusters based on NVidia Jetson hardware may be preferable. The low-cost multicore ODROID and Raspberry Pi 3 hardware are excellent ways to get hardware in students' hands and and avoid competition for a shared cluster. For these SBCs, downloadable images allow students to get started with these systems quickly.

The choice of hardware and teaching model depends on the course learning outcomes at a given institution. To illustrate, all of the microclusters presented in this paper support the coverage of learning outcomes related to OpenMP and MPI; to cover CUDA outcomes, a Jetson SBC is needed; to cover outcomes related to co-processors or FPGAs, a Parallela is needed; and so on. Likewise, the "one cluster per student" model scales well and is applicable at institutions of all sizes; the "one cluster per course" model may be limited to small institutions and/or demonstration machines.

Our collective experiences strongly suggest that microclusters are an inexpensive, accessible, cost-effective, and motivating way to introduce parallel computing concepts to undergraduates, in keeping with current CS curriculum guidelines. We hope that our positive experiences will inspire others to use microclusters to teach PDC concepts.

Features	Parallella	OROID XU4	NVidia Jetson TK1	Raspberry Pi 3
OpenMP	Y	Y	Y	Y
MPI	Y	Y	Y	Y
GPGPU	N	OpenCL	CUDA	N
Co-Processor	Y	N	N	N
FPGA	Y	N	N	N
Cores: CPU + GPU/Co-Processor	2 + 16	8 + 6	4 + 192	4 + 0
Cost (per node)	$99.00	$59.00	$192.00	$35.00

Table 1: Overview of node architectures and features.

7. ACKNOWLEDGEMENTS

The opinions 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.

The authors also with to thank the anonymous reviewers, whose comments were very helpful in finalizing this paper.

8. REFERENCES

[1] CSERD: Home. Internet Website, last accessed 05-31-16, 1994. http://www.shodor.org/cserd/.

[2] HPC university: Home. Internet Website, last accessed 05-31-16, 1994. http://hpcuniversity.org/.

[3] ODROID | hardkernel. Internet Website, last accessed 06-01-16, 2013. http://www.hardkernel.com/main/main.php.

[4] Cubietech - open source hardware. Internet Website, last accessed 05-31-16, February 2014. http://www.cubietech.com/.

[5] Adapteva status update. Internet Website, last accessed, August 2017. http://www.adapteva.com/andreas-blog/adapteva-status/.

[6] How to Build a Jetson TK1 Cluster. Internet Website, last accessed, August 2017. http://selkie.macalester.edu/csinparallel/modules/RosieCluster/build/html/.

[7] Lawrence livermore national labs high performance computing training. Internet Website, last accessed, August 2017. https://hpc.llnl.gov/training/tutorials.

[8] P. Abrahamsson, S. Helmer, N. Phaphoom, L. Nicolodi, N. Preda, L. Miori, M. Angriman, J. Rikkila, X. Wang, K. Hamily, et al. Affordable and energy-efficient cloud computing clusters: The bolzano raspberry pi cloud cluster experiment. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, volume 2, pages 170–175. IEEE, 2013.

[9] J. Adams and T. Brom. Microwulf: A beowulf cluster for every desk. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, pages 121–125, 2008.

[10] J. C. Adams. Patternlets: A teaching tool for introducing students to parallel design patterns. *Journal of Parallel and Distributed Computing*, 105:31–41, 2017.

[11] J. C. Adams, J. Caswell, S. J. Matthews, C. Peck, E. Shoop, and D. Toth. Budget beowulfs: A showcase of inexpensive clusters for teaching PDC. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 344–345. ACM, 2015.

[12] J. C. Adams, J. Caswell, S. J. Matthews, C. Peck, E. Shoop, D. Toth, and J. Wolfer. The micro-cluster showcase: 7 inexpensive beowulf clusters for teaching PDC. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pages 82–83. ACM, 2016.

[13] G. Amdahl. Storage and I/O parameters and systems potential. In *Proceedings of the IEEE International Computer Group Conference (Memories, Terminals, and Peripherals)*, pages 371–372, 1970.

[14] I. Babic, A. Weeden, M. Ludin, S. Thompson, C. Peck, K. Muterspaw, A. F. Gibbon, J. Houchins, and T. Murphy. LittleFe and BCCD as a successful on-ramp to hpc. In *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment (XSEDE'14)*, 2014.

[15] Baldzya. Bramble HPL (high-performance linpack benchmark) results. Internet Website, last accessed 2016-06-03, 2014. https://www.raspberrypi.org/forums/viewtopic.php?t=33186&p=612733.

[16] R. Brown and E. Shoop. CSinParallel and synergy for rapid incremental addition of PDC into CS curricula. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 1329–1334. IEEE, 2012.

[17] R. Brown, E. Shoop, J. Adams, C. Clifton, M. Gardner, M. Haupt, and P. Hinsbeeck. Strategies for preparing computer science students for the multicore world. In *Proceedings of the 2010 ITiCSE Working Group Reports*, pages 97–115. ACM, 2010.

[18] K. Candelario, C. Booth, A. St. Leger, and S. J. Matthews. Investigating a raspberry pi cluster for detecting anomalies in the smart grid. In *Proceedings of the 2017 IEEE Undergraduate Research and Technology Conference*, 2017.

[19] R. A. Chesebrough and I. Turner. Parallel computing: at the interface of high school and industry. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE 2010)*, pages 280–284. ACM, 2010.

[20] S. J. Cox, J. T. Cox, R. P. Boardman, S. J. Johnston, M. Scott, and N. S. OâĂŹbrien. Iridis-pi: a low-cost, compact demonstration cluster. *Cluster Computing*, 17(2):349–358, 2014.

[21] L. Dagum and R. Enon. OpenMP: an industry standard API for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55, 1998.

[22] H. Dietz. TTL papers microcluster. Internet Website, last accessed 2016-05-25, 1995. http://aggregate.org/EXHIBITS/sc95.html.

[23] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, et al. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 97–104. Springer, 2004.

[24] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22(6):789–828, 1996.

[25] N.-T. C. Initiative. Courseware management. Internet Website, last accessed 2016-06-08, 2014. http://grid.cs.gsu.edu/ tcpp/curriculum/?q=courseware_management.

[26] Intel. Server processors. Internet Website, last accessed 2016-05-25, 2016. http://ark.intel.com.

[27] W. J. Keeler and J. Wolfer. A raspberry pi cluster and geiger counter supporting random number acquisition in the cs operating systems class. In *Proceedings of the International Conference on Remote Engineering and Virtual Instrumentation (REV)*, pages 344–345, February 2016.

[28] J. Kiepert. RPiCLUSTER: Creating a raspberry pi-based beowulf cluster. Technical report, Boise State University, 2013.

[29] E. Korpela and D. Wertheimer. SETI@Home. Internet Website, last accessed 2016-05-25, 2016. http://setiathome.ssl.berkeley.edu.

[30] S. J. Matthews. Technical musings. Internet Website, last accessed 06-01-16, May 2015. http://suzannejmatthews.github.io.

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

[32] S. J. Matthews and W. Blackmon. Parallella case and cluster files. Internet Website, last accessed 05-31-16, June 2015. http://www.thingiverse.com/thing:892684.

[33] L. Mester. SETI stack and farm systems. Internet Website, last accessed 2016-05-25, 1999. https://web.archive.org/web/20130918044235/ http://bhs.broo.k12.wv.us/ homepage/staff/seti/farms.htm.

[34] R. Minnich. The ultimate linux lunchbox. *Linux Journal*, (11):3–7, 139. November, 2005.

[35] NVIDIA. *CUDA Compute Unified Device Architecture : programming guide*. NVIDIA Corporation, 1st edition, 6 2007.

[36] P. Pacheco. *Parallel Programming with MPI*. Morgan Kaufmann, San Francisco, Calif, 1st edition, Oct. 1996.

[37] P. Pacheco. *An Introduction to Parallel Programming*. Morgan Kaufmann, Amsterdam : Boston, 1st edition, Jan. 2011.

[38] A. Petitel, R. Whaley, J. Dongarra, and A. Cleary. Hpl - a portable implementation of the high-performance linpack benchmark for distributed-memory computers. Internet Website, last accessed 2016-05-25, 2008. http://www.netlib.org/benchmark/hpl/.

[39] S. K. Prasad, A. Gupta, A. Rosenberg, A. Sussman, and C. Weems. *Topics in Parallel and Distributed Computing: Introducing Concurrency in Undergraduate Courses*. Morgan Kaufmann, 1st edition, Aug 2015.

[40] S. K. Prasad, K. Kant, Y. Robert, A. Chtchelkanova, S. Das, A. La Salle, R. Le Blanc, A. Rosenberg, F. Dehne, M. Lumsdaine, et al. NSF/IEEE-TCPP curriculum initiative on parallel and distributed computing-core topics for undergraduates. In *42nd ACM Technical Symposium on Computer Science Education (SIGCSE 2011)*, 2011.

[41] Z. J. Ramirez, R. W. Blaine, and S. J. Matthews. Augmenting the remotely operated automated mortar system with message passing. *CrossTalk, The Journal of Defense Software Engineering*, 28(6), November 2015.

[42] T. Sterling, D. J. Becker, D. Savarese, J. E. Dorband, U. A. Ranawake, and C. V. Packer. Beowulf: A parallel workstation for scientific computation. In *Proceedings of the 24th International Conference on Parallel Processing*, pages 11–14. CRC Press, 1995.

[43] D. Tarnoff. Integrating the arm-based raspberry pi into an architecture course. *Journal of Computing Sciences in Colleges*, 30(5):67–73, 2015.

[44] The ACM/IEEE Joint Task Force on Computing Curricula. Computer science curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science. December 2013.

[45] D. Toth. A portable cluster for each student. In *Proceedings of the Fourth NSF/TCPP Workshop on Parallel and Distributed Computing Education (EduPar-14)*, May 2014.

[46] D. Toth. Dave toth's portable compute cluster page. Internet Website, last accessed 06-01-16, February 2016. http://web.centre.edu/david.toth/portablecluster/index.html.

[47] F. P. Tso, D. R. White, S. Jouet, J. Singer, and D. P. Pezaros. The glasgow raspberry pi cloud: a scale model for cloud computing infrastructures. In *Distributed Computing Systems Workshops (ICDCSW), 2013 IEEE 33rd International Conference on*, pages 108–112. IEEE, 2013.

[48] J. Wolfer. A heterogeneous supercomputer model for high-performance parallel computing pedagogy. In *Proceedings of the IEEE Global Engineering Education Conference (EDUCON/ITEP)*, pages 785–791, March 2015.