Designing an Independent Study to Create HPC Learning Experiences for Undergraduates

Sandino Vargas-Pérez
Department of Computer Science
Kalamazoo College
Kalamazoo, MI, USA
Email: sandino.vargasperez@kzoo.edu
https://orcid.org/0000-0002-5778-6142

***Abstract*—This paper aims to present a multi-tiered approach to designing learning experiences in HPC for undergraduate students that significantly reinforce comprehension of CS topics while working with new concepts in parallel and distributed computing. The paper will detail the experience of students working in the design, construction, and testing of a computing cluster including budgeting, hardware purchase and setup, software installation and configuration, interconnection networks, communication, benchmarking, and running parallel code using MPI and OpenMP. The case study of building a relatively low-cost, small-scale computing cluster that can be used as a template for CS senior projects or independent studies, also yielded an opportunity to involve students in the creation of teaching tools for parallel computing at many levels of the CS curriculum.**

***Index Terms*—Parallel Computing, HPC, Undergraduate Education, PDC Teaching Tools, OpenMP, MPI, Cluster**

I. Introduction

High-performance computing (HPC) and parallel programming are receiving increasing attention in the Computer Science (CS) curriculum as new technologies allow faster computations and algorithm acceleration. As a result, there is a faculty movement to include the study of these areas and a necessity to develop streamlined pedagogical approaches to teaching these topics.

For institutions like Kalamazoo College, a relatively small liberal arts college with an academic year divided into three terms of ten weeks each, introducing new courses presents a challenge. Students take a limited number of classes per term, and the department's offerings are constrained by course sequencing and faculty availability. During the height of the COVID-19 pandemic, further limitations were imposed on the number of in-person contact hours we could have with our students. Therefore, it was crucial to develop online learning resources, like a self-guided study curriculum, with meaningful exercises and accessible reading material, and appropriate ways to evaluate learning outcomes.

In the process of thinking about how to bring opportunities for students to experience HPC during their CS education, we designed an independent study focused on the idea of building low-cost, small-scale computing clusters that can be used as a teaching tool for parallel computing, as well as a template for a comprehensive CS senior project.

The objective of the independent study was to design a project with a student-led learning and research-centered approach, allowing them to develop their agency to solve problems on their own while obtaining knowledge and skills in parallel programming and design of HPC systems. The goals established for the students participating in the project were to learn:

- How to build a cluster of computing nodes for HPC, from designing and budgeting to installing and configuring hardware and software components.
- How hardware interacts with current and voltage, and the importance of understanding standards and specifications.
- How to use the HPC cluster to understand concepts such as efficiency and speedup for parallel algorithms, time complexity, cost of communication, data and task parallelism, and synchronization.
- How to use MPI and OpenMP to create programs using the C/C++ programming language, and how to use the libraries that make parallel computations possible.
- How to incorporate concepts acquired in previous CS courses, such as data structures and algorithms, computer organization and OS, computer networks, and others, to aid with the setup of interconnected networks, server administration, and data management.

Another objective of the independent study was empowering the student to influence the educational experience of their classmates and other learners. Therefore, after the project was concluded, the student was tasked to create a step-by-step guide to build and configure the cluster so that students and professors could replicate the process. This guide was intended to provide the blueprints to create a pedagogical tool for teaching HPC.

The learning outcomes of this independent study were measured using the "Taxonomy of Significant Learning" [1], a less common approach to the more known Bloom's taxonomy. The motivation was to gauge not only foundational knowledge of CS topics but also the application and integration of such knowledge, and how it intersects and affects the learners and those around them.

In the following sections, we will present the student's experience working with the independent study and show the

benchmarks and test results performed using the system. We will also suggest ways to use the cluster not only in HPC or parallel programming courses but also to aid in the teaching of concepts in computing networking, hardware configuration, operating systems, and other topics across the CS curriculum.

II. Designing the Independent Study

The independent study was designed with a senior student in mind but can be offered to any student with demonstrably proficient CS skills. The time frame to complete the study was set to be approximately 15 weeks distributed as follows:

1) Three weeks were dedicated to research and learning about HPC clusters, budgeting, and gathering materials.
2) Seven weeks were dedicated to building and configuring a traditional Bewolf cluster using RPi computers.
3) Five weeks were dedicated to running tests and benchmarks on the cluster and learning HPC concepts, models, and parallel programming paradigms.

It is important to note that the learning of some HPC concepts was interwoven into the different phases of the project. For example, learning about the types of interconnection networks used in real-life supercomputing clusters enhanced their comprehension of the required steps and helped inform their decision-making process.

The student was provided with general instructions and electronic resources to start the project and instructed to find complementary material to help fill the gaps and solve specific issues encountered during the implementation. In addition, informal evaluations such as quizzes and a list of HPC concepts and vocabulary were also given to the student as self-evaluation tools to enhance the lexicon in parallel computing. These evaluations were meant to assess their understanding of:

- Parallel computing concepts such as scalability, speedup, and efficiency, as well as the effects of Amdahl's law, cost of communication, and ways to identify computation or communication-bound problems.
- Types of interconnection networks used in HPC and their elements.
- Concepts and vocabulary used in HPC and parallel programming libraries and APIs (MPI, OpenMP, and CUDA, respectively).

The student met with the instructor every two weeks to discuss progress, challenges, possible approaches, and to ask questions about the project. To help make these meetings more substantial, the student was also asked to keep a journal in which they reflected on their weekly learning and working experience.

A. Deliverables

The independent study required two final products to be delivered. First, a functional HPC cluster capable of parallel computing and configured in a way that allows for portability, scalability, and that can be used as a tool in HPC-related courses. Second, a "How To Build a Low-Cost HPC Cluster" guide with step-by-step instructions on creating an HPC cluster to allow other learners and instructors to reproduce the entire process.

In the particular case of the student that participated in this independent study, the experience served as the basis for writing their Senior Integrated Project (SIP), a senior thesis that is part of the graduation requirements at Kalamazoo College.

III. Building a Computing Cluster for HPC

The student was tasked to build a Beowulf cluster composed of Raspberry Pi (RPi) single-board computers. The RPi (nodes) will be arranged in groups of four to form towers (the budget is the limiting factor regarding how many nodes the cluster can have). Other basic instructions to start the research and planning phase were the following:

- Build the cluster to have four towers with four nodes each. One node will be designated as the head node (with the rest of the RPi computers in that tower being workers). The remaining nodes will be workers.
- Equip the cluster with enough memory for local storage and computation.
- Provide input/output devices for the head node, such as a keyboard, mouse, monitor, and an external hard drive (to function as a shared storage unit).
- Include network interconnection devices to allow communication between towers and RPi computers.
- Design the cluster to be compact enough for easy transportation and setup.

A. Research, Design, Budgeting

The student started by designing the cluster layout after careful research [2], [3], [4], [5], creating a diagram similar to Fig. 1 with details specifying hardware configuration and interconnections. From this blueprint, they created a list of parts and began the budget and purchasing phase. During this process, finding the right components presented challenges and new learning opportunities for the student; specifications about current and voltage needs for the cluster, Ethernet cable specifications, and other cable and cord standards. The student stated: "*I chose to use only one-foot Ethernet and micro USB cables to minimize excess wiring. Finding a proper USB hub was difficult because each Raspberry Pi requires the maximum amps a USB can support (2.4 A) [. . .] I chose the Vogek 6 port USB charging stations because they were the cheapest hub that could meet the Pi's power requirements.*"

The student carefully prepared the list of components for the cluster and shared it with the independent study supervisor for further discussion. It included the parts needed, quantity, unit prices, a brief description of the part and its function within the cluster, and a URL to facilitate online purchasing. Table I shows a redacted list of the components purchased.

The budget was administered effectively, taking care of leaving enough room for unforeseen circumstances while maximizing the computational power of each node and the cluster as a whole. Regarding the purchasing process, the

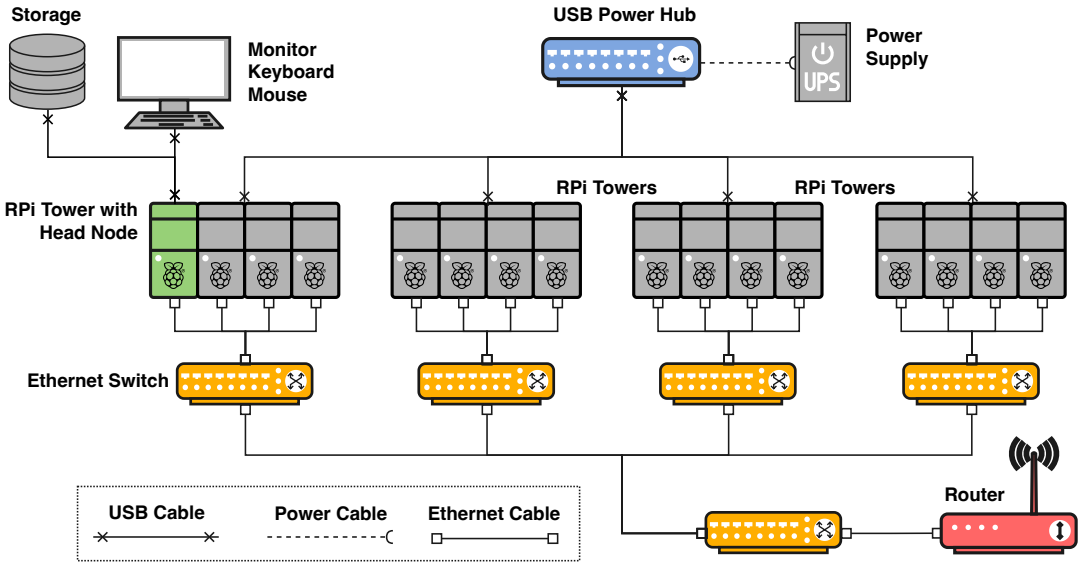

Fig. 1. Raspberry Pi Cluster Diagram.

student mentioned: "*while still under budget, I left enough money in contingency to replace something that might break during assembly. My goal was to maximize the ability of the cluster while staying within the budget allocated.*" Indeed, as designed by the student, the cluster contains 64 processing units because each RPi purchased has a quad-core processor. Although, only 60 of them will be used as working units per the student's design. They noted that: "*good parallel computing practice suggests you should not use your head node to do calculations.*"

TABLE I
LIST OF COMPONENTS FOR THE CLUSTER

Parts	Quantity	Description
Raspberry Pi 3 Model B+	16	1.4GHz 64-bit quad-core processor, 1GB LPDDR2 SDRAM
Micro SD Cards	16	64 GB w/adapter
Ethernet cable	20 (1ft), 5(3ft)	Cat 5e (1 ft), Cat 6 (3ft)
USB to Micro USB cable	18	
USB Charger Hub	4	6 ports each
Ethernet Switch	5	5 ports each
Keyboard and Mouse	1	
Monitor	1	
RPi Stackable Case	4	4 Layers, acrylic rack w/heatsinks
USB to microSD adapter	1	
Power strip	1	
USB to USB C cable	1	
USB to 2.1mm barrel jack	6	

B. Building and Configuring the Cluster

As proof of concept, the student started by building a small-scale version of the cluster with only four RPi, giving them a manageable set of variables and a more efficient troubleshooting experience. They began by installing three heat sinks on each RPi to increase the microchip's surface area and help dissipate heat. Two larger heat sinks were placed in the CPU and the RAM chip, and another smaller one on the Ethernet and USB controller. The student then proceeded to attach all the RPi to the acrylic plates with screws and connected the plates to form a tower.

After setting up the first tower, the student focused on installing and configuring the essential software needed by the RPi computers to operate. First, the latest version of Raspberry Pi OS[1] (formerly Raspbian) was selected:

- **Raspberry Pi OS with desktop**: To be installed in the head node, taking advantage of its desktop interface to manage and interact with the cluster using a GUI.
- **Raspberry Pi OS Lite**: To be installed in the worker nodes, saving space because these nodes will only be used as computational units.

Each particular OS was downloaded and installed in their respective microSD cards in a process known as flashing. A blank file named `ssh` was created and placed on each flashed microSD card to enable remote login on the nodes via secure shell (SSH). The cards were then inserted into their respective nodes, and the nodes were powered. Further configuration was necessary for the head node: changing the RPi's hostnames, passwords, language, and timezone and expanding the file system capacity. The student chose the naming convention for the nodes following the rule `node[tower##][node#]`, e.g., `node034` will represent the fourth node on the third tower. The tower number is padded with a `0` to allow the addition of towers in the future.

[1]https://www.raspberrypi.com/software/operating-systems/, Raspberry Pi OS Images

The student confirmed that the configuration of the first tower was in order and then proceeded to replicate it on the remaining nodes and towers.

C. Software Installation and Network Configuration

The next phase required the student to research the suite of software needed to administer and run an HPC cluster and to understand the necessary network interconnection configurations [6], [7], [8] to create a system capable of efficient communication and work-load distribution among processing units, nodes, and towers.

Through extensive reading, the student determined the list of applications that needed to be installed:

- MPICH, a free and open-source implementation of MPI that enables a distributed-memory model of parallel programming.
- Slurm, a free and open-source job scheduler to manage workload in the cluster. Slurm will allow multiple users connected to the cluster to submit jobs without interfering with other computations.
- Ntpdate, used to synchronize date and time across the cluster.

The student proceeded to enable passwordless authentication between the head node and worker nodes by generating and placing the public ssh key of the head node into the corresponding authorized directory on the worker nodes.

The network configuration phase was next. The student set static IP addresses on each node, stating: "*I made each node's IP correlate to their hostname, e.g.,* `node012`*'s static IP is [set to]* `10.0.0.12`*. I did not start at 0 because* `10.0.0.0` *is often reserved for the router.*" To guarantee that Slurm would work correctly, the student set up a network file system (NFS) using the external hard drive. The student configured all the nodes to automatically mount the NFS on startup by editing each node's `fstab` configuration file. Some issues were encountered during this procedure, prompting the student to: "*write a startup shell script that manually mounts every node to be used as a backup.*"

While installing and configuring Slurm, the student experienced advanced computer organization and networking concepts. Using Slurm's default configuration file as a base, the student was able to:

- Add worker nodes IP addresses and hostnames to the head node's host file.
- Configure file's permissions
- Add nodes to the cluster using the node's hostname, IP address, number of cores, and its state.
- Create partitions to determine which nodes belong to which towers.
- Allow Slurm to use the worker node's random-access memory (RAM) and the NFS in the external hard drive.
- Create a munge key, a type of authorization service to allow remote operations.

The student tested these configurations to validate that the system was working correctly and troubleshoot any issues as they appeared.

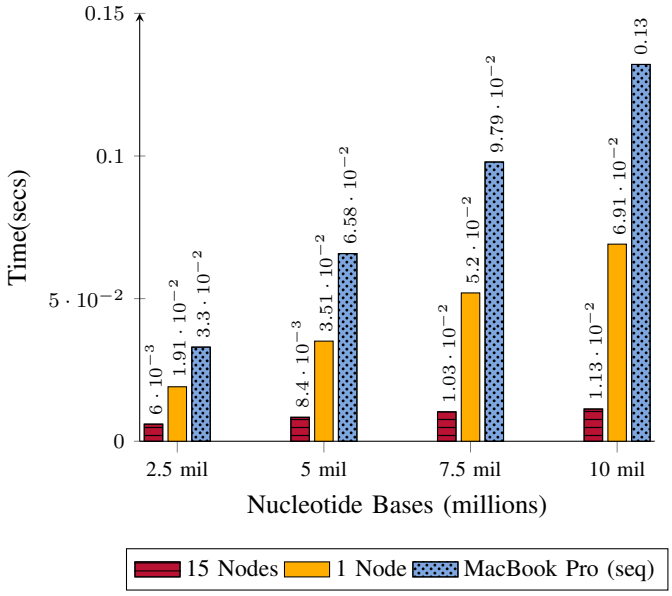

Fig. 2. Time (in seconds) to count nucleotide frequency for different DNA sequence lengths, using the HPC cluster and a MacBook Pro.

D. Running Tests and Programming in Parallel

When the construction and proper configuration of the HPC cluster were concluded, the student started performing the testing phase of the cluster's capabilities to validate how much faster parallel processing was compared to sequential processing.

The student was provided with a parallel program coded in C++ using MPI to perform the experiments and test the cluster. First, the student needed to understand the parallel code to compile and execute it in the cluster. The program counted the frequency of nitrogen bases in a DNA sequence file and used reduction MPI functions to obtain the final counts. Next, the student made modifications and improvements to the parallel code to suit the experiments better. Then, they created a sequential version of the program for comparison purposes.

The program was compiled and executed in the cluster, confirming that all nodes were configured and working as intended. To perform the tests, the student first ran experiments using all 15 working nodes (60 processing units) and compared it against one node (four processing units) and a 2015 MacBook Pro laptop (2.7 GHz Intel Core i5 processor, 8 GB 1.86 GHz, DDR3 memory, running macOS Mojave). Several runs were performed using DNA sequence lengths of 2.5, 5, 7.5, and 10 million nucleotides, and the average clock time was taken. The results can be seen in Fig. 2.

Based on the results of these experiments and other data, the student observed: "*I found that the 15 nodes cluster was substantially faster at counting nucleotide frequencies. My data shows that for a 2.5 million nucleotide sequence, the cluster was 220% faster than one RPi and 450% faster than my MacBook Pro. As the data gets bigger, the difference between the speedup increases in favor of the cluster.*"

The student further analyzed scalability issues in the cluster by learning about Amdahl's law, speedup, and efficiency. They started by performing theoretical calculations on the effect that increasing the number of processing units has on the computational speedup of the program. Using data obtained in previous experiments, they plotted a graph representing the program's theoretical speedup. Then, they proceeded to run and time several experiments keeping the input size fixed (5 million nucleotides) while increasing the processing units (4, 8, 12, 16, 20, 40, and 60). While comparing the theoretical and experimental results, the student noted: "*the results from the test were as expected and approximately followed the speedups predicted by Amdahl's Law. [. . .] it shows significant time improvements from 1 RPi to 2 RPis (4 cores to 8 cores) and significantly less improvement from 5 RPis to 10 RPis (20 cores to 40 cores).*"

After completing these tests and experiments, the student continued to learn and practice concepts in parallel computing: how to measure efficiency in parallel algorithms, time complexity, speedups, and the cost of inter-node communication. The material provided to the student for this final phase [9], [10], allowed them to gain a basic understanding of parallel algorithms and their applications, particularly how to choose the appropriate parallel programming paradigm, as well as how to parallelize a given problem, including understanding the different methodologies of parallel models, how the hardware impacts algorithmic performance, and how to ensure algorithmic efficiency when scaling a problem. However, the complete parallelization of sequential programs done solely by the student was not part of the scope of the independent study. Nonetheless, the student demonstrated an understanding of how these programs work and an ability to improve them by modifying their components.

E. Student Experience with Parallel Computing

At the conclusion of the independent study, the student was tasked to reflect on the experience of working on their self-guided, research-centered project. The student suggested: "*Taking parallel computing as an independent study drastically increased my computer science knowledge. Not only helping me round out my parallel computing knowledge, but it allowed me to explore other areas of CS.*"

The student also reflected on their experience working through the material of a parallel computing course: "*I gained much knowledge about the workings of parallel computing machines, but I had minimal experience using them. The study drastically helped me increase my knowledge. Before this course, I knew some keywords and had a rough idea of what they were, but I didn't understand how they worked. Now I feel very confident with my ability to use and understand OpenMP and have gained a lot of understanding and ability using MPI.*"

Finally, the student said: "*I have also learned a lot about the capabilities and limitations of parallel computing regarding algorithms. I think independent studies are great ways for students to be allowed to gain a level of knowledge about a subject that is either more relevant to them or overall get a deeper understanding than in a normal classroom setting.*"

Besides the actual construction, configuration, and testing of the HPC Cluster, the student produced a "How To Build a Low-Cost HPC Cluster" guide with step-by-step instructions on creating an HPC cluster using four RPi computers (instructions can be extrapolated to add more devices). The guide presents the list of components needed for the cluster, helpful pictures, instructions for assembling, software needed, commands to execute proper configuration, and more. The guide can be found following this link https://github.com/svargasperez/RPiClusterGuide

IV. Evaluating Learning Outcomes

L. Dee Fink's taxonomy [1] proposes six kinds of learning to evaluate learning outcomes: foundational knowledge, application, integration, human dimension, caring, and learning how to learn.

Understanding concepts in HPC demonstrated foundational knowledge. With readings, quizzes, and the HPC vocabulary, the student acquired a lexicon that facilitated the bi-weekly meetings and showed the student's ability to ask better questions and find solutions to the project's challenges. Foundational knowledge was also shown in the skills obtained while building and configuring the cluster, as evidenced by the students' reflections and by fulfilling the requirements for the HPC cluster.

When evaluating the integration portion, the student showed an ability to connect ideas from previous CS courses and other skills obtained while taking courses in different departments. For example, when the student worked with Amdahl's law, they said: "*[. . .] it is similar to the law of diminishing returns in economics because like the law of diminishing returns no matter how many processing units you add it will never continue decreasing the execution time.*"

To evaluate the human dimension of these learning outcomes, the student commented on time management, handling anxiety in the face of new challenges, and the care and enthusiasm put into preparing the step-by-step guide to benefit their fellow classmates. These points also intersect the caring portion of the learning outcomes and the constant motivation that the student showed to finish the independent study.

This independent study's student-led, research-centered approach developed the student's ability to solve issues and figure out better solutions to build the HPC cluster, fulfilling the learning how to learn portion of the taxonomy.

V. Conclusion

In this paper, we presented the experience of designing an independent study on HPC and parallel computing from the perspective of a student working with a self-guided, research-centered approach under the supervision of a faculty mentor. This project complemented the knowledge that a learner gathers during their academic year and provided positive reinforcement to newer areas of computer science that, due to curriculum and time constraints, are not currently possible

to obtain in many higher education institutions. The student reflections demonstrated a very effective achievement of the objectives and goals established for the study.

Using a less conventional taxonomy to evaluate learning outcomes proved to be an exciting mechanism worth considering when developing pedagogical approaches to student-led education, especially considering the disruptions caused by the COVID-19 pandemic. This approach significantly changed student motivation to learn, as evidenced by the complete commitment to the project and the valuable resources created for other students and educators.

The resulting HPC cluster and the "How To Build a Low-Cost HPC Cluster" guide can serve as powerful pedagogical tools for teaching parallel programming and enriching CS courses across the curriculum. Furthermore, the computer science department at Kalamazoo College offers students the opportunity to take the elective course "COMP-481: Applied Parallel Algorithms" [11] as an independent study, and the HPC cluster could be used as the computing hardware instead of an HPC supercomputer.

ACKNOWLEDGMENT

This paper was based partly on the work of the student Nathan H. Silverman, who engaged with the project with the highest academic standards and produced invaluable resources for Kalamazoo College and the Computer Science Department.

REFERENCES

[1] L. D. Fink, *Creating significant learning experiences: An integrated approach to designing college courses*. John Wiley & Sons, 2013.

[2] CS in Parallel Workshop. Building a raspberry pi cluster. HTML. [Online]. Available: http://selkie-macalester.org/csinparallel/modules/RPiCluster/build/html/

[3] V. A. Cicirello, "Design, configuration, implementation, and performance of a simple 32 core raspberry pi cluster," *arXiv preprint arXiv:1708.05264*, 2017.

[4] A. K. Dennis, *Raspberry Pi super cluster*. Packt Publishing Ltd, 2013.

[5] D. V. Diwedi and S. J. Sharma, "Development of a low cost cluster computer using raspberry pi," in *2018 IEEE Global Conference on Wireless Computing and Networking (GCWCN)*. IEEE, 2018, pp. 11–15.

[6] K. Doucet and J. Zhang, "Learning cluster computing by creating a raspberry pi cluster," in *Proceedings of the SouthEast Conference*, 2017, pp. 191–194.

[7] G. Mills. Building a raspberry pi cluster: Part i — the basics. HTML. [Online]. Available: https://glmdev.medium.com/building-a-raspberry-pi-cluster-784f0df9afbd

[8] Computer Hope. Definition of IP. HTML. [Online]. Available: https://www.computerhope.com/jargon/i/ip.htm

[9] P. Pacheco and M. Malensek, *An Introduction to Parallel Programming*. Morgan Kaufmann, 2021.

[10] R. Trobec, B. Slivnik, P. Bulic, and B. Robic, *Introduction to parallel computing: from algorithms to programming on state-of-the-art platforms*. Springer, 2020.

[11] Computer Science Department. COMP-481: applied parallel algorithms. HTML. [Online]. Available: http://www.cs.kzoo.edu/cs481/index.html