TEACHING WITH PARALLELLA: A FIRST LOOK IN AN UNDERGRADUATE PARALLEL COMPUTING COURSE*

Suzanne J. Matthews
Department of Electrical Engineering & Computer Science
United States Military Academy
West Point, NY 10996
845-938-5577
suzanne.matthews@usma.edu

ABSTRACT

This paper describes our experience integrating the Parallella, an energy efficient single board computer (SBC) with 18 cores, into an undergraduate parallel computing course. The board's small form-factor, high number of cores and relative cheapness makes it a very attractive option for introducing students to parallel computing. We describe and reflect on our experiences using the Parallella board, and offer novel educational materials that will assist others to incorporate the Parallella into future computing courses.

INTRODUCTION

The ubiquity of multi-core architectures in recent years makes teaching parallel and distributed computing (PDC) concepts to undergraduates more imperative than ever. Despite the fact that most laptops, tablets, and smart-phones contain multi-core chips, computer science undergraduates are still largely exposed to serial languages. The need for more extensive parallel computing education is underscored with the release of the ACM/IEEE Computer Science Curricula 2013 (CS2013) [1], which recommends that 15 hours of PDC concepts be included in the typical undergraduate computer science program. Other efforts, such as the IEEE Curriculum Initiative on Parallel and Distributed Computing (NSF/IEEE TCPP) [19] and the CSinParallel community [7], are seeking to

define the most important PDC topics to include in a computer science curriculum, and provide relevant modules and educational resources for undergraduate computing courses.

Even before the release of CS2013, many educators were looking to establish best practices for teaching PDC concepts in the classroom. Most researchers advocate using modules and other techniques that promote "hands on experiential learning" [9]. Modules can target specific courses in the CS curriculum [12], or be concentrated into a breadth-first exposure to parallel topics in a computing elective [21]. In all cases, researchers espouse the importance of demonstrating application speedup [8,9,12,21], presenting engaging modules [8,9,21], and recommend a high level of interactivity in the classroom [8, 9].

One way to promote "hands on" learning in a parallel computing course is to give each student their own multiprocessor. The shrinking nature of transistors and increasing cheapness of commodity hardware has enabled self-contained parallel architectures to enter the classroom. The earliest examples include MicroWulf [5] and LittleFe [18]. Introduced in 2008, MicroWulf provides 26.25 GFlops of performance for $2,470, making it the first Beowulf cluster to break the $100/Gflop barrier [5]. LittleFe is a six-node Beowulf cluster, with each node consisting of a multi-core processor. Modern versions of the cluster support GPGPUs [18]. The $3,000 cluster takes students around ten hours to assemble, and has been used successfully and extensively for numerous workshops [18]. Recent years have seen the rise of increasingly small single board computer (SBC) architectures, including the Raspberry Pi [20], Odroid [14], and NVidia Jetson [13] boards. Efforts from the past year have used the Raspberry Pi single board computer successfully in an undergraduate architecture course [22], and for teaching undergraduates Java programming [6].

In this paper, we discuss the integration of the Parallella board (an 18-core, credit-card sized computer) in an upper-level, undergraduate parallel computing course. We chose the Parallella for its relative cheapness, small form-factor, and marketed ease of programmability. We present an overview of the Parallella and its unique architectural features, discuss how we integrate the Parallella into our parallel computing course, share novel and valuable Parallella educational materials, and submit a collection of observations and suggestions for future use.

ABOUT THE PARALLELLA

Introduced via a KickStarter campaign in 2013, the Parallella board was advertised as a platform enabling "parallel computing for everyone". Each board has a dual-core 667MHz ARM processor, 1 GB of RAM, Gigabit Ethernet, and a 1GHz 16core Epiphany co-processor, with 32KB of local memory available to each Epiphany core (e-core), and total peak performance of 32 GFlops [2]. At $99.00, the level of performance is hard to beat, and relatively inexpensive for individual use in the classroom. A more expensive 18-core Desktop version ($145.00) enables users to connect the Parallella to a monitor. The Parallella uses a microSD card to house the operating system and user files, making

it easy to swap out the image for others containing specialized learning modules. Lastly, the board has a definite "cool" factor. Upon seeing the Parallella, many students in our department were eager to learn more.

The Epiphany co-processor is of particular interest. Co-processors and other energy-efficient hardware accelerators like GPUs are gaining increased popularity in the supercomputing world, as power becomes the dominant limiting factor to building faster machines. Most famously, the Tianhe 2 [23] supercomputer has remained at the top of the Top500 fastest supercomputer list, largely due to its incorporation of the Intel Xeon Phi co-processor, which vastly increases its computational capabilities without drastically increasing power requirements. Coprocessors like the Epiphany and Intel Xeon Phi have a multiple instruction multiple data (MIMD) architecture, similar to general-purpose CPUs. The MIMD architecture makes the co-processor more amenable to workflows that are difficult to implement on GPUs, which classically have single instruction multiple data (SIMD) architecture. Due to the co-processor's architectural similarities to the CPU, it is also argued that it is easier to program than a GPU. These reasons (along with an advertised slick developer environment for Epiphany) led us to adopt the board for the course.

COURSE OVERVIEW

Our parallel computing course was offered as an elective available to primarily seniors. The course consisted of six modules designed to give students a breadth-first exposure to parallel computing, similar to the implementation [21] at Sonoma College. Unlike the course at Sonoma, we covered POSIX threads (Pthreads), OpenMP, Epiphany, and MPI as part of our four parallel modules targeting shared memory, co-processor, and distributed system architectures.

For each module, students completed a programming project where they were provided with (or asked to write) a serial solution for a given application in C. Students then parallelized the application using one to two parallel libraries, and conducted a performance benchmarking study. Projects were supplemented with topic papers covering assigned articles that discussed recent advancements in parallel computing. Given the intensity of the assignments, the course had no exams. The course was well received, with students enjoying the combination of programming projects and topic papers.

Purchased Materials

Since the course was a pilot elective targeting a small population of students, our department purchased the necessary Parallella boards and accessories. Table 1 outlines the cost per student for the purchased components. The cost of peripherals (e.g. monitor, keyboard and mouse) is not included, as students either owned or had classroom access to the required hardware.

Table 1: Cost breakdown per student for Parallella board and required accessories.

Item	Cost
16-core Parallella Desktop Computer w/power supply and heatsink	$149.00
IoGear 4-port powered USB hub	$ 22.93
8GB MicroSD card w/Adapter	$ 6.60
MicroUSB to USB (F) Cable	$ 5.99
MicroHDMI to HDMI (F) Cable	$ 2.96
Crossover Cable	$ 3.99
Total	**$191.47**

We chose to purchase the more expensive 16-core Parallella Desktop computer because we assumed our students would prefer a more familiar desktoplike environment. Unlike the $99.00 Microserver, students can choose to work in the desktop environment or SSH into their boards remotely. We note that this type of access comes at non-trivial cost. The first four accessories (roughly $40.00) are required to enable students to connect the Parallella board to a HDMI monitor, and USB keyboard and mouse. The crossover cable is required for both the Parallella Desktop and Microserver editions. As a result, the "true" cost of the 16-core Parallella Desktop is closer to $190, roughly $10.60 per core.

Parallella Specific Mini-modules and Materials

The students were initially required to use their Parallella boards for their first four programming projects, with the last (MPI-specific) project to be completed on a campus high performance computing cluster. We created additional educational materials and "mini-modules" to aid our students in learning about the Parallella and Epiphany architecture. Each mini-module was designed to fit in a single lecture hour (55 minutes). We describe the mini-modules and materials below:

- Parallella Setup mini-module: This mini-module was designed to allow students to get their Parallella boards up and running within an hour of opening the box. Users on the Parallella forums commonly report difficulties in getting their Parallella boards to work, partially due to the lack of clarity in the provided documentation. We created custom images based on those available from the Parallella website, and provide instructions for creating an SD card using a Windows laptop. We also included trouble-shooting guidelines. The mini-module is freely available at: http://suzannejmatthews.github.io/2015/05/29/setting-up-your-parallella/

- Custom Parallella Case: Also included in the above mini-module are instructions for assembling a custom-designed, 3-D printed case for the Parallella. While the Kickstarter campaign gave original backers a plastic case for the Parallella, the case is unavailable for purchase. Our case design allows for passive cooling, and enables multiple boards to be connected either horizontally or vertically. The STL files for the case are available at: http://www.thingiverse.com/thing:892684.
- Connecting to Parallella via SSH mini-module: Our students often used SSH to complete assignments and transfer files between their Parallella boards and laptops. This mini-module describes how to use Putty and the PSFTP clients to remotely access and transfer files between a Parallella board and a Windows system. These techniques also prepared our students to connect to the campus HPC cluster, as cluster access from Windows requires the same set of software. This mini-module is freely available at: http://suzannejmatthews.github.io/2015/05/30/setting-up-ssh-forparallella/.
- Creating a Parallella Cluster mini-module: As we made the transition between the Parallella and MPI modules, we had a mini-module where students networked their Parallella boards together to a Gigabit Ethernet switch. The plan was to run the John the Ripper (JtR) password cracker in parallel by combining a JtR program written for the Epiphany architecture with MPI. Pre-flashed master node images with MPI and the necessary files were provided for this exercise. Our mini-module was adapted from the Southampton [10] tutorial for creating a Raspberry Pi cluster. The minimodule is available at: http://suzannejmatthews.github.io/2015/06/15/parallella-cluster/
- Lecture materials for the Epiphany Module: To facilitate our students' ability to complete the Epiphany programming project, we created lesson materials that discuss the Epiphany architecture and the most salient aspects of the Epiphany manuals. Our lessons include walk–throughs of basic Epiphany programs, and simple examples demonstrating how to deploy applications to the Epiphany architecture. Our students reported these lessons were critical for their understanding, since they had significant trouble understanding the Epiphany manuals on their own. The lecture materials are available at: http://suzannejmatthews.github.io/2015/05/31/epiphany/.

OBSERVATIONS

When first introduced to the Parallella, students thought the boards were very "cool", and were excited at the prospect of programming them. They really enjoyed the Parallella setup mini-module held at the beginning of the course and were very motivated to learn more. Unfortunately, their enthusiasm for the boards began to wane in the weeks that followed, largely due to issues they had connecting to the board and the quality of provided Epiphany documentation and examples.

Students Preferred SSH to Desktop Environment

The Parallella Desktop edition requires a HDMI monitor to display the desktop environment. While the classroom monitors had native HDMI support, the monitors our students personally owned did not. While this was a source of initial frustration, the students quickly adapted to connecting to the boards via SSH.

Surprisingly, students soon preferred to connect to the boards via SSH inclass instead of using the provided monitors, keyboards and mice. When queried for the reason, students said that they disliked the several minutes of setup time required at the beginning of class to connect the board to the provided peripherals, and at the time required at the end of class to disconnect everything again. This was especially inconvenient for students who had a long distance to travel between classes. The majority of the students stopped using the desktop interface six weeks into the semester, instead preferring to use SSH to connect to the boards.

Network Policies and (Lack of) Internet Connectivity a Major Obstacle

Our university network policies provided additional challenges. The Parallella boards were prohibited from being connected to the campus network. To get around this, we used static IP addresses and crossover cables to allow students to SSH into the boards from their laptops. While this was set up through an alternative configuration of their network adapter in Windows, it interfered with their ability to access the Internet while connected to their Parallella boards. This problem, unfortunately, was not resolved by the end of the semester.

Our university's wireless network bandwidth also prevented us from finishing the Parallella cluster mini-module in the allotted hour block. While the 2GB compressed cluster image requires only ten minutes to download over Ethernet, it took over an hour when students attempted to simultaneously download the file over the classroom's wireless network. The instructor instead demoed the JtR application while students fruitlessly waited for their downloads to complete.

While issues with Internet connectivity and campus network policy were a major source of frustration, we note that many of these observed issues are not unique to the Parallella board. Recent efforts [22] to incorporate the Raspberry Pi into the classroom have encountered similar difficulties with university IT policies and connecting students to the device. As new strategies are developed to integrate SBCs into classrooms, these issues will need to be taken into consideration.

Epiphany Documentation and Examples Need Improvement

There is a significant learning curve required for undergraduates attempting to program the Epiphany architecture. While the Parallella website advertises a slick Eclipse-based integrated development environment, it is currently non-functional, and too large to be run on the Parallella board. Undeterred, we used the Epiphany and Parallella

examples [15, 16] provided on GitHub and the manuals for the Epiphany Architecture [3] and Software Development Kit (SDK) [4] to figure out how to write programs for the Epiphany architecture.

The two Epiphany manuals together constitute 300 pages of reading. Most students attempted to skim the manuals, but were largely unsuccessful at finding needed information. Our students' relative inexperience at reading large software manuals is certainly a major reason for this. That said, the examples presented in the manuals are mainly designed for reference use, and contain an insufficient amount of detail to enable students to start programming the architecture quickly. Students instead relied heavily on the instructor-produced Epiphany slides to learn the SDK and complete their projects.

Lastly, the provided examples for executing simple programs in parallel on the Epiphany chip require work to be easily understandable by undergraduates. The programming model for the Epiphany chip is very similar to that of GPUs. The programmer is required to write a "host" and a "device" program. The "host" program is responsible for transferring data to and from the Epiphany chip (the device), and deploying the "device" program on the chip. The "device" program contains code that is run on each Epiphany core (e-core), and tends to be much simpler than the "host" program. The Epiphany repository [15] on GitHub surprisingly lacks many representative small examples. As of this time of writing, we have identified only four: eprime, matmul, dotproduct, and hello-world. While the Parallella repository [16] has several mature demo applications that use the Epiphany architecture, they lack sufficient documentation and tend to be too large for students new to the architecture to easily comprehend.

Due to the difficulties in programming the Epiphany chip, students wrote most of their parallel programs on the dual-core ARM chip. This included all the programs they wrote in Pthreads and OpenMP. We opened up remote access to a Linux lab containing multi-core processors to enable students to perform more meaningful benchmarking studies for their Pthreads and OpenMP projects. Students reported the Epiphany project (a prime generator requiring code similar to the dotproduct example) to be the hardest they completed in the course. At the end of the course, a few students expressed a wish that we had covered the CUDA architecture instead.

CONCLUSIONS

The Parallella board is certainly a modern marvel of computer architecture, and has a lot a promise to positively impact parallel computing education. The fully developed Epiphany applications available through GitHub are very impressive and really show off the power of the Epiphany chip. However, our experience underscores that the Parallella board is still in early development.

In particular, the documentation and provided examples require significant revision to be accessible to undergraduates. By itself, the dual-core ARM chip is very limiting in the context of a parallel computing course. Competitors to the Parallella, such as the

Raspberry Pi 2, Odroid and Jetson boards all have quad-core ARM chips. To remain competitive, we suggest that the Parallella developers consider upgrading the ARM chip to a quad-core alternative.

We also note that the Epiphany chip currently does not appear to be significantly easier to program than a CUDA chip, as both architectures require a "host" and "device" program and knowledge of custom SDKs. Epiphany developers need to make a real argument on why students should use the Epiphany architecture over CUDA at this juncture, especially given the popularity of CUDA and the rich amount of available literature for programming the CUDA architecture. The biggest question we face going forward is whether to swap out the Epiphany module with a CUDA module in future iterations. The NVidia Jetson, consisting of a quadcore ARM chip, 192 CUDA cores and priced at $192.00, represents a fierce competitor for the Parallella.

In retrospect, our incorporation of the Parallella board in our undergraduate parallel computing course was premature. We note however that there is a lot of exciting ongoing work being done to improve the programmability of the Epiphany chip, most recently the March 2015 release of the OMPi b2 OpenMP compiler [11]. Other ongoing efforts include adding support [17] for languages such as MPI, OpenCL, Erlang and BASIC for the Epiphany architecture. We anticipate our educational materials and experiences to be valuable additions to the Parallella and computer science education communities, and will enable others to explore incorporating the Parallella in their own courses.

While the Parallella board is still clearly going through growing pains experienced by all new architectures, we believe SBCs like it have a clear place in future computing courses. The increased availability of free materials related to parallel computing may make purchasing future textbooks largely unnecessary, enabling students to instead purchase a single board computer as part of required course materials. Strategies certainly need to be developed to resolve networking issues, especially as it relates to university IT policies. However, we believe that personalized parallel computing is a trend that should be encouraged, and that the Parallella board is a promising candidate for future use.

DISCLAIMER

The opinions in this paper are those of the author and do not necessarily reflect the opinions of the U.S. Military Academy, or the U.S. Army.

REFERENCES

[1] ACM/IEEE-CS Joint Task Force on Computing Curricula. 2013. Computer Science Curricula 2013. ACM Press and IEEE Computer Society Press.

[2] Adapteva Inc. E16G301 16-core microprocessor datasheet, 2013, http://www.adapteva.com/docs/e16g301_datasheet.pdf, retrieved February 2015.

[3] Adapteva, Inc. Epiphany Architecture Reference, 2013, http://www.adapteva.com/docs/epiphany_arch_ref.pdf, retrieved February 2015.

[4] Adapteva, Inc. Epiphany SDK Reference, 2013, http://adapteva.com/docs/epiphany_sdk_ref.pdf, retrieved February 2015.

[5] Adams J.C, Brom T.H, Microwulf: a beowulf cluster for every desk, *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, 121-125, 2008.

[6] Altadmri A., Brown N.C.C, Kölling M., Using BlueJ to Code Java on the Raspberry Pi (Demonstration Session), *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 2015.

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

[8] Brown R., Shoop E., Adams J., Clifton C., Gardner M., Haupt M., Hinsbeeck P, Strategies for preparing computer science students for the multicore world, *Proceedings of the 2010 ITiCSE working group reports*, 97-115, 2010.

[9] Chesebrough R.A., Turner I., Parallel computing: at the interface of high school and industry, *Proceedings of the 41st ACM technical symposium on Computer science education*, 280-284, 2010.

[10] Cox, S., Steps to make Raspberry Pi Supercomputer, 2013, https://www.southampton.ac.uk/~sjc/raspberrypi/pi_supercomputer_southampton_web.pdf, retrieved October 2014.

[11] Dimakopoulos, V.V., Hadjidoukas P.E., et. al., New version (b2) of OMPi for Parallella, 2015. http://paragroup.cs.uoi.gr/wpsite/news-posts/new-version-b2-ofompi-for-parallella/, retrieved May 2015.

[12] Ernst D.J., Stevenson D.E, Concurrent CS: preparing students for a multicore world. *SIGCSE Bulletin*, 40, (3), 230-234, 2008.

[13] Nvidia. Nvidia Jetson TK1 embedded development kit: the world's first embedded supercomputer. http://www.nvidia.com/object/jetson-tk1-embeddeddev-kit.html, retrieved May 2015.

[14] Odroid, Odroid, http://www.hardkernel.com/main/main.php, 2014, retrieved May 2015.

[15] Olafsson, A., Jeppsson O. GitHub: Simple examples showing how to program the Epiphany. https://github.com/adapteva/epiphany-examples, retrieved May 2015.

[16] Olafsson, A., Back, A., Jeppson, O. GitHub: Community created Parallella examples. https://github.com/parallella/parallella-examples, retrieved May 2015.

[17] Olafsson, A., How the @#$% do I program the Parallella?, 2015, https://www.parallella.org/2015/05/25/how-the-do-i-program-the-parallella/, retrieved May 2015.

[18] Peck C., LittleFe: parallel and distributed education on the move, *Journal of Computing Sciences in Colleges*, 26, (1), 16-22. 2010.

[19] Prasad, S.K., Chtchelkanova A.Y, Das S.K., Dehne F., Gouda M.G., Gupta A., Jaja J. *et al.*, NSF/IEEE-TCPP curriculum initiative on parallel and distributed computing: core topics for undergraduates, SIGCSE, 11, 617-618, 2011.

[20] Raspberry Pi. Teach, Learn, and Make with the Raspberry Pi. https://www.raspberrypi.org/, retrieved May 2015.

[21] Rivoire S., A breadth-first course in multicore and manycore programming, *Proceedings of the 41st ACM technical symposium on Computer science education*, 214-218, 2010.

[22] Tarnoff D., Integrating the arm-based Raspberry Pi into an architecture course, *Journal of Computing Sciences in Colleges*, 30, (5), 67-73, 2015.

[23] Top 500. Tianhe-2 (Milkyway 2):No. 1 system since June 2013. http://www.top500.org/featured/systems/tianhe-2/, retrieved May 2015.