Teaching Heterogeneous and Parallel Computing with Google Colab and Raspberry Pi Clusters

Zhiguang Xu
Valdosta State University
Valdosta, GA, USA
zxu@valdosta.edu

ABSTRACT
In this paper, we describe the process and evaluation results of teaching Heterogeneous and Parallel Computing with Google Colab and Raspberry Pi Clusters in a senior elective course in Spring 2023. The course began with an introduction to the fundamentals of designing and building parallel programs. Then, while the whole class went on to learn and practice CUDA on Google Colab for around five and half weeks, in parallel, a team of two students spearheaded a pilot project as their undergraduate research project to build, configure, and test a cluster of four Raspberry Pi’s. Then the rest of the class was divided into seven teams and spent one and half weeks to build their own Raspberry Pi clusters using the instructions and tutorials developed through the pilot project. Now that each team had a Raspberry Pi cluster fully functional and accessible through WiFi, in the next seven weeks, the class went on to learn OpenMP and MPI and developed shared-memory and message-passing based programs on various scales. In this paper, students’ performance on the course labs and assignments, their end-of-semester evaluations, and three anonymous surveys were collected as data to produce an analysis of the course. The lessons we learned from this offering of the course are: 1) Google Colab is a great and cost-saving platform for students to work on real-world alike parallel programs with modern CPUs and GPUs; 2) Students benefit substantially from the hands-on experience of building and configuring a physical cluster of Raspberry Pi’s; 3) Heterogeneous and Parallel Computing has a positive impact on student learning and gives them a new perspective on how to organize and process data in an efficient way therefore should be offered at a higher frequency to a broader range of students.

CCS CONCEPTS
• Social and professional topics → Computer science education; Student assessment; • General and reference → Empirical studies; • Computing methodologies → Parallel computing methodologies.

KEYWORDS
Heterogeneous and Parallel Computing, Google Colab, Raspberry Pi Clusters

ACM Reference Format:

1 INTRODUCTION
Heterogeneous and Parallel Computing (HPC) is now undoubtedly one of the central parts of the undergraduate Computer Science (CS) curriculum [2]. This is an echo of the intensified demand for squeezing more performance out of the contemporary computing technologies by utilizing multiple computing resources. These technologies include but are not limited to the wide deployment of multi-core processors, the rapid growth of clusters of networked and orchestrated computing nodes, and the recent emergence of General Purpose Graphics Processing Units (GPGPU) with massively parallel architectures.

In spring of 2023, we taught HPC in a senior elective course, which hadn’t been offered for several years mainly due to the lack of hardware resources. Three factors contribute to the revitalization of that course – 1) Google Colaboratory as a cloud service is now readily available with free access to modern CPUs and GPUs. This unburdens us of high financial and technical costs of establishing and maintaining a CUDA (Compute Unified Device Architecture) lab; 2) An internal funding of approximately $5k in fall of 2022 with a focus on experiential learning allows us to acquire necessary hardware components so that students in this course could build eight Raspberry Pi clusters; 3) The NSF (National Science Foundation) Affiliated Virtual Workshop on Teaching Undergraduate Collaborative and Heterogeneous Computing (TuUCH) [9] that we attended in summer of 2022 serves as a constructive inspiration. For example, some examples used in this offering of the course come directly from the TuUCH workshop.

Seventeen CS students enrolled in this course. Although all of them could program in C since it is the prerequisite to this course, most of them had little knowledge about Heterogeneous and Parallel Computing. Therefore, the course started with an introduction to the fundamentals of designing and building parallel programs such as Amdahl’s Law [2]. Students were also familiarized with the Google Colab platform including how to request a GPU as hardware accelerator, compiling and running C code both embedded in a Jupyter Notebook and persisted on Google Drive. While the former is lightweight and convenient, the latter allows students to...
use local IDEs of their preference while edited programs (especially large ones consisting of multiple files) are automatically synchronized to their cloud storage and ready for an execution using GPUs provided by Google Colab.

Then, while the whole class went on to learn and practice CUDA on Google Colab for around five and half weeks, in parallel, a team of two students worked on a pilot project as their undergraduate research project to build, configure, and test a cluster of four Raspberry Pi’s. Then in the next one and half weeks, using the instructions and tutorials developed through the pilot project and assistance from those two students, the rest of the class built their own Raspberry Pi clusters in seven teams of two/three. Proudly later, the pilot project participated a three-minute video competition at the annual Undergraduate Research Symposium in our institution and won the 2nd place award. Now that each team had a Raspberry Pi cluster fully functional and accessible through WiFi, in the rest of the semester for about seven weeks, the whole class went on to learn OpenMP and MPI and developed shared-memory and message-passing based programs on various scales.

In this paper, we will present the evaluation results of the course based on three instruments: students’ performance on the course labs and assignments, their end-of-semester evaluations, and three anonymous surveys. The lessons we learned from this offering of the course are: 1) Google Colab is a great and cost-saving platform for students to work on real-world alike parallel programs with modern CPUs and GPUs; 2) Students benefit substantially from the hands-on experience of building and configuring a physical cluster of Raspberry Pi’s; 3) Heterogeneous and Parallel Computing has a positive impact on student learning and gives them a new perspective on how to organize and process data in an efficient way therefore should be offered at a higher frequency to a broader range of students.

The rest of this paper is organized as follows. Existing work is briefly reviewed in Section II. In Section III, we describe main components in our teaching approach. In Section IV, we describe the course evaluation instruments. In Section V, we present and discuss the results. Finally, Section VI provides conclusions and future work.

2 EXISTING WORK

The Curriculum Guidelines for Undergraduate Degree Programs in Computer Science (CS2013) emphasizes the need for incorporating Heterogeneous, Parallel, and Distributed Computing into the CS curriculum [8], but makes very few recommendations about which specific topics to be covered. In the offering of our course in Spring 2023, we chose to cover three respective paradigms in the development of parallel software on multicore CPUs (OpenMP), GPUs (CUDA), and clusters of networked machines (MPI). All these topics (and some additional and more advanced ones) are comprehensively discussed in [2].

Various routes have been taken to teach HPC at the undergraduate level. Ferner et al. [1] presents how to teach parallel and distributed computing during the COVID-19 pandemic using hands-on materials for remote learning. [5] uses a pattern programming approach to teach parallel computing in five universities across the State of North Carolina. An independent professional evaluator employed to analyze their teaching results indicates that regardless of the tool being used, implementing an algorithm by first identifying its pattern and then following known implementations for that pattern have a positive impact on students’ ability to create parallel programs. Bunde [3] presents workshop modules for teaching about computational and memory heterogeneity with CUDA, a common approach to GPU programming. Many projects explore low-cost Raspberry Pi clusters to introduce parallel and distributed computing concepts. For example, Matthews et al. [6] and Doucet [4] provide step-by-step instructions on building a cluster computer using Raspberry Pi’s and use them to teach OpenMP and MPI programming. Because the undergraduate computer science curriculum includes so many topics nowadays, adding a new course as a required part of the curriculum without increasing the number of hours to graduation is difficult. Qasem et al. [7] present a module-driven approach in which coverage of the HPC topics is broken down into smaller units and dispersed throughout the lower and upper divisions of the curriculum. Such flexibility incurs encouraging results both in terms of learning outcomes and student engagement and interest.

3 MAIN COMPONENTS IN OUR TEACHING APPROACH

3.1 Programming in CUDA on Google Colab

We strive to train students to become not only an effective CUDA programmer but also a computational thinker. On the conceptual level, we use classic problems such as adding vectors and multiplying matrices to introduce basic CUDA concepts such as kernels, grids, blocks, threads, etc., then on the physical level, we use intuitive examples and diagrams to help student better comprehend complex compute and memory architectures in NVIDIA GPUs. Note-worthily, we take good use of the advanced image processing abilities in Python libraries on Google Colab to visualize the otherwise difficult-to-understand parallel algorithms. The following are images from three assignments that students worked on, each of which was rendered by a parallel program in CUDA:

- Julia set in Fig. 1 – handling multidimensional grid of threads and data
- Voronoi diagram in Fig. 2– achieving better occupancy with proper warp scheduling
- Reverse image in Fig. 3 – using shared memory to improve the efficiency of a CUDA program

3.2 Building Raspberry Pi Clusters

One of the unique components in our teaching approach is that students were given hands-on opportunities to assemble, configure, and test their own Raspberry Pi clusters. Through an internal funding of approximately $5k in fall of 2022 with a focus on experiential learning, we had the budget to acquire enough hardware components to build eight Raspberry Pi clusters, one for each team of two/three students. Table 1 below contains the Parts List for each team.

As described in the Introduction section above, a team of two students spearheaded a pilot project to assemble and configure a Raspberry Pi cluster and produced three streamlined instructional
Assemble a cluster of four Raspberry Pi’s (i.e., two Pi 4’s and two Pi 3’s) with hardware parts as listed in Table 1.

- On the Pi board sitting on the top of the cluster, which we call the manager node, install the Raspberry Pi OS and make sure it has:
  - a host name and an admin account
  - SSH enabled
  - VNC server enabled
  - connected to the Internet through WiFi (the Pi device might need to be registered with IT if required by its security policy)
  - one regular account for each team member who is working on this cluster

- On each of the other three Pi’s, which we call the worker nodes, do the same as what was done on the manager node (ensure each has a unique host name but the same admin account across all Pi’s though).

- For our convenience, assign static IP addresses to all Pi’s.

- To facilitate parallel programs to communicate through the MPI system, under both the admin and regular accounts,
  - set up SSH key pairs on the Pi’s such that a program running on the manager node could connect to each worker node via SSH keys rather than username/password.
  - set up the Network File System (NFS) on the cluster.
  - share a folder on the manager node and mount proper folders on the worker nodes to it through NFS. (These folders will hold MPI programs as described in the next section.)

- Install mpich on all nodes.

Then the rest of the students were put in seven teams, who followed these tutorials and built their own clusters as shown in Fig. 4 below.
3.3 Programming in OpenMP and MPI on a Raspberry Pi Cluster

To learn parallel computing on a Raspberry Pi cluster, OpenMP is naturally a good way to start with — on one hand, no external libraries need to be installed in order to compile the code since OpenMP is built into the GNU C++ compiler; on the other hand, threads communicate with each other through shared memory therefore only one Raspberry Pi node on the cluster is needed. Moreover, thanks to their WiFi connectivity, for the convenience of students, all clusters are kept in one room and students can SSH them from anywhere on campus at any time.

Progressively in about two weeks, students developed an OpenMP program that calculates the integral of a mathematical function using the trapezoidal rule: at beginning, they had to manually arrange which group of trapezoids each OMP thread works on, then improved it using OMP for loop and OMP reduction.

More time (about four weeks) were spent on MPI. All Raspberry Pi’s are put in use, on which running processes exchange messages to collaborate. We covered basic and dynamic P2P communications as well as broadcast and collective communications.

Students worked on several MPI programs, one of which is the classic Random Walk problem in the 2D domain. Each MPI process deals with one appropriate subdomain out of the total domain and the following is a self-explanatory algorithm by each process in the MPI world:

Algorithm 1: Random Walk by Each Process in MPI World

```c
foreach incoming_walkers do
    while incoming_walker → steps_to_walk > 0 do
        if incoming_walker → out_of_bound is true then
            add incoming_walker to outgoing_walkers;
            break;
        else
            walk to right;
        end
    end
    if process → rank is even then
        send outgoing_walkers to the next process;
        receive incoming_walkers from the previous process;
    else
        receive incoming_walkers from the previous process;
        send outgoing_walkers to the next process;
    end
end
```

4 EVALUATION INSTRUMENTS

We use students’ grades, their end-of-semester evaluations of the course, and three anonymous surveys to evaluate the effectiveness of our teaching approach.

4.1 Students’ Grades

There are eight in-class labs and seven take-home assignments in this course. All these labs and assignments are team projects and unevenly weighted. Labs where students could receive assistance from the instructor count for half of students’ final grades and assignments that students must complete independently count for the other half.

4.2 End-of-semester Student Evaluations

At the end of the semester, 70% of students filled up an anonymous online evaluation form Student Opinion of Instruction (SOI) where students rate the course using questions shown in Table 2 below. SOI is an institutional exercise and the 70% response rate in our course is way above the institutional average of 44%. These SOI questions are presented with a five-point Likert scale from “strongly disagree” (1), “disagree” (2), “neutral” (3), “agree” (4), to “strongly agree” (5). Moreover, they could also provide text responses to two additional questions “What suggestions do you have for improving the course?” and “What did the instructor do that most helped your learning?”.

Table 2: End-of-semester Student Evaluations

<table>
<thead>
<tr>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1: Course was well organized</td>
</tr>
<tr>
<td>Q2: Course assignments clearly explained</td>
</tr>
<tr>
<td>Q3: Course increased knowledge of topic</td>
</tr>
<tr>
<td>Q4: Instructor presented material clearly</td>
</tr>
<tr>
<td>Q5: Instructor provided helpful feedback</td>
</tr>
<tr>
<td>Q6: Instructor responded in timely manner</td>
</tr>
</tbody>
</table>

4.3 Anonymous Surveys

In Spring 2023, students were invited to provide anonymous feedback that would assist with the evaluation of this course offering and the improvement of the future course offerings, especially in terms of the coverage of HPC topics and our teaching approach. Feedback was collected by the instructor via three surveys: a pre-, a mid-, and a post-course survey as shown in Table 3 at the top of
the next page. The mid-course survey was conducted right after we completed our discussions on CUDA.

The purpose of the these three surveys was to assess the degree to which the students learned the material taught during this offering. Each survey question was also presented with a five-point Likert scale from “strongly disagree” (1) through “strongly agree” (5). All students participated in all three surveys.

5 EVALUATION RESULTS

All students passed the course with eleven A’s, four B’s, and two C’s. The mean and standard deviation of students’ final grades are 92.28 (out of 100) and 8.87 respectively.

The end-of-semester student evaluation results are in general quite positive. The means of Q1 to Q6 range from 3.91 to 4.82. The standard deviations of Q1 to Q6 range from 0.4 to 1.38. The mean and standard deviation of the overall *Student Opinion of Instruction* (SOI) are 4.3 (out of 5) and 0.85 respectively. All these quantitative results are good in comparison with the institutional numbers, especially for a new course preparation.

Specifically, first, students seem to enjoy the course and are particularly motivated by having convenient access to physical hardware that they build on their own. It is fun to see the positive feedback in some students’ text responses such as: “...The hands on learning with the Raspberry Pi’s was very fun and made the class enjoyable. This is one of the best CS classes I’ve taken...”.

In Fig. 5 below which shows the survey results for Q5 of the “Hands-on” subject in the survey, most students in the Pre-Course survey responded either “disagree” or “strongly disagree” due to their lack of experience with hardware. However, after successfully building their own Raspberry Pi clusters, they mostly responded “agree” or “strongly agree” in the Post-Course survey.

Figure 5: Survey Results for the Hands-on Subject

Second, for the three specific HPC topics of “CUDA”, “OpenMP”, and “MPI”, the following figures show the survey results. We can see improvements in students’ grasps of knowledge in all these three HPC topics upon their completion of this course.

- Fig. 6: Q4 of the “CUDA” subject in the survey
- Fig. 7: Q7 of the “OpenMP” subject in the survey
- Fig. 8: Q9 of the “MPI” subject in the survey

Note that survey results for Q3, Q6, and Q8 in the survey are all demonstrating similar trends therefore are skipped here.

Fourth, the survey results for students’ general familiarity with Heterogeneous and Parallel Computing and their abilities to solve problems with HPC techniques are shown in Fig. 9. We are glad to see the continuous improvements as the semester goes on. This experience has a positive impact on students learning in computer science and gives them a new perspective on how to organize and process data in an efficient way.

- Fig. 9: Q1 of the “General” subject in the survey

Note that Q2 in the survey is designed to give the instructor an understanding of students’ preparedness in terms of coding in C therefore its survey results are skipped here.
And finally, students also expressed concerns about this course and/or challenges they faced either verbally or through their answers to the “What suggestions do you have for improving the course?” question as part of the end-of-semester student evaluations. Many of them complained that “Professor was incapable of ending class on time most days”. This happened mainly because it was the first time this course was offered with such broad coverage of HPC subjects, therefore, unanticipated issues during the lectures and labs might have kept students in class longer than scheduled. We plan to address this problem through better planning and organization of class activities in the next offering of this course. Quite a few students also complained that although Google Colab is convenient and free of charge, its usage limit could cause long waits and reduce their productivity, especially when working on large notebooks with high demands on computational resources such as GPUs. Next time when we offer this course, we plan to add coverage of profiling CUDA code with Nsight while on the OpenMP/ MPI side, we will develop some additional large-scale and real-world alike projects. Our experience in the offering of the course as presented in this paper was in general very positive, which set up a solid foundation for more and better expansions in the future. On the CUDA side, we plan to add coverage of profiling CUDA code with Nsight while on the OpenMP/ MPI side, we will develop some additional large-scale and real-world alike projects. We also would like to revise our curriculum so this course could be offered at a higher frequency to a broader range of students.

6 CONCLUSION

Despite the apparent importance of HPC topics, it is a major challenge to integrate them into the undergraduate computer science curriculum at non-R1 institutions. On one hand, the rapid technological advancements have saturated the curriculum; on the other hand, hardware resources needed to cover these topics typically come with forbiddingly high costs. The sparse coverage of HPC at such institutions implies that most of their undergraduates joining the workforce will have little to no exposure to the relevant concepts. In this paper, we present our approach to confront this challenge. We use powerful yet affordable platforms of Google Colab and Raspberry Pi Clusters on which many potential HPC projects are open for students to work on, both in and out of classrooms. Our experience in the offering of the course as presented in this paper was in general very positive, which set up a solid foundation for more and better expansions in the future. On the CUDA side, we plan to add coverage of profiling CUDA code with Nsight while on the OpenMP/ MPI side, we will develop some additional large-scale and real-world alike projects. We also would like to revise our curriculum so this course could be offered at a higher frequency to a broader range of students.

REFERENCES