January 2020

Volume 11 Issue 1

# JoCsE

## Journal Of Computational Science Education

Promoting the Use of Computational Science Through Education

ISSN 2153-4136 (online)

**Volume 11, Issue 1** **January 2020**

# JOCSE

Journal Of Computational Science Education



The Journal Of Computational Science Education (JOCSE), ISSN 2153-4136, is published quarterly in online form, with more frequent releases if submission quantity warrants, and is a supported publication of the Shodor Education Foundation Incorporated. Materials accepted by JOCSE will be hosted on the JOCSE website, and will be catalogued by the Computational Science Education Reference Desk (CSERD) for inclusion in the National Science Digital Library (NSDL).

**Subscription:** JOCSE is a freely available online peer-reviewed publication which can be accessed at `http://jocse.org`.

# Contents

# Introduction to Volume 11 Issue 1: Special Issue on HPC Training and Education

Nitin Sukhija
Slippery Rock University of Pennsylvania
Slippery Rock, PA

## FORWARD

High performance computing is becoming central for empowering scientific progress in the most fundamental research in various science and engineering, as well as societal domains. It is remarkable to observe that the recent rapid advancement in the today?s and future computing and software environments provide both challenges and opportunities for cyberinfrastructure facilitators, trainers, and educators to develop, deliver, support, and prepare a diverse community of students and professionals for careers that utilize high performance computing to advance discovery. This special issue focuses on original research papers submitted to the First Workshop on HPC Education and Training for Emerging Technologies (HETET19), which was held in conjunction with ISC19 conference in Frankfurt, Germany, June 20, 2019 the Second Workshop on Strategies for Enhancing HPC Education and Training (SEHET19), which was held in conjunction with PEARC19 conference in Chicago, Illinois, U.S.A., July 29, 2019, and the Sixth SC Workshop on Best Practices for HPC Training and Education (BPHTE19), which was held in conjunction with SC19 conference in Denver, Colorado, U.S.A., November 17, 2019.

This special issue begins with an article by Holcomb et al that presents the lessons learned from the summer school and the internship program operated by University of Virginia in partnership with NASA from the year 2013 to 2018. The greatest challenge they found was to provide a good program for students with widely varying backgrounds, skills, and expectations. They accommodated the diversity by increasing hands-on exercises interspersed with lectures and by expanding their bank of programming exercises to span a range of abilities.

The article by Goodhue describes the Northeast Cyberteam Program, a 3-year National Science Foundation (NSF) funded initiative to increase effective use of cyberinfrastructure by researchers and educators at small and mid-sized institutions in Northern New England (Maine, Massachusetts, New Hampshire, Vermont). The core of their strategy is to build a regional pool of research computing facilitators (RCFs) and a process to share them across institutional boundaries, augmented by knowledge sharing and self-service learning tools that increase the effectiveness of Research Computing Facilitators. They conclude by highlighting the program management portal and the impact of the program on the smaller institutions and its relevance to the potential facilitators.



The article by Chakravorty et al presents the Summer Computing Academy (SCA), a weeklong cybertraining program offered to high school students by High Performance Research Computing (HPRC) at Texas A&M University (Texas A&M; TAMU). They provide the best practices that have been adopted in the Summer Computing Academy model and the recruitment strategies along with the selection criteria for the participants. They conclude by discussing the legal and administrative issues encountered while hosting such efforts and the sustainability of the effort.

The article by Cais et al presents a training model of the carpentries for the HPC space that provides a pathway to collaboratively created training content which can be delivered in a scalable way (serving everything from university or industrial HPC systems to national facilities). They describe some of the training material design principles and how they influence the creation process. They conclude by discussing two distinct evaluation processes, a review process that happens during material creation and learner evaluations that occur during/after training events.

The article by Lathrop et al describes the National science Foundation funded Blue Waters project, which supports an Education, Outreach and Training (EOT) program focused on preparing an HPC-capable workforce with an emphasis on petascale computing competencies. They discuss how the Blue Waters EOT team engages undergraduate students in internships, graduate students in fellowships, researchers as participants in training sessions, trainers and educators as PIs of education allocations, and underrepresented communities as PIs of broadening participation allocations. They conclude by describing the impact and benefits of the project, including directly reaching people located in many foreign countries, as well as freely disseminating materials that have been downloaded and used by thousands of people world-wide.

The article by Kunkel et al presents the current status of the certification program curated by the HPC Certification Forum. They describe the program that consists of three parts: the tree of defined competencies, the examination of practitioners to prove they possess those skills, and finally the certification demonstrating their knowledge. They conclude by discussing the benefits of the program that not only allows the re-use of existing content but also makes it possible to create a new ecosystem in which HPC centers, research labs, academic institutions and commercial companies could offer the best of their teaching material.

The article by Wong et al describes design and plan of a National Science Foundation supported hands-on Research Experiences for Computational Science, Engineering, and Mathematics (RECSEM) program at the University of Tennessee (UTK) in high-performance data sciences, data analytics, and machine learning on emerging

computer architectures. They discuss the experiences and resolutions in managing and coordinating the program, delivering cohesive tutorial materials, directing mentorship of individual projects, lessons learned, and improvement over the course of the program, particularly from the perspectives of the mentors. They conclude by discussing the outcomes and the progress of the students and overall impact of the REU program.

The article by Saravanan et al presents a novel course curriculum to teach high performance, parallel and distributed computing to senior graduate students (PhD) in a hands-on setup through examples drawn from a wealth of areas in computational biology. They provide a sample course outline that details the HPC concepts that can be covered in a one-semester course, along with the suggested bioengineering applications to introduce them. They conclude by discussing the assessment methods and the high evaluation ratings received for this interdisciplinary course.

The article by Wang et al describes an ongoing effort, FreeCompilerCamp.org, a free and open online learning platform aimed to train researchers to quickly develop OpenMP compilers. They explain the challenges faced in giving compilers training and the solutions and present the implementation of the platform. They provide an overview of the design of the web-based tutorials to take advantage of FreeCompilerCamp. They conclude by the discussing the feedback received on the design of FreeCC tutorial.

The article by Terboven et al presents a learning status survey, a developer diary to track the student?s progress in achieving the learning objectives, and an approach to enable the comparison of different HPC cluster architectures or parallel programming models. They report on the learning objectives of the labs along with their experiences with using various stimuli in their labs to increase the success rate of the learning objectives while fostering creative solutions. For example they use a competition among students to motivate them to optimize their codes for performance and show the opinions that students have towards these concepts. They conclude by discussing the feedback on the concept of self-paced learning and the evolution of the software labs in terms of obtained knowledge, training productivity and programming models, as well as students? feedback based on teaching evaluations.

The article by Colbry et al shares the lessons learned during the Computational Mathematics, Science and Engineering (CMSE) department?s development at the Michigan State University (MSU) and the initiatives it has taken on to support computational research and education across the university. They describe how the department has aided in establishing both traditional degree programs and non-traditional options to build computational competency in learners from across STEM. They conclude by discussing the CMSE department as uniquely positioned at the ?triple junction? of algorithm development and analysis, high performance computing, and applications to scientific and engineering modeling and data science.

The article by Wofford and Lueninghoener describes methodology followed at the Supercomputing Institute to teach cluster computing to undergraduate and graduate students. They described the redesigned boot camp curriculum and the two-fold approach of the new curriculum, which are to extend the content of the lectures to include more technical depth and more technical areas; and to replace the labs with practica (staged guides that have a mix of free exploration prescribed steps). They conclude by providing qualitative and quantitative results indicating the positive impact of the new curriculum on the program over recent years.

The article by Betro and Loveless describes the development of a STEM ecosystem where both the science department and math department of Baylor school have implemented an interdisciplinary approach to introduce a spectrum of laboratory and computing research skills. They outline the benefits of this ecosystem that has been an effective tool in allowing several driven and interested students to participate in collegiate-level and joint collegiate projects involving virtual reality, robotics and systems controls, and modeling. They conclude by discussing various critical factors in readying the next generation of computing leaders.

The article by Harrell et al presents the parallel computing portion of the HPC seminar series which are used as a tool to introduce students from many non-traditional disciplines to scientific, parallel and high-performance computing. They describe the two-fold approach to their curriculum: engaging students with hands-on exercises using a real-world scientific application along with regularly lecturing on more general parallel computing topics in the class. They conclude by discussing the student evaluations and the lessons learned to give undergraduate students an opportunity to explore the field of HPC and big data in a non-traditional computer science course setting and build a basic foundation of computational and data skills for their further education and research activities

The article by Chakravorty and Pham presents a review of experiences of using Google?s Qwiklabs online platform for remote and in-person training from the perspective of the HPC user. They describe the scaffolded instruction methods supported by the Qwiklabs training platform that support learners with varied skill sets. They conclude by providing recommendations on how the large-scale computing community can leverage these opportunities to work with Cloud service providers to assist researchers nationally and at their home institutions.

The article by Younts and Harrell describes the teaching methods and hardware platforms used by Purdue Research Computing to train undergraduates for HPC system roles. They present the scientific computing track, which provides students with some basic Linux skills but focuses on running and visualizing scientific codes and the HPC Systems Track, which truly focus on important aspects of building systems. They conclude by discussing the system they have designed and the failures and successes they have had teaching HPC system administrators.

The article by Kunkel discusses the current state of the developed Skill Tree in the HPC Certification Program and the process of contributing to the skills to the program. They describe the skills, organized in a tree structure from a coarse-grained to a fine-grained representation, allowing users to browse the skill based on the semantics. They conclude by reporting on the HPC certification forum which plays a virtual central authority to curate and maintain the skill tree and certificates and how the contributions to the skill definitions can be made.

# Lessons Learned from the NASA-UVA Summer School and Internship Program

Katherine Holcomb
Research Computing
University of Virginia
Charlottesville, Virginia
kah3f@virginia.edu

Jacalyn Huband
Research Computing
University of Virginia
Charlottesville, Virginia
jmh5ad@virginia.edu

Tsengdar Lee
High-End Computing Program
NASA
Washington, D.C.
tsengdar.lee@nasa.gov

## ABSTRACT
From 2013 to 2018 the University of Virginia operated a summer school and internship program in partnership with NASA. The goal was to improve the software skills of students in environmental and earth sciences and to introduce them to high-performance computing. In this paper, we describe the program and discuss its evolution in response to student needs and changes in the high-performance computing landscape. The future direction for the summer school and plans for the materials developed are also discussed.

## Keywords
computer science education, scientific computing, curriculum development, mentoring.

## 1. INTRODUCTION
The University of Virginia-NASA Summer School and Internship Program was motivated by perceived gaps in basic software-engineering and high-performance computing skills in students in science programs, particularly environmental and earth sciences. As science moves in an increasingly computational direction, the preparation of students lags further behind the demands of their future research careers. While most engineering undergraduates at the majority of higher-education institutions are required to take some form of introductory programming course, usually taught in a Computer Science department, the same is not true of science students in most cases. If they do take programming courses, they are often taught by science faculty untrained in modern software principles. A further omission is high-performance computing and parallel computing; not only is this rarely taught even to computer-science majors, but as an advanced topic it requires proficiency in basic programming before students are able to assimilate it. Our program was intended to explore ways to address these issues.

The program took place in two phases. In the first phase, which we named the Intensive Summer School for Computing in Environmental Sciences (ISSCENS), we accepted 20 students, with 10 of these selected for subsequent 8-week internships at NASA centers. In the second phase, Advanced Computing for Earth Sciences (ACES), we accepted a minimum of 20 students, all of whom were placed in 8-week internships. ISSCENS ran during the summers of 2013, 2014, and 2015. ACES took place in 2016, 2017 and 2018. Through all sessions of the Summer School we continuously modified the curriculum and emphasis while retaining the basic structure established for ISSCENS.

For both programs, applicants were required to be enrolled in or to have just completed an academic degree program at a United States institution of higher education. For undergraduate applicants, upper-division students were preferred, but this was not a requirement. During the ISSCENS phase, we accepted international students for the 10 slots that did not lead to NASA internships. For the other ISSCENS students, and for all ACES students, United States citizenship was required since this is necessary for regular NASA internships. Students were housed in a dormitory on the campus of the University of Virginia for the Summer School and were provided breakfasts and lunches on weekdays.

No prior programming experience was required, but many attendees had some minimal exposure, often through simple scripting in a language like MATLAB™, while some, particularly the ACES participants, had significant computing backgrounds. The attendees came from 57 different US academic institutions. In both programs, locally-residing students at the University of Virginia, most of whom were graduate students in the Department of Environmental Sciences, were invited to attend the full program, along with the out-of-town students. Two to five UVA students attended each year. The participants were almost equally divided between male (a total of 79) and female (total 73). We did not formally track ethnicity but the overwhelming majority would be described as white.

## 2. CONTENT
The program content was based on two courses taught through the Department of Computer Science at the University of Virginia, *Computing as a Research Tool* and *Introduction to Parallel Computing*. *Computing as a Research Tool* was aimed at graduate students who needed to apply computing to their research; basic programming was taught in the student's choice of language from the selection offered, along with Unix command-line skills and using a resource manager. *Introduction to Parallel Computing* teaches high-throughput computing, threaded computing, and MPI programming in a compiled language only (usually plain C). *Computing as a Research Too*l was taught by staff of the Research Computing support group, while *Introduction to Parallel Computing* is taught by a Computer Science faculty member.

In recognition of the fact that formal coursework can never reach all researchers who could potentially benefit, in 2008 the Department of Computer Science developed a week-long, accelerated "High Performance Computing Bootcamp" offered jointly with Virginia


DOI: https://doi.org/10.22369/issn.2153-4136/11/1/1

Tech. Taking place in early summer, it was open to all faculty, staff, and students from institutions of higher education in the Commonwealth of Virginia. For a few years this "Bootcamp" alternated between the University of Virginia and Virginia Tech venues, but in 2011 Virginia Tech developed a local version and UVA continued on its own. Staff from Research Computing then began to play an increasingly important role in teaching the Bootcamp. When the ISSCENS program was developed, we condensed our *Computing as a Research Tool* course into the introductory 1.5 weeks and utilized the "Bootcamp" as the final week. In 2016 Research Computing took full responsibility for the Bootcamp and replaced some of the more theoretical material with an introduction to data analytics for the high-performance platform.

The goal of ISSCENS and ACES was to reach students whose universities do not offer coursework such as this at all, or who cannot devote a semester to for-credit courses in computer-science topics. The format was modeled after an accelerated traditional course, with aspects of "flipped" classroom as much as possible. Each class day consisted of a morning series of lectures, with short hands-on exercises at regular intervals to the extent feasible. Afternoons consisted of lab sessions with a set of "homework" problems. A major adaptation over the length of the program was to differentiate beginner, intermediate, and advanced programming projects for each set of topics. This was particularly important for beginning students and significantly increased their satisfaction with the program. Evidence indicates that student comfort with their level of understanding of an aspect of programming correlates best with overall learning [1][2]; so, making sure students master at least the basics of the beginning topics is critical for their success. Motivation is another key factor in student learning [3] but our attendees generally were highly motivated to enhance their career skills. In fact, one student criticism of our program was that the assignments were not based on "real" research problems.

## 3. STRATEGY

We settled on teaching Python as the common language. Python is easy for most students to learn yet has sufficient power to serve as an introduction to modern programming and basic software engineering. Its popularity has grown substantially at high-ranked computer science programs in the United States, passing Java recently as the most popular language taught [4]. It is widespread in online introductory courses from EdX, Udacity, Coursera, MIT Open Courseware, etc. It has also exploded in popularity in many sciences. In particular, it is displacing the commercial software IDL for areas of interest to NASA, specifically Earth Sciences and Astronomy/Astrophysics, particularly for data analysis. The ready availability and ease-of-use of packages such as astropy (astronomy) [5], Basemap [6], xarray [7], and many others for Earth sciences, make it well suited for students in those fields. It is also free and open source so that students are able to install it on their personally-owned laptops without licensing expenses or concerns. Finally, while the critical parts of data-mining and machine-learning systems such as Theano [8] or Tensorflow [9] are generally written in a compiled language such as C++, most users interact through these packages' Python bindings. On modern computing systems, the general slowness of Python (and most other interpreted languages) is usually not a significant issue.

Once Python was chosen, it was necessary to select a version and then a distribution. We chose the older Python 2.7 simply because some packages had not been ported to Python 3.3 and up at the time, on at least one platform (e.g. Basemap was not ported to 3.N on Windows). At all points at which differences are significant, we taught both forms. By the time Python 2.7 reaches end-of-life in 2020 [10] we expect all packages to have been ported or replaced; for instance, Cartopy [11] should have implemented all features of Basemap. In 2018 we allowed students to choose but remained with a base of Python 2.7, with an increase in discussion of maintaining code for both versions [12].

Since our emphasis was on software development with an eye toward more complex code projects, we focused on the Spyder IDE [13] rather than the popular Jupyter notebook. Jupyter is oriented toward data exploration and distribution of a "narrative" of code and analysis, whereas Spyder is a lightweight but traditional IDE with many features helpful for code development, including variable and object viewers and direct access to the built-in debugger and profiler. Spyder also marks syntax errors dynamically and can look up and show names and documentation for functions in modules and packages on the fly. We demonstrated use of Jupyter during exercises with the statistical package Pandas, but mostly used Spyder. Students generally find Spyder a comfortable working environment and often prefer it to Jupyter.

For the Python language support, we quickly settled on the Anaconda distribution from Continuum Analytics [14]. This distribution is comprehensive, cross-platform, and usually very easy to install, even for novices. Over the past six years Continuum has improved the usability of their application and Anaconda now provides a graphical interface for managing packages, a significant improvement over the manual procedure using their Conda package manager from a command line [15]. Conda is still functional for standalone applications, with more powerful features such as conda environments and "pinning" package versions in an environment. There is also the pip installer for packages they do not support directly [16], but for beginners a graphical package manager is very helpful.

Modern Fortran, taught in an accelerated fashion, was used as the class compiled language for the first four sessions. Students who needed to learn it would be motivated to continue, using resources we provided as well as their own references, while students who would not need it (or would need it only occasionally) gain some exposure without being forced to spend too much time with it. Fortran is widely used in the Environmental/Earth Sciences community. There are also many legacy codes written in Fortran that students are likely to encounter if they remain in their fields. However, it is rarely taught except to advanced students majoring in atmospheric sciences or meteorology, and even then it often is not taught particularly well. Modern Fortran (the 2003 standard and up), includes many features of newer programming languages, including arrays as a container data structure, modules, subprogram interfaces, and more sophisticated data structures, including classes. We particularly emphasized array operations, which in our testing using recent compilers have proven to be remarkably fast. Obsolete constructs were described so students would recognize them, but we did not use them in examples or exercises.

For the last two years we taught Fortran and C++ side by side, allowing students to choose the one most useful to their research goals.

For the final week of each session, in which high-performance and parallel computing were introduced, we supported C/C++, Fortran, Python, and sometimes R.

## 4. CURRICULUM

The program started the Wednesday following Memorial Day and ran for two and a half weeks. We began with three days of instruction in a common scripting language (i.e., Python). The next Monday and Tuesday were devoted to object-oriented programming concepts and an introduction to software engineering. The next day was generally

focused on advanced visualization, with instruction in compiled-language programming for the last two days of the week. The last week was variable over the life of the project; but in the three ACES sessions it consisted of Unix, bash scripting, and use of a queueing system on Monday, either optimization and high-throughput computing or data analytics on Tuesday, MPI usually on Wednesday and Thursday, and programming using a multicore paradigm usually on Friday.

The curriculum evolved significantly from the first session in 2013. For the summers of 2013-2015 students applied directly to the ISSCENS program, and we then recruited NASA mentors for 10 of them. As a result, most of the applicants to ISSCENS had little to no programming experience and we spent more time on basic skills. We also included a session on scripting ESRI's ArcGIS Desktop with Python. We spent three days on a mixture of Fortran and Unix skills. The final week, the "High-Performance Computing Bootcamp," focused on high-throughput computing, using a resource manager, code optimization with an emphasis on compiled languages, OpenMP, and MPI.

For the second phase (ACES), students applied directly to NASA centers through the One-Stop Shopping Initiative. The mentors were recruited by NASA education officers, and the students were selected by the mentors. This changed the typical background of the students, since mentors usually wanted students who were more prepared to work on computational projects. Consequently, we reorganized the ISSCENS curriculum. We continued to teach three days of introductory Python and one day of object-oriented Python, but we increased the material for the "software engineering" day, and upped the sophistication of the "advanced visualization" day. Fortran was consolidated into two days, with Unix and bash moved to the first day of the "HPC Bootcamp." Coinciding with the first ACES session in 2016, Research Computing took over full responsibility for the "Bootcamp" from the Computer Science Department and drastically reduced the amount of theory in the instructional materials, substituting more practical and hands-on content in its place. The Unix/bash/SLURM session enabled students to use larger systems at NASA immediately upon arrival at their internship, if appropriate to their projects. Most ACES students, even those with computational experience, arrived never having used anything other than personal laptops or institutional desktops, so this was also essential preparation for the rest of the week. In 2017, in recognition of the growing importance of "big data" analytics and machine learning, we devoted the second day of the "Bootcamp" to this topic, including some exercises in applying Tensorflow to remote-sensing images. We discussed how cloud resources, such as Amazon Web Services, could be used for implementations of Spark Machine Learning.

In 2017 we also combined serial optimization with OpenMP/multiprocessing (the latter for Python users). Serial optimization is especially important for Python programmers; so we increased the content in that area beyond the original Computer Science material. We reduced the MPI introduction to a minimum and provided information about accelerators such as the Intel Knight's Landing and NVIDIA GPGPUs, although we had no KNL nodes and only two NVIDIA-equipped nodes at the time, which made direct experience difficult. In any case, the students' general lack of computer-science background and, often, C/C++ programming skills, as well as the short time available to devote to accelerators, prohibited any attempt to teach CUDA programming for GPUs. Therefore, our focus was on introducing OpenACC [17] instead

For 2018 we "packaged" different portions of the curriculum into workshops that we offered to the larger University of Virginia community. The only session we did not open to University attendees at that time was the "software engineering" session. It is important to note that we taught little modern software engineering. Our emphasis is on readable code, careful consideration of code data and algorithmic structures, team development, and testing and debugging skills.

The HPC "Bootcamp" for 2018 was broken into three logical portions, each of which was accessible to all UVA students and faculty as well as to ACES students. By that time we also had accelerator hardware available for hands-on access, and so had the opportunity to restructure appropriate sections to take more advantage of these devices, including KNL. We also acquired more GPUs on our HPC system so that we demonstrated Tensorflow in its most usual environment (in 2017 we used a core-only version). The "big data on HPC" day was also a workshop available to UVA affiliates. The final segment was a three-day block; the first day covered serial optimization, OpenMP, and OpenACC. The two following days introduced MPI, including mpi4py [19]; the introduction to the Intel Knight's Landing MIC occurred on the final day, along with PGAS concepts such as co-array Fortran [20][21] and UPC++ [22]

# 5. OUTCOMES

Our effectiveness was variable and depended very much on the background of individual students. The biggest challenge we faced throughout the program was that the participants' experience, interests, and aptitudes ranged from complete beginners who had never programmed in any language to students finishing a master's degree in computer science. The intention of the program was to train students in HPC, but students who have not mastered basic serial programming have very little ability to absorb parallel computing, particularly in an intensive setting with a short timeframe.

Assessments administered to the students showed that the majority felt that the pace was reasonable; it was fast, but they felt they could go back later to pick up more details. In aggregate, approximately 10% thought the pace was too fast, and 5% thought it too slow.

In the 2017 session we forced students to work in small teams on a programming project for the "software engineering" session. Somewhat to our surprise, most of the students had never worked collaboratively on programming, where one programmer must code to an API that other programmers established. Many of the students felt that this was one of the most valuable learning experiences of the entire program.

The most important assessment is their performance in their later research and internships. For the first three summers only 10 students went to NASA internships sponsored by us, but many went to internships with other organizations. Others returned to their research. For the final summers all but one non-UVA attendee went to NASA internships. Student assessments were quite positive of their internships with one exception. Assessments were anonymous but several alumni contacted us personally to state that their internships would have been much more difficult without the training program.

Example quotes:
"I was able to make more progress than my mentor had expected."
"I know that I would have been woefully unprepared for this summer if not for the lessons and guidance provided to us through ACES."

Several NASA mentors also requested students who had participated in our program in previous years. In at least one case, a mentor also sent an intern from the previous summer to ACES before the student's second summer internship.

In a few cases, our program was career-altering. During the first ISSCENS session, an attendee who had never programmed before

discovered she had aptitude in that area, and reoriented her area of interest in her field of hydrology toward computational modeling.

Another outcome of this project was a professionally-produced video series on introductory Python, aimed specifically at scientists and engineers and therefore emphasizing NumPy, SciPy, Pandas, and other packages and techniques used heavily by scientific applications. Seven students in an advanced chemistry course at the University of Virginia were recruited as a focus group to use this series to learn to program using these videos. Even those who had no programming experience were able to contribute to a final group programming project for the course.

## 6. FUTURE OF THE PROGRAM

Once we opened the "packaged" sessions to University of Virginia affiliates, we found that they were extremely popular, so we have continued the program as the Research Computing Summer Education Series (SES). For 2019, with no ACES students attending, we condensed the Python sessions to 3 days, largely by eliminating some topics that are primarily of interest to environmental-science students as well as a few more advanced topics. Response was so enthusiastic that we had to find a larger auditorium. The C++/Fortran short course also proved popular. Inspired by our success with Python, we added short courses in R and MATLAB™. We opened the "software design" short course and it attracted considerable interest as well. We augmented the last week's HPC training with more data analytics, including machine learning, and added short courses in image processing and bioinformatics, the latter emphasizing HPC applications.

A major difference between the Summer School students and the UVA-only attendees was the level of motivation and distraction. Summer School attendees expected to devote their entire day to the material, were motivated to learn in order to do well at their internships, and maintained focus. Summer Education Series attendees were not as reliable at returning for lab sessions. Lab attendance was very good for Python and R, less so for compiled languages and advanced topics, and highly variable for HPC-specific topics. SES students also tended to be low skill. We will adjust our material to reflect that, by incorporating some of our "beginner" level material as exercises within the lecture/hands-on sessions.

We currently have all Summer Education Series course materials online but not in a polished or readily accessible form. Our plan for the next year is to convert the lectures and hands-on materials into a combination of Markdown and, where appropriate, Jupyter notebooks. These will be posted to a public site which will also host our Python video series; this site currently is https://learning.arcs.virginia.edu but this will be consolidated in the near future with other education sites managed by UVA Research Computing. This will make self-guided learning possible. It will also allow us to disseminate the coursework to any institution that would be interested in replicating our series. Only the "Introduction to HPC" session is particularly site-specific, and even that could be easily modified for other sites and resource managers.

## 7. SUMMARY AND LESSIONS LEARNED

For six sessions over five years we operated a successful summer program to train students, primarily in Earth and environmental sciences, in programming skills, scientific visualization, software design, and high-performance computing. The greatest challenge was to provide a good program for students with widely varying backgrounds, skills, and expectations. We accommodated the diversity by increasing hands-on exercises interspersed with lectures and by expanding our bank of programming exercises to span a range of abilities. We also encouraged students to work more in small groups rather than individually.

The most important conclusion we have drawn is that it is possible to provide students a "crash course" in programming that enables even beginners to handle research-level scientific programming tasks, and for more advanced students to produce near professional-quality software. From our experience, a one- to three-day targeted course seems to well serve the needs of researchers and research students. They rarely have the time or opportunity to take for-credit courses in programming, yet self-teaching or the very short (less than a day) workshops frequently offered often do not adequately prepare them for real-world programming. Many of our students have also expressed dissatisfaction with well-known online courses and found they achieved more when assistance was available, even if they were responsible for most of the material on their own. Over the course of the programs we also moved to include more team-based programming projects, more basic to intermediate projects so that beginners can progress more smoothly without too large a jump in difficulty, and greater integration of hands-on exercises and projects within the "lectures" rather than having dedicated lecture sessions each day. These changes considerably enhanced student satisfaction and assimilation. One clear pattern emerged, however, and that is the importance of the programming exercises, ideally including the more complex "homework" problems. For those, the availability of expert assistance is extremely helpful, and we would advise other groups wishing to undertake a similar experiment to prioritize face-to-face assistance.

## 8. ACKNOWLEDGMENTS

We thank the education officers and intern coordinators at the NASA centers that have hosted our students, especially Blanche Meeson and Mablelene Burrell of Goddard Space Flight Center in Greenbelt, Maryland, who worked to recruit mentors for our students and to make sure they had an enriching internship experience. We are grateful to the many NASA scientists who mentored our students. We also thank Professors Andrew Grimshaw and Aaron Bloomfield for developing and delivering the original material for the "HPC Bootcamp." This work was supported by NASA grants NNX16AB18G and NNX12AP99G.

## 9. REFERENCES

[1] Bergin, Susan, and Reilly, Ronan 2005. Programming: factors that influence success. *ACM SIGCSE Bull.* 37, 1, 411-415. DOI=http://dx.doi.org/10.1145/1047124.1047480.

[2] Wilson, Brenda 2002. A study of factors promoting success in computer science including gender differences. *Computer Science Education* 12,1-2 , 141-164.
DOI=http://dx.doi.org/10.1076/csed.12.1.141.8211

[3] Liu, Ou Lydia, Bridgeman, Brent, and Adler, Rachel M. 2002. Measuring learning outcomes in higher education: motivation matters. *Educational Researcher* 41, 9, 352-362.
DOI=https://doi.org/10.3102/0013189X12459679

[4] Guo, Philip 2014. Python is now the most popular introductory teaching language at top US universities. [Online] Available: https://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/fulltext. [Accessed 26 July 2019]

[5] The Astropy Project. 2016. [Online] Available: http://www.astropy.org/ [Accessed 26 July 2019]

[6] Whitaker, Jeffrey. 2011. Basemap. [Online] Available: https://matplotlib.org/basemap [Accessed 26 July 2019]

[7] Hoyer, Stephan, and Hamman, Joe 2017. "xarray: N-D labeled arrays and datasets in Python." Software Sustainability Institute: *Journal of Open Research Software* 5, 1, 10, 2017. DOI: http://doi.org/10.5334/jors.148

[8] Theano Development Team. 2017. [Online] Available: http://deeplearning.net/software/theano/ [Accessed 26 July 2019]

[9] Abadi, Martin et al. 2015. *Tensorflow: large-scale machine learning on heterogeneous distributed systems*. Preliminary White Paper. Google Research. [Online] Available:

https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45166.pdf [Accessed 26 July 2019]

[10] Python Software Foundation 2008. PEP 373 – Python 2.7 release schedule. [Online] Available: https://www.python.org/dev/peps/pep-0373/ [Accessed 26 July 2019]

[11] Met Office (UK). 2017. Cartopy. [Online] Available https://scitools.org.uk/cartopy/docs/latest [Accessed 26 July 2019]

[12] Python Software Foundation. 2017. [Online] Available https://docs.python.org/3/howto/pyporting.html [Accessed 26 July 2019]

[13] Pierre Raybaut. 2017. Spyder: The Scientific Python Development Environment [Online] Available: https://spyder-ide.org [Accessed 26 July 2019]

[14] Anaconda, Inc. , 2019. The Anaconda Distribution [Online] Available: https://docs.anaconda.com/anaconda [Accessed 16-Aug-2017]

[15] Anaconda, Inc. 2017. "Conda." [Online] Available: https://docs.conda.io/projects/conda/en/latest/ [Accessed 26 July 2019]

[16] Anaconda, Inc. 2017. "Managing Packages." [Online] Available:. https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-pkgs.html [Accessed 26 July 2019]

[17] OpenACC-standard.org. 2015. "OpenACC programming and best practices guide." [Online] Available: https://www.openacc.org/sites/default/files/inlinefiles/OpenACC_Programming_Guide_0.pdf [Accessed 26 July 2019]

[18] Lu, Xiaoyi, Shankar, Dipti, Gugnami, Shashank, and Panda, Dhabaleswar K 2016. High-performance design of Apache Spark with RDMA and its benefits on various workloads. *IEEE International Conference on Big Data (Big Data)*, 253-262. DOI=10.1109/BigData.2016.7840611.

[19] Dalcin, Lisandro. 2019. "MPI for Python." [Online] Available: http://mpi4py.readthedocs.io/en/stable/ [Accessed 26 July 2019]

[20] Reid, John, and Numrich, Robert W. 2007. Co-arrays in the Next Fortran Standard. *Scientific Programming*, 15, 1, 9-26.

DOI=10.1155/2007/954503

[21] Fanfarillo, Alessandro, Burnus, Tobias, Cardellini, Valeria, Filippone, Salvatore, Nagle, Dan, and Rouson, Damian 2014. OpenCoarrays: Open-source Transport Layers Supporting Coarray Fortran Compilers. In *Proceedings of the 8th International Conference on Partitioned Global Address Space Programming Models* (PGAS '14). ACM, New York, NY, USA, Article 4 , 11 pages.

DOI=http://dx.doi.org/10.1145/2676870.2676876

[22] Zhen, Yili, Kamil, Amir, Driscoll, Michael B., Shan, Hongzhang, and Yelick, Katherine 2014. UPC++: A PGAS extension for C++, *IEEE 28th International Parallel and Distributed Processing Symposium*, pp. 1105-1114.

# Northeast Cyberteam Program – A Workforce Development Strategy for Research Computing

John Goodhue
MGHPCC
Holyoke, MA
jtgoodhue@mghpcc.org

Julie Ma
MGHPCC
Holyoke, MA
jma@mghpcc.org

Adrian Del Maestro
University of Vermont
Burlington, VT
adrian.delmaestro@uvm.edu

Sia Najafi
Worcester Polytechnic Institute
Worcester, MA
snajafi@wpi.edu

Bruce Segee
University of Maine
Orono, ME
segee@maine.edu

Scott Valcourt
University of New Hampshire
Durham,NH
scott.valcourt@unh.edu

Ralph Zottola
University of Alabama
Tuscaloosa, AL
rzottola@uab.edu

## ABSTRACT

Cyberinfrastructure is as important for research in the 21st century as test tubes and microscopes were in the 20th century. Familiarity with and effective use of cyberinfrastructure at small and mid-sized institutions is essential if their faculty and students are to remain competitive.

The Northeast Cyberteam Program is a 3-year NSF-funded regional initiative to increase effective use of cyberinfrastructure by researchers and educators at small and mid-sized institutions in northern New England by making it easier to obtain support from Research Computing Facilitators.

Research Computing Facilitators combine technical knowledge and strong interpersonal skills with a service mindset, and use their connections with cyberinfrastructure providers to ensure that researchers and educators have access to the best available resources. It is widely recognized that Research Computing Facilitators are critical to successful utilization of cyberinfrastructure, but in very short supply. The Northeast Cyberteam aims to build a pool of Research Computing Facilitators in the region and a process to share them across institutional boundaries. Concurrently, we are providing experiential learning opportunities for students interested in becoming Research Computing Facilitators, and developing a self-service learning toolkit to provide timely access to information when it is needed.

## Keywords

workforce development, research computing facilitator, project portal, Ask.CI, MGHPCC, Northeast Cyberteam

## 1. INTRODUCTION

The Northeast Cyberteam Program is a National Science Foundation (NSF)-funded initiative to increase effective use of cyberinfrastructure by researchers and educators at small and mid-sized institutions in Northern New England (Maine, Massachusetts, New Hampshire, Vermont). The program combines direct assistance to computationally-intensive research projects; experiential learning opportunities that pair experienced mentors with students interested in research computing facilitation; sharing of resources and knowledge across large and small institutions; and tools that enable efficient oversight and possible replication of these ideas in other regions.

## 2. STRATEGY AND METHODS

The core of our strategy is to build a regional pool of research computing facilitators (RCFs) and a process to share them across institutional boundaries, augmented by knowledge sharing and self-service learning tools that increase the effectiveness of Research Computing Facilitators. To encourage the face-to-face communication necessary for effective mentoring and cross


DOI: https://doi.org/10.22369/issn.2153-4136/11/1/2

institution resource sharing, we have maintained a regional focus, with oversight from anchor institutions in each participating state. For efficiency, and to open the possibility of replicating these ideas in other regions, we have developed a portal for management of project workflows.

## 2.1 Building a Regional Pool of Research Computing Facilitators

Research Computing Facilitators combine technical knowledge and strong interpersonal skills with a service mindset, and use their connections with cyberinfrastructure providers to ensure that researchers and educators have access to the best available resources. It is widely recognized that Research Computing Facilitators (RCF) are critical to successful utilization of cyberinfrastructure, but in very short supply[1].

Since most small and mid-sized institutions cannot individually support a research computing department, the Northeast Cyberteam aims to develop a sustainable pool of facilitators who can work across institutions in the region.

The project gains further leverage by partnering with the large research universities in the Massachusetts Green High Performance Computing Center (MGHPCC) consortium, and with national programs such as the Campus Champions.

To deliver direct assistance to research and education projects while giving students experiential learning opportunities, we developed a model where researchers are paired with student facilitators, typically individuals with an affinity for computationally intensive research, but often with little or no domain expertise relevant to the project. Mentors provide subject matter expertise, and guide the project in a direction that will yield results over a 3-6 month period. This gives the student an opportunity to practice facilitation skills, gain some hands on experience with advanced computing resources, and learn a new domain.

This method of exposing a student to a new scientific domain, with a mentor who provides a safety net of subject matter expertise while modeling how facilitation should be provided, expands the student's domain knowledge and ability to apply computing skills in new situations (a common *modus operandi* for Research Computing Facilitators).

By matching students, mentors, and projects across institutional boundaries, the program expands the skill sets available to all participants in the pool, and provides 'bench depth' that makes it easier to manage turnover, handle bursts of activity, and foster communication among peers to accelerate professional growth.

## 2.2 Knowledge Sharing and Self-Service Learning Tools

Providing peer-validated tools to enable self-service learning is a key to our strategy of developing facilitators through experiential learning. We recognize that one of the most fundamental skills of successful facilitators is their ability to quickly learn enough about new domains and applications to then be able to draw parallels with their existing knowledge and help to solve the problem at hand. There is usually not enough time to enroll in a traditional training course or attend a seminar when a new domain or application is encountered. This is especially true of researchers who may face a particular computational roadblock in their pursuit of a result.

The Cyberteam Portal is used to access the self-service learning resources developed to provide just in time information delivery to participants as they embark on projects in unfamiliar domains. The goal of these learning resources is to reduce the need for direct assistance, and reduce duplication of effort, by adapting and building awareness of available documentation, training, application software, and software utilities, and by supplementing these resources where there are high impact opportunities.

Using a common tagging infrastructure and voting capabilities modeled after crowd-sourced repositories such as StackExchange, we are building a uniform underlying structure. This allows a user to click on a tag from any part of the portal and obtain a listing of all content, including mentor profiles, project profiles, frequently asked questions, and training resources.

The self-service learning section of the portal is designed to accommodate three types of information commonly needed by research computing facilitators:

**1. Frequently-asked questions whose answers evolve over time as technology advances.** We partnered with the Campus Champions and research computing groups at large and small institutions to develop Ask.CI (https://ask.ci), a collaborative, crowd-sourced Q&A site specifically curated for the research computing community. Principal goals for the site are to: 1) reduce RCF workload at institutions of all sizes by pooling questions and answers on an open, searchable, archived site, and 2) make Q&A content available to smaller institutions that do not have the resources to maintain their own internal repositories. We address the evolution of answers over time by including a voting mechanism that allows users to indicate the "best" answer to a question, which might change as new information emerges.

**2. Relatively static information such as introductory training modules on Linux clusters, programming languages and schedulers.** We are developing a resource repository designed to help facilitators come up to speed on particular topics when needed by providing pointers to publicly available, relevant, and vetted training resources. The modules that we are collecting are self-paced, and clearly defined, requiring varying levels of expertise.

**3. Dynamic, situation specific information needed to solve an immediate problem, typically handled by a Help Desk at larger institutions.** We are piloting a Regional Help Desk that is accessible via the portal. Any user in the region can submit a ticket that is then handled by Northeast Cyberteam participants.

## 2.3 Regional Focus

National scale initiatives are an important starting point, but cannot efficiently reach thousands of smaller institutions. On the other hand, expecting every small and mid-sized institution to develop advanced computing capacity on its own invites unsustainable cost and duplication of effort. The Northeast Cyberteam strategy is based on the premise that larger institutions

[1] [1] Gregory E. Monaco, Gwendolyn Huntoon, David Swanson, Donald F. McMullen, Henry Neeman, Jennifer Leasure, Joni Blake, Kate Adams. The Role of Regional Organizations in Improving Access to the National Computational Infrastructure. National Science Foundation, June 2016.

with robust advanced computing resources and experienced facilitators can anchor *regional* efforts to increase the use of cyberinfrastructure and advance science throughout the area.

## 2.4 Oversight

Program direction is set by a Steering Committee that includes leaders from each of the larger institutions that serve as "anchors" for the Northeast Cyberteam Program, in this case, University of Maine, University of New Hampshire, University of Vermont, and MGHPCC. The steering committee also includes a program manager who coordinates day to day activity, and key personnel from other institutions that have provided students and mentors. The Steering Committee as a whole approves all projects undertaken. For selection of projects, the Steering Committee relies less on competitive applications (though merit will naturally play a role), and more on outreach to faculty at smaller institutions who can benefit from access to cyberinfrastructure but are either unaware of available resources or have given up after a poor experience. Care has been taken in sourcing and monitoring projects to ensure that they lead to results that might not otherwise have been achieved, and blaze trails that others can follow.

## 2.5 Program Management Portal

The program relies heavily on the Northeast Cyberteam Portal for management of project workflows, recruitment of mentors and student facilitators, and recording results. The management section of the portal also encapsulates the experience that we are gaining, with the goal of making it possible to replicate the methodology in other regions.

The process for managing a project through its life cycle follows a standard set of steps, all of which are managed via the portal.

1) A Steering Committee member introduces the project, usually planned to be 3-6 months in duration, for approval.

2) If approved, the project is posted on the portal and Steering Committee members collaborate to recruit a mentor and a student RCF. The student and mentor both register on the Portal and become members of the Northeast Cyberteam. Individuals can also register on the portal in advance of a project assignment and become part of the Cyberteam pool that are considered first when new projects are recruiting.

3) The student RCF executes the project with support from the mentor, reporting on progress at monthly Cyberteam videoconference meetings.

4) At the end of the project, the Cyberteam Program Leader conducts exit interviews and the Steering Committee reviews lessons learned.

# 3. RESULTS/LESSONS LEARNED

We have launched 28 projects over the past two years, most of them lasting 3-6 months, and many of them supporting generation of publishable results. We are also beginning to see impact beyond the individual project level, with some smaller institutions starting to treat research computing as an ordinary part of the research and education toolkit instead of a distant luxury item.
Although there is still much to do, we have enough experience to draw some preliminary conclusions.

**1. Value of Research Computing Facilitators to research and education at small and mid-sized institutions:** Consistent with the findings of the report that inspired the Northeast Cyberteam Program[1], the number of research projects that can benefit from Research Computing Facilitators is limited only by our ability to find and recruit them, which is improving over time.

Based on feedback from exit interviews, we are starting to think more systematically about how to assess project readiness. We have seen a spectrum of readiness levels - at one end there are faculty who have a clear idea about what they need to get to a new level of sophistication, while at the other end there are faculty who need help but are unable to engage productively. Over time, we expect to develop an explicit set of readiness criteria, and will gain more experience on how to respond when a project is not yet ready.

**2. Ability of finite-length student projects to fill the need:** Overall, we have been impressed by the quality and responsiveness of the students who have participated in the program. Interestingly, we have had success with grade levels ranging from sophomore to post-doctoral. We have almost always been able to structure an assignment that moves the project from one reasonably well-defined state to another. Examples include (1) moving from a workstation to a cluster for greater throughput; (2) improving the performance or throughput of a workflow in order to generate results with faster turnaround or in greater volume; and (3) adopting a new computing tool such as Jupyter notebooks.

**3. Willingness of mentors to participate:** Experience over the past two years has validated our hypothesis that experienced Research Computing Facilitators would be willing to serve as mentors as part of their regular jobs. The opportunity to evaluate potential new hires is a practical motivator, but it also helps that people who become RCFs generally enjoy teaching others, and that teaching is central to the culture of academic institutions.

**4. Ability to apply students and mentors across institutional boundaries:** This aspect of the program has been critical to success. We are pleased that two initial concerns have not been significant impediments. Our first concern was distance – while occasional face-to-face meetings are possible (and necessary), most work must be done remotely, even if the student is separated from a project by just a few miles. We have found that tools for collaboration, such as high quality desktop videoconferencing, shared document repositories, and flexible source control systems, are sufficient to maintain communication and trust when combined with face-to-face contact. The second concern was administrative, as grant administrators understandably lean toward applying funds in ways that benefit students and faculty at their home institutions. While every co-PI has needed to spend some extra effort explaining the purpose and benefits of the program, this has not delayed or prevented cross-institution assignments.

**5. Willingness of larger institutions to share information:** The Ask.CI project has received considerable support from Research Computing groups at larger institutions, both for the initial idea of building a shared Q&A list, and the more recent idea of "sandboxes" that expose internal Q&A lists outside their home institutions. In a similar vein, the regional help desk and the

training information repository have benefitted from contributions by research computing groups at larger institutions.

**6. Importance of active program management:** The second largest expense category for the project (after student support) is support for a project lead at each Anchor Institution and the Program Manager who manages the overall program. While the value of program management is often overlooked, this investment has been critical to success. It has enabled several important outcomes, including: (1) efficient recruiting of projects, students and mentors; (2) development of process, tools, and strategy; (3) effective communication across the anchor institutions; and (4) the ability to explain the purpose and benefits of the program to grant administrators who have expressed initial skepticism about supporting this kind of collaboration across institutions. We have gained some recruiting momentum, and developed processes and tools that will reduce the need for active management and coordination. However, it seems likely that at least some active management will be required for ongoing success.

## 4. REPRODUCIBILITY

The Northeast Cyberteam Program has been underway for just over two years. It took some time for our steering committee to get into a regular rhythm of meeting times, project submissions and approvals, but we now have a reasonably well-established system that is delivering on the goals of moving science forward while giving potential student facilitators real world experiential training in the field of research computing.

All of the tools that we have developed, including the Portal, Ask.CI Q&A site, Regional Help Desk, and Training Resources Wiki, have been designed with an eye towards reproducibility/expansion. Even the logo was designed to be easily adapted to other geographic regions or domains.

## 5. Northeast Cyberteam and SEHET

Our goal in participating in the SEHET19 workshop is to find opportunities to collaborate with other groups focused on workforce development for the Research Computing community. Collaboration can take many forms, beginning with small steps such as posting a topic on Ask.CI or adding links to our Training Resources Wiki. A more ambitious collaboration would involve launching cyberteams in other areas of the country, anchored by a large institution (or group of institutions) where advanced research computing is a priority, and outreach to the surrounding institutions is encouraged. Leveraging the Northeast Cyberteam model and tools will allow researchers at surrounding smaller institutions to take advantage of cyberinfrastructure when their work requires it. Simultaneously, it will expose a new generation of potential facilitators to this exciting and dynamic field earlier in their careers, significantly expanding the available pool of candidates.

## 6. ACKNOWLEDGEMENTS

This work was partially supported by the National Science Foundation under grant award ACI-1659377. The authors thank the many Cyberteam mentors and student facilitators for their participation in this effort, the Ask.CI moderators whose tireless efforts to build and maintain the site are beginning to yield significant results, and the hundreds of contributors who have generously shared knowledge and experience on Ask.CI.

# Incorporating Complexity in Computing Camps for High School Students – A Report on the Summer Computing Academy Program at Texas A&M University

Dhruva K. Chakravorty
High Performance Research Computing
Texas A&M University
College Station, TX, 77843
chakravorty@tamu.edu

Marinus "Maikel" Pennings
High Performance Research Computing
Texas A&M University
College Station, TX 77843
pennings@tamu.edu

Honggao Liu
High Performance Research Computing
Texas A&M University
College Station, TX 77843
honggao@tamu.edu

Xien Thomas
High Performance Research Computing
Texas A&M University
College Station, TX 77843
xien.thomas@tamu.edu

Dylan Rodriguez
High Performance Research Computing
Texas A&M University
College Station, TX 77843
dylan@tamu.edu

Lisa M. Perez
High Performance Research Computing
Texas A&M University
College Station, TX 77843
perez@tamu.edu

## ABSTRACT

Summer computing camps for high school students are rapidly becoming a staple at High Performance Computing (HPC) centers and Computer Science departments around the country. Developing complexity in education in these camps remains a challenge. Here, we present a report about the implementation of such a program. The Summer Computing Academy (SCA) at is a weeklong cybertraining[1] program offered to high school students by High Performance Research Computing (HPRC) at Texas A&M University (Texas A&M; TAMU). The Summer Computing Academy effectively uses cloud computing paradigms, artificial intelligence technologies coupled with Raspberry Pi micro-controllers and sensors to demonstrate "computational thinking". The program is steeped in well-reviewed pedagogy; the refinement of the educational methods based on constant assessment is a critical factor that has contributed to its success. The hands-on exercises included in the program have received rave reviews from parents and students alike. The camp program is financially self-sufficient and has successfully broadened participation of underrepresented groups in computing by including diverse groups of students. Modules from the SCA program may be implemented at other institutions with relative ease and promote cybertraining efforts nationwide.

## CCS CONCEPTS

•**CS**→Computer Science; •**Cybertraining**→training on using cyberinfrastructure; •**HPC**→high performance computing

## Keywords

HPC training, summer camps, broadening participation, assessment strategies, best practices. diversity, high school students, computational thinking, artificial intelligence

## 1. INTRODUCTION

The prevalence of computing in everyday life has led to an extensive interest in computing in general and high performance computing in particular. While several efforts cater to the needs of graduate students and professionals, there is a significant drop-off in computing training at the high school level. Despite the Computer Science for All initiative, GenCyber, and CyberPatriot, issues such as insufficient access to computing resources and a lack of proficient trainers have resulted in low computer science adoption at the high school level.**[1]** Indeed, a recent report on the state of K-12 CS curricula in Texas, entitled "Building the Texas Computer Science Pipeline" discussed


DOI: https://doi.org/10.22369/issn.2153-4136/11/1/3

the effects of the CS Advanced Placement exam and the state educational guidelines for CS course **[2]**. The report found significant deficiencies in CS education and made a series of recommendations to prepare students for future careers in computer science (CS) related fields. Recommendations included developing parent demand for CS courses, and the advanced placement CS course, by including additional engaging, project-based courses. Finally, the report emphasized the need to inform students, teachers, and administrators to careers in CS by connecting them to CS experts and practitioners. In addition, the prevalent threats to personal information, prevalence of cyber-bullying, the increasing need for confidentiality also make it essential that students learn aspects of the internet, digital citizenship, and cybersecurity at an early age.

The availability of inexpensive devices like the Raspberry Pi significantly lower the entry price point for computing-based education. Use of accessible physical assets, such as cloud computing resources, easy availability of artificial intelligence technologies, and Raspberry Pi clusters coupled with robotics and visualization technologies give educators opportunities to engage learners in complex or problematic scenarios. In this paper, we report on advancements made by the TAMU HPRC Summer Computing Academy program. This paper is presented with a view to help HPC units and departments of Computer Science and Engineering that are either currently hosting similar programs or are interested in developing similar programs. This paper is organized into the following sections. We first describe the SCA program. followed by best practices that have been adopted in the Summer Computing Academy model. We next describe our recruitment strategies and selection criteria for participants followed by our cybertraining model. The following sections provide an overview of how to adopt emerging technologies and a description of our assessment model. The paper next describes legal and administrative issues encountered while hosting such efforts and concludes with a discussion of the sustainability of the effort.

## 2. HISTORY OF THE SCA

The SCA seeks to enable aspiring computer scientists, developers, and engineers in their pursuit of computing fields by offering weeklong cybertraining programs. The goal of this effort is to help usher the next generation of cyber-practitioners in the country. The SCA introduces high school students to various aspects of computational thinking by employing hands-on exercises and active learning activities using Raspberry Pi microcontrollers and sensors [3-10]. The camps are further designed to introduce high school students to concepts in cyber security and promote safe cyber behavior as well. The program is steeped in well-reviewed pedagogy; the refinement of the educational methods based on constant assessment is a critical factor that has contributed to its success. The camp format and has received rave reviews from parents and students alike. In its recent evaluation, Campus programs for Minors (CPM) for the Texas A&M System has designated the SCA program as a “Model Camp”. Since its inception in 2017, the SCA has offered an introductory camp that is geared for beginners who have not benefited from an opportunity to learn about computing. Beginning in 2018, the SCA also offered an intermediate camp that is geared toward students who have had some exposure to programming in school and are keen to learn about how computing integrates with STEM disciplines. In 2019, the SCA offered camps in Cybersecurity and Artificial Intelligence that were funded by the GenCyber and Governor’s Summer Merit Program of the Texas Workforce Commission. In 2019, on completion of these camps, high students will be able to demonstrate the use of algorithms and loops to a class and explain fundamental principles of cybersecurity and artificial intelligence. They will be able to write code in Scratch or Python that uses variables within algorithmic thinking and loops. As such, the objectives of the SCA program are to:

- Use research-based methods to develop a high-impact, high-immersion opportunity that introduces students to concepts in computing including software, hardware, networking, cybersecurity and data-management practices.
- Engage students in cybertraining skills using hands-on approaches utilizing high-technology and low-technology avenues.
- Reinforce and develop further knowledge of cyber skill sets through hands-on exercises.
- Introduce concepts of cybersecurity and artificial intelligence
- Retain student interest after the camp by offering access to series of free in-person and online cybertraining-themed short courses and seminars organized by Texas A&M HPRC.

Diversity is a core tenet of the SCA program. The 2017 SCA cohort had an even distribution of male and female participants and included a significant number of students from groups that are traditionally underrepresented in computing. In 2018, a total of 55 high-school students are targeted for participation in this high-immersion program. In 2019, a total of 145 high school students will participate in this program. A significant proportion of these attendees will be females and other underrepresented minorities in computing.

## 3. BEST PRACTICES

Working with minors presents a unique set of administrative and educational challenges. This is particularly challenging while working with often overlooked groups such as Foster and home-schooled children. Planning a schedule that includes presentations, capstone assessments, and evaluations, a process involving

numerous faculty, staff, and classroom instructional materials, while maintaining the legal and administrative requirements of the university is a difficult challenge. Here we present a list of best-practices that were adopted by the SCA program.

## 3.1 Pedagogy

Proficiency in computer science widely differs across students. One goal of the registration form is to identify high-achieving or highly knowledgeable students who might need additional frameworks or scaffolds of instruction to be available. To ensure that the learning outcomes of the SCA program are met, we use the information from the student's registration packets and pre-camp package to ensure that each camp has students who are at a similar level of computing proficiency. In addition to asking students to self-evaluate themselves, in 2019, the SCA requested letters of recommendation from teachers. Owing to sponsor requirements, the SCA also collected transcripts (or year-end report cards) and birth-certificates for the first time.

The conceptual framework implemented for training incorporates elements of active learning, such as exploratory learning via research projects, where mentors provide guidance to help focus mentees activities in productive directions and group discussions in research seminars. Active learning has been shown among high-ability trainees to produce significantly higher levels of metacognitive activity than procedural training, leading to the development of higher adaptive transfer. In addition, the training

provided through the introductory camp incorporates several elements of the experiential learning cycle in which:

A. Students are introduced to several aspects of cyber security. (New experiences)

B. Exercises encourage students to integrate and apply previously developed to specific problems. (Critical thinking)

C. Students must select and apply skills from their repertoire to problems by developing hypotheses and validating them. (Hands-on experimentation)

## 3.2 Advancing Knowledge and Understanding in Computing [3-10]

Computing is a constantly evolving landscape. Disruptive computing technologies result in new threats that appear on extremely short timescales. Therefore, computing education must prepare learners to stay abreast of both current and emerging technologies along with effective responses. To develop desired capabilities, the SCA program focuses on the described attributes:

(a) Defining desired capabilities – Effective CS education begins with iteratively refining desired learning outcomes at the Analyzing, Evaluating, and Creating levels of the Revised Bloom's Taxonomy. The SCA defined learning outcomes will provide the foundation for the next three phases.

(b) Operationalizing learning outcomes – While SCA learning outcomes articulate expectations for learner achievement, we also develop assessment activities to provide opportunities for learners to demonstrate achievement.

(c) Evaluating learner development – Once learners have completed an assessment activity, their learning must be evaluated both formatively and summatively. Consistent and detailed evaluation information is facilitated through development of scoring schemes, also known as rubrics. Explicit descriptions of levels of achievement will help learners understand expectations for their performance as well as help SCA team members provide helpful feedback to the students.

(d) Facilitating learning – With the foundation of learning outcomes, assessment activities, and scoring schemes, the project team can develop learning activities using research-based instructional approaches. An emphasis on cooperative teamwork and active engagement will be the basis of the camp's learning activities

## 3.3 Security and Child Protection Measures

The SCA program is guided by TAMU CPM guidelines. These include providing camp counselors at a ratio of 1 counselor per 12 campers. All counselors and instructors will complete Child Protection training and undergo Background Checks. Campers will be monitored and accompanied by counselors while on campus at all times. There will also be personnel trained in first-aid and CPR. Campers and counselors wear identifiable SCA camp specific T-shirts and badges. These badges include information about the camp's location, emergency camp contact and the student participant's emergency contact. As part of the camp's accreditation process at TAMU, the camp makes accommodations for students with disabilities, and all camp locations are accessible to students with disabilities.

## 3.4 Administrative Best Practices

TAMU staff and faculty form the SCA team are charged with delivery of the camps. In compliance with TAMU Campus Programs for Minors guidelines, there will be a minimum of three instructors present at any time for the twenty-five children. At all times, at least one male and one female camp supervisor will be present. All instructors have previously been K-12 instructors and/or have previously instructed K-12 summer camps. Preparation for the SCA begins three months prior to the event. All faculty

and volunteers meet on a bi-weekly basis to plan the curriculum, schedule, and the materials for the program. Based on program assessments and participant reviews, a significant number of materials were adopted from the 2017 SCA offering. These meetings help all parties review compliance and regulatory issues as well. All SCA teaching material and requisite forms, are posted online and will be available to camp participants a week prior to the camp. Typical days begin at 8:00 AM and end at 4:30 PM. Each day will end with a de-brief so that instructors may reflect about the day's activities and coordinate events for the next day. The program provides a final session on each day will end with a brief recap. On Friday, the cohort is dismissed at 3:30 PM following an informal reception during which certificates are awarded. Parents and guardians are invited to the reception and get to talk with the instructors.

## 4. RECRUITMENT AND SELECTION

Student recruitment is a critical aspect of a program's success. In order to effectively recruit youths who have not been exposed to computing but yield high promise, we developed a novel recruitment strategy. Rather than focusing on students working with computer science teachers and computing clubs, we found participants from language clubs, food clubs, dance groups, and online gaming communities. A high degree of academic performance and intrinsic motivation were, however, a must. The diverse participant body at our camp, and long waitlists amply demonstrate the effectiveness of our recruiting program. The SCA program is advertised online and interested attendees fill out an online application, which includes a statement of interest and a letter of recommendation. Outreach efforts are made through existing contacts with a number of Independent School Districts in the Houston, Dallas, Waco, Austin and San Antonio regions. The effectiveness of our recruiting system is best demonstrated by applications coming in from across the Southern seaboard (Florida to California). In 2018, both SCA camps are significantly oversubscribed. We started pre-registering for the 2019 Summer Computing Academy in 2018 and are now on track to offer 145 scholarship positions spread out over 4 camps. For our merit camps, the following criteria are currently used to evaluate applications:

- The participant's academic achievement (Target: high)
- Impact statement describing the student's interest in computing. (Target: high enthusiasm)
- Program participant desired ratio (Target: high)
- Ratio of students belonging to an underrepresented group in computing (Target: high)
- Ratio of male to female participants (Target: balanced)

2019 represents a departure from our merit-based approach. As such, the criteria used to evaluate applications were:

- The participant's academic achievement (Target: adequate)
- Impact statement describing the student's interest in computing. (Target: curious about computing)
- Ratio of students belonging to an underrepresented group in computing (Target: high)
- Ratio of male to female participants (Target: balanced)

Information about the applicant's previous exposure to computing is collected during the camp application process via the application form and mentor's recommendation letters. This information is used to judge the best possible camp for every participant.

The SCA program is particularly successful at addressing challenges in broadening access and adoption of cybersecurity skills to the nation's scientific and engineering workforce by ***(a)*** Producing course material that utilizes established pedagogical methods such as research project-based learning to prepare a community of students; ***(b)*** Developing and disseminating online modules for continuing and remote education; ***(c)*** Leveraging existing collaborations to recruit participants from groups that are traditionally underrepresented in the STEM and computing fields. As a measure of success in this capacity, the 2017 SCA boasted of incredible diversity in terms of socioeconomic classes of camp participants, female: male ratio of participants (11:11) and instructors (5:5) and richness of computing experience. These students self-identified as being Hispanic, African American, Asian, Mixed-race and/or Caucasian.

## 5. CYBERTRAINING FOR FUTURE STEM PROFESSIONALS

The SCA program is designed on the principles of engagement, training, retention, and sustainability to promote the CI professional career path. The SCA is unique as it adopts a novel instructional approach that was developed to provide a holistic view of the computing landscape in STEM rather than merely impart programming skills. With a view toward broadening the learning and understanding of students through further diversification of learning approaches, we designed the educational process using the backward design approach. We first identified the learning objectives and competencies that participants were expected to learn and built each exercise around them. Modules are developed using a philosophy of "deliberate practice", i.e. practice with essential feedback. We employ a hybrid method of instruction that relied on the principles of guided discovery via observation and hands-on based laboratory-type learning. The students will develop new skill-sets through this high-impact learning community experience. Students will be taught a grounds-up approach towards cybersecurity on Raspberry Pi computers using sensors and LED devices.

During in-class lectures, students are introduced to the First Principles via lectures, programming, and games. An emphasis is placed on data hygiene, safe cyber behavior, protection from cyber pornography and cyber bullying; and perhaps most importantly cyber ethics. The idea that cyber activities could have deleterious consequences are reinforced. Since students are likely to gravitate towards exercises that are attached to something that they can see or touch, computing exercises are based on sensors that they were able to see, hear, and otherwise interact with. We encourage students to debug by introducing features in activities that would cause them to fail. Over the course of the program, these exercises introduced students to various aspects of computing while simultaneously encouraging computational thought. To encourage student participation while ensuring that we successfully covered the material, we include structured and unstructured components to each session. Instruction sessions begin with a three to five-minute topic introduction that employs real-world issues to begin discussions. Connections between the real-world issue and the class topic are pointed out during this time. Students perform activities with brief pauses to answer questions on the topic. After the first half of the session, students will be allowed to work on activities at their own pace, allowing students with advanced skills to move to more challenging exercises. The instructor and assistants are available to help students as they worked through these exercises. To foster networking and build a student community, participants who successfully completed an exercise are encouraged to help others. Students are taught to use the Github repository for code. The instructor will facilitate progress but do not lecture or direct progress. Finally, the instructor, emphasizing the relationship between the key concepts and the relevant real-world examples, would lead a brief recap discussion that was followed by a capstone exercise. To ensure that students receive a different view of topics, review sessions will be taught by a different instructor. Moreover, the review sessions also integrate concepts covered in previous sessions. Each session is guided by how the students participated in previous review sessions. Concepts covered in the classroom will be reinforced by demonstrations at facilities that underscore the importance of the taught material in real-world scenarios.

In-class exercises and instruction are backed up by visits to on-campus facilities such as the “Teague Super Computing Data Center” that houses the Ada and Terra compute clusters and the state-of-the-art “Engineering and Innovation Center” fabrication laboratory. These site visits to facilities allow students to experience cybersecurity principles applied to various aspects of day-to-day computing.

## 6. OPTING EMERGING TECHNOLOGIES

The rapidly changing nature of the cybersecurity landscape underscores the need for young adults to be prepared to identify and mitigate new threats in their daily lives. The SCA program offers a blend of cybersecurity and visualization technologies to students in an innovative learning environment. Modeling and visualization are participatory technologies that provide the means to achieve engagement while legitimizing the role of computing in scientific discovery and research. Participating students will use our newly developed NSF-funded CiSE-ProS learning modules and virtual reality interfaces. Students will learn about multiple aspects of cybersecurity in both of our camps. Of particular note are the exercises on the last day of the Introductory SCA camp. Students play the well-reviewed “capture the flag” exercise on a Raspberry Pi equipped with a LED display board, called a pi-hat. In this exercise, each student is assigned a specific color that is displayed on their pi-hat. Students can then choose to “defend” their Raspberry Pi and ensure that the pi-hat keeps the same color. Else, they could “capture” someone else’s Pi by changing the captured Pi hat’s colors to show their color. This exercise allows students to experiment with defensive strategies, probe others for vulnerabilities in their defensive structure and puts all that they learned during the camp to test. Students can form groups to capture a pi and change the color on the raspberry pi-hat display to display their color. Previously, students have worked as a group, by running a password guessing program in parallel on each other’s machines, while others have employed similar maneuvers to constantly change their network settings. This exercise is much enjoyed; it is followed by a reminder about ethical behavior and a how to be a good cyber citizen. In 2018, we we particularly excited about introducing students to exercises using AI, cryptography and the TAMU virtual reality simulator that emphasizes cybersecurity principles from a hardware perspective. Students had the opportunity to freely explore a data center in a virtual reality environment with haptic controls. Students are free to “walk” in the VR data center and understand the various aspects of data center security as well. The exercise directs to replace a compromised node for a new node. This exercise was followed by a programmatic evaluation survey that showed that students were comfortable with the use of VR technologies, understood key concepts of cybersecurity and appreciated the physical interactions in the data center room. A prototype of the CiSE-ProS VR simulator was demonstrated at the TAMU HPRC booth SuperComputing 17 Conference in Denver, CO, and again at the TAMU HPRC booth SuperComputing 18 Conference in Dallas, TX.

## 7. ASSESSMENTS AND EVALUATIONS

The SCA has received highly positive reviews from students and parents alike. Post-camp surveys and assessments reported 100% improved attitudes to computing and STEM and computing. After attending the SCA, 30% of female (Hispanic and African American) participants said that they were more likely to major in Computer Sciences! A number of students from the 2017 SCA have gone on to apply to the Computer Science program at TAMU and STEM programs at other universities. As further evidence of success, a significant number of students from the 2017 SCA cohort who have yet to graduate from high school will be attending the 2018 SCA Intermediate camp.

Prior to engagement with the academy, potential participants complete an application which is designed to acquire a sense of each applicant's predilection towards computer science topics. The application along with accompanying letters of reference from teachers help guide the staff's understanding of each participant's interest and experience with microcomputers and programming. The following questions are examples of the types of questions used for the pre-camp evaluation.

***(i)*** How familiar are you with the Raspberry Pi?
***(ii)*** How familiar are you with Python programming?
***(iii)*** Rate your prior level of coding experience.
***(iv)*** Please describe how this program will help achieve your personal and academic goals.

After the conclusion of the camp, participants are given the opportunity to evaluate their experience. This evaluation is conducted in the form of several questions which aim to convey the participant's expectations as well as whether or not the computing academy met said expectations. The following questions are a select few of the more than twenty questions used in the evaluation.

***(i)*** Are you planning to pursue a degree in a STEM field? If you answered yes, did attending the camp have any influence on that decision?
***(ii)*** Did you use any of the skills gained from the computing academy (for example started programming as a hobby, experimenting with your Raspberry Pi, used it in school projects etc.?
***(iii)*** Did you take any computer related courses in school before the summer camp and did you take any extra computer related courses after attending the camp and if so, did attending the camp have any influence on it?
***(iv)*** Did you participate in any other STEM-related extracurriculars? (e.g other camps) Are you still using your Raspberry Pi?

Throughout this experience, we have found that teaching students new computing concepts, such as navigating a Linux environment, using a command line, and writing code, is more productive when done through an interactive format rather than using a lecture format that is interspersed with a few activities. With an interactive format, students are more motivated to follow along with the instructor and other students by participating in the activities. This promotes the practice of a new concept or technique and allows for greater retention of the new information by the student. Capstone exercises found that lectures which did not follow this interactive format were much less successful than their interactive counterparts. Non-interactive lectures often left many students behind and afraid to ask questions. This led to students becoming bored and inattentive, causing them to retain little to no information from the lecture. Those lectures that were interactive promoted student engagement with the instructor and the rest of the class. The students were more attentive and willing to ask questions when they did not understand a topic. These lectures also provided ample opportunity for the students to practice the topics they were learning which promoted an understanding of the application of their newfound knowledge and greater retention thereof. While most students were familiar with Google Drive, Git/GitHub adoption in workflows remained a challenge. In a surprise to us, participants were successfully able to perform basic operations using Gedit in a matter of minutes. Students (and parents) were more likely to participate in a distributed (online) training session after an in-person activity as compared to those without the face to face interaction.

In future iterations, we hope to assess student skills and competencies both pre-and post-camp using a SLAG type evaluation scheme. Representative samples of students and their parents will be interviewed. Some of the topics will include their experience in the camp, motivation to pursue careers in STEM and cybersecurity and ways to refine the offerings. These assessments will be used to inform future iterations of camps. The administration of these instruments is designed to be anonymous and the assessment will demonstrate the knowledge acquisition from the camp. Follow up will be through surveys such as the popular STEM Semantics Survey and STEM Interest Survey, which will be administered at the beginning and end of the camps and again 6 months later. Open-ended questions allow campers to explain their perceptions of how the camp experience impacted their interest in cybersecurity and STEM-related careers. In 2019, the SCA started a post-camp survey that was common to the GenCyber program.

## 8. LEGAL AND ADMINISTRATIVE ISSUES

The requirements for hosting the SCA stem from a shared commitment to provide a safe environment and meaningful experience for participants that not only meet the minimum legal requirements but also reflect the Texas A&M University's core values of Excellence, Integrity, Leadership, Loyalty, Respect, and Selfless Service. To

make sure these requirements are met, SCA staff works together with the Campus Program for Minors and the Internal Review Board at Texas A&M University [11]. In 2019, Texas A&M's IRB exempted the SCA. The requirements for compliance are outlined by the Texas A&M University Rule for Campus Programs for Minors [12]. These requirements can be classified into two categories; Program logistics and Program Compliance. It is important to have supervision ratios in place that help ensure the safety of the participants and the quality of the SCA. The American Camps Association **[13]** (ACA) recommends having at least one qualified staff member for every 12 SCA participants. The SCA typically maintains at least one qualified staff member for every 10 participants and ensures that there is always at least one female and one male staff member present at all times. Lunch during the SCA is provided by the nearest campus dining facility to limit exposure to the Texas summer heat and to limit traveling times. Furthermore, campus dining facilities meet the requirements of the University catering policy, CPM requirements, and ADA standards. The SCA will check with participants before the beginning of the program to determine whether there will be any specific dietary needs, restrictions, or requests.

**Table 2: Information, Consent, and Authorization forms required by Campus Programs for Minors at TAMU.**

| | |
|---|---|
| Liability Waiver | informs participants and their guardians about the potential risks associated with the camp as well as waives Texas A&M University of liability in the event of an accident |
| Media Release | informs participants about their release of their likeness and image for use in media to promote future computing academies |
| Code of Conduct | establishes an expectation of behavior to be adhered to by the participants |
| Github Account Consent | informs participants of their use of Github, obtains consent from their guardians to allow their participant to create a Github account for use with version control exercises |
| Participant Pick-up Authorization | informs guardians about pick-up procedure and establishes rules for authorizing participant's daily departure from the academy |
| Medication Disbursement | obtains information regarding the disbursement of medication to students |
| Online Gaming | Permission to play online games |
| Research information | Informs parents that the students may be asked to participate in a research program |

Every staff member and volunteer is required to pass a criminal background check performed by Texas A&M Human Services. In addition, to comply with the Texas Education Code, every staff member must successfully complete a comprehensive child protection training. The Texas A&M University System created an online training course that meets the requirements of the Education Code [14] and has been approved by the Texas Department of State Health Services. SCA also needs to identify a hospital/urgent care facility to refer participants to in the event of an emergency. and SCA will send a letter to the chosen medical facilities letting them know the dates of the program, approximate number of participants, and the policy information for the insurance coverage. the SCA will purchase general liability and accident medical coverage through System Risk Management as required by Texas A&M University System regulations.

Instruction for the Student camp takes place at the state-of-the-art Interdisciplinary Life Sciences Building teaching facility and the state-of-the-art Zachry Engineering Education Complex at Texas A&M University. This training avenue is compliant with ADA standards and ensures that we can support students with specific individual requirements. This unique teaching space is designed to accommodate the facilitation of hands-on training and education providing a flexible workspace with a modular group working environment layout. The room is a flexible space that can seat over 300 students. This facility supports not only classroom instruction, but is ideal for supervised hands-on-exercises as well.

## 9. SUSTAINABILITY AND SCALABILITY

Every aspect of the SCA program was designed with sustainability in mind. Student training in computing is a critical area where demand currently outweighs supply. The camp's format was well received by the community and the camps have been oversubscribed since 2017. Owing to the demand from the community, these camps are inherently sustainable. In 2019, we demonstrated this aspect via funding from the GenCyber and Texas Workforce Commission programs. Lennovo, Dell, Google, and MarkIII were corporate sponsors of the event. In addition to need, we have encouraged continuity by adopting camp training materials in our HPRC training sessions and describing these efforts in our proposals. Indeed, knowledge repositories of training materials are made available free-of-charge to future camp providers. These camps also provide opportunities to work with Assistant Professors who are preparing their portfolio for CAREER or YIP awards. Finally, the key elements to sustaining the educational, organizational and cyber aspects of the effort

include plans to seek external funding and commercializing technologies. While the program remains oversubscribed in 2018 as well, it has a fee-based structure. Federal support for these programs would help eliminate fees and help the camps as well.

The proposed project can be scaled to reach a broader audience in the future. In 2018, the SCA extended its offering to include an Intermediate Camp as well. In 2019, we are offering camps with a concentration on Cybersecurity and Artificial Intelligence. We are currently exploring possibilities of organizing an advanced camp in 2020 and teacher training camps as well.

Schools districts in Texas are currently implementing a computing-based curriculum. **[2-10]** The improvements to the curriculum are part of Texas's education vision that describes the path for teacher student success. Funds budgeted by School Districts for teacher professional development will support these training initiatives as well.

## 10. CONCLUSIONS

Texas A&M University SCA program currently offers two merit-based summer camps designed to promote a hands-on cyberlearning experience for high-school students. The camps 1) enhance participant engagement 2) enhance participant understanding of complex computing concepts using observational experiences, and 3) provide participants with a learning environment that utilizes state of the art technology. In the future, we hope that the camps will follow the spirit of the Stanford Transcription model, where participating students will receive a letter confirming participation, that can accompany their applications to pursue a degree program at TAMU. Despite having participants at different levels of computational proficiency, this novel program maintained a retention rate of 100% during the week. Currently, efforts are underway to deploy scoring schemes and rubrics that were created by the GenCyber program to accompany each assessment activity. Identifying the skills of a registered participant and incorporating them as a class remains a significant challenge for such efforts. Toward this, we will offer a registration quiz for advanced camps to accurately identify the student's proficiency before the start of the program. In future iterations, the SCA program will offer research opportunities to contribute during the product development phase and the design of instructional activities. Finally, a select number of SCA counselors have completed Internal Review Board (IRB) training in order to study and report on student learning outcomes.

All SCA materials are available free-of-charge to the national CI training community at our website hprc.tamu.edu. [15-17] Agendas, registrations forms, sample announcements, templates to track participants, Trello event boards and other such materials will be made available by the authors on request.

## 11. ACKNOWLEDGMENTS

The authors would like to thank staff, student workers and researchers at Texas A&M HPRC, Yang Liu, Michael Dickens, the Laboratory for Molecular Simulation, Dr. Steve Johnson, TAMU Engineering Innovation Center, TAMU IT, TEES IT, TexGen, Division of Research, the Texas Engineering Experiment Station IT, TAMU CPM and TAMU Provost IT for supporting the HPRC SCA program at Texas A&M. Portions of this research were conducted on the Ada and Terra compute clusters provided by TAMU HPRC. We gratefully acknowledge support from the NSF Award OAC #1730695 *"CyberTraining: CIP: CiSE-ProS: Cyberinfrastructure Security Education for Professionals and Students",* and NSF Award OAC # 1925764 "*CC: Cybterteam: South West Expertise in Training Education and Research"*

## 12. REFERENCES

[1] Research on Learning in Formal and Informal Settings, National Science Foundation, URL - https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=505359

[2] Texas Regional Collaboratives, "Building the Texas Computer Science Pipeline Strategic Recommendations for Success | theTRC.org," 2014

[3] A. Yadav, N. Zhou, C. Mayfield, S. Hambrusch, and J. T. Korb, "Introducing computational thinking in education courses," in Proceedings of the 42nd ACM technical symposium on Computer science education, pp. 465–470, ACM, 2011.

[4] J. J. Lu and G. H. Fletcher, "Thinking about computational thinking," in Proceedings of the 40th ACM Technical Symposium on Computer Science Education, SIGCSE '09, (New York, NY, USA), pp. 260–264, ACM, 2009.

[5] K. Brennan and M. Resnick, "New frameworks for studying and assessing the development of computational thinking," in Annual American Educational Research Association meeting, (Vancouver, BC, Canada), 2012

[6] S. Y. Lye and J. H. L. Koh, "Review on teaching and learning of computational thinking through programming: What is next for K-12?," Computers in Human Behavior, vol. 41, pp. 51–61, 2014.

[7] J. M. Wing, "Computational thinking and thinking about computing," Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences, vol. 366, no. 1881, pp. 3717–3725, 2008.

[8] M. Prince, "Does active learning work? a review of the research," Journal of engineering education, vol. 93, no. 3, pp. 223–231, 2004.

[9] H. Eshach, "Bridging in-school and out-of-school learning: Formal, non-formal, and informal education," Journal of science education and technology, vol. 16, no. 2, pp. 171–190, 2007.
[10] J. Parsons and L. Taylor, "Improving student engagement," Current issues in education, vol. 14, no. 1, 2011.
[11] Campus Program for Minors: https://cpm.tamu.edu
[12] Texas A&M Campus Programs for Minors Rules: http://rules-saps.tamu.edu/PDFs/24.01.06.M1.pdf
[13] American Camp Association: https://www.acacamps.org/
[14] Texas Education Code: https://statutes.capitol.texas.gov/Docs/ED/htm/ED.51.htm#51.976
[15] D. K. Chakravorty, M. Pennings, H. Liu, Z. Wei, D. M. Rodriguez, Levi T. Jordan, D. F. McMullen, N. Ghaffari, and S. D. Le. "Effectively Extending Computational Training Using Informal Means at Larger Institutions," Journal of Computational Science Education 2018, 40-47 DOI 10.22369/issn.2153-4136/10/1/7.
[16] D. K. Chakravorty, M. Pennings, H. Liu, Z. Wei, D. M. Rodriguez, L. T. Jordan, D.F. McMullen, N. Ghaffari, S. D. Le, D. Rodriquez, C. Buchanan, and N. Gober. "Evaluating Active Learning Approaches for Teaching Intermediate Programming at an Early Undergraduate Level," Journal of Computational Science Education 2018, 61-66 DOI 10.22369/issn.2153-4136/10/1/10.
[17] D. K. Chakravorty, D. F. McMullen, N. Gober, J. H. Seo, M. Bruner, and A. Payne. "Using Virtual Reality to Enforce Principles of Cybersecurity," Journal of Computational Science Education 2018, 81-87 DOI 10.22369/issn.2153-4136/10/1/13.

# Expanding user communities with HPC Carpentry

Alan Ó Cais*
Jülich Supercomputing Centre
Jülich, Germany
a.ocais@fz-juelich.de

Peter Steinbach
Scionics Computer Innovation GmbH
Dresden, Germany
Max Planck Institute of Molecular Cell Biology and Genetics
Dresden, Germany
steinbach@scionics.de

## ABSTRACT

Adoption of HPC as a research tool and industrial resource is a priority in many countries. The use of data analytics and machine learning approaches in many areas also attracts non-traditional HPC user communities to the hardware capabilities provided by supercomputing facilities. As a result, HPC at all scales is experiencing rapid growth of the demand for training, with much of this at the introductory level.

To address the growth in demand, we need both a scalable and sustainable training model as well as a method to ensure the consistency of the training being offered. Adopting the successful training model of The Carpentries (https://carpentries.org/) for the HPC space provides a pathway to collaboratively created training content which can be delivered in a scalable way (serving everything from university or industrial HPC systems to national facilities).

We describe the ongoing efforts of HPC Carpentry to create training material to address this need and form the collaborative network required to sustain it. We outline the history of the effort and the practices adopted from The Carpentries that enable it. The lessons being created as a result are under active development and being evaluated in practice at sites in Europe, the US and Canada.

## KEYWORDS

Carpentries, Software Carpentry, Education, Training, HPC

## 1 INTRODUCTION

Many countries are now spending substantial budgets on high performance computing (HPC) related research initiatives [1]. These initiatives are creating computational laboratories of unprecedented scale. At the same time, big data analytics, cloud computing and deep neural networks are influencing the development of HPC to the extent that the possible future convergence of HPC and big data applications is being discussed [10]. Cross-domain interest in Deep Learning alone is driving new sets of users to seek out access to the latest hardware, and in many cases this hardware is to be found in the national and regional supercomputing facilities. The explosion of interest in all of these fields brings with it many challenges, not least of these is providing researchers with adequate training so they can effectively and efficiently leverage these computational laboratories for their research.

To highlight a specific example of this growth, we consider EuroHPC (https://eurohpc-ju.europa.eu) which contains 25 participating EU states. In EuroHPC, each of the participant states is expected to have an *HPC Competence Centre* which will provide HPC services to industry, academia and public administrations. These HPC Competence Centres will be a gateway into the European HPC landscape and, in many cases, the first landing point for HPC interest in those countries. Since it is a European initiative one must strive for consistent levels of service across all states, despite greatly varying experience of the HPC domain. From a training perspective, how does one achieve this? Addressing this challenge is of critical importance to the long-tail impact of the investment in EuroHPC.

In this paper we will address the initial training requirements of a new user who is freshly exposed to HPC resources. Within academic research there is always a constant flow of early career researchers entering the field and accessing resources at all levels of the hardware pyramid. For this reason, we see the "HPC novice" profile as something that is relevant across the spectrum of HPC facilities, from institutional resources all the way up to national and international facilities.

A scalable, collaborative training model is an effective and sustainable way to tackle large increases in demand for "HPC novice" training. We propose to adopt and adapt the model developed by the Software Carpentry initiative [11] and apply it to the novice HPC learner. In Section 2, we introduce aspects of that model and why it has the potential to map well to the HPC space. In Section 3, we outline some of the training material design principles and how they influence the creation process. In Section 4, we look at two distinct evaluation processes, a review process that happens during material creation and learner evaluations that occur during/after training events. Finally, in Section 5, we consider future development efforts in light of progress and outcomes that have occurred to date.

*Authors are listed alphabetically

[1] See for example,

- The Exascale Computing Project, https://www.exascaleproject.org/;
- The EuroHPC Joint Undertaking, https://eurohpc-ju.europa.eu/;
- The Collaboration of Oak Ridge, Argonne, and Livermore (CORAL), https://www.energy.gov/downloads/fact-sheet-collaboration-oak-ridge-argonne-and-livermore-coral/;

but also comparable initiatives in China, Taiwan, Japan and India (see https://en.wikipedia.org/wiki/Exascale_computing).


https://doi.org/10.22369/issn.2153-4136/11/1/4

## 2 SOFTWARE CARPENTRY AND HPC CARPENTRY

In the lessons-learned review of Software Carpentry initiative [12], the author draws a distinction between the "minority who do high-performance computing" and other scientists who use computing as a research tool. However, many who have been involved with training of novice HPC users will recognise that the philosophy of Software Carpentry to teach "researchers the computing skills they need to get more done in less time and with less pain" is just as relevant to the HPC novice, but that the scope of necessary skills is wider. The goal of HPC Carpentry is to target the skills gap as well as the technical and conceptual barriers faced by the HPC novice as they transition from desktop/server computation to HPC resources.

At it's core, Software Carpentry is a volunteer project dedicated to teaching basic computing skills to researchers. This is done by treating lessons the same way you would an open source software project: collaboratively created and free to access or use. This approach is inherently scalable as the lesson itself is a shared resource. Most importantly, Software Carpentry provides training on modern research in education, and associated evidence-based teaching practices [3]. In short, the instructor is taught how to deliver training material effectively and the community provides them with the high quality training content that they will teach. Just as importantly though, a shared lesson and a shared approach to teaching opens up the possibility to create a community around that material. Such a community is a resource in and of itself, and provides a platform for further collaboration on more complex training topics.

HPC Carpentry seeks to crystallise such a community in the HPC domain. This approach resembles the same process that has already been undertaken with the Data Carpentry and Library Carpentry initiatives[2].

### 2.1 The Importance of Collaboration

For some time, there has been significant interest among the HPC training community in the Software Carpentry approach to generating and delivering training content. There have already been two distinct efforts to develop the type of HPC novice material that is being considered here:

- HPC in a day (https://psteinb.github.io/hpc-in-a-day/),
- Introduction to High-Performance Computing (https://github.com/hpc-carpentry/hpc-intro/releases/tag/v9.1.2).

Further interest has included a Birds of a Feather session at SC17 [9].

HPC Carpentry is not, currently, a formal part of the Carpentries but the current development drive is being carried with the knowledge and participation of the Carpentries. At CarpentryCon 2018, HPC Carpentry had 2 sessions [4, 7] (each with ~40 participants), where much of the discussion centred around how to merge existing efforts and form a single group of collaborators to drive the lesson development forward. The creators and maintainers of the previously listed novice material were central to this discussion and have agreed to engage in a unification effort under the HPC Carpentry umbrella. This has resulted in the creation of a

[2]Software Carptentry, Data Carpentry and Library Carpentry are now collectively termed *The Carpentries*, https://carpentries.org/

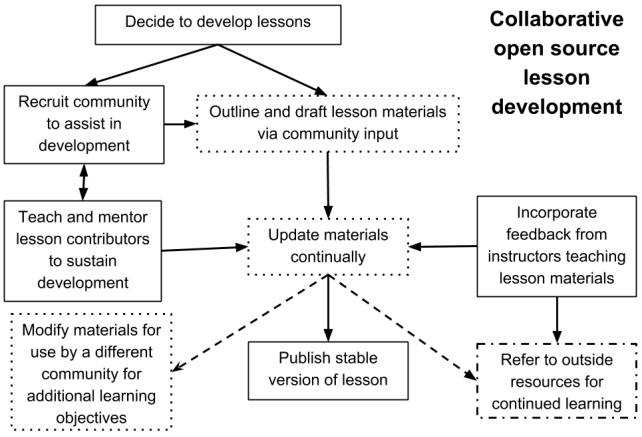



**Figure 1: Collaborative open lesson development (reproduced from [6]).**

`maintainers-hpc` mailing list under the Carpentries umbrella[3] to coordinate the activities of HPC Carpentry, and contributions to the lessons.

The importance of this step is perhaps best represented if we consider [6], where the authors succinctly capture collaborative lesson development in one of their figures (which we reproduce in Fig. 1). Fig. 1 emphasises the crucial step of recruiting a community to

- define the target audience
- assist in material development
- provide feedback and continuous quality control
- link to other projects or resources

among other aspects. Without a community, the continuous improvement, practicability and sustainability of the lesson is in jeopardy (since the maintenance burden falls on a single set of shoulders). Defragmenting previous initiatives within a single collaborative group was an important first step, and a signal to other interested parties that it would form a cohesive effort.

## 3 LESSON DESIGN

The "rules" that govern lesson integration and development are reproduced from [6] in Fig. 2.

One of the first things the interested parties did was to clarify our audience by outlining a set of *Learner Profiles* that are representative of the intended audience (Rule 1)[4]. Furthermore, we restricted the training window to be a single day since this is what many sites already do in practice for the targeted level of this material. In addition, there was lengthy discussion about what should, and should not, be included in the lesson. In particular, there is the issue of what should be considered as required prerequisite knowledge.

The most significant potential prerequisite is the UNIX shell, whether it should be incorporated or not was intensely discussed.

[3]https://carpentries.topicbox.com/groups/maintainers-hpc
[4]https://github.com/hpc-carpentry/hpc-carpentry.github.io/blob/master/why-hpc-carpentry.md#learner-profiles


**Figure 2: 10 rules for collaborative lesson development (reproduced from [6]).**

Teaching UNIX shell fundamentals is already core material within Software Carpentry [2] but there are some additional topics (such as `ssh` sessions, for example) that are specific to remote computing. Having introduced the restriction that there be only one full day to address all the material, the decision was made to split the lesson in two equal parts: the first part dealing with the shell (in the context of remote computing)[5] and the second part with the introduction to working with an HPC resource[6]. Our decision aligns with Rule 2 about modularity of lessons, it provides the instructor with an opportunity to omit the first part if the prerequisite knowledge exists among the participants. In addition, every lesson is structured into chapters. For example:

(1) Why use HPC?
(2) Logging in to a cluster with `ssh`
(3) Submitting jobs
(4) Environment modules
(5) Transferring data

By virtue of creating chapters with high independence and low to no overlap, an instructor is free to skip or remove portions of the content without loss of material coherence.

To respect Rule 3 as regards best practices, we take the Carpentries Instructor Training [1] as our standard[7]. We begin from the end: what do we want our learners to be able to do by the end of the lesson? We work backwards from this to create a *concept map* and give the lesson structure. The map also helps us decide where to introduce formative assessments to help learners commit each concept to memory. Formative assessment ties into Rule 7 of Fig. 2 to infer the lesson audience knowledge intake for a given topic. Furthermore, the HPC Carpentry community strives to implement continuous feedback loops during a workshop, as has become a standard with the Carpentries. In practice this is done through the use of an Etherpad[8] to create collaborative notes and collect questions within the group without interrupting the teaching flow. Anonymous feedback on sticky notes at the end of every half-day session is also collected.

[5]https://hpc-carpentry.github.io/hpc-shell/
[6]https://hpc-carpentry.github.io/hpc-intro/
[7]In particular, please see https://carpentries.github.io/instructor-training/05-memory/
[8]https://etherpad.org/

Lesson design is also influenced by the instructional approach of Carpentries' instructors, who use *guided practice* as the instructional tool by means of *participatory live coding*[9]: each lesson is a set of prepared/faded examples that is done together with the user.

During the lesson design and the enabling collaborative workshops [4, 7], it was repeatedly made evident, by the vigorous discussions on what to include into an introduction to HPC, that the HPC community is very heterogeneous. Expectations by traditional HPC users with a Fortran/C/C++ background and experience in shared or distributed memory parallelism were confronted at equal measure with expectations from a high-throughput community with a map-reduce based understanding of parallelism. This evidence of the convergence or divergence of HPC and big data is mapped directly into a community like HPC Carpentry. This can be considered as evidence for Rule 10, but also emphasises the challenge within HPC Carpentry to implement Rule 1 to 9 in such a environment. It is an ongoing effort to dissect clearly the central topics that HPC carpentry wants to focus on and what motivates them.

## 3.1 Lesson Portability

A key technical issue for lesson portability is the fact that the learner environment can vary significantly between HPC sites. For example, a truly portable lesson should be possible to be configured for different resource managers, scheduler queue configurations, MPI launchers, …

Since the Carpentries lessons are delivered via GitHub pages using Jekyll[10], we leverage liquid templates[11] to enable portability. For example, in the YAML configuration file for the Jekyll website, we can set a variable with:

```
scheduler: "slurm"
```

which can then be referenced throughout the lesson material with:

```
{{ site.scheduler }}
```

Lesson portability then means adding a YAML configuration file appropriate for the target site. This approach is based on that originally taken in [8].

[9]see https://carpentries.github.io/instructor-training/14-live/
[10]see https://help.github.com/en/articles/about-github-pages-and-jekyll
[11]https://jekyllrb.com/docs/liquid/

It should be noted that this also influences lesson design since we must make an additional effort to avoid referencing specific features of tools that may not be available in all of the expected configurations. While this increases the burden for contribution and makes the lesson source code harder to read, we choose to embrace the heterogeneity of HPC as it exists and thus enlarge the number of possible lesson users.

## 4 EVALUATION

There are two different types of evaluation of the lesson material: the evaluation of contributions to the lessons themselves and the evaluation from learners in workshops where the lessons are delivered.

In the case of evaluation of contributions, the contributions themselves are made via Pull Requests[12] to the lesson repositories. These contributions are reviewed by lesson maintainers and by any interested parties who wish to engage in discussing the contribution. The intention is that this process is not only open (*anyone* can make a Pull Request) but also provides a platform for exposing new contributors to the design principles used in the lessons. It also gives us a mechanism by which we can follow Rule 4 of Fig. 2: we can actively encourage new contributors and incorporate them into the community.

From the learners, there are three methods of gathering feedback[13]:

(1) pre- and post-workshop surveys;
(2) minute cards - *anonymous* notes gathered at lunch time and the end of the day which have one positive and one negative comment;
(3) the "one up, one down" technique - the instructor asks the learners to alternately give one positive and one negative point about the day, without repeating anything that has already been said.

Summaries of these evaluations are communicated during the meetings of the lesson collaborators. The surveys help us to assess whether the learning goals have been achieved, while the minute cards and the "one up, one down" technique are usually an evaluation of the instructor as much as the lesson content. Where appropriate, matters arising from the evaluations are raised as issues in the relevant repository.

## 5 CONCLUSIONS

The "HPC novice" lessons https://hpc-carpentry.github.io/hpc-shell and https://hpc-carpentry.github.io/hpc-intro are both undergoing significant development currently. These lessons have been delivered by a number of people within the HPC Carpentry community and initial feedback has been largely positive. There has not, as yet, been a more objective evaluation of the lessons as the lessons are still (in software terms) in an "alpha" state. The immediate goal is to continue the development process and stress-test the lessons with new learners, instructors and teaching environments to be found in Canada, the US and Europe (home to many of the collaborators).

While doing so, a community of contributors is being created, nourished and expanded. It is the development of this community, based on open and transparent governance structures (motivated by [5]), that is key to generating the capacity and energy to sustainably develop HPC training material based on modern teaching methods and flexible enough to be adopted by the heterogeneous HPC community of the 21st century. This community effort is, however, unfunded volunteer effort which restricts the possibility of providing concrete timelines for achieving lesson maturity. The lesson repositories on GitHub are the best place to keep track of recent developments and contributions[14].

Much in the same way that Software Carpentry developed, there is a desire from collaborators to ultimately move beyond novice lessons to more advanced topics. Priority has been given to novice content because it is common ground before we approach the branching of learner profiles expected when we consider more advanced topics (software users, software developers, hardware-specific training, domain-specific training,…).

[12]https://help.github.com/en/articles/about-pull-requests
[13]https://carpentries.github.io/instructor-training/06-feedback

## ACKNOWLEDGMENTS

Alan Ó Cais acknowledges support from the European Union's Horizon 2020 research and innovation program, under grant agreement No. 676531 (project E-CAM) and grant agreement No. 823964 (project FocusCoE).

## REFERENCES

[1] Aron Ahmadia, James Allen, Piotr Banaszkiewicz, Erin Becker, Trevor Bekolay, John Blischak, Andy Boughton, Erik Bray, Abigail Cabunoc Mayes, Steve Crouch, Neal Davis, Matt Davis, Jonah Duckles, Rémi Emonet, FÃĺlix-Antoine Fortin, Ivan Gonzalez, Chris Hamm, Michael Hansen, Rayna Harris, Felix Henninger, Konrad Hinsen, Amy Hodge, Mike Jackson, W. Trevor King, Justin Kitzes, Christina Koch, Tom Liversidge, FranÃğois Michonneau, Bill Mills, Lex Nederbragt, Aaron O'Leary, Elizabeth Patitsas, Aleksandra Pawlik, Fernando Perez, Jon Pipitone, Timothée Poisot, Ariel Rokem, Raniere Silva, Tracy Teal, Tim TrÃűndle, Fiona Tweedie, Jill-Jênn Vie, Jordan Walker, Alistair Walsh, Belinda Weaver, Ethan White, Chandler Wilkerson, Jason Williams, Greg Wilson, and Anelda van der Walt. 2016. Software Carpentry: Instructor Training. (June 2016). https://doi.org/10.5281/zenodo.57571

[2] Inigo Aldazabal Mensa, Harriet Alexander, James Allen, Areej Alsheikh-Hussain, Daniel Baird, Piotr Banaszkiewicz, Pauline Barmby, Rob Beagrie, Trevor Bekolay, Evgenij Belikov, Jason Bell, Kai Blin, John Blischak, Simon Boardman, Maxime Boissoneault, Jessica Bonnie, Andy Boughton, Ry4an Brase, Amy Brown, Dana Brunson, Orion Buske, Abigail Cabunoc Mayes, Daniel Chen, Kathy Chung, Gabriel A. Devenyi, Emily Dolson, Jonah Duckles, Rémi Emonet, David Eyers, Filipe Fernandes, Hugues Fontenelle, Francis Gacenga, Matthew Gidden, Ivan Gonzalez, Norman Gray, Varda F. Hagh, Michael Hansen, Emelie Harstad, Adina Howe, Fatma Imamoglu, Damien Irving, Mike Jackson, Emily Jane McTavish, Michael Jennings, Dan Jones, Alix Keener, Kristopher Keipert, Tom Kelly, Jan T. Kim, W. Trevor King, Christina Koch, Bernhard Konrad, Sherry Lake, Doug Latornell, Philip Lijnzaad, Eric Ma, Joshua Madin, Camille Marini, Kunal Marwaha, Sergey Mashchenko, FranÃğois Michonneau, Ryan Middleson, Jackie Milhans, Bill Mills, Amanda Miotto, Sarah Mount, Lex Nederbragt, Daiva Nielsen, Aaron O'Leary, Randy Olson, Adam Orr, Nina Therkildsen, Kirill Palamartchouk, Adam Perry, Jon Pipitone, Timothée Poisot, Hossein Pourreza, Timothy Povall, Adam Richie-Halford, Scott Ritchie, Noam Ross, Halfdan Rydbeck, Mahdi Sadjadi, Pat Schloss, Bertie Seyffert, Genevieve Shattow, Raniere Silva, Sarah Simpkin, John Simpson, Byron Smith, Nicola Soranzo, Ashwin Srinath, Daniel Standage, Meg Staton, Peter Steinbach, Marcel Stimberg, Bartosz Telenczuk, Florian Thoele, Tiffany Timbers, Stephen Turner, Jay van Schyndel, Anelda van der Walt, David Vollmer, Jens von der Linden, Andrew Walker, Josh Waterfall, Ethan White, Carol Willing, Greg Wilson, Donny Winston, Lynn Young, and Lee Zamparo. 2016. Software Carpentry: The Unix Shell. (June 2016). https://doi.org/10.5281/zenodo.57544

[3] Susan A. Ambrose, Michael W. Bridges, Michele DiPietro, Marsha C. Lovett, and Marie K. Norman. 2010. *How Learning Works: Seven Research-Based Principles for Smart Teaching*. Jossey-Bass. https://www.amazon.com/How-Learning-Works-Research-Based-Principles/dp/0470484101

[14]https://hpc-carpentry.github.io/hpc-shell and https://hpc-carpentry.github.io/hpc-intro

[4] Alan Ó Cais and Daniel Smith. 2018. Breakout 8: HPC Carpentry. https://github.com/carpentries/carpentrycon/tree/master/CarpentryCon-2018/Sessions/2018-05-31/05-Breakout-8-HPC-Carpentry. (2018). Breakout session at CarpentryCon 2018.
[5] The Carpentries. 2019. Governance. https://carpentries.org/governance/. (2019).
[6] Gabriel A. Devenyi, Rémi Emonet, Rayna M. Harris, Kate L. Hertweck, Damien Irving, Ian Milligan, and Greg Wilson. 2017. Ten simple rules for collaborative lesson development. *CoRR* abs/1707.02662 (2017). arXiv:1707.02662 http://arxiv.org/abs/1707.02662
[7] Christina Koch and Peter Steinbach. 2018. Workshop 5: HPC Carpentry. https://github.com/carpentries/carpentrycon/tree/master/CarpentryCon-2018/Sessions/2018-06-01/05-Workshop-5-HPC-Carpentry. (2018). Workshop at CarpentryCon 2018.
[8] Peter Steinbach, Aaron O'Leary, Abigail Cabunoc, Andy Boughton, Ashwin Srinath, Bill Mills, Francois Michonneau, Greg Wilson, James Allen, John Blischak, Jon Pipitone, Michael Hansen, Olav Vahtras, Piotr Banaszkiewicz, Raniere Silva, RÃĭmi Emonet, Stephan Janosch, TimothÃĭe Poisot, Toby Hodges, Trevor Bekolay, and W. Trevor King. 2019. HPC in a day. (March 2019). https://doi.org/10.5281/zenodo.2612065
[9] Andrew Turner, Christina Koch, Tracy Teal, Robert Freeman Jr, Chris Bording, and Martin Callaghan. 2017. HPC Carpentry - Practical, Hands-On HPC Training. https://sc17.supercomputing.org/index.html%3Fpost_type=page&p=5407&id=bof125&sess=sess359.html. (2017). BoF session at SC17.
[10] Theo Ungerer and Paul Carpenter. 2018. Eurolab-4-HPC Long-Term Vision on High-Performance Computing. *CoRR* abs/1807.04521 (2018). arXiv:1807.04521 http://arxiv.org/abs/1807.04521
[11] Greg Wilson. 2010. Software Carpentry web site. http://software-carpentry.org. (2010). Main web site for Software Carpentry, replacing http://swc.scipy.org.
[12] G Wilson. 2016. Software Carpentry: lessons learned [version 2; peer review: 3 approved]. *F1000Research* 3, 62 (2016). https://doi.org/10.12688/f1000research.3-62.v2

# Blue Waters Workforce Development
# Delivering National Scale HPC Workforce Development

Jennifer Houchins
Shodor Education Foundation, Inc.
Durham, NC
jhouchins@shodor.org

Scott Lathrop
Shodor Education Foundation, Inc.
National Center for Supercomputing Applications (NCSA)
University of Illinois
Urbana-Champaign, IL
lathrop@illinois.edu

Robert Panoff
Shodor Education Foundation, Inc.
Durham, NC
rpanoff@shodor.org

Aaron Weeden
Shodor Education Foundation, Inc.
Durham, NC
aweeden@shodor.org

## ABSTRACT

There are numerous reports documenting the critical need for high performance computing infrastructure to advance discovery in all fields of study. The Blue Waters project was funded by the National Science Foundation to address this need and provide leading edge petascale computing resources to advance research and scholarship. There are also numerous reports that identify the lack of an adequate workforce capable of utilizing and advancing petascale class computing infrastructure well into the future. From the outset, the Blue Waters project has responded to this critical need by conducting national scale workforce development activities to prepare a larger and more diverse workforce. This paper describes those activities as exemplars for adoption and replication by the community.

## KEYWORDS

HPC, education, training, computational science education, petascale computing, broadening participation

## 1 INTRODUCTION

The National Science Foundation funds the Blue Waters project, which supports an Education, Outreach and Training (EOT) program focused on preparing an HPC-capable workforce with an emphasis on petascale computing competencies. The Blue Waters EOT team engages undergraduate students in internships, graduate students in fellowships, researchers as participants in training sessions, trainers and educators as PIs of education allocations, and underrepresented communities as PIs of broadening participation allocations. All of these communities benefit from access to on one of the most advanced computing environments available to the open science research community. Educators, researchers and students are asked to present their research via conference presentations (e.g. the annual Blue Waters Symposium) and publications (e.g. the Journal of Computational Science Education).


https://doi.org/10.22369/issn.2153-4136/11/1/5

## 2 EDUCATION ALLOCATIONS

The initial proposal for the Blue Waters project requested that 1% of the available computing resources be devoted to educational activities to prepare a larger and more diverse, computationally literate workforce [12]. At that time, 1% of the system was a substantial commitment - providing more computational resources than were available to researchers via all of the other NSF funded HPC systems. This portion of the system is allotted through an education allocation application process that is available to faculty and staff at any US institution. Applications may be made to support undergraduate and graduate courses, training sessions, workshops, webinars, institutes, and Research Experiences for Undergraduates (REUs). We encourage innovative approaches to educating the community. Requests range from one day training events to a full year program of structured learning, such as through internships and fellowships. Allocation requests typically range from 5,000 to 25,000 node hours, although allocations of larger amounts have been granted for programs serving large numbers of participants or for conducting more complex semester course requirements.

## 3 BROADENING PARTICIPATION ALLOCATIONS

In order to engage a more diverse community of researchers, the Blue Waters project created a new allocations category for Broadening Participation [11]. The purpose was to encourage principal investigators who were women, minorities, individuals at NSF designated EPSCoR institutions [2], andor individuals at Minority Serving Institutions (MSIs) to apply for an allocation of time on the Blue Waters system. These allocations were intended as start-up allocations of up to 200,000 node-hours to allow each research team to scale up their codes to Blue Waters. Twenty-one teams from across the United States were selected in the first year. Included among the Principal Investigators [1] (PIs) were ten females and two underrepresented minorities. In addition, there were four female co-PIs and eight underrepresented minority co-PIs. Among

the lead institutions, five were Minority Serving Institutions and ten were within EPSCoR jurisdictions. After working on the Blue Waters system for nearly a year, one of the teams received a PRAC allocation [3] from NSF, which will allow the team to significantly advance their computational research in the future.

## 4 STUDENT ENGAGEMENT

The Blue Waters project directly engaged students through the Blue Waters Graduate Fellowship Program and the Blue Waters Student Internship Program. Both programs provide these students with a full year of financial support and access to the Blue Waters system to conduct computational research. Information about each of these student programs follows.

### 4.1 Graduate Fellowships

The Blue Waters project offers a unique, federally supported program that provides PhD students with a full year of computational science and engineering research support. Each fellow receives an allocation of 50,000 node-hours to pursue their computational and/or data-enabled research on the Blue Waters system [13]. The fellows also receive a $38,000 stipend. Each fellow is able to request up to $12,000 in tuition allowance to help offset their educational expenses. Each fellow is invited to attend the annual Blue Waters Symposium to make a formal presentation and display a poster to share their research progress with the other attendees. They are also encouraged to give a presentation of their Blue Waters supported research at a domain conference of their choosing. There are between four and 10 fellows selected each year through a competitive application process that is open to students in all US academic institutions. After their fellowship year ends, the fellows are encouraged to continue using the Blue Waters system to pursue their research while completing their PhD. They are also encouraged to continue using the system during a subsequent postdoctoral appointment. Many of the fellows have gone on to faculty positions, postdoctoral positions, and professional positions in academia, government agencies and academic institutions. We continue to track the progress of each fellow to facilitate a longitudinal analysis of the impact of the fellowship program.

### 4.2 Undergraduate Internships

The Blue Waters Student Internship Program is designed to motivate and prepare the next generation of computational researchers by engaging them in year-long research projects. The Internship Program [14] supports about 20 students each year, with a $5,000 stipend spread out over the full year of their appointment. The program welcomes applications from undergraduates at all degree granting US institutions. Each year, the program kicks off with a two-week intensive petascale institute at NCSA that also engages the interns in learning to make effective use of the Blue Waters system. The students are matched with faculty who mentor them through their year-long research projects. Towards the end of their year-long research endeavors, the students apply to present a poster on their research project at the annual Blue Waters Symposium in the May/June timeframe. The faculty report that the combination of a two-week institute and support for a full year have proven to be very effective for the students? learning outcomes and the projects the students pursue. To date, the internship program has benefitted 120 undergraduate students and resulted in numerous papers being published by the students in the Journal of Computational Science Education (JOCSE) [8] as described in the next section.

## 5 JOURNAL OF COMPUTATIONAL SCIENCE EDUCATION

All of the interns are encouraged to publish their research in the peer-reviewed Journal of Computational Science Education (JOCSE) [8]. In this publication, the students are encouraged to describe their experiences and the impacts of the program on their academic pursuits and career goals. JOCSE accepts articles from the international community that address the teaching and learning of computational science and engineering, the development and applications of instructional materials, projects, and innovative approaches for conducting workforce development. The editors welcome articles that address the assessment of materials or programs, methods for achieving improved learning outcomes, and innovative computational science programs. The journal articles and instructions for submissions are available at http://jocse.org/.

## 6 EDUCATION AND TRAINING

The Blue Waters EOT team [15] has brought together experts to offer a variety of education and training sessions throughout the year to assist researchers and educators with incorporating state-of-the-art resources, tools, and methods within their research and education endeavors.

The Blue Waters project conducts a variety of training events throughout the year to assist participants in learning computational and data-enabled science and engineering methods, tools, and resources. The training is designed to prepare participants to make effective use of computing resources, with an emphasis on petascale computing. The training events include webinars, workshops, symposia, tutorials, hackathons, and other related activities. These are delivered as in-person events, webcasts, and as self-paced tutorials.

The Virtual School of Computational Science and Engineering (VSCSE) [5] delivered graduate level computational science and HPC workshops and courses to students at colleges and universities across the United States, and to students at international locations. The VSCSE workshops were delivered using high-definition video conferencing to as many as 7 remote sites simultaneously. In total, the 20 VSCSE [5] workshops served over 5,000 people at 54 institutions. The semester courses were led delivered by HPC experts in the field and were conducted in collaboration with faculty at participating institutions in order to provide students with access to course content and mentoring that would otherwise not have been available to them. A total of 7 semester courses were delivered to over 600 graduate students at 29 institutions.

### 6.1 HPC University Repository

The HPC University portal [7] was established to provide a mechanism for disseminating HPC training and education material. It is built on the foundation and principles established by the Computational Science Education Reference Desk (CSERD) [6], which is among the collections funded by the National Science Digital Library (NSDL) [4] funded by NSF. The Blue Waters EOT team

recruited faculty and staff from across the United States to develop and classroom-test 30 “Undergraduate Petascale modules" [9] appropriate for teaching parallel computational modeling to undergraduate or graduate students in STEM disciplines. Training materials that were developed by Blue Waters were also posted to the repository. Education and training materials developed by the community have also been made accessible via the repository.

In addition to adding substantial education and training content to the HPC University repository, Blue Waters committed staff time to improving the infrastructure and promoting this resource. The repository has proven to be an effective tool for facilitating broad dissemination of the materials. The materials have been downloaded and used to support workforce development at the high school, undergraduate and graduate levels.

### 6.2 Undergraduate Petascale Curriculum Modules

To facilitate inclusion of petascale computing in undergraduate education, the Blue Waters project funded the development of a set of curricular materials that include classroom-ready, domain-specific examples of HPC applications in science and engineering [9]. These materials were developed by faculty with experience teaching HPC and designed to enable the teaching and use of HPC in undergraduate science and engineering classrooms. The materials are presented in self-contained modules that include instructor materials for domain-specific applications, starter source codes, and sample problem solutions. The modules range in topic (or content area) from parallel simulation of n-body problems to dynamic programming with CUDA. All of the modules have been catalogued in the HPC University repository [7] and are freely available for use in the classroom.

## 7 THE BLUE WATERS SYMPOSIUM

The Blue Waters Symposium [10] is an annual gathering of Blue Waters staff, researchers, students, and professionals from the computational science and engineering community. The Symposia participants share successes and challenges in utilizing large-scale heterogeneous computing systems. Each of the scientific teams using the Blue Waters system are asked to provide updates on how the petascale system has helped to advance their research. Nationally and internationally recognized leaders are invited as keynote speakers to present innovative, impactful, and at times controversial ideas that advance knowledge and provoke interactions among the attendees. There are numerous opportunities for the participants to discuss challenges, opportunities, and the future of scientific computing. The discussions often times result in new collaborations and cooperative ventures.

## 8 SUMMARY

The Blue Waters project actively recruits students, faculty, professionals, and mentors in these activities from across the United States, with an emphasis on engaging women, minorities, and people with disabilities. Since going into full-service operations in 2013, over 200 education and training allocations have been utilized for activities ranging from one-day workshops to two-week institutes. The Blue Waters project has engaged more than 3,700 people in learning to make effective use of computational and data-enabled science and engineering tools, resources, and methods. The participants in the activities came from 219 academic institutions, of which 65 are within EPSCoR jurisdictions. The impact and benefits have been widespread, including directly reaching people located in many foreign countries, as well as freely disseminating materials that have been downloaded and used by thousands of people world-wide.

The Blue Waters project places a high importance on sharing what we have learned to help others to be even more successful in their own endeavors. We look forward to sharing our experiences with the community and fostering an ongoing exchange of lessons learned and good practices.

## REFERENCES


[1] National Center for Supercomputing Applications. 2019. Blue Waters Broadening Participation Awards. Retrieved May 31, 2019 from http://www.ncsa.illinois.edu/news/story/blue_waters_awards_21_broadening_participation_allocations

[2] National Science Foundation. 2019. EPSCoR Program. Retrieved May 31, 2019 from https://www.nsf.gov/od/oia/programs/epscor/

[3] National Science Foundation. 2019. PRAC Allocations. Retrieved May 31, 2019 from https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503224

[4] NSDL. 2019. The National Science Digital Library. Retrieved May 31, 2019 from https://nsdl.oercommons.org/

[5] The University of Michigan. 2019. Virtual School of Computational Science and Engineering. Retrieved May 31, 2019 from http://www.vscse.org

[6] Inc. Shodor Education Foundation. 2019. Computational Science Education Reference Desk. Retrieved May 31, 2019 from http://www.shodor.org/refdesk/

[7] Inc. Shodor Education Foundation. 2019. HPC University. Retrieved May 31, 2019 from http://hpcuniversity.org

[8] Inc. Shodor Education Foundation. 2019. Journal of Computational Science Education. Retrieved May 31, 2019 from http://jocse.org

[9] Inc. Shodor Education Foundation. 2019. Undergraduate Petascale Curriculum Modules. Retrieved May 31, 2019 from http://www.shodor.org/petascale/materials/modules/

[10] Blue Waters. 2019. Blue Waters Symposium. Retrieved May 31, 2019 from https://bluewaters.ncsa.illinois.edu/blue-waters-symposium

[11] Blue Waters. 2019. Broadening Participations Allocations. Retrieved May 31, 2019 from https://bluewaters.ncsa.illinois.edu/broadening-participation-allocations

[12] Blue Waters. 2019. Education Allocations. Retrieved May 31, 2019 from https://bluewaters.ncsa.illinois.edu/education-allocations

[13] Blue Waters. 2019. Graduate Fellowships. Retrieved May 31, 2019 from https://bluewaters.ncsa.illinois.edu/fellowships

[14] Blue Waters. 2019. Student Internship Program. Retrieved May 31, 2019 from https://bluewaters.ncsa.illinois.edu/internships

[15] Blue Waters. 2019. Training Offerings. Retrieved May 31, 2019 from https://bluewaters.ncsa.illinois.edu/training

# One Year HPC Certification Forum in Retrospective

Julian Kunkel
University of Reading
Reading, United Kingdom
j.m.kunkel@reading.ac.uk

Kai Himstedt
Universität Hamburg
Hamburg, Germany

Weronika Filinger
EPCC, The University of Edinburgh
Edinburgh, United Kingdom

Jean-Thomas Acquaviva
DDN
Paris, France

Anja Gerbes
Goethe-Universität
Frankfurt am Main, Germany

Lev Lafayette
University of Melbourne
Melburne, Australia

## ABSTRACT

The ever-changing nature of HPC has always compelled the HPC community to focus a lot of effort into training of new and existing practitioners. Historically, these efforts were tailored around a typical group of users possessing, due to their background, a certain set of programming skills. However, as HPC has become more diverse in terms of hardware, software and the user background, the traditional training approaches became insufficient in addressing training needs of our community. This increasingly complicated HPC landscape makes development and delivery of new training materials challenging. How should we develop training for users, often coming from non-traditionally HPC disciplines, and only interested in learning a particular set of skills? How can we satisfy their training needs if we don't really understand what these are? It's clear that HPC centres struggle to identify and overcome the gaps in users' knowledge, while users struggle to identify skills required to perform their tasks.

With the HPC Certification Forum, we aim to clearly categorise, define, and examine competencies expected from proficient HPC practitioners. In this article, we report the status and progress this independent body has made during the first year of its existence. The drafted processes and prototypes are expected to mature into a holistic ecosystem beneficial for all stakeholders in HPC education.

## 1 INTRODUCTION

There is a generally accepted set of skills and competencies necessary to efficiently use HPC resources. This skill set depends on the role and domain of the practitioner but also on the available infrastructure of the centre providing the computing resources. For example, a scientist needing to run an application on a specific machine may need basic skills in Linux, MPI, environment modules, and knowledge about the batch scheduler, e.g., Slurm. Now rather than providing that scientists with a list of instructions to be followed in a copy and paste fashion or a never-ending list of things they should know, we want them to understand each of the required steps without having to learn everything about it, but at the same time understand how it fits into the bigger picture of using an HPC system. For instance, understanding Slurm is a good example of a very fine-grained skill that fits under a more generic skill defined as "resource management", illustrating concepts across the rich variety of available resource managers.

Most data centers operating HPC systems offer regular training events focusing on general aspects of their supercomputer's hardware architecture, software stack, application development environment and various tuning and debugging tools. It is understandable why the materials they offer are geared towards the special demands of the specific HPC environment and the institutions they support. The problem is that teaching content typically covers only a fraction of the HPC skills necessary to use another HPC system. This approach of teaching only specific implementations, tools or workflows—instead of concepts behind them—makes a move to a different system or tool unnecessary complicated. Narrowing the scope of training events make sense from the HPC provider perspective, but is not very conducive to development of a more comprehensive learning environment that provides assessment and certifies the newly acquired skills. Although certificates are used widely in IT industry to verify certain knowledge, until now there was no similar approach for HPC training. It is, however, clear that a mapping of competences and certification scheme could address some of the challenges stemming from the constantly growing training needs of our community,

This article describes the current status of the certification program curated by the HPC Certification Forum. Our previous work provided a brief overview of the evolution from the project that sparked the HPC Certification Forum [2] while this article provides details on adopted mechanisms and design decisions.

The article is structured as follows: First, in Section 2 we introduce the HPC Certification Forum and the certification program. Then, an overview of the status is given in Section 3, followed by the organisation of competencies discussed in Section 4. This leads to the proposal for the certification process in Section 5. In Section 6 the whole learning ecosystem is described. Related work is presented in Section 7. Finally, the article is concluded in Section 8.

## 2 THE HPC CERTIFICATION FORUM

The HPC Certification Forum (HPCCF) has the role of a (virtual) central authority to curate and maintain the proposed certification program. The program consists of three parts: the tree of defined competencies, the examination of practitioners to prove they posses those skills, and finally the certification demonstrating their knowledge. Although, the forum is not involved in development of any


https://doi.org/10.22369/issn.2153-4136/11/1/6

training materials or tools, it supports the ecosystem around the competencies.

The HPCCF aims to support existing activities and complements them by providing a unified and clear way of mapping out the relevant HPC competencies. Thus, the HPCCF does not regulate the content of training material; we purposely separate the definition of skills, the examination and the certification from content delivery. Similarly, the program does not prescribe a curriculum or any fixed order by which skills need to be obtained. It eases the navigation between different competencies without being overly restrictive.

The forum is organised around a Webpage[1], a GitHub repository[2], and a team collaboration tool (Slack) for communication and monthly meetings. The membership is divided into three categories: associate, full and steering board. Anyone can become an associate member free of charge, allowing a passive involvement – e.g., observing the status, being listed on the webpage. Full members must actively contribute to the program and, in return, gain voting rights for the roles of the steering board. The steering board directs the overall activities and is organised in different responsibilities, i.e., into topic-specific chairs. At the moment the steering board has 9 members.

## 3 OVERVIEW OF THE HPCCF ACTIVITIES

The first year of the HPCCF forum was rich in discussions, both internally and with external partners, resulting in a number of processes and prototype tools being put in place. These are described briefly below, and in more details in the subsequent sections.

*Management.* The first steering board has been elected at ISC-HPC 2018. We established communication channels via webpage, email and Slack for regular discussion. Slack is also used to conduct our monthly open meetings between members and the steering board. At beginning, video conferencing tools have been used but the geographical diversity of the attendees influenced the decision to move towards chat-based tools. The asynchronous nature and automatic message retention proved to be more productive, and provided a more open and inclusive platform for discussion. Meeting notes with action items are extracted from Slack and documented using Google Doc (an online word processor allowing real-time and multiple-user collaboration).

The ongoing conversation with various stakeholders involved in HPC education and training, keeps producing more engagement in the forum's activities. The member's contributions are voluntarily and follow the schedule agreed between the contributor and the HPCCF board. The prospective contributions are managed in Trello (a web-based list-making collaborative application), in which each member can manage their own card.

*Technical tasks.* The tasks we set out to accomplish are: 1) the definition of competencies; 2) the examination of practitioners; 3) the creation of certificates; and 4) the reinforcement of an ecosystem of tools supporting them.

For all these sub-goals we made substantial progress. While the definition and organisation of competencies was the main focus, we prototyped various tools and processes that embed the competencies into the wider education ecosystem.

Firstly, considering the pedagogic literature from higher education we finalised the templates for the skills description. The effort to map out all of the relevant competencies is still on-going, but the feedback received during a number of events organised by the HPCCF and other members of the HPC education community resulted in the addition of new topics to the first version of the certification program.

The definitions of the competencies are version controlled with Git, and are publicly available in XML and Markdown format. A Wiki[3] is being used to edit the skills' descriptions, and a navigable and widely customisable JavaScript[4] makes the skill tree easily accessible to the forum members and the end-users alike.

Preliminary processes and tools have been developed and deployed for curation of the examination questions and for conducting an online multiple-choice examination.

## 4 SKILLS

A skill is defined as a set of learning outcomes and relevant metadata. Within a single skill, there can also be multiple levels (basic, intermediate and expert level) building upon each other and further distinguishing the expertise. We expect the practitioners to acquire the lower levels before progressing to more complex levels.

The basic level should cover the most relevant aspect of the skill (needed by anyone that uses the skill), whereas the intermediate (used for common exceptional cases) and expert levels (used in special circumstances) are needed only for a subset of users. Typically, an institution has only few experts, if any at all.

This model can be compared to the classification of school knowledge, for example, a skill with the name "basic arithmetic operations" would include the math skills of addition (being able to add numbers) and subtraction (being able to subtract numbers), multiplication (being able to multiply numbers) and division (being able to divide numbers). Operating on two numbers with ± would be a basic level of the skill, whereas including multiplication/division and mentally operating on hundreds of numbers would be an expert level. Another example could be a skill called "technical drawing", in which being able to draw simple geometric figures would be counted as a basic level, and an ability to draw a complicated mechanical objects would be an expert level.

The skills are organised in a tree structure from a coarse-grained to a fine-grained representation, allowing users to browse the skill based on the semantics. The root level of our current[5] skill tree is shown in Figure 1. The basic idea is that the skills from the root level simplify the navigation by providing an indication about their scope, e.g., core knowledge, usage of HPC environments, or about programming. This should allow the user to rapidly drill into the skill representation. The closer a skill is to the root, the more abstractly it is defined, while leaf nodes cover the knowledge for a specific skill.

As the tree serves the purpose of organising the skills, the references from one branch to a skill in another branch are allowed. This

[1] http://hpc-certification.org
[2] https://github.com/HPC-certification-forum
[3] https://www.hpc-certification.org/wiki/
[4] https://www.hpc-certification.org/skills/map/
[5] The goal is to finalise the tree this year.

important feature makes the reuse of the skill definitions possible, at the same time allowing users to navigate the tree according to the semantics.

## 4.1 Description of a skill

Each skill on the tree, including the inner nodes, is described in more detail as follows:

- **ID**: Identifier according to its position in the skill tree. The last character indicates the level of the skill (Basic, Intermediate, or Advanced).
- **Name**: A name capturing the essence of the skill.
- **Background**: Provides brief information motivating the need for the skill and how it fits into the bigger picture with other skills.
- **Aims**[6]: Describe the purpose of the skills, but doesn't really include a list of what a practitioner will learn or do. Explaining what a skill is trying to achieve is not the same as saying how it should be done.
- **Learning outcomes (LOs)**: Defines briefly what practitioners will learn. The objectives are statements what prospective learners are able to do. They should clearly describe or define an action bringing about a measurable/quantifiable increase in understanding of that skill.

On the leaf level, a skill is fine-grained and orthogonal to other skills – their narrowed scope means they can be taught in sessions ranging from a 1.5 hour lecture up to a 4 hour workshop. We believe this granularity allows practitioners to cherry-pick the skills relevant to their circumstances, and lecturers and examiners to prepare small lectures with well-defined content. For technology-dependent skills on the leaf level (e.g., a specific file system or workload manager) the introductory skills are often provided, as they contribute to the foundation of many specialised skills representing a specific hardware or software technology.

The aggregation within the tree is similar to that found in the education circles. Hussey et al.[1] categorised the aggregation of skills into the following levels: individual teaching events, specified for modules or short courses, and those specified for whole degree programmes. Hussey et al. conclude that LOs on the coarse grained level are not useful for high-level education because they "would state little more than an annotated list of contents" and relevant for those familiar with the subject. To overcome these issues, we provide a relaxed level of granularity and lower level of abstraction of the learning outcomes on inner nodes.

## 4.2 Skill Examples

This subsection presents an example of an inner-node skill called "Executing parallel applications", as well as its two leaf skills - "Workload manager introduction" and "Slurm workload manager".

- **ID**: USE4.2-B
- **Name**: Executing parallel applications
- **Background**: Parallel computers are operated differently than a normal PC – all users share the system. To ensure a fair and efficient use of shared resources, various operative procedures are put in place. Users must understand these concepts and procedures to be able to run a parallel application on the resources available to them. Moreover, different HPC systems adopt different solution to manage their resources.
- **Aim**: To enable practitioners to comprehend the concepts and procedures for running parallel applications in HPC environments; to run and monitor the execution of parallel applications on HPC systems.
- **Learning outcomes**:
  - explain the concepts and procedures related to resource allocation and job execution in an HPC environment;
  - run interactive jobs and batch jobs;
  - understand and describe the content and expected behaviour of job scripts;
  - change provided job scripts and embed them into shell scripts to run a variety of parallel applications;
  - analyse the output generated from a job scheduler and understand the cause of typically generated errors.

The next leaf-level skill describes how workload management works in general, regardless of the specific software implementing it. The aim and outcomes defined by the parent skill (described above) are expected to be covered to a certain extend and refined by this leaf skill.

- **ID**: USE4.2.1-B
- **Name**: Workload manager introduction
- **Background**: There is a wide range of different workload managers in use. This skill on a conceptual level explains how to use them.
- **Aim**: To enable practitioners to comprehend and describe the basic architecture and concepts of resource allocation on an HPC system.
- **Learning outcomes**:
  - comprehend the exclusive and shared usage model in HPC;
  - explain the generic steps required to run and monitor a single job;
  - differentiate between the batch and interactive job submission;
  - comprehend the generic concepts and structure of resource manager, scheduler, job and job script;
  - explain the role of environment variables as a mean to communicate certain settings of your job;
  - comprehend job budgeting and accounting principles.

The following skill describes a leaf-level skill for the usage of Slurm at the basic level. While there is no formal dependency to the introduction (skill USE4.2.2-B), it is expected that practitioners have obtained the other skill before learning this one.

- **ID**: USE4.2.2-B
- **Name**: Slurm Workload manager
- **Background**: Slurm is a widely used open-source workload manager, which also provides various advanced features.
- **Aims**:
  - To enable practitioners to use relevant tools to run and monitor parallel applications using Slurm.

[6]The definition of aims and outcomes follows literature for higher education [5] and https://www.heacademy.ac.uk/system/files/assessment-learning-outcomes.pdf.

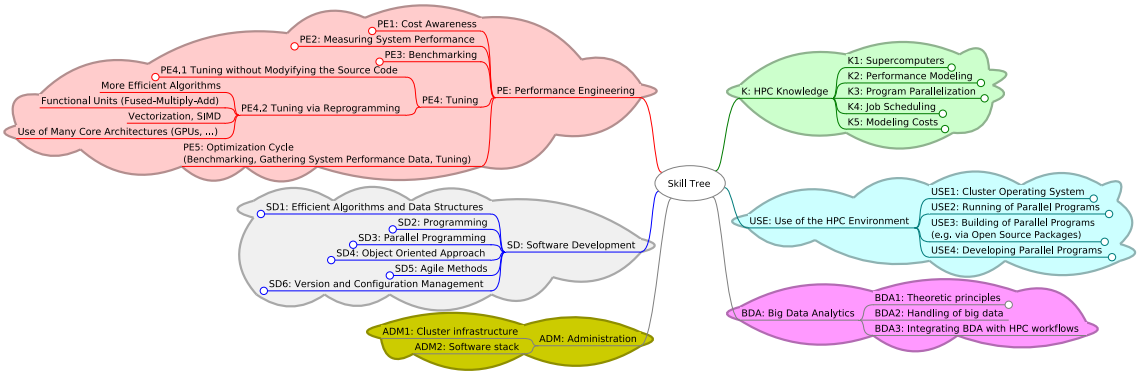



**Figure 1: Skill tree: top level competencies with the PE4 branch expanded.**

  - To enable practitioners to comprehend and describe the basic structure of Slurm and the related suite of tools.
- **Learning outcomes**:
  - run interactive jobs with *salloc*, a batch job with *sbatch*;
  - explain the architecture of Slurm, i.e., the role of *Slurmd*, *srun* and the injection of environment variables;
  - explain the function of the tools: *sacct*, *sbatch*, *salloc*, *srun*, *scancel*, *squeue* and *sinfo*;
  - explain time limits and the benefit of a backfill scheduler;
  - comprehend that environment variables are set when running a job;
  - comprehend and describe the expected behaviour of a simple job script;
  - comprehend how variables are prioritised when using command line and a script;
  - change a provided job template and embed them into shell scripts to run a variety of parallel applications;
  - analyse the output generated from submitting to the job scheduler and typically generated errors.

The learning outcomes of this skill, marked with B to indicate its basic level, are focusing on the most fundamental knowledge required to use Slurm effectively. An intermediate version of the skill could, for example, cover how to create reservations or special queues.

## 5 CERTIFICATION

A certificate shall serve as a confirmation that a user obtained the expertise in the relevant skills. Special considerations needs to be given to the certification integrity. We must ensure that the learning outcomes of the covered skills are examined to a satisfactory degree, and prevent self-cheating or dishonesty. Students can always cheat during an examination – the susceptibility to cheating depends on many factors, including but not limited to the type of exam, methods used for assessment, motivation for taking it, consequences of failing or scoring poorly, and the possibility and frequency of resits.

We want to enable a cost-effective assessment (free for the practitioner), so we chose an online examination. This increases the opportunity for cheating [3] but we plan to deploy a several strategies to minimise the risk of cheating, such as raising the awareness of the examinees, using a pool of questions, time limits for each question and a delay between registering for and taking the examination.

Since knowledge can age, each certificate needs to indicated when (month and year) the qualifying examination took place. Also, because the examination of a single fine-grained leaf-level skill would be too easy for short-term memorisation and more prone to self-cheating, the certificates bundle multiple skills together. We believe the incentive to deliberately fake the assessment, e.g., by having the exam filled by someone else, and thus, being awarded a certificate, is low. Therefore, we address this issue in a lightweight fashion only.

A process has also been created for prospective contributors of the examination questions. The questions are the only proprietary component for the HPCCF – using restrictive license terms for authors while giving them credit. This is considered necessary for managing the database containing solutions to the examination questions.

In the next section, we discuss the process proposed for conducting the examination.

### 5.1 Examination Process

The examination process can be started at any time but a delay is employed before the examination actually starts. Firstly, a user has to register using their name, email address and optionally an affiliation. This process is explained, together with privacy policies, on the examination website. Next, they need to read a paragraph on the integrity policy defining cheating and encouraging honesty. Practitioners are asked to opt-in for the policy statement. Upon receiving this registration, we generate an encrypted token on the server that is returned to the user by email. Up to this stage, we do not store any information about the user on the server.

The email contains a link that will start the examination, and the user chooses to start the examination by clicking on it. Only then, the information about the user and the examination start time is stored in a temporary database.

The summative assessment itself is conducted using multiple choice questions. Therefore, for each skill, we will develop a pool of questions and answers, and accept external contributions. Each question has a pool of possible answers (e.g., 10). An examination

consists of randomly selected questions with five of their answers, and the practitioner has to decide if each statement is true or false.

Once the user submits the completed exam, the selection of answers is stored on the server for validation. At the moment, we envision the validation process includes a final manual step to trigger the certificate generation once provided with all information. If the user meets the pass criteria (typically 70% of correct answers) the earned certificate is issued and sent to the user by email. All personal information about the user is then deleted from the server, but the affiliation and raw responses are preserved. The affiliation is used for promotional purposes, while the raw responses will be analysed to optimise the questions. For example, if we identify that most users make the same mistake it may be an indication that either that question or its answer is too ambiguous and should be improved.

If the user didn't meet the pass criteria, they will be informed about their score. The user can then retry to obtain the certificate after a cool-down period (typically one week), but not immediately afterwards to prevent success via brute force methods. To enforce this, the information about when the exam was undertaken is linked to the user's name and email. As the sole purpose of the multiple choice questions is the examination, the incorrectly answered questions will not be revealed. We assume that high quality training materials available within the HPC community, and covering the individual skills being part of this certification program, offer constructive feedback and other ways of consolidating the newly acquired knowledge. Therefore, the Forum's only concern is to provide the mechanism of certifying whether the learners posses that knowledge or not; As opposed to pointing out gaps in their understanding.

## 5.2 Certificates

The examinees that pass the examination are awarded a corresponding certificate. Such certificate consists of two parts: a PDF and a text file. The PDF contains the key information making the certificate meaningful. An example of how it may look like is presented in Figure 2. The name and identifier of the certificate is found in the centre, on our example these are "HPC Driving License" and ID 1, respectively. Similar to a driving license, this particular certificate could provide the minimum set of knowledge required to understand and use a typical supercomputer.

The text file contains the same information, as well as a verification URL that can be given to a third-party to confirm the certificate's credibility. It is also PGP signed using the private key of the HPC Certification Forum to allow verification with the public key. An example file looks as follows:

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512
HPC Certification Forum Certificate
This text confirms that "Julian M. Kunkel" has
successfully obtained the certificate
"HPC driving license" (id: 1) at 02/2019.
Verification URL: https://hpc-certification.org/[...]
-----BEGIN PGP SIGNATURE-----
[...]
-----END PGP SIGNATURE-----
```


**Figure 2: Draft for an awarded certificate**

Together both documents should be able to provide enough credibility for the certification to be fully functional within the HPC academic and professional domains. Further improvements will be adopted as required.

# 6 ECOSYSTEM

The Forum encourages the development of an ecosystem around the HPC classification by supporting training and tool development. Many of its members are involved in those activities in a professional capacity, so the development of training materials and tools, and the development of the certification program have elements of co-design in them.

## 6.1 Training Delivery

The HPC Certification Forum is not developing training material directly or competing with providers of training material. However, we support individuals and institutions by endorsing and promoting their training materials and courses in two ways.

Firstly, an author is allowed to indicate on the training material itself or a related promotional material which skills are covered either fully or partially. We provide a seal that can be used for that purpose (see Figure 3). The reference to the HPCCF and the seal can be used free of charge under the condition that the developer of the training material registers a link to the material (or course) on our webpage using an online form. That way, the HPCCF is informed about the usage of the seal.

Secondly, we will link from our webpage the endorsed training material covering the individual skills and certificates. By using JavaScript and dynamic webpages, we will provide various views of the skills – with and without links to suitable training materials.

Note that we are not intending to verify the correct usage of the seal explicitly. However, in case the training material or course doesn't deliver the expected material practitioners may complain and we will remove the link to that training material from our webpage.

We expect that this strategy will make a good range of free training material available for most skills, while various institutions


**Figure 3: Draft for the seal for teaching material**

and individuals can still charge for effective training courses. Ultimately, the catalogued training materials will complement each other, leading to a rich variety of content suitable for each individual practitioner, e.g., addressing different learning styles and languages. This becomes increasingly important as more in-time personalised training in needed by our ever-growing HPC community.

### 6.2 Navigation

Thanks to the contributions of the PeCoH project[7], a JavaScript prototype embeds a navigable skill tree into a webpage. The script can be used by different stakeholders to adjust and present the skill tree in the most suitable for them view. By view we mean the selection and organisation of the skills, and the additional information provided when looking at the tree. For instance, you could have a tree for a specific scientific domain, for a specific data centre, or even for the role of the tester of a specific application. This feature could be used to indicate which skills are mandatory or beneficial for these use cases, providing links to training material or adding further supplementary information.

## 7 RELATED WORK

Relevant work can be divided into two categories: efforts to establish an HPC curriculum, and systematic efforts to design, develop and deliver HPC teaching and training materials.

In academia, many universities offer their own curriculum around scientific computing and HPC, as part of their undergraduate programmes, covering a range of theoretical and practical aspects relevant to the students field of study, e.g., software development of numerical domain-specific applications. Such courses rarely cover the efficient and effective use of HPC systems. The number of taught modules focusing on HPC is small, and the number of the whole programmes dedicated to HPC is even smaller. We are aware such programmes are run by: EPCC at the University of Edinburgh[8], the University of Liverpool[9], the International School for Advanced Studies (SISSA) and ICTP in Italy[10], the Trinity Collage Dublin[11] in Ireland, the Polytechnic University of Catalonia[12] and BSC in Spain and the University of Cote d'Azur[13] in France. These 1-2 year master programmes are fully accredited by the universities that run them, but that also makes them fairly expensive and their comprehensiveness is not suitable for the majority of HPC users.

Data centres offer their own teaching materials and run a number of training courses to support their own users. Although, these are becoming more varied in content and more accessible through the adoption of online, remote and asynchronous delivery methods, they are still not enough to fulfil the training needs of our community. Due to a variety of reasons some training opportunities are still only available at a specific time or place, or simply do not scale well enough to satisfy the demand.

Several projects, organisations and initiatives continue to put their efforts into developing and sharing of teaching HPC resources. The EuroLab-4-HPC project establishes training in form of online courses[14]. The Barcelona Supercomputing Centre (BSC) aims to develop a professional training curriculum [4]. Both XSEDE[15] (The Extreme Science and Engineering Discovery Environment) and PRACE[16] (Partnership for Advanced Computing in Europa), serving similar purpose in US and Europe, provide a variety of training opportunities and resources to make HPC more accessible to researchers across many scientific domains. The SciNet Certificate Program[17] provides course material for three major Computer Science topics: Scientific Computing, High Performance Computing, and Data Science each with several subcategories. Whereas the initiatives such as the HPC University[18], the Carpentries (the HPC Carpentry especially)[19] and ACM SIGHPC Education Chapter[20] strive to make sharing of teaching resources and practices simpler, and more accessible to trainers and practitioners alike.

Finally, many companies offer training courses making their products and services easier to use by their end-users, e.g. NVIDIA offering courses in deep learning[21] or Arm offering courses on various arm technologies[22].

## 8 CONCLUSION

The HPC Certification Program allows the re-use of existing content but also makes it possible to create a new ecosystem in which HPC centres, research labs, academic institutions and commercial companies could offer the best of their teaching material. According

[7] https://www.hhcc.uni-hamburg.de/pecoh.html

[8] https://www.epcc.ed.ac.uk/msc
[9] https://www.liverpool.ac.uk/study/postgraduate-taught/taught/big-data-msc/overview/
[10] https://www.mhpc.it/
[11] https://www.maths.tcd.ie/hpcmsc/
[12] https://masters.fib.upc.edu/masters/miri-high-performance-computing
[13] http://univ-cotedazur.fr/education/training?AIHPC18&lang=en
[14] https://www.eurolab4hpc.eu/
[15] https://portal.xsede.org/web/xup/training/overview
[16] http://www.training.prace-ri.eu/
[17] https://www.scinethpc.ca/scinet-certificate-program/
[18] http://hpcuniversity.org
[19] https://hpc-carpentry.github.io/
[20] https://sighpceducation.acm.org/
[21] https://www.nvidia.com/de-de/deep-learning-ai/education/
[22] https://www.arm.com/support/training

to our proposal, the existing and newly created teaching resources should be marked accordingly to indicated which skills they cover. In the future, the program may provide means to register and reference existing content of third-parties allowing users to browse the skills and navigate to teaching material.

### 8.1 Benefits

The program brings multiple benefits to everyone involved in HPC teaching and training. It's obvious that making clear what skills are required or recommended for a competent HPC user would be helpful to both the HPC service providers and practitioners. Training providers could bundle together skills that are most beneficial for specific user roles and scientific domains, which would allow practitioners to browse through skills to quickly identify and learn the skills required to perform their tasks. The variety of training offered within our community makes finding the right resources more complicated than it should be – the certification program will provide useful information where the desired skills could be learned. The examination confirming that a certain set of competencies has been acquired makes the learning process more complete and meaningful.

By participating in the program the HPC training providers can increase the visibility of their teaching opportunities and share their resources more effectively. The mapping of the skills defined by the program onto the existing training materials should also help to identify any potential gaps and improve the integrity of offered training. Finally, the certificates recognised by the whole HPC community simplify inter-comparison of independently offered courses and provide additional incentive for participation. Overall, the flexibility of the program allows to construct more personalised and just-in-time pathways to learning about HPC.

## ACKNOWLEDGMENTS

We are thankful for the contributions and discussions with the members of the HPCCF and for the contributions made by the PeCoH project particularly Nathanael Hübbe for developing tools. PeCoH was supported by the German Research Foundation (DFG) under grants LU 1353/12-1, OL 241/2-1, and RI 1068/7-1.

## REFERENCES

[1] Trevor Hussey and Patrick Smith. 2008. Learning outcomes: a conceptual analysis. *Teaching in higher education* 13, 1 (2008), 107–115.

[2] Julian Kunkel, Kai Himstedt, Nathanael Hübbe, Hinnerk Stüben, Sandra Schröder, Michael Kuhn, Matthias Riebisch, Stephan Olbrich, Thomas Ludwig, Weronika Filinger, Jean-Thomas Acquaviva, Anja Gerbes, and Lev Lafayette. 2019. Towards an HPC Certification Program. *Journal of Computational Science Education* (01 2019), 88–89. https://doi.org/10.22369/issn.2153-4136/10/1/14

[3] Neil C Rowe. 2004. Cheating in online student assessment: Beyond plagiarism. (2004).

[4] Maria-Ribera Sancho. 2016. BSC best practices in professional training and teaching for the HPC ecosystem. *Journal of Computational Science* 14 (2016), 74–77.

[5] DM Williamson. 2011. Good Practice Guide on Writing Aims and Learning Outcomes. *Queen Mary, University of London, Pub 7282* (2011).

# Project-Based Research and Training in High Performance Data Sciences, Data Analytics, and Machine Learning

Kwai Wong
University of Tennessee
Knoxville, TN
kwong@utk.edu

Stanimire Tomov
University of Tennessee
Knoxville, TN
tomov@icl.utk.edu

Jack Dongarra
University of Tennessee
Knoxville, TN
dongarra@icl.utk.edu

## ABSTRACT

This paper describes a hands-on project-based Research Experiences for Computational Science, Engineering, and Mathematics (RECSEM) program in high-performance data sciences, data analytics, and machine learning on emerging computer architectures. RECSEM is a Research Experiences for Undergraduates (REU) site program supported by the USA National Science Foundation. This site program at the University of Tennessee (UTK) directs a group of ten undergraduate students to explore, as well as contribute to the emergent interdisciplinary computational science models and state-of-the-art HPC techniques via a number of cohesive compute and data intensive applications in which numerical linear algebra is the fundamental building block.

The RECSEM program complements the growing importance of computational sciences in many advanced degree programs and provides scientific understanding and discovery to undergraduates with an intellectual focus on research projects using HPC and aims to deliver a real-world research experience to the students by partnering with teams of scientists who are in the forefront of scientific computing research at the Innovative Computing Laboratory (ICL), and the Joint Institute for Computational Sciences (JICS) at UTK and Oak Ridge National Laboratory (ORNL). The program also receives collaborative support from universities in Hong Kong and Changsha, China.

The program focuses on scientific domains in engineering applications, image processing, machine learning, and numerical parallel solvers on supercomputers and emergent accelerator platforms, particularly their implementation on GPUs. The programs also enjoy close affiliations with researchers at ORNL. Because of these diverse topics of research areas and backgrounds of this project, in this paper we discuss the experiences and resolutions in managing and coordinating the program, delivering cohesive tutorial materials, directing mentorship of individual projects, lessons learned, and improvement over the course of the program, particularly from the perspectives of the mentors. [1]

## KEYWORDS

Computational Science, Educational Outreach, Research Experiences for Undergraduates, Data Analytics, Machine Learning (ML), Hands-on Experiences and Education, HPC

## 1 INTRODUCTION

Computational science is an emerging field of study that is truly interdisciplinary, involving researchers from mathematics, computer/information science, and many domain science areas. Computational modeling and simulation have become indispensable tools in nearly every field of science and engineering. The RECSEM predecessor, called CRUSE (2013-2016), and the current RECSEM (2017-2019) programs give students a synergetic set of knowledge and skills that are useful for them to perform scientific research in HPC. These programs aim to deliver a synergetic hands-on research experience to the students by combining the expertise at the Joint Institute for Computational Sciences (JICS) [12] and the Innovative Computing Laboratory (ICL) [4] at the University of Tennessee, focusing at HPC simulation in engineering applications, emergent schemes of numerical mathematics, and state-of-the-art numerical linear algebra software, and data intensive computing. ICL is leader in enabling technologies and software for scientific computing, developing and disseminating high-quality numerical libraries like LAPACK, ScaLAPACK, PLASMA, and MAGMA [15].

The RECSEM program focuses on scientific domains in engineering applications, images processing, machine learning, and parallel numerical solvers on HPC and emergent platforms. Figure 1 shows the principle idea of the REU program. In general, the program starts with a two-week training session, introducing the students to the supercomputing environment and the common computational methods and tools to be used later. Each student is assigned a project complemented to his/her academics background and computing skill level and solves a computational modeling problem under the supervision of a team of mentors and advisors.

From 2013 to 2018, these programs have admitted a total of 92 students. Forty of them are international students from our four collaborating institutes from Hong Kong and three local students are supported under a separate REU grant from other colleges at UTK. The CRUSE and RECSEM programs have attracted students from 28 different colleges across the nation. Out of the 52 domestic students, 15 are women and 11 are African Americans (3 females). The students worked on a total of 55 different research projects with a total of 23 different lead advisors and 18 mentoring research staff and student associates at JICS and ICL. The program also enjoys tight collaborations with researchers at the Oak Ridge National Laboratory (ORNL). Given the scope of activities and size of students

[1] This paper describes in detail the work presented at the ISC'19 *Workshop on HPC Education and Training for Emerging Technologies* [17].


https://doi.org/10.22369/issn.2153-4136/11/1/7

and staff in making this program a fruitful experience for the participants, we discuss the experiences and resolutions in managing and coordinating the program, directing mentorship of individual projects, and lessons learned via exemplary data science projects building on native linear algebra from ICL.

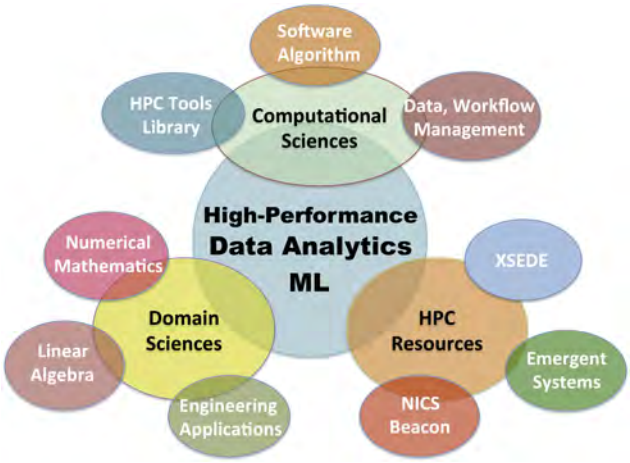



**Figure 1: Design of the RECSEM Computational Science program.**

## 2 PROGRAM DESIGN AND PLAN

The CSURE program started in 2013 and lasted for four years. The revised RECSEM program streamlines the operation and begins in 2107. These programs draw from the computational sciences experiences of JICS staff and the expertise of numerical linear algebra building on the HPC platform from ICL.

The principle goal was to promote the ability of undergraduate students to succeed in a research-oriented program in computational sciences. *Hence the REU programs seek to mimic the pace and intensity of graduate-level or industrial-level projects with well-defined deliverable deadlines.* The intention is to provide the participants a good knowledge of how a graduate project is organized and executed. In addition, its intellectual focus is not only to push for publishable research outcomes, but also to expose the students to research experiences through appropriate levels of motivations and accomplishments. These are major reasons we choose to do a ten-week long research program, giving students enough time to master the skills in accomplishing their research goals.

While the primary goal of these programs is to develop students' interest in pursuing research careers in computational sciences, we also provide strong professional development, post-program development opportunities, and social networking for the REU participants among themselves. Students are encouraged to continue their research activities at their home institutions afterward. There are several major tasks that the students are asked to follow. These tasks start with an informal in-class presentation, a midterm lecture presentation, an open poster presentation, and conclude with a final presentations and a final report in the last week. These tasks aim to gradually assist the students towards finishing their research goals in time. A detailed listing of the program is available at the program's webpage, www.jics.utk.edu/recsem-reu.

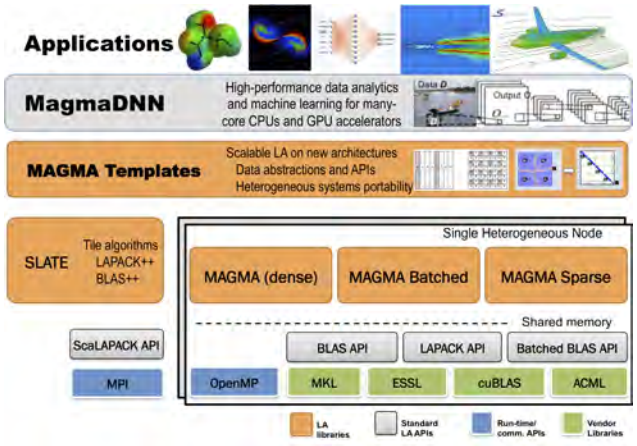



**Figure 2: Software stack for high-performance data sciences in RECSEM using linear algebra, data analytics and ML**

### 2.1 Schedule of the REU program

To deliver such a diverse program, a well-planned step-by-step schedule for the entire summer is desirable to be in place by early December. Event items for the preparation period include logistical arrangements, program announcement and recruitment, selection of students, payroll registration, social activities, preparation of training materials, evaluation instrumentation, mentor selection and training, and most importantly identification of research projects and mentoring teams. Following that will be the ten-week summer program starting the first week of June and ending the first week of August. A typical daily schedule for the last three years can be found on the RECSEM webpage [10]. A typical timeline of the program is listed in Table 1.

*The last week of the program is reserved for reporting, presentations, surveys, and meetings with students. It is important to have a detailed check-out list for each student and a cordial discussion session with each student.* The discussion session involves soliciting general impressions from each student, including upsides and downsides of the program, ideas for improvement, and future opportunities for project work and graduate school. These discussion sessions provide valuable insights to the advancement and improvement of the program.

### 2.2 Recruitment and student selection

The NSF Computer and Information Science Engineering (CISE) directorate has a joint recruitment program for REU students [11] but the program opts to do additional recruitment because of the diverse, interdisciplinary nature of the program. We rely on recruitment through emails and contacts with collaborative institutes of JICS and ORNL, particularly with an established outreach partner, Morehouse College in Atlanta, and the campus champions of the XSEDE program. Many of the applicants are highly recommended students through the contacts of our collaborators.

**Table 1: Timeline of the REU program**

| | |
|---|---|
| Jan.-March | Student recruitment and research project identification |
| March | Student selection and research project selection |
| April-May | Prepare training materials, setup research plan, post detailed schedule |
| May | Mentor training, prepare reference materials, coordinate travel & logistics |
| First day | Goal statement, projects assignment, schedule, survey, social issues, Q&A |
| $1^{st}$ week | Training and hands-on workshop, meeting mentor, define and formulate research goals and plan of projects |
| $2^{nd}$ week | Students finalize research plan with mentors, 1st social gathering |
| 2–$4^{th}$ week | Preliminary study, in-class presentation |
| $5^{th}$ week | Mid-term presentation, 2nd social |
| 6–$8^{th}$ week | Research and HPC implementation, final poster |
| $9^{th}$ week | Prepare for final presentation, extending work |
| $10^{th}$ week | Final presentation, project report, concluding |
| Last day | Survey, Q&A, retrospective movie, summary |
| August | Summarize results, follow up with students for possible extended work |
| September | Survey report, final report, project continuation |
| October | Final NSF yearly report submission |

Candidates considered for the program fill out an application form and write a short essay describing their background, interests in science, and their goal statements. This information is used to select students and then to assign them to work on the proposed core science domains, to ensure that the specific proposed projects are beneficial to the students and matched to their interests, background knowledge, and skills.

Student selection is not always a straightforward process because of the diverse, multidisciplinary nature of this program and the challenge in finding participants that match for the various research topics. A group of mentors and the PI team meets to iterate over the applicants, ranking the students for their suitability to the program and the research topics. The deadline for applications is in the late February but generally moved back depending on the need for more applicants interested in specific research topics.

Participants are selected based on three major factors: the nature of their home colleges, their interests and background, and their letters of intent and references. Students from smaller schools with fewer research opportunities are preferred in order to expand the national research community. Rising senior students are preferred. GPA is a deciding factor only if two candidates have comparable qualifications. Over the course of CSURE and RECSEM, we have not seen that GPA is necessarily predictive of success with the program.

Two students are usually assigned to work on one research subject. Each pair starts off together but often splits up to work towards separate research aspects of the same topic at the midpoint of the program.

Acceptance letters will be sent out as soon as the first deadline is passed. Getting a written commitment from each accepted student is important. A second set of acceptances is always needed as there are always students declining to attend. Declination letters also need to be sent out in a timely manner; however, itâĂŹs wise to keep in contact with a few applicants in case of unexpected availability. There are cases that students withdraw late in April for sundry reasons.

## 2.3 International students

Having a group of foreign students is an enlightening element to this program. The goal is to bring together students from different backgrounds in cultural thinking and education pathways, hoping to broaden the perspectives and understandings of ways of approaching solving problems. Hong Kong students have participated in the UTK summer research program under the direction of Dr. Kwai Wong over a decade. The international students are funded by their universities. Unlike the selection of the domestic students, these students are selected competitively primarily based on their academic achievements. The students are usually highly academically motivated and look for attending graduate school in the US, a fact confirmed by tracking their options after graduation. Pairing the domestic and the foreign students is done whenever a project permits. We had six pairs of students, so far and all of them worked out wonderfully, complementing each other in research efforts. Foreign students generally have better methodical skills while domestic students are more resourceful and investigative. Overall the domestic students are impressed by the mathematics and algorithmic training of others, while the foreign students gain tremendously in open minded ideas and team efforts. As the program goes by, the students mix extremely well and enjoy sharing thoughts as well as social activities beyond the academic ones, such as cooking, music, and playing video games together.

## 2.4 Logistic support and social activities

There are complicating issues for the summer program. The program includes students from a foreign country. It has mentors from UTK and ORNL. The students will have access to supercomputers at NICS and XSEDE [16]. Conference and meeting rooms have to be arranged. Visitor badges to ORNL must be processed. The students each have separate travel plans. Housing must be arranged. Reservation of venues for the planned activities such as group photo sessions, lecture presentations, poster presentations, and social gatherings are done early to ensure availability. All of these issues

require timely efforts, coordination, patience, and most importantly, a good supporting team for a successful program.

It is hard to foresee some of the logistic issues for a new program and usually takes two to three years to find good solutions for them. We recognize that getting the support from the school and the involvement of the research office do help tremendously.

Housing for students is the most urging logistical issue to be resolved and must be prepared early. The entire group should stay together in the arranged housing to help them to blend together socially. It helps to grow a solid bond among the students by organizing group activities and encouraging the students to create their own activities. We determine the housing in early February and proceed to place the students as soon as we have finalized the list.

Over the years, we have tried many options to minimize the cost but eventually settled in on-campus housing at a reduced rate negotiated with the help of our research office. In addition, a number of offices and meeting rooms are arranged to host the students in close proximity. Co-locating the entire group and student helpers in one or two rooms strongly enhances the cohesiveness of the program. There are also meeting rooms available nearby for private discussions. JICS and ICL have reserved three large offices and two conference rooms for the REU students. These rooms are located side by side in the same hallway where Kwai Wong's office is, making him readily available for any questions anytime.

JICS and ICL have research staff, students, and administrative staff. The JICS mentors include UT faculty, staff members in ORNL research groups, and joint faculty with appointments at both UTK and ORNL. The REU program has benefited tremendously from this infrastructure and staff support.

The program starts off with a campus walk and a group lunch on the first day. There are also two organized gatherings in the apartment complex. The students also participate in activities organized for undergraduate summer interns by the UTK office of research. Such activities include a tour of the Neyland football stadium and a few luncheon talks about graduate school application and scholarship information. Some highlight activities include a trip to the Great Smoky Mountain National Park or the Fall Creek Falls State Park, the Knoxville zoo and a tour to the Spallation Neutron Source facility at ORNL. These social activities help to bring the group together and improve morale.

Importantly, we have arranged a local student to serve as the lead of the group, helping the group to resolve some of the logistic issues in town.

### 2.5 Computing resources

The JICS facility represents an investment by the state of Tennessee and features a state-of-the art auditorium, conference rooms, and suites for students and visiting staff. It also provides the access to different parallel computing platforms available at NICS and XSEDE [16]. The ICL has expertise in the fundamental building block of numerical libraries on HPC systems, with emphasis on GPUs. In particular, RECSEM uses the MAGMA libraries to build new data analytics and ML capabilities, e.g., MagmaDNN [1–3, 6–9, 13], as well as computational support for applications in various fields, as illustrated on Figure 2.

**Table 2: Timeline of the program**

| Stage | Week | Project targets |
|---|---|---|
| Training | 1.5 | Lectures, exercises, research skills |
| Science Study | 1.5 | Overview and set research plan |
| Formulation | 1.0 | Objectives and algorithm, short talk |
| Prototyping | 2.0 | Description, midterm presentation |
| Implementation | 2.0 | Results, poster presentation |
| Concluding | 1.0 | Final presentation and report |

In the RECSEM program, we turn to XSEDE to support the computing need for the research projects. An educational allocation is obtained to access resources in PSC, SDSC, and TACC. Such arrangement has huge impact to the multi-discipline nature of the program that we organize, not just the variety of hardware platforms ranging from traditional core-based component to various types of accelerators, but also the availability of software and the interactive access for development and testing purposes. Matlab is openly available on XSEDE's bridges system, which helps tremendously. The GPU platform available on XSEDE's comet and bridges platforms provide excellent computing platforms for data science projects. In addition, we have also arranged individual multicore workstations fitted with a low end P100 compatible GPU card used for code development. These workstations provide a good alternative to accessing supercomputing remotely.

## 3 RESEARCH WORK AND MENTORSHIP

This REU program addresses the growing importance of computational sciences in many advanced degree programs. The agenda of the program is organized around a synergistic set of ideas and practices that are common to many scientific domains. The focus of the projects leverages the multidisciplinary expertise of the staff in JICS, UTK, and ORNL.

In order to provide students with the most valuable and realistic experience in computational sciences we have identified several different areas of significant interest and expertise within our organization. A participant will select a scientific area in which he/she would like to be involved. Students are paired to work as a team together with their assigned scientific mentors and advisors.

One of the major theme in the RECSEM program is to deliver the fundamental concepts of numerical linear library which is the major building block of computational intensive and data driven sciences. We will provide exemplary data science projects using the home grown numerical libraries, LAPACK and Magma. The other major theme is to deliver a set of software tool and workflow frame that can facilitate the development and launching different simulation applications on scalable HPC platforms. These projects breed cross interactions among team of students as well as promoting computation performance of simulating programs.

### 3.1 Stages of the research plan and mentor experiences

The schedule of the research program is organized into six progressing stages shown in Table 2.

The program begins with a kick-off meeting to highlight the agenda of the program and introduce the team of researchers and staff working in the project. A set of tutorials containing a series of lecture materials and a clear calendar of schedule of work and activities is listed on the program website [10] and is available to the participants at start.

The first day is reserved for payroll paperwork, initial survey, introduction of students, exchanging email addresses, Q&A, introducing a local student team leader, and a campus walk. *A list of safety reminders, health concerns, complaints, and emergency contact information is discussed in detail.* During the program sessions, occasional health issues arise and absence and sick policy will be given.

The first stage of the program includes an in-depth introduction to the use of supercomputers, including programming languages and compiling procedures, batch queuing systems, and I/O tools. Training activities include classroom instruction, hands-on exercises, research and modeling design, and computational studies. Tutorials come with hands-on exercises that put them to work as teams. Recognizing there is an uneven level of expertise in computing, we always pair the team up to compensate for their knowledge in computer and domain sciences. The introductory sessions intersperse lectures with discussion questions, emphasis on group effort on problem solving, and hands-on exercises.

Research topics are assigned to students ahead of time; however, should they change their mind, they may do so in the first week. An important task of the first week is to give specific assignments to students to help them begin making progress on their research topic.

The second week of work includes an introduction to the domain science areas and the specific project content assigned to each team of students. This involves hour-long talks by the subject mentors. We avoid asking students to spend time on learning materials that they will not use. The rest of the week moves to scientific study with literature review, reading and discussing relevant articles, and hands-on practice with relevant computational methods and tools. The first two weeks are crucial to set the right path for each project of the program. The PI team will discuss with each project mentors and determine if the set of goals of a project is not clearly defined or too difficult for the students to achieve in ten weeks. The discussion and meeting with mentors will also help the students to draw their research plan.

In later stages, students start to identify their research plans under the direction of their mentors. Every student will conclude the research plan and project goal in three weeks. Student progress toward their planned goals is evaluated frequently during the program. Mathematical formulation and algorithmic prototyping and testing are then followed. The last week will be reserved for concluding the project, presenting the final results, and finishing the final report.

## 3.2 Progress oversight and deliverables

*The program has five deliverables.* These are designed to steer the students to finish their projects on time. The timeline of these deliverables is listed clearly on the webpage and emphasized in the first week of the program. *The first deliverable is a short in-class summary talk* of the research topic and the approach. *The second deliverable is an open presentation* of their research work and initial results. It is aimed to orient the students to focus on their works, help crystallize the approach, and make students aware of the project timeline.

*The third deliverable is a public poster presentation*, organized with other groups of REU students. The posters help students to organize their results. Students also have the opportunity to review other projects and potentially seek ideas to improve theirs.

The last two weeks of the program have the students working toward *concluding their projects with a final presentation.* Each presentation lasts for 40 minutes and usually receiving a number of questions from their peers and attendants. Final presentations are great experiences for the participants and represent a concluding milestone for their research endeavors.

*The last piece of work is a report.* This is, in fact, a continuous process from the beginning, with students organizing their weekly summaries and articulating their results in detailed reports. Each student is encouraged to keep a weekly summary report. The final report will be a combined work that documents the studentâĂŹs progress and findings. *Yet in fact, this turns out to be the most demanding part of the 10-week program. Hence, it is very important to keep reminding students throughout the program to work on documenting their efforts and results. Every presentation and report of the projects in the REU program can be found on www.jics.utk.edu/recsem-reu.*

## 3.3 Research projects

The research topics available for the participants span a wide range of scientific and engineering domains yet follow a synergistic theme of numerical computation or data analytics. Each of the areas corresponds to significant capabilities at UTK or ORNL with active researchers and projects. All projects include hands-on experience and use of parallel computing in the various scientific and engineering domains of the program.

Research projects are selected based on the expertise of the core team of mentors and the backgrounds of applicants. Descriptions of previous research projects, from traditional domain sciences to cross-disciplinary data computation are listed in the following sections.

Overall, the projects selected in the RECSEM program circle around a synergistic theme of simulation and implementation on the HPC platforms. They involve many cross team collaboration and effort. Although a team of students concentrates on their own project, they are exposed to research elements that help them to explore helpful ideas and seek suggestions from other teams to expand or advance their research scope.

*3.3.1 Data Analytics and Machine Learning.* A common theme for all projects is the use of high-performance numerical libraries, data analytics, and machine learning. Several projects are specifically targeting the development of such capabilities. Examples are the development of MagmaDNN [1–3, 6–9, 13], a high-performance data analytics and deep neural networks (DNN) framework for

manycore GPUs and CPUs. Students learn state-of-the-art algorithms and performance optimizations techniques for data analytics and machine learning, implement them in open source library, and also help other students use these capabilities for data-driven science projects. Projects have included the development of the MagmaDNN DNN framework [7], extensions with convolution algorithms [2], including Winograd [6], mixed-precision FFTs using the new FP16 Tensor Cores units on Nvidia GPUs [3, 13], parallelization and addition of new features [9], hyper-parameter optimization framework [1], and scalability improvements [8]. A number of applications using DNN in general start off with using Keras [5] and TensorFlow [14] for algorithmic development and move to using MagmaDNN to enhance programming efficiency on HPC platforms. Such cross team interaction among students encourage exchanges of ideas and inject values of research collaborations. Students with strong CS and Mathematics background are assigned to these projects.

*3.3.2 Computational Engineering and Sciences Applications.* Engineers have been using supercomputers to analyze and resolve many challenging problems for many years. Nowadays, computer simulation has become a mandatory step in the process of design and development for many industrial applications. Projects completed by the participants include climate and pollution transport simulations, biomechanics modeling, traffic flow computation, and power system evaluation.

Computational chemistry, physics, and geography have big footprints on large scale supercomputers. Participants have completed a number of projects in quantum mechanics, molecular dynamics, neutron image reconstruction, and GIS modeling.

In general, these projects fall in the category of solving systems of ordinary or partial differential equation of the conservation laws systems. A variety of open source community or commercial codes are used. The goals are primarily to analyze the response of specific modeling systems and enhancement of the simulation methodology on the HPC systems. Although students will not involve in developing the code, they are required to understand the models, the mathematical equations and the numerical implementation of the project. As the simulation is carried out on HPC platforms, the usage and understanding of various numerical libraries will determine the performance of the simulations. These projects will work well for students in engineering and applied mathematics.

*3.3.3 Numerical Mathematics and data science projects.* Numerical mathematics is the building block of computer simulation of every scientific application. A science problem can usually be modeled by a set of mathematical equations and then numerically solved on computers. The effectiveness of these solvers is often determined by the combination of the specific choice of numerical schemes and implementations, which is particularly true on HPC platform. A theme of the research projects is to develop efficient numerical schemes for equation-based and data-based applications, generally needed in a lot large-scale engineering simulations. Projects completed by the participants include implementation and comparison of continuous and discontinuous discretization formulations, machine learning algorithms for images and signal processing problems, topological analysis of high-dimensional data, variational inequity problems, and functional MRI, electroencephalogram (EEG) and traffic statistical data analytics. Students with strong computing or mathematics and background will be good candidates for these type of projects.

*3.3.4 Software Tuning and Implementations.* Linear algebra is the backbone of HPC. These are the core computing functions used in almost every project of the REU program. Projects completed by the participants include mixed precision parallel dense solve implementation on GPU and multi/many-core CPU processors, randomized SVD calculations, Fast Fourier Transform implementation on the new Nvidia Tensor Core architecture. Large scale simulations in engineering and sciences applications are mostly composed of a sequence of functional steps that can be done either serially or concurrently. These steps or programming modules often involve I/O tuning, optimization of numerical libraries, piping the output for data analysis or visualization, and re-submission of jobs that constitute a cohesive workflow procedure. One of the core projects of the REU program is the construction of an efficient parallel workflow framework that is suitable to launch both computational and data intensive types of job on HPC platforms. These projects will require students with senior level of programming skill. These projects always involve the transfer knowledge of other applications to design and showcase their implementation.

## 3.4 Mentorship

The lead mentors are designated persons committed to the program. Mentors are selected based on their availability and commitment. They are leading researchers in their domain science working at UTK and/or ORNL. The team of mentors defines the major element of success of the program. They are chosen early and are involved in the selection of students. The student research projects vary every year but fall in the scope of the major program subject areas. In general mentors meet with their students at least twice a week and are available for questions. Graduate students of the mentoring team are in general also available to provide constant guidance and direction to the students. Given the reality that travel for conferences, reviews, or other purposes makes it likely that mentors will be occasionally absent, having additional advisors is important to ensure steady progress. General oversight of the research progress by the program director is also recommended. Regular discussions between the program director and the mentoring team are also helpful.

RECSEM enjoys a large pool of mentors with suitable discipline of expertise as well as new project ideas from UTK and ORNL. As the nature of computational sciences slowly migrate to other emerging topics, one or two new mentors are solicited to enhance the diversify of the research scope. The PI team bears the responsibility to explain in details the nature of the program. The scope of the RECSEM program involves element of computing that an undergraduate student has the ability to understand and expand, which is a primary criterion to be judged by the PI team. Overall, the PIs will determine if they or another mentor is available to help out in case the primary mentor is out for unforeseeable reason.

Mentors or their graduate students usually meets at least once a week, likely more often at the beginning of the project. The PI team will also be present in the first few meetings to give general guidance

and make suggestions particularly in computing requirement and resources.

## 4 ACCOMPLISHMENTS, CHALLENGES, AND LESSONS LEARNED

### 4.1 Survey

Evaluation of the program is centered on the toolkit distributed by the NSF CISE REU program as published by the University of North Carolina, Charlotte [11]. The evaluation provides the mentoring team with regular feedback for ongoing assessment of the program via in-person meetings along with formal mid-program and annual reports. Reports include evidence-based recommendations for program improvements in the form of clear actions items that program directors can apply directly to further program improvement. A final summary report examines and determines to what extent the program succeeded in meeting its stated goals.

*Surveys for the students are performed at the start of the program and at the end of the program. We use the standalone A La Carte student survey from the CISE REU toolkit [11].* In order to evaluate the project's impact on participants, students are given pre- and post-evaluation surveys that assess their attitudes toward and interests in computational science, as well as their knowledge of computational science and its use in the domain focus area. The results of these surveys each year guide modifications to the project for future years. Surveys and summative evaluations are independently instrumented either professionally by a contract agency or a person that is familiar with the process. The REU program engaged Dr. Christian Halloy, a retired computational science leader to conduct the summative program evaluations. Dr. Halloy conducted pre- and post-participant surveys, a personal discussion with each participant, and provided a detailed final report. He also attended and critiqued the progress of the students' final lecture and poster presentations.

In summary, U.S. students' scores for survey constructs of self-efficacy, graduate school intentions, computing attitudes, help-seeking and coping, scientific leadership, and scientific identity were favorable at the start and end of the program (means above 4.00 on a 1 to 5-point scale for pre- and post-surveys). The largest improvement gain for REU participants after the 10-week period was found for the research skills and knowledge scale with a mean increase from 3.9 to 4.4. Overall, participants were satisfied with the program with M = 4.25, and their mentor M = 4.12.

Participants rated their mentors quite highly for all the indicators, the highest average score being for "approachable" (M = 4.89), while the lowest average score is seen for "accessible" (M = 4.56) which is nevertheless quite high per se.

In general, the following recommendations were provided as examples of practices the REU may consider to include or maintain to ensure continued and future success of the program.

(1) Expand the evaluation to include feedback from additional stakeholder groups (i.e., faculty advisors/mentors and program administrators) in order to gain an additional understanding of the REU program.
(2) If possible, create a system to follow the student participants over time to assess additional project impacts on a long-term basis. (e.g., graduate school attendance, career choice, presentation and publications, awards and honors, etc.).
(3) Continue to integrate strategies that will enhance the experience across diverse backgrounds, considering that students in the program possess differing academic backgrounds and research preparation.
(4) Carefully recruit faculty and graduate students who will be available throughout the duration of the program. Consider a back-up strategy to support students if a volunteering mentor is unavailable during parts of the program.
(5) Continue to include and potentially increase hands-on instruction at the beginning of the program to engage and motivate participants.
(6) Continue to provide opportunities for students and mentors to network at the start of the summer and throughout the research experience.

### 4.2 Challenges and lessons

The success of the program builds on the foundation of many elements. The most important one rests on the cohesiveness of the program and the mentorship.

The program lasts for ten weeks, a good plan before the start of the program is essential for every project to set up a solid research path by the end of the third week. The selection of topics and mentors for the students with the right background will be an important factor.

Although The RECSEM program enjoys the variety and a diverse pool of mentors, the expertise and directorship of the PI team is important to thread the projects together under a synergistic theme of idea. Being said, the selection of projects and matching projects to a diverse background of students requires efforts of thoughtful examination and discussion with mentors. The following list gives some helpful insights to a typical REU program.

(1) Each year program and projects could vary but would work out best to set a common goal to unite the idea;
(2) The PI team has basic level of knowledge to give advice to every single project throughout the program;
(3) Targets of work has to set in place after three weeks of the program to get research work finish on time. Do not hesitate to work with the mentors to ensure its timeline if problems arise;
(4) Pair students with comparable skills, not GPA, that can complement each other. Often REU program enjoys the pool of large amount of students that could be matched;
(5) Listen to what students suggest and comment early, and make adjustment as needed, switching or adding student partners when needed;
(6) Avoid selecting project without a specific goal, unachievable goal, nor too many tasks;
(7) Prepare to spend extra time with the students to sort out problems and question. The PI of the RECSEM program meets with the entire group of students every morning at 9:00am;

(8) Complement skills set from other team to augment the deficiency of some teams. Identify a list of students that could help out in programming, software tools, mathematics, or some common topics. After all, it is a group activity, more interactions are better. If possible, prepare a group of local undergraduate students to help out in programming.

### 4.3 Program outcomes and impacts

The success of the program counts on dedications and efforts of our mentors. We have instituted a total of 73 different projects. Selection and availability of mentors are constant subjects of concern even we enjoy having a large pool of volunteer scientists. As this REU program continues, we learn to streamline the dimension of projects and maintain the core subject areas the team of resident mentors and PIs are familiar with. Often the program director has to be prepare to spend over half of his time a day answering questions for the entire group.

Parallel computing to many participants has a steep learning curve, pairing students in their comfort knowledge backgrounds is essential to get a project done in time. In addition, to avoid duplication effort within a team, we often design a team project with multiple themes allowing every student has his/her research own contribution.

Human dynamics, emotion, frustration, and conflicts among students, however rare, are unavoidable issues. Listening, patience, caring, and professionalism are appropriate answers to most. After all, we put research experiences as the primary theme of the program. Having international students gives a good mix of cultural interaction, in fact, improves overall group dynamics.

Over the last seven years, we have instituted a multidisciplinary computational sciences REU program that encompasses 73 different projects, including a total of 124 students from 39 colleges. This program has established a continued relationship with undergraduate institutions such as Morehouse College in Atlanta, Maryville College in Knoxville, New Mexico State University in NM, and Slippery Rock University in PA. This is important in sustaining long-term viability of the REU program, which can continue to evolve and improve from listening the feedback and suggestions from our partner colleges. The outcomes of the studentsâĂŹ research work included six sponsored conference presentations, three conference papers and a number of conference and journal papers to be submitted. A list of their reports is posted in the RECSEM website [12]. Close to 75% of the students have gone to or are applying for graduate schools. The program director has maintained yearly contacts with the participants. This is important to our sponsor. It helps to track the progress of the students and overall impact to the REU program.

## 5 CONCLUSIONS

This REU program intends to provide participants with an experience with a similar level of effort as in graduate school. The program provides students an exposure to research with high performance computing applied to a variety of scientific applications. In three summers, we have resolved many problems and met even more challenges. In particular, the following items summarize the highlights of the program:

(1) A well-defined step-by-step timeline leading to the end of the program is in place in early December.
(2) The participants are selected based on three major factors: the nature of their home college, their interests and background, and their letters of intent and references.
(3) The project assignments are sent to students ahead of time.
(4) Getting a written commitment from each enrollee is important.
(5) A midterm preliminary presentation of the research topic and the approaches of the research, is very important.
(6) Housing for students must be prepared in the early stage of the program. The entire group stays together in the arranged housing to get them to blend together socially.
(7) A program director is important, with regular availability to the participants.
(8) Co-locating all students and helpers in a multi-purpose lecturing room enhances the cohesiveness of the program.
(9) A list of safety reminders, health concerns, and emergency contact information is discussed in detail in the first day.
(10) An effective team of mentors represents a major element of success of the program. They are chosen early and are also involved in the selection of their students.
(11) We have arranged a local student to serve as the site lead to the group, particularly for social activities.
(12) The most demanding part of the 10-week experience is the final report. The program director should keep reminding participants and constantly check for progress.
(13) A detailed checkout list for each student and a meeting with each student before the program ends are needed.
(14) Surveys for the students are performed at the start of the program and at the end of the program.

## ACKNOWLEDGMENTS

This work was conducted at the Joint Institute for Computational Sciences (JICS), sponsored by the National Science Foundation (NSF), through NSF REU Award #1262937 and #1659502, with additional Support from the University of Tennessee, Knoxville (UTK), and the National Institute for Computational Sciences (NICS). This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562. Computational Resources are available through a XSEDE education allocation award TG-ASC170031.

## REFERENCES

[1] Frank Betancourt, Kwai Wong, Efosa Asemota, Quindell Marshall, Daniel Nichols, and Stanimire Tomov. 2019. OpenDIEL: A Parallel Workflow Engine and DataAnalytics Framework. In *Practice and Experience in Advanced Research Computing (PEARC '19)*. ACM, ACM, Chicago, IL.
[2] S. Chen, A. Gessinger, and S. Tomov. 2018. *Design and Acceleration of Convolutional Neural Networks on Modern Architectures.* Technical Report. Joint Institute for Computational Sciences (JICS), UTK. 2018 Summer Research Experiences for Undergraduate (REU), Knoxville, TN, 2018.
[3] Xaiohe Cheng, Anumeena Sorna, Eduardo D'Azevedo, Kwai Wong, and Stanimire Tomov. 2018. Accelerating 2D FFT: Exploit GPU Tensor Cores through Mixed-Precision. (11-2018 2018).
[4] ICL [n. d.]. Innovative Computing Laboratory (ICL). http://icl.cs.utk.edu.
[5] Keras [n. d.]. Keras: The Python Deep Learning library. https://www.keras.io.
[6] Lucien Ng, S. Chen, A. Gessinger, D. Nichols, X. Cheng, A. Sorna, Kwai Wong, Stanimire Tomov, Azzam Haidar, Ed DâĂŹAzevedo, and Jack Dongarra. January, 2019. MagmaDNN 0.2: High-performance data analytics for manycore GPUs

and CPUs. In *MagmaDNN, 2018 Summer Research Experiences for Undergraduate (REU)*.

[7] Lucien Ng, Kwai Wong, Azzam Haidar, Stanimire Tomov, and Jack Dongarra. 2017. MagmaDNN – High-performance data analytics for manycore GPUs and CPUs. In *MagmaDNN, 2017 Summer Research Experiences for Undergraduate (REU)*.

[8] Daniel Nichols, Natalie-Sofia Tomov, Frank Betancourt, Stanimire Tomov, Kwai Wong, and Jack Dongarra. 2019. MagmaDNN: Towards High-Performance Data Analytics and Machine Learning for Data-Driven Scientific Computing. In *ISC High Performance*. Springer International Publishing, Springer International Publishing, Frankfurt, Germany.

[9] Daniel Nichols, Kwai Wong, Stanimire Tomov, Lucien Ng, Sihan Chen, and Alex Gessinger. 2019. MagmaDNN: Accelerated Deep Learning Using MAGMA. In *Practice and Experience in Advanced Research Computing (PEARC '19)*. ACM, ACM, Chicago, IL.

[10] RECSEM [n. d.]. RECSEM REU Program at UTK. http://www.jics.utk.edu/recsem-reu.

[11] REU [n. d.]. Research Experience for Undergraduates (REU): Socially Relevant Computing. https://reu.uncc.edu/cise-reu-toolkit/results-cise-reu-toolkit.

[12] REUReports [n. d.]. Computational Science for Undergraduate Research Experience, 2013-17 internal reports. http://www.jics.utk.edu/csure-reu/csure13/projects,
http://www.jics.utk.edu/cure-reu/csure-14/projects,
http://www.jics.utk.edu/csure-reu/csure15/projects,
http://www.jics.utk.edu/csure-reu/csure16/projects,
http://www.jics.utk.edu/recsem-reu/recsem17/projects.

[13] A. Sorna, X. Cheng, E. D'Azevedo, K. Wong, and S. Tomov. 2018. Optimizing the Fast Fourier Transform Using Mixed Precision on Tensor Core Hardware. In *2018 IEEE 25th International Conference on High Performance Computing Workshops (HiPCW)*. 3–7. https://doi.org/10.1109/HiPCW.2018.8634417

[14] TensorFlow [n. d.]. TensorFlow: An end-to-end open source machine learning platform. https://www.tensorflow.org/.

[15] S. Tomov, J. Dongarra, and M. Baboulin. 2010. Towards Dense Linear Algebra for Hybrid GPU Accelerated Manycore Systems. *Parellel Comput. Syst. Appl.* 36, 5-6 (2010), 232–240. DOI: 10.1016/j.parco.2009.12.005.

[16] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. R. Scott, and N. Wilkins-Diehr. 2014. XSEDE: Accelerating Scientific Discovery. *Computing in Science Engineering* 16, 5 (Sep. 2014), 62–74. https://doi.org/10.1109/MCSE.2014.80

[17] Kwai Wong, Stanimire Tomov, and Jack Dongarra. 2019. Hands-on Research and Training in High Performance Data Sciences, Data Analytics, and Machine Learning for Emerging Environment. In *ISC High Performance, Workshop on HPC Education and Training for Emerging Technologies*. Springer International Publishing, Springer International Publishing, Frankfurt, Germany.

# Computational Biology as a Compelling Pedagogical Tool in Computer Science Education

Vijayalakshmi Saravanan[i]
Rochester Institute of Technology
Rochester, USA
vsavse@rit.edu

Anpalagan Alagan
Ryerson University
Toronto, Canada
alagan@ee.ryerson.ca

Kshirasagar Naik
University of Waterloo
Waterloo, Canada
snaik@uwaterloo.edu

## ABSTRACT

High-performance computing (HPC), and parallel and distributed computing (PDC) are widely discussed topics in computer science (CS) and computer engineering (CE) education. In the past decade, high-performance computing has also contributed significantly to addressing complex problems in bio-engineering, healthcare and systems biology. Therefore, computational biology applications provide several compelling examples that can be potent pedagogical tools in teaching high-performance computing. In this paper, we introduce a novel course curriculum to teach high-performance, parallel and distributed computing to senior graduate students (PhD) in a hands-on setup through examples drawn from a wealth of areas in computational biology. We introduce the concepts of parallel programming, algorithms and architectures and implementations via carefully chosen examples from computational biology. We believe that this course curriculum will provide students an engaging and refreshing introduction to this well-established domain.

## Keywords

Pedagogical Tools · High-Performance Computing (HPC) · Parallel and Distributed Computing (PDC) · Computational Biology.

## 1. INTRODUCTION

Over the last few years, computational biology has revolutionized medical research, bringing in novel analysis tools that accelerate diagnosis and drug discovery. The enormous amount of experimental data generated by the human genome project, proteomics, and clinical research has fostered this revolution by enabling extremely accurate, albeit complex models for various biological phenomena. The analysis of these models requires high processing power and time to find accurate solutions, making them attractive candidates for parallelization. For example, high-performance parallel computing has successfully contributed to the understanding of protein dynamics [1], ion channels and cellular reaction kinetics [2], resulting in several specialized high-throughput tools such as GROMACS, a parallelized molecular simulation toolkit [3]. Further, novel projects such as

Folding@home [4] have enabled the pooling of distributed computing resources from around the world to analyze proteins.

Recently, bioengineers have begun focusing on reverse engineering biological systems, by reconstructing gene and metabolic networks that describe the interactions between various genes and protein from experimental data. This relatively new area of research requires novel computational tools due to the vastly heterogeneous nature of the data involved [5]. While computer scientists have been able to contribute to improving the performance and accuracy of biological analysis, the striking applications found in the domain can also serve to provide a wealth of motivation for computer scientists. In addition, there are several different methods of implementing these applications, some more easily parallelized than others did (and the best implementation can depend on the application). Therefore, they provide an excellent opportunity for computer science students to gain insight into issues faced by programmers of parallel algorithms. For this reason, we believe that biomedical applications can be a powerful tool in teaching parallel and distributed computing. In this paper, we propose a novel course curriculum that introduces parallel and distributed computing to senior graduate students in a hands-on manner through a set of carefully chosen computational biology applications. We also propose several sample research term projects that can be carried out as a direct extension of the learning outcomes of this course.

### 1.1 Contribution and Related Work

The ACM and NSF/TCPP guidelines recommend that parallel computing is introduced in CS and CE courses from early stages [28][29]. As parallelism and multi-core computing becomes more accessible, academic institutions in India are exploring the introduction of interdisciplinary concepts in CS and CE education. In this context, several courses have been developed to teach the parallel computing programming concepts with real-world examples [30] [31] [32] [33]. The first author has also introduced a course teaching parallelism with hands-on experimental learning activities as a member of the Board of Studies (BoS)/Curriculum Design Committee at Amrita/VIT University, India in 2005-2009. In this course, the author piloted a new course introducing certain concepts in HPC and PDC using real-world applications, including those in computational biology. Drawing upon this experience, the key contribution of this paper is the design of an interdisciplinary course curriculum that uses problems in computational biology as educational tools in computer science education. Currently, several courses designed for biology majors focusing on the fundamentals of parallel and distributed computing [6] [7]. Recently, courses incorporating high-performance computing for medical applications have also been developed [8]. Advanced courses in


DOI: https://doi.org/10.22369/issn.2153-4136/11/1/8

computational biology have also been targeted towards CS graduate students specializing in biological computation [9]. However, there has been little attention to the pedagogical value that computer scientists can draw from the biological application domain. The curriculum proposed in this paper fills this gap, while bringing in several advantages. First, it serves to provide students with an insight on programming choices regarding how much parallelization is required, based on the application. Second, it promotes interdisciplinary thinking among computer science graduate students and will be particularly valuable to computer scientists who wish to make a career transition into the computational biology domain. Finally, this course does not require expensive parallel computing resources, and can easily be taught using FPGAs and commonly available desktop/laptop GPUs. Further, the use of biological problems such as protein folding as examples to teach parallel computing enables the use of a wealth of tools that pool worldwide distributed computing resources such as the Folding@home project [4].

# 2. HPC IN COMPUTATIONAL BIOLOGY

While several fields can be used to supply examples for the application of HPC and PDC, computational biology provides a large variety of problems that are both complex and challenging even at a research level. This diversity in the availability of HPC and PDC applications in computational biology is the primary reason for choosing this domain. In addition, since the course is taught to senior graduate students, we hope that the large number of open problems in this emerging area will inspire students to explore this interdisciplinary area for their research.

Over the last two decades, there have been significant advances in biomedical sciences, computational biology, drug discovery and systems biology with the introduction of high-performance computing technologies. The tremendous increase in computational power of the desktop to high-end computing devices such as supercomputers, clusters, and grids due to the end of Dennard scaling and Moore's law has opened up great opportunities in the simulation of relevant biological systems for applications in bioinformatics and computational biology. In this section, we introduce the applications of high-performance systems in computational biology. Among them, we introduce relevant biomedical problems such as heterogeneous computing, GPU architecture and large-scale distributed computing has been successfully applied [11]. We also point out the pedagogical value associated with these problems for computer science students. We will later draw upon these areas to construct specific application examples to teach high-performance computing.

## 2.1 Big Data Analytics

Owing to advances in genome projects, proteomics, high-resolution imaging and the rapid digitization of patient clinical records, there has been an explosion in the volume of biological and medical data sets. The optimal use of this data is a major challenge in biomedical research, requiring the development of more sophisticated architectures and tools. A key question that remains in computational biology research is how to extract information, construct models and reverse-engineer biological networks from the massive amount of vastly heterogeneous data [5]. However, dealing with biological networks, while extremely effective, is also computationally intensive. Here, the integration of big data tools with high-performance and parallel processing techniques have been proposed [10]. From a pedagogical perspective, biomedical big data applications can be effective in teaching efficient parallelization and introducing massively parallel processing (MPP) tools.

## 2.2 GPU Computing

The recent explosion in the availability of cheap graphical processing units (GPU) from PCs to heterogeneous computing platforms where they perform massively parallel have introduced a new facet of high-performance computing. This fact has attracted many researchers to use GPU computing platforms for wider applications, in particular, biomedical engineering. Similarly, bio-inspired algorithms such as the genetic algorithm and ANT Colony optimization [12] [13] [14] have been effectively implemented on GPUs which drastically reduce the communication overhead between the CPU and GPU. Problems like the analysis of biological DNA sequences can be effective in teaching CUDA [15] [16] [17] as well as a good source for discussion of the concerns involved in GPU programming and parallel programming in general. They can also be used to motivate a framework for easily parallelizing genetic algorithms. CUDA makes it simple for programmers with only a basic understanding of genetic algorithms to code their own genetic algorithms to run on NVIDIA GPUs.

## 2.3 Distributed Computing

Distributed computing allows for the utilization of vast amounts of computational power to tackle challenges in medical and computational biology. In particular, the biggest challenge in computational biology is simulating proteins due to their great complexity. Analysis of the folding of complex proteins requires the fastest CPUs and supercomputers. Recently, idle computing resources worldwide have been pooled to carry out this analysis, creating a massively distributed computing setup [4]. The key challenge in this distributed computing setup is to exploit parallelism, where it is often difficult to subdivide jobs and extract work from all jobs. Similarly, another challenge is to have efficient algorithms to exploit the available computing power. We believe that the protein-folding problem can help introduce concepts such as large-scale distributed computing architectures and algorithms, as well as network security, novel simulation methods and client-server architectures.

# 3. COURSE DESCRIPTION

In this section, we outline the proposed course curriculum, "High Performance Computing through Computational Biology"(HPCCB), where students will be introduced to various concepts in high-performance computing in a hands-on manner using examples from computational biology. This course will allow students to learn high-performance computing through direct application as well as carry out design projects from concept to realization. First, we provide an outline of the learning objectives, teaching methodology and a brief description of the biological applications chosen, with particular emphasis on their value in illustrating specific HPC concepts. Second, we provide criteria for course evaluation, including laboratory work and research term projects. Finally, we outline the process of evaluating the success of the course via direct feedback from students.

## 3.1 Prerequisites

This course is mainly designed for senior graduate students at the PhD level. Postgraduate students pursuing Master degrees may also register for this course if they have an individual study plan where this particular course is relevant. This course will require basic Linux competence and advanced programming skills. Basic knowledge and/or exposure to biological, high data-intensive and computationally intensive applications will be useful.

## 3.2 Course Development and Learning Outcomes

Figure 1 shows the design and development cycle of the HPCCB course. This course can be taught at the senior graduate (PhD) level as well as at the Masters level, with slightly different approaches. We begin by analyzing the preparedness of the

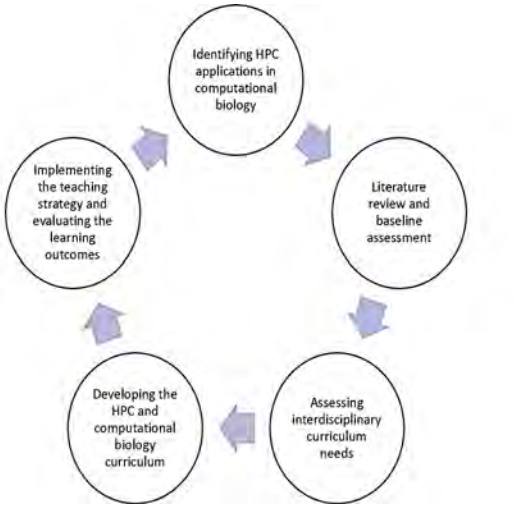



**Figure 1** HPC through computational biology: course design cycle class, in terms of prior course work in CS as well as exposure to interdisciplinary domains. For graduate classes with a strong fundamental background in CS, the course can be tailored to be predominantly application-inspired (focusing on interdisciplinary research problems) to provide a different flavor to traditional concepts. For classes that are composed primarily of students at the Masters level, bridging material needs to be provided to supplement the application-based teaching of concepts with traditional CS problems and examples. Based on the above assessment, we define the learning outcomes of the course. While specific learning outcomes may vary, a general baseline can be as follows:

Students should be able to:

- Define different kinds of parallel architectures, like processor arrays, shared and distributed memory multiprocessors, reconfigurable computing processors and supercomputer architecture,
- Analyze an application for parallelization potential,
- Design/select algorithms for the high-performance computing requirements of the application,
- Assess the scale of bio-specific tools and libraries for parallel and distributed computing,
- Implement commonly used HPC platforms and parallel programming models using appropriate programming languages,
- Measure, assess and analyze the performance of the designed HPC solution and optimize HPC codes, and,
- Perceive the larger role of HPC in computational biology, through examples and detailed term research projects.

Other learning outcomes for application-oriented classes may include:

Students should be able to:

- Effectively utilize the visualization techniques to present the results,
- Provide experience in technical communication (both oral and written),
- Design prototype for computational biology software or bioengineering devices,
- Evaluate the economic considerations related to HPC-based bioengineering designs, such as market analysis and budgeting,
- Apply the regulatory rules important in biomedical devices such as FDA regulations etc., and,
- Value the ethical considerations of biological research, devices, and treatments on individuals, industries and society, as well as ethical considerations involved in collecting and processing big data.

At this stage, interdisciplinary programs may choose to provide an increased weightage to biological applications, while traditional CS programs may want to balance out the application examples with core CS examples. As a baseline, we recommend that at least 60% of the examples chosen in the course be based on biological applications, to exploit the full pedagogical value of application-based teaching. We also recommend that interdisciplinary HPC and CS instructors are trained in the pedagogical perspectives as shown in Fig. 2. In particular, instructors must be exposed to laboratory-based hands-on teaching techniques, which they must employ to guide students towards developing parallel thinking by working on specific application studies in the laboratory. Further, the instructors must obtain exposure to the state-of-the-art computational techniques used in biological applications prior to teaching the course. Once HPC is introduced to the class through the suggested computational biology applications, research areas that can be extended into term projects are selected. In the final stage, the proposed curriculum is implemented with regular feedback to evaluate learning outcomes to further fine-tune the course.

## 3.3 Infrastructure

An important aspect of course development involves assessing the computational infrastructure necessary to conduct the course laboratory and research term projects. With the widespread availability of desktop and laptop GPUs, the infrastructure requirements for this course are minimal. The following are the minimum infrastructure requirements for the course.

- LAM/MPI, OpenMP, OpenCL, and CUDA
- 16 dual 450Mhz Pentium II Linux PCs

We recommend that computers/clusters with higher specifications be made available when possible.

# 4. SAMPLE COURSE OUTLINE AND DESCRIPTION

We now provide a sample course outline that details the HPC concepts that can be covered in a one-semester course, along with the suggested bioengineering applications to introduce them.

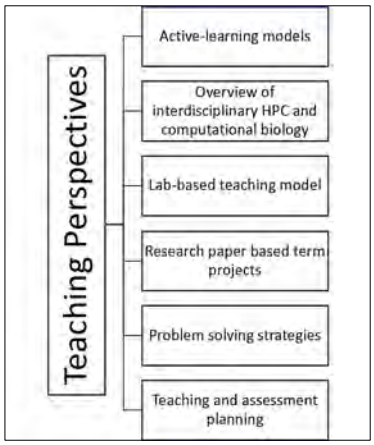



**Figure 2. Teaching perspectives for HPC and Computational Biology course**

## 4.1 Parallel Computation in Biological Applications

This section will comprise of a literature review on the basics of high-performance and PDC applications. Here we will introduce the real-world applications of HPC, with specific focus on computational biology and bioengineering, with surveys and extracts from texts like [18]. We will also introduce the basic concepts and terminology in HPC and parallel processing.

*Section Summary*

- High-performance computing and biological networks
- HPC architecture and Parallel processing

## 4.2 Parallel Implementation

In this section, we will introduce the problem of analyzing DNA sequences collected from large genome projects. One problem in the analysis of biological DNA sequences is the alignment of long sequences to identify regions that are matched and mismatched [19] [20]. This is accomplished by implementing a dynamic programming problem that provides a pair-wise comparison of sequences. The major problem when processing a long sequence of the whole genome is to meet the requirements of computer memory and execution time i.e. memory and CPU intensive application. Therefore, parallel implementations of this dynamic programming algorithm are necessary, as described in [19]. Through this problem, we introduce the concepts of parallel architectures and programming, along with data parallelization like SIMD and MIMD.

*Section Summary*

- Biological sequence analysis algorithm
- Parallel sequencing analysis methods: SIMD, MIMD
- Parallel biological tools
- References: [19], [20]

## 4.3 Massively/Embarrassingly Parallel Solutions to Computational Biology Problems

In this section, we use the problem of molecular sequence analysis to introduce the concept of hybrid SIMD-MIMD architectures. We attempt to provide insight into the process by which a programmer must determine the extent and scale of parallelization required for an application. We also introduce several mapping techniques to reduce the complexity of the sequence alignment. Through this example, we also briefly introduce the concept of performance evaluation.

*Section Summary*

- Hybrid SIMD-MIMD architecture
- Levels of parallelization fine and coarse-grained, and implementation choices
- Mapping techniques,
- Reference paper: [21]

## 4.4 Multithreaded/Multi-core Parallel Implementation

In this section, we return to [19] motivate the idea of multithreaded parallel programming implementations. We further use the problem in [22], where a protein threading problem is formulated as a Mixed Integer Program (MIP) to describe the advantages of multithreading. In this example, students will learn multithreading by decomposing the MIP into sub-tasks and implementing them in a multi-core parallelized setting. Further, this MIP can also be viewed as the shortest path problem. Through this problem, students will gain an in-depth insight into parallel multithreading/multi-core parallel implementations as well as a flavor for optimization problems such as linear programming (LP), MIP and shortest path problems.

*Section Summary*

- A multithreaded parallel implementation, parallel execution model
- Solving protein threading problem in parallel [22]
- Reference paper: [19], [22]

## 4.5 Performance Improvement using Distributed Computing Environments

In this segment, we introduce performance optimization for parallel codes by considering the example of the Clustal W algorithm for multiple sequence alignment. This algorithm involves a pairwise comparison stage followed by the construction of a guided tree, which is then used for progressive alignment. Each stage of this algorithm needs to be optimized to reduce computational complexity. Through this example, students will learn to measure the performance of their parallel implementations and optimize the code to improve performance. Further, parallel clustering algorithms will also be introduced. This problem will also use OpenMP, OpenCL and CUDA programming, thereby providing a succinct overview of these techniques.

*Section Summary*

- Measure and assess the performance of parallel implementations
- Parallel code optimization, parallel clustering
- OpenMP, OpenCL, and CUDA programming techniques
- Reference paper: [23]

### 4.6 Big Data and Parallel Computing using Genomics and Computational Biology

In this section, a large-scale search problem involving big-data will be introduced with the help of genomics data sets. Various search algorithms like GeneWise and GeneMatcher [24] will be introduced, with emphasis on performance analysis for parallel big-data implementations.

*Section Summary*

- Genomics introduction
- Various genome algorithms
- Performance analysis
- Reference paper: [24]

### 4.7 High-performance algorithms in Computational Biology

The examples suggested in the above segments will be revisited to understand SMP and CMP deployments and introduce advanced concepts in algorithm complexity analysis.

*Section Summary*

- SMP and CMP machine deployment
- Complexity algorithm analysis

### 4.8 Parallel and Distributed Memory Architecture

In this section, we use the problem of gene linkage analysis to introduce the concept of distributed memory architectures. Linkage analysis software like Genehunter [25] efficiently distribute both computation and memory. Students will learn these parallelization approaches, including assessment of memory requirements and memory architectures through this problem.

*Section Summary*

- Memory architecture for parallel bioengineering
- Memory requirements
- Parallelization approach and methods
- Reference paper: [25]

## 5. RESEARCH PROJECT

This advanced elective course focuses on interdisciplinary teaching approach similar to other existing courses. Our unique approach is to satisfy the current computing demand and train our students in the research and scientific-orientation. In this context, we provide the final project based on the research problems with the perspectives of computational biology. The references provided in each of the above sections can be extended into excellent implementation projects in HPC as well as computational biology. The design project component is intentionally kept open-ended, with no predefined solution. Project outcomes can be hardware, software or review-based.

For example, Fig. 3 represents the basic flow of genetic information and how it gets manifested at the population level in bacteria. At each stage we need high-throughput computational programs to simulate various processes involved.

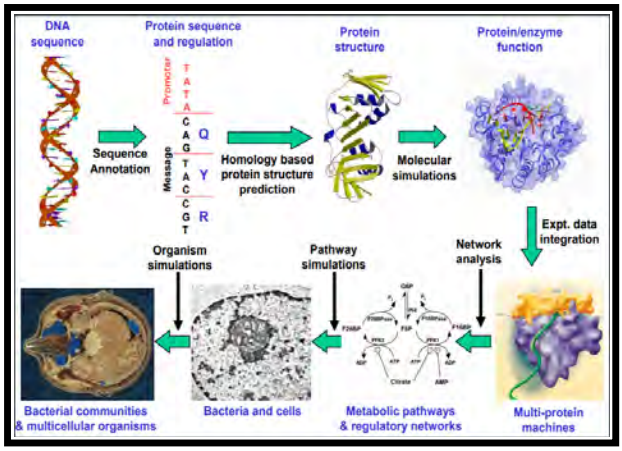



**Figure 3. Basic flow of genetic information in bacteria [34]**

One of the main challenges in implementing biological high-performance computing is to develop platforms for efficient analysis by choosing the right architectures and implementations. The implementation project will provide students with hands-on experience in applying the learning outcomes of this course to real-world computational biology problems. Students may, in consultation with the course instructor, choose research projects related to various topics in course outline or topics that are of interest to their doctoral research, in the case of students at the PhD level. Depending on the size of the class, research projects may be carried out in teams or individually. A research design project typically includes collecting data/information, using tools to analyze the data, using various methods and algorithms, and quantifying the effectiveness of the proposed solution. Potential research topics will be listed during first two weeks of a term, considering the preparedness and specific research interests of the graduate students taking the course. This research project will help students in the following activities:

- Apply theoretical knowledge to identify, formulate and solve real-world biological problems,
- Design devices, software, or experimental apparatus related to biomedical applications or research,
- Experience the end-to-end design process in real-world applications,
- Test solutions by designing suitable experiments,
- Plan and manage a research project, including building teamwork strategies,
- Gain experience in working independently or as part of interdisciplinary teams, and
- Effectively communicate and document research progress and outcomes.

## 6. ASSESSMENT METHODOLOGY

Students are evaluated on their progress towards the course learning outcomes based on the following criteria:

- Demonstration of a computational biology software, device-prototype or study as part of a term research project,
- Submission of assignments and laboratory reports,
- Submission of a final project report that includes theoretical modelling and related observations,

– Short oral presentation of the outcomes of the term project, and
– Final examination.

In Table 1, we summarize the weightage assigned to different facets of the course in the evaluation process.

**Table 1. Grading Breakdown**

| Item | Weightage |
|---|---|
| Assignments | 10% |
| Research Project | 40% |
| Laboratory Modules | 20% |
| Final Exam | 30% |

**Table 2. Laboratories and Assignments**

| Topic | No. of Hours | Assignments/Labs |
|---|---|---|
| Parallel architecture and system models | 2 | Reference [27] |
| Introduction to HPC in Biological applications | 5 | Paper review |
| Teaching parallel programmingtechniques MPI, Pthreads and OpenMP, Biotools/libraries | 6 | Lab 1-Lab 4 |
| Analyze data intensive and computationally intensive applications | 3 | Lab 5 |
| Data visualization | 3 | Lab 6 |
| Big Data Infrastructure | 4 | Hadoop Eco system, HDFS |
| HPC in Big Data | 4 | Lab 7 |
| Statistical modeling for data | 2 | Linear, logical regression, Bayesian models |
| Research project | 40 | Chosen by student in consultation with instructor |

In Table 2, we list various laboratory modules along with the number of hours assigned to them. Note that students are expected to spend at least 40 hours over the semester working on their chosen research project. We also provide the rubrics for assessment of the course research project in Appendix A.

# 7. REFERENCE MATERIAL

To accomplish the learning outcomes in Section 3.2, we choose a set of examples, drawn primarily from current literature in computational biology, and supplemented from traditional textbooks like [26], to cover the essentials of parallel and distributed high-performance computing. We note that specific references pertaining to the chosen biological applications and their HPC implementations have been provided under each subsection of the course outline in Section 4. This material can be replaced or augmented with current literature relevant to the research interests of the students taking the course. Considering the practical and demonstrative nature of the course, the use of online interactive books is worth mentioning. While we do not recommend any specific online textbooks, some interactive programming books can be used as a good source of practice material [27]. With several iterations of this course, we will work towards the design of an interactive online book where we will introduce parallel and distributed computing through examples and exercises from biological applications. Students will then be able to access this material both on-campus as well as remotely and will be able to work on the exercises and projects designed for them online.

# 8. FEEDBACK MODEL

The level of student engagement and satisfaction with the learning outcomes of the course can be analyzed by conducting two surveys during the term, and one survey after the end of the term. The first survey can be conducted in one month of the course and the second survey in three months of the course. The first two surveys may be replaced by a single survey for universities following the quarter system with shorter terms. Students can be asked to rate the course on a scale of 0 to10 on a variety of topics, including but not limited to, their satisfaction with the course instructor, laboratories, assignments, choice of research projects, amount of time invested into the course, difficulty level of the course and usefulness of the assigned textbooks and reference material. In addition, students may also be asked to provide suggestions on how the course can be improved both in the short term that is, during the same semester, and in the long term, that is, in subsequent iterations. This feedback will then be used as a pivot to improve the course structure to cater the students' needs in the particular term, as well as to refine the course development process as laid out in Fig. 1 in the subsequent terms.

# 9. COURSE EXPERIENCES AND EVALUATION

We now detail the experiences and evaluation results of the first author in introducing a similar curriculum, and describe how these experiences have contributed to the shaping of the curriculum presented in this paper. The first author introduced an interdisciplinary course on HPC and PDC when she was a faculty and member of the Board of Studies (BoS)/curriculum design committee at Amrita/VIT, India in 2005-2009. In this course, senior undergraduate engineering students in their final year were introduced to HPC and PDC concepts through real-world examples drawn from multiple domains, including computational biology. As the course involves learning HPC and PDC concepts directly via implementation of problems at the research-level, the

feedback received indicated that this course was too involved at the undergraduate level. Therefore, the author reintroduced this curriculum in 2010 as an advanced elective for graduate students (PhD) who have taken prior courses on the foundations of HPC, and could draw value from the interdisciplinary flavor of this course. This course, comprising of 9 students, was well-received with an average rating of 95%. Several students attending the course also chose to pursue interdisciplinary HPC-based term and thesis projects supervised by the first author. Initially, the author faced various difficulties such as getting proper training in HPC, lack of proper textbooks and lack of HPC tools to support the computational needs at the university. However, now the author's institute has received the funding from Intel to support the infrastructure needs and the management streamlined the other issues as well. We also taught the project based proposed course, which is mentioned in Section.5, was well received by Ph.D. students with an average rating of 90%.

## 10. CONCLUSION

In this paper, we present a novel course designed to teach high performance parallel and distributed computing to graduate students directly via hands-on applications drawn from computational biology. While HPC successfully contributed to solving several biological problems, we believe that computer scientists can conversely draw enormous pedagogical value from biological examples and problems. Motivated by this, we presented a course curriculum where HPC is introduced almost entirely via hands-on application examples. We first summarize the main trends in HPC applied to biomedical engineering and computational biology. We demonstrate several successful stories and application fields, in which relevant biological problems have been solved (or are being targeted) using the computational power available in current processors. We then provide a detailed description of the course and suggested examples to be used in the course. We would also like to point out some potential implementation challenges in terms of the high learning curve in emerging programming models. We believe that this course can inspire students to undertake investigations into improving the performance on HPC systems motivated by computational biology problems. This will be of high technical interest in the computer science community as biological applications have novel computational patterns that can lead the next generation of high-performance heterogeneous computing systems.

## 11. References


[1] Sanbonmatsu, K. Y., and C-S. Tung. High performance computing in biology: multimillionatom simulations of nanoscale systems. Journal of structural biology 157.3 (2007): 470-480.

[2] Stevens, R. (2002, September). Biology and High-Performance Computing. In UK-HPCUsers Meeting. http://www.cels.anl.gov/about/people/files/UK-BIO-HPC-final1.pdf

[3] Pronk, Sander, et al. GROMACS 4.5: a high-throughput and highly parallel open sourcemolecular simulation toolkit. Bioinformatics (2013).

[4] Pande, Vijay S., et al. (2003). Atomistic protein folding simulations on the submillisecondtime scale using worldwide distributed computing. Biopolymers 68.1 91-109.

[5] Lee, W.P., Tzou, W.S. (2009). Computational methods for discovering gene networks from expression data. Briefings in Bioinformatics 10 (4):408-423.

[6] http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-047computational-biology-fall-2015/.

[7] https://biology.ufl.edu/files/ZOO4926-6927-CompBio-Basic-Res-Comp-Skills.pdf.

[8] Fundamentals of High-Performance Computing for Public Health. Columbia University School of Public Health. https://www.mailman.columbia.edu/public-healthnow/news/big-data-academy-public-health-supercomputing.

[9] CMSC 838T: Advanced Topics in Programming Languages - Systems Software for High-performance Computing, Emphasis on Bioinformatics Applications. University of Maryland. http://www.cs.umd.edu/class/spring2003/cmsc838t/.

[10] Marx, Vivien. Biology: The big challenges of big data. Nature 498.7453 (2013): 255-260.

[11] Garland, M., Kirk, D.B.: Understanding throughput-oriented Architectures. Communications of the ACM 53, 5866 (2010)

[12] Wong, M. L., Wong, T. T., and Fok, K. L. (2005, September). Parallel evolutionary algorithms on graphics processing unit. In 2005 IEEE Congress on Evolutionary Computation (Vol. 3, pp. 2286-2293). IEEE.

[13] Manfrin, M., Birattari, M., Stutzle, T., Dorigo, M.: Parallel ant colony optimization forthe traveling salesman problem. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stutzle, T. (eds.) ANTS 2006. LNCS, vol. 4150, pp. 224234. Springer, Heidelberg (2006)

[14] Cecilia, J.M., Garcia, J.M., Ujaldon, M., Nisbet, A., Amos, M.: Parallelization strategiesfor ant colony optimization on GPUs. In: NIDISC 2011: 14th International Workshop on Nature Inspired Distributed Computing. Proc. 25th International Parallel and Distributed Processing Symposium (IPDPS 2011), Anchorage, Alaska, USA (May 2011)

[15] Wong, M. L., and Wong, T. T. (2009). Implementation of parallel genetic algorithms ongraphics processing units. In Intelligent and Evolutionary Systems (pp. 197-216). Springer Berlin Heidelberg.

[16] Pospichal, P., Jaros, J., and Schwarz, J. (2010, April). Parallel genetic algorithm on thecuda architecture. In European Conference on the Applications of Evolutionary Computation (pp. 442-451). Springer Berlin Heidelberg.

[17] Garland, M., Le Grand, S., Nickolls, J., Anderson, J., Hardwick, J., Morton, S., Phillips, E., Zhang, Y., Volkov, V.: Parallel Computing Experiences with CUDA. IEEE Micro 28, 1327 (2008)

[18] Aluru, Srinivas, ed. Handbook of computational molecular biology. CRC Press, 2005.

[19] Martins, Wellington S., et al. "Whole genome alignment using a multithreaded parallel implementation." Symposium on Computer Architecture and High Performance Computing. 2001.

[20] Yap, Tieng K., Ophir Frieder, and Robert L. Martino. "Parallel computation in biologicalsequence analysis." IEEE Transactions on Parallel and Distributed Systems 9.3 (1998): 283-294.

[21] Schmidt, Bertil, Heiko Schrder, and Manfred Schimmler. "Massively Parallel Solutions forMolecular Sequence Analysis." ipdps. 2002.

[22] Yanev, Nicola, and Rumen Andonov. "Solving the protein threading problem in parallel."Parallel and Distributed

Processing Symposium, 2003. Proceedings. International. IEEE, 2003.
[23] Mikhailov, Dmitri, Haruna Cofer, and Roberto Gomperts. "Performance optimization ofclustal w: Parallel clustal w, ht clustal, and multiclustal." SGI ChemBio (2001).
[24] Mo, Yi, Moira Regelson, and Mike Sievers. "A Study of GeneWise with the DrosophilaAdh Region." 13th Annual Genome Sequencing and Analysis Conference. 2001.
[25] Conant, Gavin C., et al. "Parallel genehunter: Implementation of a linkage analysis packagefor distributed-memory architectures." Journal of Parallel and Distributed Computing 63.7 (2003): 674-682.
[26] Jeffers, Jim, and James Reinders. High Performance Parallelism Pearls Volume Two: Multicore and Many-core Programming Approaches. Morgan Kaufmann, 2015.
[27] K. Hwang, G. C. Fox, and J. J. Dongarra, Distributed and Cloud Computing: From ParallelProcessing to the Internet of Things, Elsevier, 2012.
[28] Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society, Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. New York, NY, USA: ACM, 2013.
[29] S. K. Prasad, A. Chtchelkanova, S. Das, F. Dehne, M. Gouda, A. Gupta, J. Jaja, K. Kant, A. La Salle, R. LeBlanc, M. Lumsdaine, D. Padua, M. Parashar, V. Prasanna, Y. Robert, A. Rosenberg, S. Sahni, B. Shirazi, A. Sussman, C. Weems, and J. Wu, NSF/IEEE-TCPP curriculum initiative on parallel and distributed computing: Core topics for undergraduates, in Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education, ser. SIGCSE 11. New York, NY, USA: ACM, 2011, pp. 617618.
[30] 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, ser. ITiCSE-WGR 10. New York, NY, USA: ACM, 2010, pp. 97115.
[31] A. Fitz Gibbon, D. A. Joiner, H. Neeman, C. Peck, and S. Thompson, Teaching high performance computing to undergraduate faculty and undergraduate students, in Proceedings of the 2010 TeraGrid Conference, ser. TG 10. New York, NY, USA: ACM, 2010, pp. 7:17:7.
[32] J. Adams, R. Brown, and E. Shoop, Patterns and exemplars: Compelling strategies for teaching parallel and distributed computing to CS undergraduates, in IEEE International Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), May 2013, pp. 12441251.
[33] Eduardo Csar, Ana Corts, Antonio Espinosa, Toms Margalef, Juan Carlos Moure, AnnaSikora, Remo Suppi, "Introducing computational thinking, parallel programming and performance engineering in interdisciplinary studies," J. Parallel Distrib. Comput. 105, 2017, pp. 116-126.
[34] Rick Stevens, Biology and High-Performance computing https://pdfs.semanticscholar.org/presentation/cd70/536ba6011018539eda74dddaede95cca893b.pdf

**Appendix A: Rubric for Research Project Evaluation**

| Criteria | Poor | Average | Excellent |
|---|---|---|---|
| **Abstract** | Not precise or succinct, one or more elements missing | Contains the topic under investigation, but does not clearly summarize the conclusion and/or methodology of investigation. | Clearly states the topic of investigation, how the investigation was undertaken and the conclusions. |
| **Introduction and relevance of application, device, prototype or study** | Little or no description, little or no attempt to explain the significance of the application, device, prototype or study | Some description of device, some attempt to explain the significance of the application, device, prototype or study | application, device, prototype or study is clearly described, clearly explains the significance and application of developed application, device, prototype or study, places the problem in the context of relevant literature |
| **Blue prints and Block Diagrams** | Unreasonably sized and spaced, either incorrectly captioned or not captioned at all | Most are appropriately sized and spaced, most are properly captioned | All block diagrams have a specific purpose, all appropriately sized and spaced, all properly captioned |
| **Procedure** | Method not described clearly, omits crucial details | Method described fairly clearly, some important details omitted | Provides a detailed and comprehensive description of the implementation |
| **Demonstration** | Unclear demon-stration/questions not answered | Better description/some questions answered | Excellent and clear demonstration/ most questions answered |

[i] Corresponding/First Author: Vijayalakshmi Saravanan, Rochester Institute of Technology, NY. Email:vsavse@rit.edu

# FreeCompilerCamp.org: Training for OpenMP Compiler Development from Cloud

Anjia Wang
Lawrence Livermore National Laboratory
Livermore, California, USA
University of North Carolina at Charlotte
Charlotte, North Carolina, USA
awang15@uncc.edu

Alok Mishra
Lawrence Livermore National Laboratory
Livermore, California, USA
Stony Brook University
Stony Brook, New York, USA
alok.mishra@stonybrook.edu

Chunhua Liao
Lawrence Livermore National Laboratory
Livermore, California, USA
liao6@llnl.gov

Yonghong Yan
University of North Carolina at Charlotte
Charlotte, North Carolina, USA
yyan7@uncc.edu

Barbara Chapman
Stony Brook University
Stony Brook, New York, USA
Brookhaven National Laboratory
Upton, New York, USA
barbara.chapman@stonybrook.edu

## ABSTRACT

OpenMP is one of the most popular programming models to exploit node-level parallelism of supercomputers. Many researchers are interested in developing OpenMP compilers or extending existing standard for new capabilities. However, there is a lack of training resources for researchers who are involved in the compiler and language development around OpenMP, making learning curve in this area steep.

In this paper, we introduce an ongoing effort, FreeCompilerCamp.org, a free and open online learning platform aimed to train researchers to quickly develop OpenMP compilers. The platform is built on top of Play-With-Docker, a docker playground for users to conduct experiments in an online terminal sandbox. It provides a live training website that is set up on cloud, so anyone with internet access and a web browser will be able to take the training. It also enables developers with relevant skills to contribute new tutorials. The entire training system is open-source and can be deployed on a private server, workstation or even laptop for personal use. We have created some initial tutorials to train users to learn how to extend the Clang/LLVM and ROSE compiler to support new OpenMP features. We welcome anyone to try out our system, give us feedback, contribute new training courses, or enhance the training platform to make it an effective learning resource for the HPC community.

## 1 INTRODUCTION

Due to the increasing complexity of supercomputer node architectures for high performance computing (HPC), high level programming models are used to improve the productivity of using supercomputers. OpenMP is considered by many as the de-facto portable programming model for exploiting node-level parallelism for supercomputers. Compiler support for OpenMP has been added in many open source compilers, such as GNU compiler collection, Clang/LLVM, and ROSE source-to-source compiler frameworks, as well as vendor compilers from Intel, Cray, NVIDIA and AMD. More and more researchers are interested in conducting research using OpenMP as a vehicle in the area of parallel programming models, compiler technologies and computer systems. However, one of the major challenges in developing an OpenMP compiler and to extend OpenMP language is the steep learning curve of compiler implementation and the development efforts of adding compiler support for language extensions.

Fundamentally, compiler development is a complex and time consuming task. Although many cloud-based, online learning platforms [3, 21, 25, 30, 31] have been created for computer science education, focusing on entry-level programming courses, there is a clear lack of such resources to teach compiler development. Even with the developer manuals of a compiler framework, it is difficult for beginners to teach themselves how to modify compilers which contains millions of lines of code. Training beginners by proficient compiler developers consumes lots of time, human efforts and cost, which is not scalable in the long term.

In this paper, we introduce an ongoing effort, FreeCompilerCamp.org, a free and open online learning platform aimed to train researchers to quickly develop OpenMP compilers and help them learn the skills of compiler development. FreeCompilerCamp.org has several distinct features: 1) It allows anyone who is interested in developing OpenMP compilers to learn the necessary skills for free; 2) A live training website is set up so a web browser and an Internet connection are the only requirements for anyone to take the training; 3) It enables those who have the relevant skills to contribute new tutorials; and 4) The entire training system is open-source so it can be be deployed on a private server, workstation or even personal laptop.

The remainder of the paper is divided as follows: Section 2 gives background information for our work. Section 3 explains the challenges faced in giving compilers training and our solutions. Section 4 presents the implementation of the framework. Section 5 gives an overview of the design of the tutorials with a few examples. Section


https://doi.org/10.22369/issn.2153-4136/11/1/9

6 covers work related to this paper. Finally Section 7 consists of the conclusion and our future plans.

## 2 BACKGROUND

The goal of this work is to improve the effectiveness and scalability of compiler development training for researchers, developers and graduate students. We choose two OpenMP compilers, Clang/LLVM and ROSE as examples. This section gives a brief introduction of background information.

### 2.1 Compilers

Compilers are essential for HPC. As opposed to interpreted languages, programs written in compiled languages gives a better performance and are more favorable towards high performance computing. A compiler takes high-level human readable programs written in programming languages, such as C/C++ or Fortran, and converts them into low-level binary machine codes for a specific architecture. The entire process of this transformation is complicated. A compiler need to parse the code, check for syntax correctness, gather necessary semantic information (like type checking or variable declaration before use and so forth), then convert the source from high level language to intermediate representation and before transforming them into machine codes [10].

Today a compiler can do much more than converting a program into machine instructions. As HPC hardware designs are evolving, machines are becoming more and more complex, and issues which need to be address by the programmers are also getting convoluted. This raises the question about what more can a compiler do for the programmers. Compilers have very complex designs so that the work of an application developer becomes simpler. Owing to the complexity of design, extending a compiler to add a new feature is a very time consuming job. The development cycle of a compiler is at least 3-5 years. Training programmers to do compiler development is challenging to both the trainer and the trainee.

### 2.2 Clang/LLVM

LLVM [18] is the prime environment for developing new compilers and language-processing tools. HPC programmers rely on compilers and analysis tools. LLVM is the environment of choice for the development of such tools, and thus should be of interest to many HPC programmers. LLVM makes it easier to not only create new languages, but to enhance the development of existing ones. Its primary C/C++ compiler frontend is Clang. Today most supercomputing clusters deploy LLVM as one of their compilers due to the following reasons:

(1) It provides a high-performance and up-to-date C/C++ compiler frontend Clang.
(2) Many researchers in HPC community enjoy Clang's diagnostic abilities and static-analysis framework.
(3) It allows for tapping other languages that have an LLVM back-end like Intel's ISPC [27] and different scripting languages.
(4) It makes for compelling compiler research, as evident by the plethora of projects built using LLVM [22].

Ever since its first release in 2003, LLVM has gone through a plethora of changes and updates. With every release new features are added and older features are deleted or updated. Owing to these diverse set of features and many more, using Clang/LLVM for developing a tool or a plugin is a very complex task. There are lots of tutorials which are available for Clang/LLVM, but they are all just text based tutorials and come with their own set of challenges.

### 2.3 ROSE

ROSE is an open source compiler infrastructure developed at Lawerence Livermore National Laboratory (LLNL). It is designed to build source-to-source program transformation and analysis tools for Fortran, C, C++, OpenMP, and UPC applications[28]. Internally, ROSE generates a uniform abstract syntax tree (AST) as its intermediate representation (IR) for input codes. Sophisticated compiler analyses, transformations and optimizations are developed on top of the AST and encapsulated as simple function calls, which can be readily leveraged by tool developers. The ROSE AST can be optionally unparsed to human readable and compilable source files, which in turn can be compiled into final executable by a traditional compiler such as GCC or Intel compiler.

However, for users who are not familiar with the ROSE compiler, it's not easy to customize the framework because of the complexity of ROSE. ROSE has more than two millions lines of code, including tests, built-in projects and tutorial examples. Creating a new transformation module could involve multiple functions, which are located in different files that far away from each other. Like any other compiler frameworks, ROSE compiler exposes its own API functions for developers to traverse, analyze, and modify its abstract syntax tree. Users not only need to learn the general knowledge of compilers but also have to understand how ROSE API functions work.

### 2.4 OpenMP

In HPC, OpenMP is the de-facto portable programming interface for exploiting node-level parallelism [13]. OpenMP uses C/C++ directives and Fortran comments to annotate base language programs written in C/C++ and Fortran, respectively. These annotations express additional semantics related to parallelism, worksharing, synchronization, tasking, and so on. A compiler supporting OpenMP can recognize OpenMP annotations and transform the annotated input code into multi-threaded code calling some OpenMP runtime functions.

There are multiple compilers implementing OpenMP, such as GCC[4], Intel[5], Cray[12], IBM XL[7], Clang/LLVM and ROSE[20]. Most of the parallel constructs in OpenMP are realized through compiler directives. This allows a serial program to be very easily converted into a parallel one by just adding the necessary preprocessor directives.

Figure. 1 is an OpenMP program to calculate PI in parallel. User inserted an OpenMP `parallel for` directive at line 14-15 right above the loop (Fig. 1a). An OpenMP compiler transforms (or lowers) the program into multi-threaded code with calls to runtime library functions (Fig. 1b). In the lowered code, at line 11-23 the loop block is outlined as a function containing the original statements in the loop. At line 15, a runtime function call is used to split loop iterations among several threads. At line 5 the `main` function passes

```c
#include <omp.h>
#include <stdio.h>

int num_steps = 10000;

int main() {
    double x = 0;
    double sum = 0.0;
    double pi;
    int i;
    double step = 1.0/(double) num_steps;

    // Run the code in parallel
    #pragma omp parallel for private(i,x) \
            reduction(+:sum) schedule(static)
    for (i=0; i<num_steps; i=i+1) {
        x = (i+0.5)*step;
        sum = sum + 4.0/(1.0+x*x);
    }

    pi=step*sum;
    printf("%f\n", pi);
}
```

**(a) OpenMP program to calculate PI**

```c
... // omitted headers and a data structure declaration storing variable addresses
static void OUT__1__2189__(void *__out_argv);
int main(int argc, char **argv) {
  ... // omitted variable declarations
  XOMP_parallel_start(OUT__1__2189__,&__out_argv1__2189__,1,0,"demo.c",10);
  XOMP_parallel_end("demo.c",15);
  pi = step * sum;
  printf("%f\n",pi);
  XOMP_terminate(status);
}
static void OUT__1__2189__(void *__out_argv) {
  ... // omitted variable declarations
  double *sum = (double *)(((struct OUT__1__2189___data *)__out_argv) -> sum_p);
  double *step = (double *)(((struct OUT__1__2189___data *)__out_argv) -> step_p);
  XOMP_loop_default(0,num_steps - 1,1,&p_lower_,&p_upper_);
  for (p_index_ = p_lower_; p_index_ <= p_upper_; p_index_ = p_index_ + 1) {
    _p_x = (p_index_ + 0.5) * *step;
    _p_sum = _p_sum + 4.0 / (1.0 + _p_x * _p_x);
  }
  XOMP_atomic_start();
   *sum = *sum + _p_sum;
  XOMP_atomic_end(); XOMP_barrier();
}
```

**(b) Transformed (or Lowered) code**

**Figure 1: PI calculation using OpenMP and its corresponding multi-threaded code generated by ROSE**

the outlined function's pointer to another runtime function which will spawn multiple threads to execute the outlined function.

The initial OpenMP standard in 1997 only specified a handful of directives. Since then, substantial amount of new constructs have been introduced and most existing APIs have been enhanced in each revision [14]. The latest version of OpenMP 5.0, released in 2018, has more than 60 directives. Compiler support thus requires more efforts than before [19]. A full compiler implementation of the latest OpenMP standard for both C/C++ and Fortran would involve a large amount of development efforts spanning multiple years. Furthermore, more and more researchers and developers are interested in designing various extensions to OpenMP in order to tame the increasing complexity of heterogeneous node designs in high performance computing. Such extensions could be used to enhance the expressiveness, performance or productivity of OpenMP. Support for those extensions requires significant amount of compiler development.

## 3 CHALLENGES AND SOLUTIONS

Table 1 summarizes the main pain points for compiler training. For example, one of the first problems for developers is getting hands on a machine which is suitable for compiler development. Getting access to a supercomputing cluster could be a challenge and a potential deterrent for many. The second, and the most frustrating challenge for beginners is making sure that all the software packages necessary for developing a compiler are met on the said machine. Sometimes user might not have suitable access to install certain dependencies. Or sometimes the dependencies are just too complex to resolve on a particular machines. One solution to these

| Pain Points | Description | Proposed Solution |
|---|---|---|
| Accessibility | Paperwork to get accounts on suitable machines | Online sandbox terminal open to anyone |
| Installation | Many software packages are needed | Docker images |
| Effectiveness | Traditional text tutorials are not effective | Learning by doing, testing, certification |
| Content | No single person/group knows all details of OpenMP compiler development | Self-made tutorials + crowd-sourcing to accept external contributions |
| Design trade-offs | One compiler cannot demonstrate all options | Hosting tutorials for multiple compilers |
| Costs | Hosting websites with containers costs money | Open-source, self-deployable framework |
| Security | Online websites have inherent risks | Containers + Cloud machines |

**Table 1: Pain points and solutions for training OpenMP compiler developers**

two problems is to provide a free online sandbox terminal which will already have an environment setup for compiler development.

Based on our experiences, traditional text tutorials are not as effective for compilers development, as hands-on tutorials. If a framework is provided which gives its users an option to learn by hand-on practice, freedom to dig deep and perform self experiments, then such a framework will be most efficacious way of teaching compilers.

Another problem of creating the content of compiler development, especially for OpenMP, is that no one person or group knows all the details of OpenMP implementations since they involve many compilation and runtime stages including parsing, AST, transformation, as well as runtime support. No one implementation demonstrate all the options of OpenMP. This generally results in incomplete or unproductive tutorials. Having an open source environment where multiple users can submit tutorials for multiple compilers can resolve such complications.

Finally hosting tutorials on website costs money. Having containers can result in larger disk space which means more expenses. Having an open sourced, self-deployable framework can help users host their tutorials for free.

FreeCompilerCamp.org is aimed to build a free and open cloud-based training platform integrating the solutions mentioned above. This platform aims to facilitate the training of researchers to quickly develop compilers for OpenMP and help them learn the skills of compiler development. We will elaborate the design and implementation of this platform in the next sections.

## 4 FREECOMPILERCAMP.ORG PLATFORM

FreeCompilerCamp.org is a learning system with several distinct design principals:

- It aims to allow any developer, who is interested in understanding the internal working of OpenMP compilers, to learn the necessary skills for free.
- It provides a pre-configured compiler development environments in an online sandbox, which eliminates the burden of beginners' tedious and error-prone software installation processes.
- A live training website based on the system is set up, so a web browser and an Internet connection are the only requirements for anyone to get the training.
- The entire training system is open-source, so it can also be deployed by anyone on a private server, workstation or even personal laptop.
- It enables anyone who has the relevant skills to contribute new tutorials as well.

There are two components in the FreeCompilerCamp.org platform (or FreeCC as an abbreviation) as displayed in Figure 2 – a web-based framework with all tutorials and a Play-With-Compiler (PWC) engine for the sandbox environment. The website provides a browser-based interactive interface with two panels: the left panel contains the training instructions in text, and the the right panel connects with the PWC engine, which creates a live terminal sandbox for real-time practice.

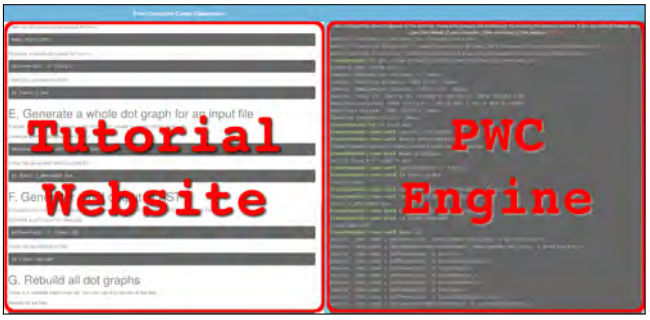



**Figure 2: Two components of FreeComplierCamp.org**

### 4.1 Tutorial Website

The tutorial website is created as the major interface of FreeCC. It provides easy-to-understand document in multiple tutorials organized by categories. Users can choose any entry on demands or learn in order.

### 4.2 Play-With-Compiler Engine

The Play-With-Compiler engine is based on Play-With-Docker (PWD) [26], which is an online sandbox platform for visitors to learn basics about container techniques using Docker [23]. Docker uses OS-level virtualization to deliver software, libraries and configuration files in packages called containers, which are isolated from one another though there are defined channels to enable their communication. Containers on a same machine shares a single operating-system kernel and are thus more lightweight than virtual machines.

Play-With-Docker uses a so-called Docker-in-Docker technique. While the host service is running in an outer docker, the component of this service runs in an isolated inner docker so that multiple components won't affect each other[16, 33]. In the case of PWD, each user has their own sandbox and won't get interrupted by others' activities. PWD uses Apline Linux, which is widely used in docker images due to its lightweight and security.

### 4.3 Customization

We encountered several technical issues during the development of FreeCompilerCamp.org and subsequently resolved them. Most of these issues may not be new in web development, but our target audience is mostly people with a HPC background, who may not have a flair for web development. Also these issues are common and will be faced by anyone who would like to deploy our framework. Hence mentioning these issues here is vital.

**4.3.1 Same-Origin Policy.** The same-origin policy [29] restricts resources loaded from one origin to interact with resources from another origin. This prohibits training website and PWC to be deployed on different servers. We had to apply Cross-Origin Resource Sharing [32] mechanism that uses additional HTTP headers to enable resources on PWC server to be accessed by training website.

**4.3.2 Port Conflict.** Later to simplify management and lower the cost, we decided to deploy both the training website and PWC

on the same server. This caused port conflict since they both use port 80 by default. We set up an HTTP server using Apache and non-default ports redirection to resolve this conflict.

**4.3.3 Alpine Linux.** The PWD sandbox had dockers built from Alpine Linux, which was unfit for compiler training. Compilers are sensitive to the host system environment. Alpine Linux is not supported for the development of either ROSE or LLVM. Therefore, we created new docker images based on Ubuntu for better compatibility with both ROSE and LLVM. Ubuntu has a much wider application support, hence if future even more compilers can be added in the tutorial.

**4.3.4 Security.** The PWD sandbox by default gives users root access inside the terminal. This is a security risk since a malicious user may hack into web hosting directories where they are not supposed to access. As a solution we create a user/group (freecc/freecc) in our sandbox and let all process run in that user account instead of root. This way we have more control over what access we want to provide the users.

## 5 TUTORIAL DESIGN

We have created several initial tutorials to take advantage of FreeCompilerCamp. The goal is to have a good mix of text and commands for users to read and practice essential compiler skills.

### 5.1 Concepts

Tutorials of FreeCC are designed based on the principle of experimental learning or learning-by-doing. Learning-by-doing was introduced by John Dewey and it promotes the idea that students should learn by actively interacting with environments[15]. Kolb reviewed the major experimental learning models and created his own comprehensive structural model[17]. He also explored the application of experimental learning in higher education. Students not only read static texts but also apply the theoretical knowledge into practical cases. They learn the skills by solving problems, working on small projects, and so on.

Under the guidance of this theory, FreeCC hosts the tutorials to let users start from any point they like with a ready environment, with the following major features:

- We make users practice as much as possible with detailed instructions, by providing an easy-to-use sandbox for users to test given code or conduct their own experiments.
- FreeCC covers different topics in compiler development, including parsing, AST generation, OpenMP programming, compiler extension, and so on.
- We split larger learning tasks into smaller ones to fit each tutorial into a 10-15 minutes session. The goal is to ensure that we can grab sufficient attention from visitors.
- The tutorial not only lists the steps but also explain why each step should be conducted and how it works.
- FreeCC supports clickable code snippets, which can be tested in the sandbox right away by clicking.
- Video instructions are not included currently because more students prefer static tutorials to video tutorials[24]. Using static tutorial is easier to seek and pick different sections of tutorial and learn at a comfortable pace for themselves.

### 5.2 Example Tutorials

FreeCompilerCamp.org provides a flexible learning experience based on the concepts mentioned above. In particular, we split the training content into several tutorials with incremental complexity so visitors can jump into the right levels they are comfortable with. We start with simple ones to let visitors play with input and output of compilers and get familiar with compilers' internal representations for input programs. After that, we let them try out how to traverse the tree representations and finally how to change the tree for writing transformations.

**5.2.1 Tutorial for Learning AST.** Taking ROSE as an example, we designed the following tutorials:

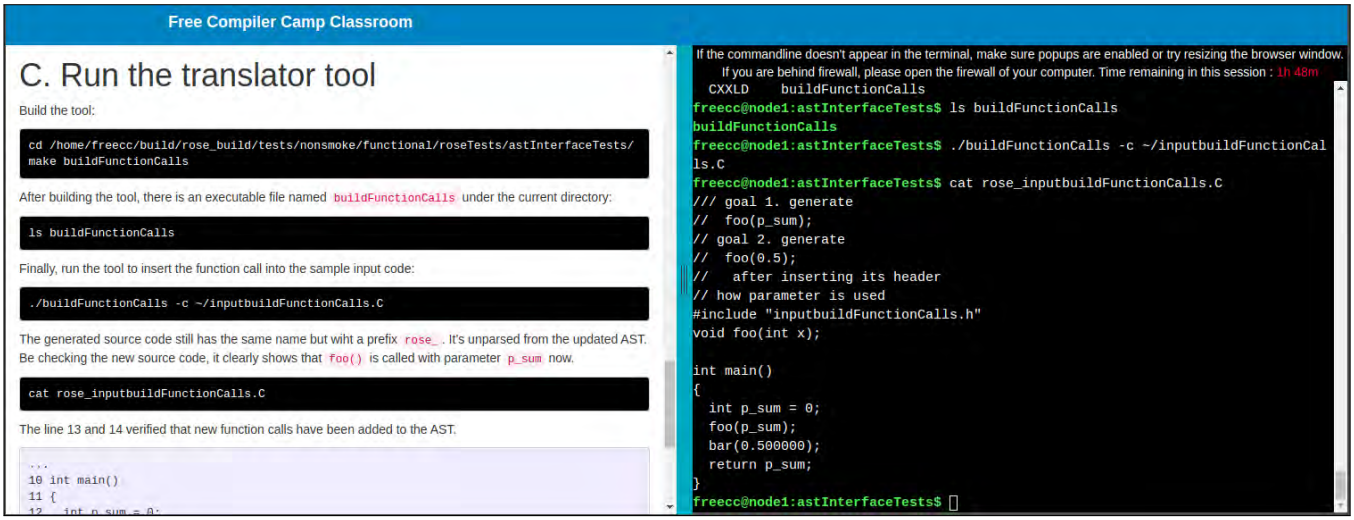



**Figure 3: The tutorial for teaching AST modification**

- AST/IR Generation. For a given input source file, an AST will be generated and represented visually in a graph. This tutorial shows how information is retrieved from source code and organized internally inside ROSE for future use.
- AST/IR Traversal. After AST generation, this tutorial shows how to traverse the tree to search for certain information of interests, such as loops or functions.
- AST/IR Modification. This tutorial demonstrates the method to add function call nodes into AST. Unparsing the AST will result in an output source file with the inserted function calls.

For example, the AST modification tutorial teaches users how to insert a functional call node into AST and check the updated AST by looking into the corresponding unparsed source code (Fig. 3). User can click the corresponding code snippets to download those files without leaving the page. All necessary source files can be downloaded in the sandbox on demand. In the sample input, there's no function calls in the `main` function. The tutorial explains how a function call subtree is constructed in the compiler and showed all steps to create the subtree and attach it to the AST to complete the task. The input and expected output are both provided in the tutorial so that users can compare their results with the correct solution.

**5.2.2 Tutorial of Fixing a Compiler Bug.** Developers often learn many things by fixing real bugs. Figure 4 is an example tutorial to fix a user-reported bug in ROSE. A PI calculation program in OpenMP compiled by ROSE generated some wrong value. Upon debugging it was found that during ROSE's transformation of the loop body of `'omp parallel for'`, the loop stride was miscalculated due to incorrect operand nodes were retrieved in the AST. The tutorial first highlights the bug and describes the steps to reproduce it. It then explains how compiler transformation and a runtime library function collaborate to schedule loop iterations among multiple threads. After that, it gives specific instructions on which source files should be modified to fix the bug. At last, with a few simple clicks, the modified ROSE is re-built to compile the test program and correct execution output is generated. Thus in a wholesome way this tutorial gives an example of a real OpenMP implementation bug and explains how to reproduce, debug and resolve it.

**5.2.3 Tutorial for Writing a Clang Plugin.** We take Clang as another example to show our tutorials. This is a self-contained tutorial about how to write a short plugin in Clang which modify the source code as required.

Let's say that we want to analyze a simple C file as shown in Listing 1. Suppose we want to do some simple fixes on this C file. We would like to change the name of `func1` to `add` and `func2` to `multiply`. Then we would also like to change the function calls of `func1` and `func2` to `add` and `multiply` respectively. This will result in a code as shown in Listing 2. We can write a plugin which will parse through the AST and make the above changes to the file.

**Listing 1: Example input code**

```
int func1(int x, int y) { return x+y; }
int func2(int x, int y) { return x*y; }
int saxpy(int a, int x, int y) {
    return func1(func2(a,x),y);
}
```

**Listing 2: Expected output code**

```
int add(int x, int y) { return x+y; }
int multiply(int x, int y) { return x*y; }
int saxpy(int a, int x, int y) {
    return add(multiply(a,x),y);
}
```

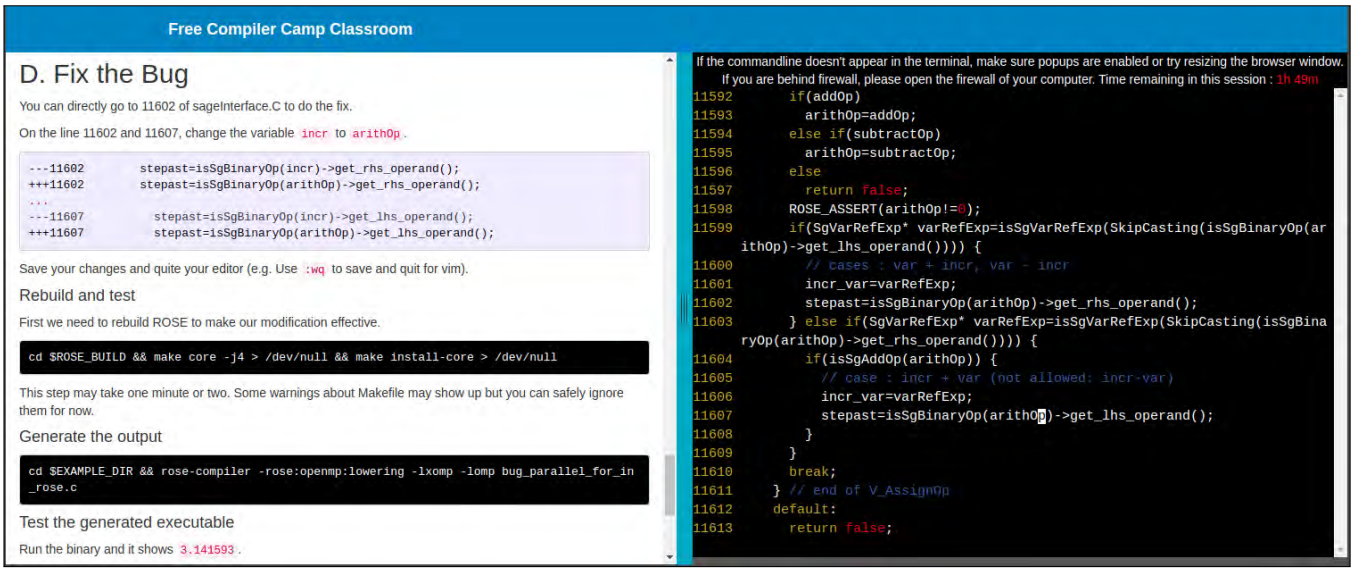



**Figure 4: The tutorial for fixing an OpenMP translation bug in ROSE**

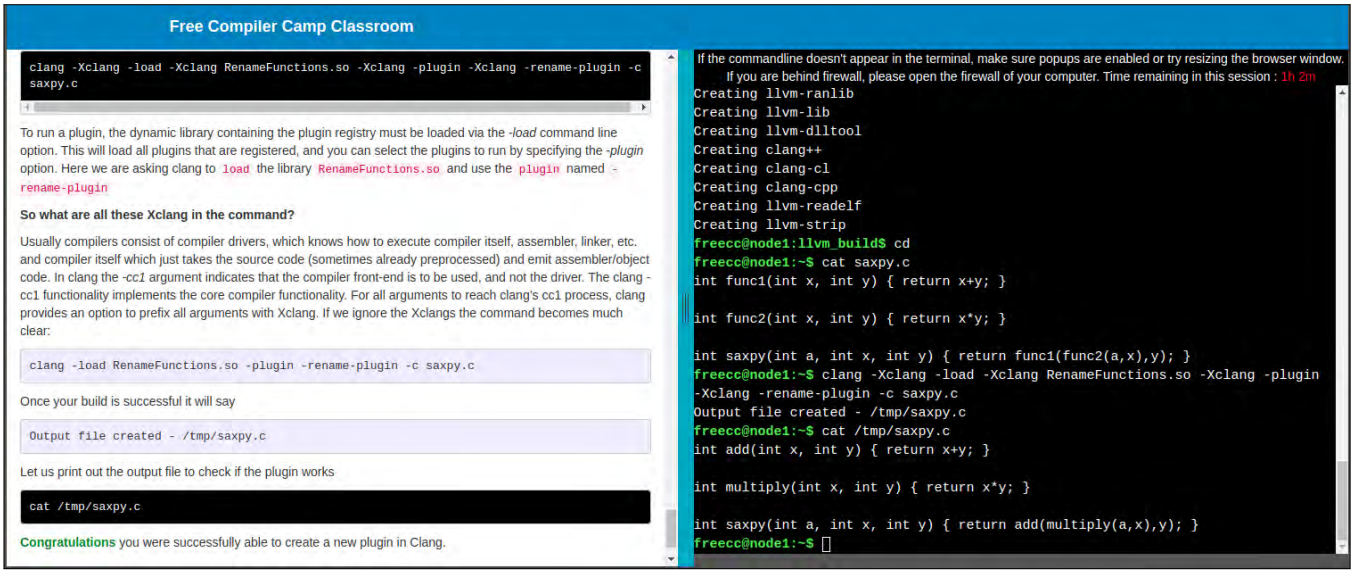



**Figure 5: The tutorial for writing a Clang Plugin**

This tutorial explains in details the steps that need to be taken to write this plugin. It starts with giving an overview of what is a clang plugin. Then it goes to explain what this plugin intends to do. Then it explains how to setup the source code structure of the plugin and which files need to be written or modified in order to write this plugin. The tutorial also provides an option to the user to download a reference plugin or to write the plugin by themselves. In the end it helps the user to build and test out the plugin. Figure 5 is a screenshot of this tutorial where the user tests the plugin.

## 5.3 Trial and Feedback

We have asked a group of students who major in Computer Science but with only basic compiler knowledge to take a trial of FreeCC. To assure the most accurate feedback, no pre-training ahead of the trial was provided. Students picked one tutorial based on their interests and completed it all by themselves without any other guidance. Then they filled a survey form about their experiences of using FreeCC. The feedback from the survey is summarized as follows:

- They feel comfortable with the length of each tutorial with 10-15 minutes.
- All steps of tutorial are completed without any issue.
- Students prefer to use clickable code snippets rather than type they manually.
- Providing a choice from multiple code editors will be helpful.
- Additional video instructions are not needed.
- The sandbox and clickable code snippets attracted the most attention. They make FreeCC unique comparing to conventional tutorials.
- Some students tried to conduct their own experiments in PWC as we expected.
- The overall appearance of FreeCC could be improved.
- They want to retrieve files from the sandbox (ssh or git might help).
- Support for X11 forwarding might be needed to display graphics.
- The tutorials can use some links to external courses for fundamentals about OpenMP and compilers.
- GPU support is needed for extending tutorials running on GPU.

Based on the feedback, we conclude that the current design of FreeCC tutorial is a very good start point. All testers are satisfied with the features of FreeCC. The sandbox, PWC, is highly rated since students don't need to configure any complicated environment but a modern browser on any system. Criticism mostly came from the website appearance, customization and cloud-machine resources for GPUs, which can be addressed in the future.

# 6 RELATED WORK

*Existing Compiler Tutorials.* Both ROSE [9] and Clang [1] already have abundant documentation on their official websites, including user guides, tutorials, and Doxygen generated API webpages, etc. There is also a ROSE wikibook which is open for anyone to contribute. Clang's official page provides documentation ranging from how to obtain and build clang, to how to write plugins and create tools, etc. Along with that there are several free and open source tutorial blogs which are available for Clang. OpenMP's official page provides links [8] to several open tutorials available on the internet. However, all the existing documentation is written in the traditional text format. It is still up to the readers to find a machine to install and configuration the development environment. The entire preparation phase may take hours to finish. Many learners simply give up due to the tedious steps or the lack of access to a suitable machine.

*Online Education systems.* There is a large amount of online learning systems [21], including Khan Academy [31], Coursera [2], edX [3] and so on. These learning systems mostly are aimed for

general education and training purposes. They are not specially targeting compiler development. A closely related website is freeCodeCamp [6], which is an online training platform for training web developers. Play-with-Docker is an online sandbox for people to learn docker. Our work builds on top of this framework with customization for compiler training.

Although several cloud-based tools have been leveraged for computer science education, there is a clear lack of such tools to teach compiler development. Ngo et. al [25] use CloudLab, a national experimentation platform for advanced computing research, to teach cluster computing to students. Bisbal [11] provides an outline of what topics need to be taught to computational scientists in a logical order to train them in open-source software. Shin et. al. [30] developed a web-based MOOC system related to computational science education which could hold various resources and efficient programming practices. Many such tools and resources are available across several domains of computation, but compiler development is still devoid of such online tools.

## 7 CONCLUSION AND FUTURE WORK

In this paper, we have introduced an ongoing effort, FreeCompilerCamp.org, a free and open online learning platform aimed to train researchers to quickly develop OpenMP compilers. FreeCompilerCamp.org is built on the Play-with-Docker platform to relieve learners' burden of finding suitable machines and installing software. The tutorials of FreeCompilerCamp are entirely web-based with both text content and a live embedded sandbox terminal in which learners can immediately practice compiler development skills. Instructors or students can customize this platform easily and deploy it on any local server, workstation or even personal laptop.

In the future, we will include more tutorials about how to develop OpenMP compilers for HPC. We will also design online examinations to help learners evaluate the effectiveness of their learning process. We welcome anyone to try out our system, give us feedback, contribute new training courses, or enhance the training platform to make it an effective learning resource for the HPC community.

## ACKNOWLEDGEMENT

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344, and partially supported by the U.S. Dept. of Energy, Office of Science, ASCR SC-21), under contract DE-AC02-06CH11357. IM Release Number: LLNL-CONF-791339. This material is also based upon work supported by the National Science Foundation under Grant No. 1833332 and 1652732. This research was also supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

## REFERENCES

[1] 2019. Clang Documentation. Retrieved Sep 26, 2019 from https://clang.llvm.org/docs/
[2] 2019. Coursera. Retrieved Sep 26, 2019 from https://www.coursera.org/
[3] 2019. edX. Retrieved Sep 26, 2019 from https://www.edx.org/
[4] 2019. GCC Support for the OpenMP language. Retrieved Jul 22, 2019 from https://gcc.gnu.org/wiki/openmp
[5] 2019. Intel C++ Compiler Code Samples. Retrieved Jul 22, 2019 from https://software.intel.com/en-us/code-samples/intel-c-compiler
[6] 2019. Learn to code with free online courses, programming projects, and interview preparation for developer jobs. Retrieved Sep 26, 2019 from https://www.freecodecamp.org/
[7] 2019. OpenMP support in IBM XL compilers. Retrieved Jul 22, 2019 from https://www.ibm.com/developerworks/library/l-openmp-support/index.html
[8] 2019. OpenMP Tutorials & Articles. Retrieved Sep 26, 2019 from https://www.openmp.org/resources/tutorials-articles/
[9] 2019. Rose Documentation. Retrieved Sep 26, 2019 from http://rosecompiler.org/ROSE_HTML_Reference/index.html
[10] Alfred V Aho, Ravi Sethi, and Jeffrey D Ullman. 1986. Compilers, principles, techniques. *Addison wesley* 7, 8 (1986), 9.
[11] Prentice Bisbal. 2019. Training Computational Scientists to Build and Package Open-Source Software. *Journal of Computational Science Education* 10, 1 (Jan. 2019), 74–80. https://doi.org/10.22369/issn.2153-4136/10/1/12
[12] C Cray. 2019. C++ Reference Manual, S-2179 (8.7). Cray Research. Retrieved Jul 22, 2019 from https://pubs.cray.com/content/S-2179/8.7/cray-c-and-c++-reference-manual/openmp-overview
[13] Leonardo Dagum and Ramesh Menon. 1998. OpenMP: An industry-standard API for shared-memory programming. *Computing in Science & Engineering* 1 (1998), 46–55.
[14] Bronis R. de Supinski, Thomas R. W. Scogland, Alejandro Duran, Michael Klemm, Sergi Mateo Bellido, Stephen L. Olivier, Christian Terboven, and Timothy G. Mattson. 2018. The Ongoing Evolution of OpenMP. *Proc. IEEE* 106, 11 (2018), 2004–2019.
[15] John Dewey. 1938. *Experience and Education.* Kappa Delta Pi.
[16] Tom Goethals, Dwight Kerkhove, Laurens Van Hoye, Merlijn Sebrechts, Filip De Turck, and Bruno Volckaert. 2019. FUSE : a microservice approach to cross-domain federation using docker containers. In *Proceedings of the 9th International Conference on Cloud Computing and Services Science*. Scitepress, 90–99. http://dx.doi.org/10.5220/0007706000900099
[17] David A. Kolb. 2014. *Experiential Learning: Experience as the source of learning and development.* Pearson FT Press.
[18] Chris Lattner and Vikram Adve. 2004. LLVM: A compilation framework for lifelong program analysis & transformation. In *Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization*. IEEE Computer Society, 75.
[19] Ilias Leontiadis and George Tzoumas. 2001. OpenMP C Parser.
[20] Chunhua Liao, Daniel J Quinlan, Thomas Panas, and Bronis R De Supinski. 2010. A ROSE-based OpenMP 3.0 research compiler supporting multiple runtime libraries. In *International Workshop on OpenMP*. Springer, 15–28.
[21] Tharindu Rekha Liyanagunawardena, Andrew Alexandar Adams, and Shirley Ann Williams. 2013. MOOCs: A systematic study of the published literature 2008-2012. *The International Review of Research in Open and Distributed Learning* 14, 3 (2013), 202–227.
[22] LLVM. 2019. Projects Built with LLVM. Retrieved Aug 31, 2019 from https://llvm.org/ProjectsWithLLVM/
[23] Dirk Merkel. 2014. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal* 2014, 239 (2014), 2.
[24] Lori S. Mestre. 2012. Student preference for tutorial design: a usability study. *Reference Services Review* 40, 2 (2012), 258–276.
[25] Linh B. Ngo and Jeff Denton. 2019. Using CloudLab as a Scalable Platform for Teaching Cluster Computing. *Journal of Computational Science Education* 10, 1 (Jan. 2019), 100–106. https://doi.org/10.22369/issn.2153-4136/10/1/17
[26] Marcos Nils and Jonathan Leibiusky. 2019. Play with Docker. Retrieved Jun 18, 2019 from https://training.play-with-docker.com
[27] Matt Pharr and William R Mark. 2012. ispc: A SPMD compiler for high-performance CPU programming. In *2012 Innovative Parallel Computing (InPar)*. IEEE, 1–13.
[28] Dan Quinlan and Chunhua Liao. 2011. The ROSE source-to-source compiler infrastructure. In *Cetus users and compiler infrastructure workshop, in conjunction with PACT*, Vol. 2011. Citeseer, 1.
[29] Jörg Schwenk, Marcus Niemietz, and Christian Mainka. 2017. Same-origin policy: Evaluation in modern browsers. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*. 713–727.
[30] Junghun Shin, Jason Cholhoon Jang, Huiseung Chae, Gimyeong Rvu, Jaejun Yu, and Jongsuk Ruth Lee. 2018. A Web-Based MOOC Authoring and Learning System for Computational Science Education. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*. IEEE, 1028–1032.
[31] Clive Thompson. 2011. How Khan Academy is changing the rules of education. *Wired Magazine* 126 (2011), 1–5.
[32] Anne Van Kesteren and et al. 2014. Cross-origin resource sharing. *W3C REC-cors-20140116, latest version available at< https://www.w3.org/TR/cors/* (2014).
[33] Chanho Yong, Ga-Won Lee, and Huh Eui-Nam. 2018. Proposal of container-based HPC structures and performance analysis. *Journal of Information Processing Systems* 14, 6 (2018), 1398–1404.

# Self-paced Learning in HPC Lab Courses

Christian Terboven
Chair for High-Performance Computing
RWTH Aachen University, Germany
terboven@itc.rwth-aachen.de

Julian Miller
Chair for High-Performance Computing
RWTH Aachen University, Germany
miller@itc.rwth-aachen.de

Sandra Wienke
Chair for High-Performance Computing
RWTH Aachen University, Germany
wienke@itc.rwth-aachen.de

Matthias S. Müller
Chair for High-Performance Computing
RWTH Aachen University, Germany
mueller@itc.rwth-aachen.de

## ABSTRACT

In a software lab, groups of students develop parallel code using modern tools, document the results and present their solutions. The learning objectives include the foundations of High-Performance Computing (HPC), such as the understanding of modern architectures, the development of parallel programming skills, and course-specific topics, like accelerator programming or cluster set-up.

In order to execute the labs successfully with limited personnel resources and still provide students with access to world-class HPC architectures, we developed a set of concepts to motivate students and to track their progress. This includes the learning status survey and the developer diary, which are presented in this work. We also report on our experiences with using innovative teaching concepts to incentivize students to optimize their codes, such as using competition among the groups. Our concepts enable us to track the effectiveness of our labs and to steer them for increasing sizes of diverse students.

We conclude that software labs are effective in adding practical experiences to HPC education. Our approach to hand out open tasks and to leave creative freedom in implementing the solutions enables the students to self-pace their learning process and to vary their investment of effort during the semester. Our effort and progress tracking ensures the achieving of the extensive learning objectives and enables our research on HPC programming productivity.

## KEYWORDS

HPC education, software lab, parallel programming, programming effort, training productivity

## 1 MOTIVATION

With the intent to make the dedication of our chair–the High-Performance Computing (HPC)–popular among students, to attract the best and highly-motivated students, and in general to engage with students early on and to foster their skills, we have created a series of HPC software labs. These support a diverse group of students in self-paced learning and are meant to accompany theoretical education in the field of HPC with a practical component.

As a requirement to execute the labs successfully, we have to be able to stem the course with very limited personnel resources. Nevertheless, we want to give students the opportunity to work on world-class HPC architectures. To support the students to reach the given learning objectives, we developed a set of concepts to motivate students and to track their progress. This also enabled our research on HPC development productivity.

This paper presents our *learning status survey* and the *developer diary* to track the student's progress in achieving the learning objectives, and our approach to *enable the comparison* of different HPC cluster architectures or parallel programming models. We also report on our experiences with using innovative teaching concepts such as using a *competition* among students to motivate them to optimize their codes for performance and show the opinions that students have towards these concepts.

Thus, the paper is structured as follows: In Section 2, we describe the structure and content of three different kinds of software labs that we have conducted at the HPC chair of RWTH Aachen University. Section 3 summarizes the learning objectives of our labs–classified into general and course-specific goals. To motivate our students and increase the success rate, we have created various stimuli that are presented in Section 4. Section 5 covers the methodology on how we track development effort and progress. In Sections 6 and 7, we evaluate the software labs in terms of obtained knowledge, training productivity and programming models, as well as students' feedback based teaching evaluations. Finally, we conclude in Section 8.

## 2 HPC SOFTWARE LABS

Within the Computer Science curriculum at RWTH Aachen University, a software lab is a mandatory part of Bachelor studies and expected to be completed in the 4th or 5th semester. It teaches practical skills. As part of the actual work within a software lab, students have to come up with a precise outline of the task at hand, develop code using modern tools, document the results and prepare a final presentation. Special emphasis is put on the experience to work in a group, including the challenging tasks to self-organize the development project throughout the semester. At RWTH, students have the opportunity to select from 10 to 15 different software labs that are offered each semester. These span a wide range of topics,


https://doi.org/10.22369/issn.2153-4136/11/1/10

covering the whole computer science domain at RWTH. While the students are not expected to have prior knowledge in the particular topic of a given software lab, they might have obtained knowledge based on their individual selection of optional courses. This diversity requires a flexible and self-paced approach towards HPC education. The following provides our concepts and experiences of the three different labs regularly carried out at our institute.

The first lab is called *Parallel Programming Models for Applications in the Area of High-Performance Computation* (PModels lab). Each group has to parallelize a serial code given as skeleton of a Conjugate Gradient (CG) solver for sparse matrices with three different parallel programming models on different hardware architectures. In the past, these were OpenMP for CPUs, and CUDA and OpenACC for GPUs. The different student groups start each with another programming model so that (amongst others) the chance to copy performance tuning steps from others is minimized. Before starting the course, the students get a basic introduction to these programming models. Then, they will work independently usually starting with a performance analysis (using corresponding tools) and investigating the hotspot of the application. It is key to efficiently parallelize this hotspot, i.e. the sparse matrix-vector product, to achieve good performance. For grading, basic parallel versions with reasonable performance are sufficient. However, as part of the competition, students are strongly encouraged to continue their performance tuning and apply tools and performance engineering of their codes until the end of the semester. This also includes thinking about (and implementing) performance models, new data structures for sparse matrices, minimization of data transfers, or other solutions to improve convergence of the solver. The final results are evaluated with respect to overall solving time for a given matrix.

The second lab is titled *Parallel Programming for Many-Core architectures with OpenMP* (OpenMP lab) that is offered every summer term. Each group has to solve three tasks and provide implementations with the parallel programming model OpenMP. In all tasks, the students are encouraged to use performance and correctness tools to analyze their intermediate and final solution. Task one is the parallelization of a *sparse-matrix-vector-multiplication*. The identification of the loop that has to be parallelized and the implementation of a first-parallel version is rather simple, but the main challenge is to achieve a load balance among the participating threads since the non-zero entries are irregularly distributed among the matrix. Thus, the students have to identify that the number of nonzero elements has to be the same for each thread, not the number of rows in the matrix. Further improvements of the execution performance have to be achieved by enabling SIMD vectorization and aligned allocation of the data. Task two is a *merge-sort* code, for which tasking has to be employed due to the recursive nature of the algorithm. Again, a first-parallel version can be implemented with little effort, but the challenge is to find a cut-off strategy to limit the overhead induced by the recursion. Both tasks have to be implemented for a many-core architecture with two different kinds of memory, namely regular DDR memory and high-bandwidth memory, and the students have to decide where the data elements have to be placed. Furthermore, both kinds of memory exhibit NUMA characteristics that have to be respected appropriately. In task three, a *k-means* code has to be ported to an accelerator (GPU) architecture. In this task, the effort to implement an initial-parallel version is higher, because the offloading requires more programming work and is more error-prone than the constructs that have to be used in the first two tasks. The key performance optimization challenge is to minimize the data transfer between host and GPU. Furthermore, the students have to understand which type of kernel and data volumes are profitable when offloaded to a GPU and deliver a performance improvement over the parallel execution on the host. In summary, the solution of the three tasks teaches students the key skills to program for contemporary HPC architectures and to perform performance analysis and optimization on these architectures.

The third lab is titled *HPC Cluster Challenge* (Cluster lab) and is offered every winter term. Each group receives a box of hardware with the task to construct a cluster from its contents. Afterwards, they have to parallelize and optimizing a Jacobian solver code aiming towards fully utilizing their respective cluster. The four sets of hardware for the four different groups are very different, so that the groups' experiences can be compared at the end of the software lab. All groups receive the same network equipment, which is pre-configured to allow connections to the university network and, thus, enabling remote work. We provide an overview lecture about what constitutes a cluster and provide general hints (not step-by-step instructions) on how to configure the network, a shared filesystem, etc. The first group receives a set of Intel-based desktop PCs with NVIDIA GPU cards. In consequence, the code has to exploit message-passing, multi-core and many-core parallelism. Although this is the most standard hardware, we found the groups are challenged to choose from the different configuration options and descriptions found online. The second group receives a set of NVIDIA Jetson boards, and again message-passing, multi-core, and many-core parallelism has to be exploited and there are different possible software configurations. The third group receives a set of Huawei Kirin boards. These are limited to message-passing and multi-core parallelism because the programming options for the so-called AI engine is not well-documented. The fourth group receives a set of 64-bit Banana Pi board, and again these are limited to message-passing and multi-core parallelism, this time because of limited capabilities of the graphics processor. In summary, the hardware ranges from low-power ARM-based SoCs to desktops equipped with GPUs. In all cases, the hybrid parallelization has to employ message-passing between the nodes, threading with each node and partly offloading to exploit accelerator units. The results are compared with respect to effort and price-performance.

Table 1 provides an overview of the three labs discussed in this work with the number of participating students, the group size and their average semester. It is noteworthy that we have improved our material and methodology (cf. Section 5) over time by taking feedback and new insights into account. To this end, we (still) used manual developer diaries in summer 2015 instead of the more advanced electronic approach introduced in Section 5. Furthermore, we focused on oral attestations and final grades to evaluate the students' gained knowledge in summer 2015. In later semesters, we added knowledge surveys (cf. Section 5) to extend and improve this concept.

**Table 1: Overview of the three software labs with the number of participating students, the group size and their average semester.**

| Term | Lab | # Students | # Groups | Semester |
|---|---|---|---|---|
| Summer 2015 | PModels | 14 | 7 | 4.5 |
| Summer 2016 | PModels | 12 | 6 | 4.1 |
| Summer 2017 | OpenMP | 18 | 6 | 5.4 |
| Winter 2017 | Cluster | 15 | 4 | 5.5 |
| Summer 2018 | OpenMP | 17 | 5 | 4.4 |
| Winter 2018 | Cluster | 15 | 4 | 6.4 |
| Summer 2019 | OpenMP | 16 | 4 | 4.8 |

## 3 LEARNING OBJECTIVES

The learning objectives of our labs can be classified into a general and a course-specific set of goals. The generic foundation of our HPC education lies in a thorough understanding of modern multi- and many-core processor architectures including CPUs (with various instruction set architectures) and accelerators. This foundation is paired with theoretical knowledge including parallelism, scalability and performance modeling to form analytical and assessment skills for a wide range of hardware architectures. We build upon this foundation by teaching software engineering skills and best practices geared towards developing parallel software. These include generic skills such as software requirements and design, documentation, version control and development diaries (cf. Section 5) as well as more specific tasks such as the correctness of parallel programs, debugging and performance analysis. Furthermore, we foster self-organization and collaboration through team work. Presentations of the results teach the students the visualization and description of software solutions, performance results and algorithms.

After completing the PModels lab, the students have a general understanding of shared-memory and GPU programming using different parallel programming models, i.e., OpenMP, CUDA and OpenACC. They know about methodologies to leverage the available parallelism and can clearly identify differences between low-level and high-level programming approaches. Moreover, students have an idea how to treat sparsity and how to apply optimization techniques to typical numerical solver such as the CG.

The goals of the OpenMP lab are similar: The students have a broad understanding of shared-memory and accelerator programming with OpenMP and its various techniques to map parallelism to hardware such as the concepts fork-join, tasking and accelerator offloading. Furthermore, the students are able to make profound decisions on how to parallelize scientific tasks for specific hardware architectures and optimize its hardware utilization.

The HPC Cluster Challenge lab focuses on a broad understanding of HPC clusters including the structure of clusters, networks, shared storage and the cluster management. A key goal is the understanding and analysis of power demands and energy efficiency of clusters. On the software side, the objective is to port and parallelize scientific tasks to a target cluster and the design and implementation of the task with suitable programming models (mainly OpenMP, MPI, CUDA-C/C++, OpenCL).

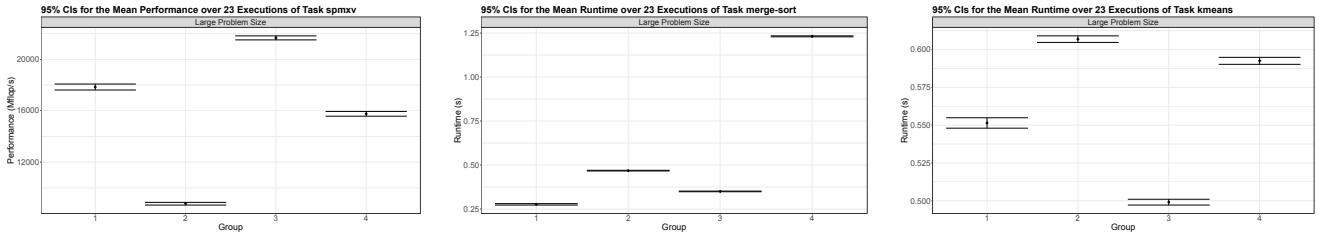
## 4 STIMULI

We use various stimuli in our labs to increase the success rate of the learning objectives while fostering creative solutions. We found that some of the tasks are especially challenging to derive from the objectives without providing a step-by-step guide which would hinder self-pacing, creativity and planning aspects. Thus, we define generic tasks based on the expected outcome such as 'implementing, parallelizing and optimizing a specific algorithm for a target architecture' and combine these with stimuli to increase the achievement of the learning objectives while fostering creative solutions.

The main stimulus we use is competition through group work in which prices (HPC-related books) are awarded to the team which achieves the fastest solution for all three tasks of the OpenMP lab, or all three parallel CG code versions in the PModels lab. As was outlined above, each lab partitions the students into three to five groups. Each group has to solve the same tasks in the same order, but the competition successfully ensures that solutions are not freely exchanged between the groups. In all three software labs, the solution of a task results in a parallel program for which in the execution on the target architecture the time can be measured. Each tasks offers a certain degree of freedom in the solution and the execution parameters so that the performance results differ between the groups. The winning group is determined via the formula-1 system: for each task, the fastest solution is awarded 25 points, the second fastest is 18 points, then 15 and 12 points. This ensures a fair and thrilling competition even in the presence of one group delivering a much better or worse solution than the rest in one particular task. In no instance of the software labs we have observed a single group clearly dominating the competition.

Figure 1 represents a typical result of the competition. It shows the result of all three tasks from the competition in summer term 2019. For each of the four groups, the resulting runtime of 23 repetitions is plotted. All four groups have delivered a working solution, applied the correct techniques to achieve reproducible performance with little variation, and achieved results in the same performance class. The third group won the competition since they achieved the highest throughput for task 1, the second-lowest runtime for task 2 and the lowest runtime for task 3.

Furthermore, we award creative solutions through presentation time during the oral attestations and the final presentations with all students. In order to expose the work of the HPC Cluster Challenge software lab to the IT Center, which also operates national HPC infrastructure, we selected a public and frequently visited space in between two building parts for the setup the clusters. In consequence, the clusters are on public displays and interested visitors can see the systems in operation, include power measurements, and the students at work. While this is certainly not comparable to the public display of the Cluster Challenge activities at ISC or SC conferences, the students reported that they like that atmosphere after a certain time of getting used to it.




**(a) Task 1** **(b) Task 2** **(c) Task 3**

**Figure 1: Competition results of the OpenMP lab in summer 2019.**

## 5 EFFORT AND PROGRESS TRACKING

One of our main learning objectives is the ability to document and present software projects in an especially performance-oriented context. These goals are coupled with our own motivation of investigating the progress of the students and the effectiveness of our labs. Therefore, we established a thorough data collection methodology which is used throughout the labs. While we do not grade the quality of said data, we regularly collect the data and provide students with feedback on how to improve their documentation skills.

We use the collected data to steer the lab for increasing sizes of divers groups of students as well as to develop and assess innovations in teaching techniques. The main output of our teaching curses is knowledge obtained as measured by the degree of achieved learning objectives while the input (cost) is the effort invested by the students in completing the course. Thus, the overall training productivity $TP$ is quantified by $\frac{\text{Degree of achieved learning objectives}}{\text{Training effort}}$. Furthermore, the data supports research into programming productivity on a increasingly heterogeneous set of computing hardware and programming models. The main output is hereby the achieved performance of the implemented parallel software: programming productivity $PP = \frac{\text{Achieved performance}}{\text{Training effort}}$. For better comparison among a cohort, we typically normalize the obtained data.

The following provides our methodology for collecting the three main productivity metrics knowledge, performance and training effort. The learning objectives include propositional knowledge obtained through preparational and supplemental materials such as programming courses, lectures[1], literature, best practice guides, etc. and procedural knowledge obtained through completion of design, implementation, programming etc. tasks. We use knowledge surveys (KS) [4, 7] where students rate their confidence in solving tasks on a three-point scale to quantify the changes in knowledge. This allows for a much higher assessment throughput than in traditional tests and KS provide a comprehensive self-assessment to the students. We assess our set of learning objectives with 40–50 tasks (approx. 30 minutes to answer) which has shown to be preferable regarding the overhead for the students and their participation rates. The KS are combined with oral attestation to capture additional learning objectives such as team-oriented skills, software-engineering methods and decision processes. The resulting confidence ratings of the KS are combined with average grades from all oral attestations (typically 2–3 over the course of the lab).

The performance of the software is typically captured by runtime or throughput on a specific system. Hereby, the system consists of pre-defined types of HPC cluster nodes for the OpenMP and PModels lab or the self-built cluster for the Cluster lab. The students may use all available resources of the system towards their performance goals. To increase the reliability of the collected performance data, we use multiple benchmark repetitions coupled with mean and standard deviations. To simplify the data collection, we provided run scripts, makefile targets and data collection spreadsheets.

The cost of the training is the effort in person-hours which consists mainly of the time attending the lab and the development effort for completing the task(s). While the attendance time is clearly defined, lots of the development is carried out outside the lab's presence hours. Thus, we use development diaries to record the quantity and type of effort carried out by the students. To maximize the accuracy and comparability of the data while minimizing the intrusion of the data collection, we develope the electronic development diary EffortLog[2] [6]. It uses strict input forms, precise questionnaires and fixed intervals (60-minutes has proven well) to achieve highly accurate data. Large efforts were recently put into increasing the usability of the tool which include a simplified layout, auto-completion, notifications to further improve data quality and summaries of the current project including performance results. Moreover, experience has shown that an operating system agnostic implementation with a minimal set of dependencies is key for the usability of the tool. The recorded effort data is related to the achieved performance by reminding the students to collect performance data and append it to the development activities. The resulting data contains the development of the achieved performance over the development effort as shown in the evaluation in Chapter 6. The main challenge of the tool is that it is currently not well-integrated into the typical development tool chain of the students. Harrell et al. [1] target this challenge by integrating data collection into git hooks. While this method promises high adaption through commonly used tooling, the accuracy of this method needs to be investigated further especially for student setups where we have observed that version control is often used sparsely and with low commit frequency. We intend to investigate the integration of

[1] See the list of courses and lectures of the HPC group of the RWTH Aachen University: https://www.i12.rwth-aachen.de

[2] The sources are publicly available on Github: https://github.com/RWTH-HPC/effortlog

**Table 2: Overview of the collected knowledge data during the OpenMP lab in summer 2018 and 2019. A KS rating of 3 means the ability to answer the question for grading purposes and 1 means not answerable by the trainee.**

| | 2018 | | 2019 | |
|---|---|---|---|---|
| | pre-KS | post-KS | pre-KS | post-KS |
| Mean | 1.97 ± 0.22 | 2.40 ± 0.30 | 1.32 ± 0.27 | 2.35 ± 0.33 |
| Median | 1.95 | 2.40 | 1.20 | 2.40 |

such tools or the combination of both kind of tools into a future version of our labs.

## 6 EVALUATION OF THE LABS

This section provides an overview of the results obtained with the data collected during the labs including investigations into the obtained knowledge and training productivity, as well as, differences in parallel programming models. The obtained knowledge is mainly captured by knowledge surveys carried out before and after the lab as described in Section 5. The participation is voluntarily and does not contribute to the grading of the labs. This protects the privacy of the students but often leads to incomplete data sets. Therefore, only the meaningful data is discussed which was obtained from the OpenMP labs in summer 2018 and 2019. We collected 19 valid surveys (6 people finished both pre- and post-KS) in summer 2018 and 12 valid surveys (4 people finished both pre- and post-KS) in summer 2019. The observed mean, median and standard deviations are provided in Table 2. The labs show to be very effective in achieving the learning objectives by a large increase in the confidence rating of the post-KS over the pre-KS for both labs. To investigate statistical significance of this data, we applied one-sided Wilcoxon signed rank tests to the collected data of people completing both pre- and post-KS. It shows statistically significant increases in the knowledge of the students with p-values of 0.00014 and 0.01046 for the OpenMP labs 2018 and 2019 respectively.

The collected productivity data of the students opens up wide areas of research into HPC programming productivity such as the estimation of software costs of HPC projects. While most of this research is ongoing and will require more data, some early results can be found in [2, 3, 6]. Figure 2 provides an example for the analyses carried out on the productivity data collected during the OpenMP lab in summer 2019. It shows the normalized performance (in relation to the best-effort solution) over the normalized development effort (in relation to each group's total effort). The anonymized data is provided by four groups of students for three tasks. It shows two distinct developments of the performance over the effort: A linear increase and a step-wise increase in performance. Linear increases are typical for groups working on small, incremental changes or only parts of the code which can lead to missed tuning opportunities. The collected data aids in identifying these groups early-on and supporting them in identifying the main performance limiters. Step-wise increases are often observed in groups who radically change parts of their code, algorithm or their launch configuration. A few of these changes (1–2 for these small projects) often lead to

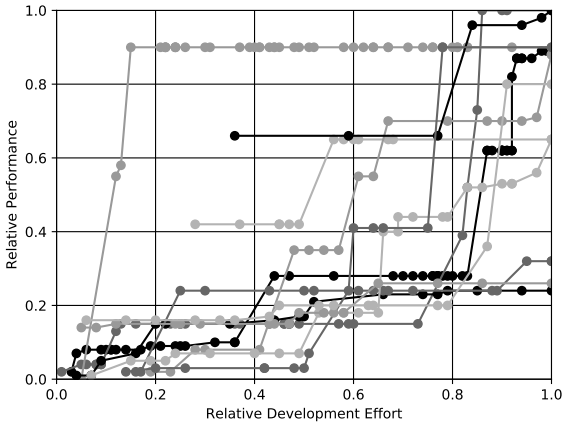



**Figure 2: Achieved performance over the development effort of $n = 12$ solutions during the OpenMP lab.**

| | OpenMP | OpenACC | CUDA |
|---|---|---|---|
| OpenMP | | 0.6812 | 0.0737 |
| OpenACC | 0.0737 | | 0.0210 |
| CUDA | 0.0161 | 0.2598 | |

runtime — effort

**Figure 3: $p$-values of one-sided Wilcoxon rank sum test with respect to students' development effort (upper triangle) and runtime (lower triangle) [5]. Significant differences (on a 5 % significance level) are marked in grey. Results are based on (valid) data, i.e. 11 student teams, from the PModels labs.**

large performance increases while most of the changes do not provide performance benefits. Our research focuses on modeling these functions, the understanding of the triggers for a sharp change and estimation methods.

To this end, it is important to also investigate different impact factors on development time and runtime. For example, results from the PModels labs show that the choice of the parallel programming model may affect the productivity [5]. Figure 3 illustrates the significant differences between OpenMP, OpenACC and CUDA in terms of development effort (upper triangle) and runtime (lower triangle). It expresses corresponding $p$-values in the way that the row item is significantly lower than the column item (on a 5 % significance level) where results are based on the one-sided signed Wilcoxon rank sum method. We find that the effort to use OpenACC is significantly lower than the one needed for CUDA programming. In contrast, the runtime achieved when using CUDA is significantly lower than with OpenACC. For the comparison of other programming models, we cannot draw any conclusions with this data. Nevertheless, this kind of evaluation of software lab data supports the hypothesis that parallel programming models affect productivity results so that this factor should be kept fixed for future investigations of other impact factors.

**Table 3: Teaching evaluation results averaged over all our software labs (except winter 2017 and summer 2019). BG = necessary background knowledge available, SOL = able to solve exercises alone or contribute to solutions in a group, MOT = exercises motivate student to solve them. Results given as percentage of students answering the question with 1 = strongly agree, ..., 5 = strongly disagree.**

| scale | BG [%] | SOL [%] | MOT [%] |
|---|---|---|---|
| n | 69 | 71 | 69 |
| 1 | 37.7 | 67.6 | 47.8 |
| 2 | 34.8 | 23.9 | 33.3 |
| 3 | 15.9 | 7.0 | 14.5 |
| 4 | 5.8 | 1.4 | 2.9 |
| 5 | 5.8 | 0 | 1.4 |
| average | 2.09 | 1.45 | 1.82 |

## 7 STUDENT FEEDBACK (BASED ON TEACHING EVALUATIONS)

To investigate whether our teaching concepts appeal to the students, we take students' feedback into account. For that, we use the results of the official teaching evaluations that RWTH Aachen University implements for each course at the end of a semester. These teaching evaluations consist of three main parts: statistical information, questions on a Likert scale (mostly ranging from 1 = strongly agree to 5 = strongly disagree), and free-form fields where students have the chance to mention anything that they liked and disliked about the course. Since the questions in all lab teaching evaluations are (mostly) the same, we can easily average or aggregate the results over all our software labs. The only exceptions are the evaluations in winter 2017 and summer 2019 where some of the questions did not appear. Thus, these questions have a reduced population (cf. the corresponding values of $n$). During our software labs, the students get a dedicated time slot to fill out the corresponding questionnaires. Nevertheless, participation is voluntary so that the number of responses $n$ may differ for each question. For this evaluation, we focus on questions that may reflect the effectiveness of our teaching concepts (instead of presenting the complete results).

In our seven labs, we supervised 107 students from which 103 took part in the teaching evaluation. These students are mostly, i.e. 87 %, in their second or third year of the Bachelor program in computer science. Furthermore, 72.5 % of the voting students, i.e., students who rated the question with one and two, denoted that they have the necessary background knowledge to complete this course where the average scoring is at 2.09 (cf. *BG* in Table 3). Thus, we assume that the students' feedback stems mainly from their experiences gained throughout our software labs—instead to their (missing) pre-knowledge.

First, we evaluate the overall concept of our software labs. Here, the students have rated the PModels lab on average with 1.8 and 1.5 in summer 2015 and summer 2016, respectively. The OpenMP lab was scored on average with 1.4 in summer 2017, with 1.2 in summer 2018, and 1.6 in summer 2019. The Cluster lab has been rated with 1.7 in winter 2017 and 1.9 in winter 2018. For comparison, we (only) have the average scores across all courses within the computer science department at RWTH Aachen University (for 2016 and 2017) available. In summer 2016, this overall average was at 1.8, in summer 2017 at 2.0, and, in winter 2017 at 1.9. To this end, our corresponding software labs are better rated than the average. Nevertheless, this interpretations should be taken with care since the computer science average also includes (compulsory) lectures that usually score worse than labs or seminars.

**Table 4: Teaching evaluation results aggregated aver all our software labs. Answers to the questions what students particularly liked or disliked, respectively, about the lab (in free-text form). Top three answers are presented if they have more than one vote.**

| like | | dislike | |
|---|---|---|---|
| topic | # | topic | # |
| independent working/flexibility | 11 | unclear goal | 14 |
| concept of tasks | 7 | little instructions | 11 |
| competition | 6 | | |

Getting more specific, we look at the feedback for our concept of tasks. 81.2 % of the voting students state that the exercises motivate them to solve the tasks. The corresponding question is rated with an average score of 1.82—as given by *MOT* in Table 3. Moreover, seven students particularly praise this concept in the free-form comments of the questionnaire (cf. Table 4). The competition as part of the exercises concept is explicitly mentioned positively six times. Contrarily, the fourteen comments that the goals of the software labs are not clearly given (cf. Table 4) is thought-provoking. Thus, we are continuously improving our corresponding material and goal statements without limiting the student's creative freedom in solving the tasks.

Finally, we investigate the concept of independent working and self-paced learning. From the free-text comments in Table 4, we see that it is received with mixed feelings. Eleven students particularly mention that they like the independent working and the flexibility that comes with self-paced learning. On the other hand, eleven students state that they do not like working without detailed instructions and developing (performance tuning) steps themselves. Assuming different learning types and the fact that students are not very familiar with this way of working through other courses, we still find these results a balanced relationship. This is especially true considering that 91.5 % of the voting students rated that they were (still) able to solve the exercises alone or contribute to solutions in a group (cf. *SOL* in Table 3) with an extremely good average score of 1.45. Correspondingly, 94.4 % of the voting students find the degree of difficulty appropriate (cf. Table 6).

Overall, students work mostly between one and five hours for the software lab outside of the classroom sessions. Taking the respective median from the different time intervals in Table 5, this comes to an average of 4.1 hours. As comparison, this is more than double

**Table 5: Teaching evaluation results averaged over all our software labs. Time for preparation and follow up work given as percentage of students answering the question ($n = 99$).**

| time | [%] |
|---|---|
| < 1 hr | 0 |
| 1 to 3 hrs | 26.3 |
| 3 to 5 hrs | 49.5 |
| 5 to 7 hrs | 18.2 |
| 7 to 9 hrs | 3.0 |
| > 9 hrs | 3.0 |

**Table 6: Teaching evaluation results averaged over all our software labs (except winter 2017 and summer 2019). Degree of difficulty reported, given as percentage of students answering the question ($n = 71$).**

| degree | [%] |
|---|---|
| appropriate | 94.4 |
| too difficult | 5.6 |
| too easy | 0 |

the time that students spend for exercises attached to regular HPC lectures taking place in the same semesters as the labs. Since grading does not require the best performing code version, these results indicate that students make use of the provided flexibility and are motivated to spend extra time for scoring well in the competition.

Given the students' feedback from the official teaching evaluations, we conclude to continue with our concept of self-paced learning while improving our material, e.g., with respect to elaborating on the goals of the software labs.

## 8 CONCLUSION

Software labs are effective in adding practical experiences to the HPC education and in enabling access to and hands-on experiences on world-class HPC systems. Our approach to hand out open tasks and to leave creative freedom in implementing the solutions enables the students to self-pace their learning process and to vary their investment of effort during the semester. These conclusions are also supported by students' feedback given through teaching evaluations. Our effort and progress tracking ensures the achieving of the extensive learning objectives and enables our research on HPC programming productivity.

## REFERENCES


[1] Stephen Lien Harrell, Joy Kitson, Robert Bird, Simon John Pennycook, Jason Sewall, Douglas Jacobsen, David Neill Asanza, Abaigail Hsu, Hector Carrillo Carrillo, Hessoo Kim, et al. 2018. Effective performance portability. In *2018 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*. IEEE, 24–36.

[2] Julian Miller, Sandra Wienke, Michael Schlottke-Lakemper, Matthias Meinke, and Matthias S Müller. 2018. Applicability of the software cost model COCOMO II to HPC projects. *International Journal of Computational Science and Engineering* 17, 3 (2018), 283–296.

[3] Marco Nicolini, Julian Miller, Sandra Wienke, Michael Schlottke-Lakemper, Matthias Meinke, and Matthias S Müller. 2016. Software cost analysis of GPU-accelerated aeroacoustics simulations in C++ with OpenACC. In *International Conference on High Performance Computing*. Springer, 524–543.

[4] Edward Nuhfer and Delores Knipp. 2003. 4: The knowledge survey: A tool for all reasons. *To improve the academy* 21, 1 (2003), 59–78.

[5] Sandra Wienke. 2017. *Productivity and Software Development Effort Estimation in High-Performance Computing; 1. Edition.* Dissertation. RWTH Aachen University, Aachen. https://doi.org/10.18154/RWTH-2017-10649 Apprimus Verlag, Published on the publication server of RWTH Aachen University 2018.

[6] Sandra Wienke, Julian Miller, Martin Schulz, and Matthias S Müller. 2016. Development effort estimation in HPC. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 107–118.

[7] Karl R Wirth and Dexter Perkins. 2005. Knowledge surveys: An indispensable course design and assessment tool. *Innovations in the Scholarship of Teaching and Learning* (2005), 1–12.


## A ARTIFACT DESCRIPTION: SELF-PACED LEARNING IN HPC LAB COURSES

### A.1 Abstract

This paper does not contain or rely on any computational results.

# Computational Mathematics, Science and Engineering (CMSE): Establishing an Academic Department Dedicated to Scientific Computation as a Discipline

Dirk Colbry
Michigan State University
East Lansing, MI
colbrydi@msu.edu

Michael Murillo
Michigan State University
East Lansing, MI
murillom@msu.edu

Adam Alessio
Michigan State University
East Lansing, MI
aalessio@msu.edu

Andrew Christlieb
Michigan State University
East Lansing, MI
christli@msu.edu

## 1. ABSTRACT

The Computational Mathematics, Science and Engineering (CMSE) department is one of the newest units at Michigan State University (MSU). Founded in 2015, CMSE recognizes computation as the "triple junction" of algorithm development and analysis, high performance computing, and applications to scientific and engineering modeling and data science (as illustrated in Figure 1). This approach is designed to engage with computation as a new integrated discipline, rather than a series of decentralized, isolated sub-specialties. In the four years since its inception, the department has grown and flourished; however, the pathway was sometimes arduous. This paper shares lessons learned during the department's development and the initiatives it has taken on to support computational research and education across the university. By sharing these lessons, we hope to encourage and support the establishment of similar departments at other universities and grow this integrated approach to scientific computation as a discipline.

## Keywords

Computational science; academic department administration.

## 2. INTRODUCTION

Establishing an entirely new department is no trivial task. One immediate hurdle is finances: the traditional university funding model would require that existing departments give up a portion of their budget in order to free up funds to create a new unit. At Michigan State University (MSU), the concept of Computational Mathematics, Science and Engineering (CMSE) was discussed at length and many faculty and administrators could see the potential for positive impact – but no one wanted to lose their existing funding. As part of these discussions, many alternatives to creating a new department were considered. For example, CMSE could have developed as a new focus area within an existing department (such as Computer Science & Engineering) or could have been the central theme for a new cross-disciplinary center or institute within the University's research unit. These solutions do not provide the same long-term foundation and commitment that results from the formation of a new department, however.

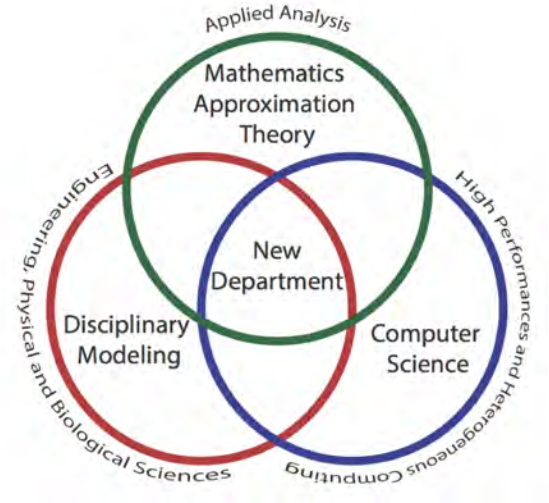



**Figure 1: Triple Junction of algorithm development and analysis, high performance computing, and applications to scientific and engineering modeling and data science**

Fortunately, in the early 2010s MSU announced its Global Impact Initiative (GII) [2], which offered new resources to bring more than 100 additional faculty to the university to pursue solutions to "Grand Challenges." One of these grand challenges was the continued advancement of computation in science, and the proposal to create a new CMSE department was an obvious fit for the MSU GII. Almost all faculty in the new department have joint appointments with other units, which created opportunities to leverage the GII funding to simultaneously create CMSE and grow the faculty in programs across campus.

The idea of "jointness" has been ingrained into the culture of CMSE from the beginning. The department is shared between the College of Natural Science and the College of Engineering. Faculty wear multiple "hats," typically in CMSE and in another STEM (science, technology, engineering, math) unit on campus. The department was designed from the start to encourage faculty to speak from two valued perspectives: the common language we are developing in CMSE, and the traditional language of their STEM departments.


DOI: https://doi.org/10.22369/issn.2153-4136/11/1/11


**Figure 2: The new department is shared between two colleges and almost all faculty have joint appointments between CMSE and another department.**

The CMSE department is home to computational thinkers from many fields and actively fosters discussion and collaboration across disciplines. 10 existing MSU faculty transferred (part of) their appointments into CMSE and the department conducted numerous searches to bring in 25 new, external hires. These faculty members have expertise in a range of science and engineering areas, as illustrated in Figure 2, as well as a variety of experience. Of the 35 faculty members of CMSE in Summer 2019, 22 were Assistant Professors, 3 were Associate Professors, 7 were Full Professors, and 3 were Academic Specialists (faculty not on a tenure track). In Fall 2019, 3 new hires are slated to join the faculty and the plan is to grow CMSE to 50 faculty members over the next few years.

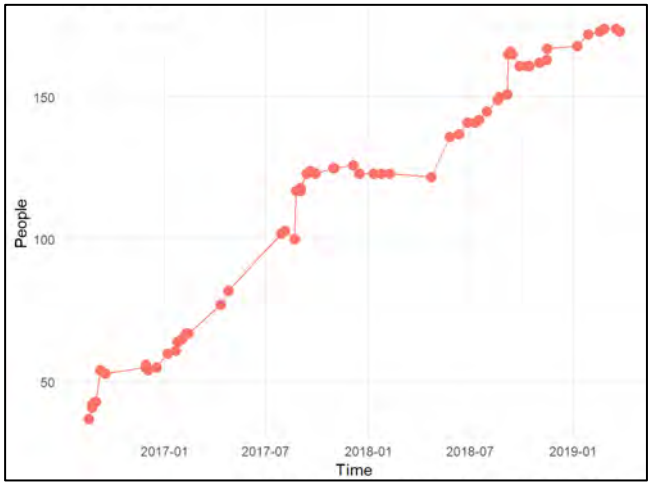



**Figure 3: Figure 3: Growth rate of the CMSE community. Measured by tracking faculty, staff, researchers and full-time students within the department email list.**

The rapid growth within CMSE (see **Error! Reference source not found.**) has not always been easy. Early on, senior faculty were burdened with abnormally high service requirements, in part because the many junior faculty needed to focus on earning tenure. The joint-appointment standard within CMSE meant that the department had to overcome many bureaucratic hurdles to ensure that tenure processes were aligned for faculty whose appointments spanned multiple colleges and/or departments. The diverse backgrounds of our faculty brought the benefit of different perspectives, but also meant that we had to work hard to establish effective communications across disciplines and research efforts. Several years in, however, the department seems to have hit its stride. While these problems have not gone away entirely, we have strong leaders and effective plans in place to help ensure the continued growth and success of the department and its academic and research programs.

## 3. UNDERGRADUATE EDUCATION

At MSU, all STEM undergraduates are expected to take some combination of common "gateway" courses (e.g., calculus, chemistry, physics, biology). CMSE has developed two scientific modeling courses: Introduction to Scientific Modeling (CMSE 201) and Tools for Scientific Modeling (CMSE 202). These courses focus on learning to program in the context of solving scientific and engineering problems and contribute to a parallel effort across MSU to add "computational competency" to the "gateway" learning goals for all STEM majors. Ideally, all STEM students will learn basic programming concepts within their first two years at MSU, which will enable instructors in higher level courses to use programming as a tool to more effectively teach other STEM concepts. For example, computational competency is now a requirement for all Physics majors and is a prerequisite in courses such as Linear Algebra (Math/CMSE 314), which uses real world examples and computational methods to teach Linear Algebra

The CMSE 201/202 course use a Flipped Classroom style of teaching that focuses on hands on learning inside of the classroom, with accompanying lectures provided in videos watched outside of class (see Figure 5). This course pedagogy is grounded in learning sciences [1] and is growing rapidly (see Figure 5).

Students who are excited by what they learn in CMSE 201/202 now have the option of earning an undergraduate minor in Computational Modeling and Data Science. This minor is targeted primarily at STEM students but is open to undergraduates from across the university. This minor gives students a solid background in programming and computational science through a 2-3 semester introductory course sequence; additional exposure to a breadth of methods in computational and data science, including a disciplinary-specific computational course; and options for a research experience or project-focused "capstone" course.

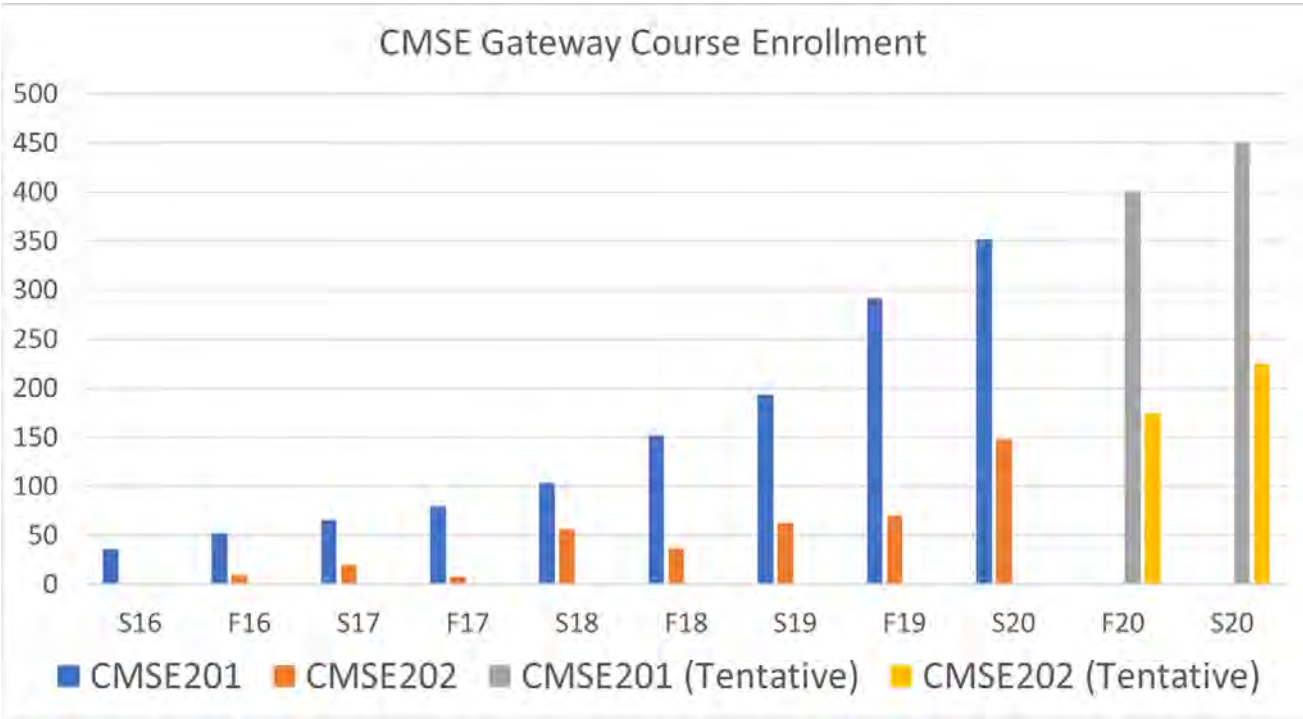



**Figure 4: Growth of 201/202 Enrollment by year as the CMSE department grows to serve the MSU STEM community**

Starting in Fall of 2019, MSU is offering a new undergraduate major in Data Science. This program is a collaboration between multiple departments (CMSE, Statistics, Computer Science), and MSU is working with other institutions to develop a common set of competencies for data science programs [3], We are very cognizant of the current hype surrounding data science and machine learning, and the explosion of "data science" programs across higher education. Unfortunately, the term data science is not well defined and there is not yet a standard understanding of the content of a "data science" degree. As part of its overarching mission to help establish data science as a discipline, CMSE is working across disciplines within MSU – and across institutions more broadly – to help define the disciplinary standards for data science programs.

## 4. GRADUATE EDUCATION

In 2016, CMSE launched three program options for graduate students: the Master's of Science, the Doctor of Philosophy, and a dual-major PhD program that allows students to combine CMSE with another doctoral major at MSU. These graduate programs are designed to help students develop broad skills for solving problems through computational modeling, data exploration, and high-performance computing techniques. Our graduate alumni will have acquired a broad range of computational skills, as well as substantial expertise in solving mathematical and statistical problems using scientific methods.

In Fall 2019, CMSE included 43 PhD students; 2 dual enrollment BS+MS students; approximately 25 dual-major PhD students; and 15 postdoctoral researchers. The CMSE graduate curriculum features a core set of courses in mathematical, numerical and computational methods: numerical linear algebra; numerical differential equations; parallel computing; and the mathematical foundations of data science. With this foundation, students may choose additional coursework that is tailored to their research interests; common examples include graduate courses in physics, applied mathematics, engineering and/or computer science. In addition to completing coursework, PhD students must pass qualifying exams in the four areas covered by the core curriculum and must write and defend a dissertation research plan for their comprehensive exam. The PhD is awarded upon completion and successful defense of their research dissertation.

The dual-major PhD option is administered by the MSU Graduate School and is open to all doctoral students at the University [4]. Students matriculate in one primary major, and then work with advisors to develop a cross-disciplinary program of coursework and research spanning an additional major area of study. Individual program plans are developed for each student pursuing a dual-major PhD and are generally put into place within the first 12-18 months of graduate studies. Upon successful completion of the individualized program, students earn a single Doctor of Philosophy diploma that reflects both majors.

CMSE has established a dual-major PhD pathway that allows students to pursue a substantial, novel, computationally-focused research program in consultation with at least one advisor (committee member) in CMSE; other advisors (committee members) may be drawn from any appropriate unit on campus [5]. CMSE already has the most dual-major PhD students of any department in the College of Engineering (the department's administrative home), and we anticipate that this interdisciplinary PhD option will be an advantage in recruiting new graduate students with novel research paths. For example, a typical dual-major PhD student in CMSE might be developing algorithms that are more computationally in-depth than is typical in their home discipline. By creating a dual-major PhD program, these students can craft a set of course and research requirements specific to their area of interest and gain access to the faculty and university resources to support their success.

Beyond the foundational graduate courses, the CMSE curriculum is designed to be nimble and allow faculty and students to explore new topics and research challenges as they arise. For example, in Fall 2019, the department offered nine special-topics courses on the following topics:

- Optimization
- Mathematical reasoning
- Foundations of computational science and engineering
- Applied machine learning
- Programming foundations for bioinformatics
- Statistical analysis and visualization of biological data
- Gaps and errors in statistical data analysis
- Applied linear algebra
- Applied calculus for bioinformatics

To illustrate the utility of this special-topics model, consider a specific example from Fall 2018, when several faculty jointly offered a course entitled "Algorithms for next-generation architectures." Students explored several different types of general-purpose graphical processing units (GPGPUs) and field-programmable gate arrays (FPGAs) and the software technologies required to use this hardware efficiently. The course encouraged students to think carefully about how to choose and develop algorithms that efficiently use a specific type of hardware to solve their problems. This course, like most graduate and undergraduate courses in CMSE, was taught in a "flipped" manner: students did substantial reading and other preparation prior to class, then during class they discussed these assignments, solved mathematical problems and proofs, wrote software, and analyzed data (see Figure 5). This teaching method has been shown to be very effective in a range of undergraduate STEM courses and has also been well received in our graduate courses.

**Figure 5: Example of flipped classroom, where students use classroom time to solve real world problems in groups.**

## 5. COMPUTATIONAL EDUCATION

In addition to establishing traditional undergraduate and graduate programs, CMSE seeks to support the development of computational competency more broadly. For example, the department has developed two graduate certificate programs, one in Computational Modeling and another in High Performance Computing [6]. These stand-alone certificates are earned by completing at least three courses (9 credits) from a list of approved options. Working professionals may enroll as lifelong learners (non-degree students) and pursue a certificate program to enhance their skills, or graduate students in other MSU programs may choose to complete a CMSE certificate in addition to their Master's or Doctoral program requirements. In the longer term, we hope to create a pathway that would allow non-traditional students to earn a Master's degree by completing several standalone certificates over time along with a culminating capstone experience. This could provide additional flexibility for working professionals who are not interested or able to pursue a full-time graduate program.

CMSE has also created a Bioinformatics Program to offer short, modular, introductory courses focusing on the development of basic skills in computation and bioinformatics. This program addresses the needs of MSU graduate students in biological sciences, who often seek additional training in how to work with the very large data sets now common in the life sciences. These short courses are designed to help students gain skills that can be immediately applied to their coursework and research, as well as helping to build computational competency and skills that can be leveraged if the students wish to pursue more advanced CMSE coursework.

## 6. CONCLUDING DISCUSSION

CMSE is uniquely positioned at the "triple junction" of algorithm development and analysis, high performance computing, and applications to scientific and engineering modeling and data science. In the four years since its inception the department has grown and flourished, establishing both traditional degree programs and non-traditional options to build computational competency in learners from across STEM. As the department continues to mature, we hope to support the formation of similar units at other institutions and to help shape the emerging discipline of scientific computation.

## 7. ACKNOWLEDGMENTS

CMSE would like to acknowledge the many people that were a part of the process of building this department: Stephen Hsu and the MSU GII Initiative of the MSU Office of the Vice President for Research and Innovation; the MSU Institute for Cyber-Enabled Research (iCER); members of the original CMSE Proposal Committee (Titus Brown, Departments of Computer Science and Microbiology & Molecular Genetics; Robin Buell, Department of Plant Biology; Andrew Christlieb, Departments of Mathematics and Electrical and Computer Engineering; Ian Dworkin, Department of Zoology; Michael Feig, Department of Biochemistry; Kathy Hunt, Department of Chemistry; Mark Iwen, Departments of Mathematics and Electrical and Computer Engineering; Ben Levine, Department of Chemistry; Vince Melfi, Department of Statistics and Probability; Filomena Nunes, National Superconducting Cyclotron Laboratory / FRIB; Brain O'Shea, Lyman Briggs College and Department of Physics and Astronomy; Charles Ofria, Department of Computer Science; Bill Punch, Department of Computer Science; Shin-Han Shiu, Department of Plant Biology; Yang Wang, Departments of Mathematics; GuoWei Wei, Departments of Mathematics; John Verboncoeur, Department of Electrical and Computer Engineering); and the CMSE faculty, staff and students.

## 8. REFERENCES

[1] Brian Danielak, Brian O'Shea, and Dirk Colbry. 2016. Using Principles from the Learning Sciences to Design a Data-Driven Introduction to Computational Modeling. In *Workshop on Teaching Computational Science (WTCS)*.

[2] Global Impact Initiative | Research at Michigan State University. https://research.msu.edu/global-impact/

[3] 2019 Data Science Leadership Summit, Nov 7-9, Santa Fe, https://sites.google.com/msdse.org/datascienceleadership2019/home

[4] Interdisciplinary Programs | MSU Graduate School. https://grad.msu.edu/interdisciplinaryprograms

[5] Dual PhD in Computational Mathematics, Science and Engineering. https://cmse.msu.edu/academics/graduate-program/dual-phd-in-cmse/

[6] Graduate Certificates | Computational Mathematics, Science and Engineering. https://cmse.msu.edu/academics/graduate-program/grad-certificates/

# The Supercomputer Institute: A Systems-Focused Approach to HPC Training and Education

J. Lowell Wofford
Los Alamos National Laboratory
Los Alamos, NM
lowell@lanl.gov

Cory Lueninghoener
Los Alamos National Laboratory
Los Alamos, NM
cluening@lanl.gov

## ABSTRACT

For the past thirteen years, Los Alamos National Laboratory HPC Division has hosted the Computer System, Cluster and Networking Summer Institute summer internship program (recently renamed "The Supercomputer Institute") to provide a basis is cluster computing for undergraduate and graduate students. The institute invites 12 students each year to participate in a 10-week internship program. This program has been a strong educational experience for many students through this time, and has been an important recruitment tool for HPC Division. In this paper, we describe the institute as a whole and dive into individual components that were changed this year to keep the program up to date. We also provide some qualitative and quantitative results that indicate that these changes have improved the program over recent years.

## KEYWORDS

training, education, recruiting, student programs, system management

## 1 INTRODUCTION

For the past thirteen years, Los Alamos National Laboratory HPC Division[10] has hosted the Computer System, Cluster and Networking Summer Institute (CSCNSI)[4][1] summer internship program to provide a basis is cluster computing for undergraduate and graduate students. The institute invites 12 students each year to participate in a 10-week internship program. The program is aimed at students interested in a broad range of HPC related fields, but provides a systems design and management focused curriculum. A number of recent articles have proposed training programs in HPC[15, 18], but these programs have been focused on applications and have only scratched the surface of lower–level HPC systems. We believe that the inclusion of a systems focused program can provide depth and perspective to many students, regardless of the HPC related field they intend to enter.

The institute breaks the program into two parts: (1) a "boot camp" running approximately two weeks covering fundamentals of cluster computing; and, (2) an eight-week-long guided research project. At the end of the program, students present their research in both a short talk and a poster.

The boot camp curriculum was significantly redesigned in the last year around a new methodology. Whereas in previous years, the boot camp largely consisted of guided projects to set up a small compute cluster, this year took a more directed education approach to teach the fundamentals of cluster computing before starting research. The theory in these changes was: (1) a ground-up foundation in cluster computing, starting with basic Linux skills, building to how clusters are designed and built, and then building and running parallel applications on them will provide a strong basis for any area of future HPC related research; (2) a curriculum based on practical guides with occasional theory lectures will provide a stronger foundation than self-guided projects; (3) frequent feedback through anonymous as well as named survey evaluations allow for day-by-day adjustments to the curriculum.

This approach has proven very successful based on comparative analysis of survey results from both students and project mentors, as well as the quality and complexity of the research results achieved in this institute. In this paper, we will layout the structure of the institute, the motivations, and changes made to the boot camp curriculum and qualitative and quantitative analysis of the institute outcomes. Our focus will be the evaluation of the impact this ground-up foundation in cluster computing has on subsequent student research. Because the new curriculum has only been provided for one year, the sample size of students is relatively small, however, the results suggest a strong positive impact on both students' assessment of the program and students' productivity in the research portion.

## 2 THE CSCNSI

### 2.1 Overview

The CSCNSI summer program is a 10-week paid summer internship sponsored by the High Performance Computing Division at Los Alamos National Laboratory (LANL). Each year's program starts in the fall with a recruitment and application process, from which 12 participants are selected based on qualities such as their existing skills, their current progress in school, and interests they express in their application materials. In parallel, HPC Division staff members propose projects that they would like to have CSCNSI students work on during the upcoming summer. Four projects are selected each summer, and each project is assigned a team of three students. When the participants arrive for the program, their teams and their project/mentor matches are already defined and they are immediately ready to start the program.

[1] Since August 2019, the CSCNSI has been renamed "Supercomputer Institute."


https://doi.org/10.22369/issn.2153-4136/11/1/12

The first two weeks of the program consist of a "cluster boot camp". This portion of the program focuses strongly on building base HPC systems knowledge, and includes work with bare metal hardware; booting and provisioning systems; system configuration and management; developing and running parallel applications; and looking at current and future HPC technology. For this portion of the program, each student team is given a 10-node HPC cluster to work with. Each team starts with an uncabled cluster with blank disks, and by the end of the two weeks they have built a fully-functioning 10-node, Infiniband–connected cluster that is capable of running real-life HPC applications.

During the remaining 8 weeks of the program, each team works with their assigned mentors on the project that was selected for their team. These projects normally make extensive use of the clusters the teams have just built, and may involve building and benchmarking parallel filesystems; writing and testing parallel applications; writing system software related to monitoring, containers, scheduling, or other operating system level topics; testing known security flaws for exploitability and to find mitigations; evaluating new networking technology; or almost any other topic of interest to HPC researchers. At the conclusion of this directed research period, each team gives a 20-minute presentation on their work to the HPC community at the Laboratory as part of the yearly HPC Division Student Showcase. A catalog of past projects and posters can be found at [5].

### 2.2 Process

The process of preparation for each year's CSCNSI session consists of three main sections: the student selection process, including recruiting, interviewing, and selecting participants in the program; the mentor and project proposal process, which results in the projects that will be worked on in the program; and the project matching process, in which selected students are matched with mentors and projects that fit their interests and skills.

The student selection process begins with recruiting in the fall of the year before a particular CSCNSI session. This recruiting occurs in conjunction with HPC Division's regular student recruitment activities at conferences, including the Grace Hopper Celebration[6], the Richard Tapia Celebration of Diversity in Computing[1], the International Conference for High Performance Computing, Networking, Storage, and Analysis[8], and at university site visits. These recruiting trips include on-site interviews and the ability for HPC Division staff to make spot offers to highly qualified candidates.

Meanwhile, the CSCNSI program is open to applications from other students via its website[4]. Applications typically open at the start of the fall semester, and application materials are typically due by early December. At the close of this application period, a selection committee from HPC Division reviews all applicants and performs phone interviews with the strongest candidates. Offers are made to selected candidates, and these participants are combined with any spot offers made at recruiting trips to make that summer's 12-member CSCNSI class.

In parallel with this process, potential mentors from HPC Division's technical staff propose projects for teams within the program. This process begins with a call for proposals from across the Division requesting the project title and a short abstract describing the project's goals and benefits, as well as the proposed mentors, any extra hardware that would be required by the project, and skills that are needed by students who would work on this project. These proposals are evaluated based on their technical merit, their ability to produce results by the end of the 10-week program, and their ability to expose the students to new technology. They are also evaluated alongside the applicant pool to ensure the skills needed by each project can be met by the selected students each summer. At the end of this process, four projects are selected to be worked on that summer.

After the participants and projects have been selected, the final step of the process is to build four three-person teams and assign them to the selected projects. This is done by comparing knowledge of students' backgrounds and interests gathered from their interviews, resumes, and application materials with the project required skills specified by the mentors, with the goal of matching students with projects that will offer them an opportunity for growth and an opportunity to be successful.

### 2.3 Technical

The CSCNSI is a multi-discipline program that starts with a two-week "bootcamp" that focuses on HPC systems hardware and software. Each team is supplied with a 10-node, Infiniband–connected cluster, and over the course of the bootcamp they learn to build their cluster from scratch. Their clusters start out as bare hardware: the nodes are racked, but all of the network and power cables are in a box in the rack. Starting with how to properly label and cable a rack of computers, the students spend their bootcamp period installing the operating system, installing scientific libraries, automating the node build process, and finally running MPI applications across all of their nodes. The process gives the students a strong understanding of the underlying technology that makes an HPC system work. The bootcamp curriculum is described in more detail in sections 3 and 4.

With their clusters built, the student teams are ready to work on their main project. The technologies used in these projects vary greatly depending on their focus. Recent projects include transparently running user application containers; testing the overhead incurred by compute node health checks; finding security anomalies in network flows; and testing the overhead introduced by speculative execution exploit fixes. Each of these projects digs deeply into individual aspects of system hardware and software, building on the base that the students learned during their bootcamp session.

## 3 THE CURRICULUM

### 3.1 History & motivation

The CSCNSI boot camp curriculum ("the curriculum") has grown organically during its long history, sometimes going for multiple years with only small changes, while at other times receiving large rewrites to update the material to better match updated technology. The program's instructor role has passed between multiple people in recent years, resulting in a series of updates that weren't necessarily self consistent, and this year a decision was made to do a major rewrite of the material. Using the existing material as a

topical guide, a new curriculum was built that included update technologies, removed outdated information, and more closely matched the realities of today's HPC environments.

Additionally, the previous approach to the boot camp left the students to mostly explore the different topics informally on their own, with little guidance. While this informal approach has some strong learning benefits, student surveys and previous instructor feedback indicated that this left some teams struggling to have viable systems for their subsequent research. Additionally, given the rapid pace of the boot camp, this approach severely limited the depth to which certain topics could be explored. All of these factors suggested that a new approach to the curriculum that merged both formal an informal learning may be beneficial.

To achieve a more guided approach to the curriculum, a significant amount of new material was required. For the 2019 curriculum, over 200 pages of technical guides and roughly 300 lecture slides were developed. These materials have been made public, and can be found at [9].

### 3.2 Methodology

The objective of the new curriculum, aside general updates and improvements, was to provide more formal learning components than the previous curriculum. This would allow the students to achieve the practical objectives of the boot camp—getting their teams' compute clusters deployed into a useable state—while also allowing more depth to be explored in more topics. Meanwhile, we did not wish to lose the learning benefits of the previous largely informal learning approach.

The previous curriculum split the boot camp into lecture and lab segments. The lecture segments were generally very short, with one to two presented per day. The lab segments would consist of an unguided list of tasks. Teams would go off to achieve these tasks with as-needed assistance by the instructors.

The alterations in the approach of the new curriculum were two fold: (1) to extend the content of the lectures to include more technical depth and more technical areas; (2) to replace the labs with "practica." These practica take the form of staged guides that have a mix of free exploration prescribed steps. These guides will be explained in more detail below.

At a high level, the boot camp curriculum builds the students' skills in stages. Since students come from diverse backgrounds with varied experience, we start with basic skills in using and installing the GNU/Linux operating system. By the end of the 12-day curriculum, the students have fully functional Linux compute clusters controlled through configuration management and using industry-standard HPC tools for provisioning, monitoring, and resource management. Students are introduced to a combination of facilities, systems, programming and visualization concepts, and tools.

Organizationally, the curriculum was divided into chapters. Each chapter begins with a theory lecture, followed by practical written guides, or practica. Most of the time is spent working through these guides. The guides are further subdivided into steps. It is expected that all of the students work through the guides and synchronize at the end of each step. This helps ensure that the entire class stays roughly on the same content throughout. Maintaining synchronization of the students is important for both efficiency in teaching and assistance, as well as making sure that students are focused on the same kinds of tasks at the same time. Keeping students in sync means that questions from other students remain timely and relevant, and other students are actively working on the same projects, and therefore are more prepared to assist fellow students. During each step the instructor and assistants help teams that had questions or were stuck with portions of the guide. At the end of each step, the instructor summarizes the step, performs the step on an example cluster, handles any high-level questions related to that step, and briefly introduces the goals of the next steps. For most guides, each step has an accompanying slide with additional notes for that step.

To keep the more advanced students occupied as well as introduce more advanced concepts such as advanced shell scripting, for most guide steps a "challenge" problem was assigned. These challenge problems leverages material from the section, as well as requiring some outside information that the students must research. Examples of challenge problems include: using the "find" command to do a recursive find-and-replace operation and writing a shell script to do a ping scan on a network. Teams that finished the challenge were asked to present their solutions to the group, along with explanations, and group discussion of the different solutions was encouraged.

## 4 CURRICULUM OVERVIEW

The curriculum for the boot camp is divided into 11 chapters. See Table 1 for a syllabus of the curriculum. For the condensed two-week boot camp, each chapter approximately represents the curriculum content for one day. Each chapter is designed to both add relevant HPC skills and further the process of bringing the teams' 10-node compute clusters into a usable and maintainable state for the later research portion. Below we outline the curriculum by chapter, for each chapter summarizing the structure, content and motivations for that chapter. The chapters fall into logical groupings based on their overarching objective. We have broken them out by these groupings below.

### 4.1 Introducing HPC

The first two chapters of the curriculum provide a general introduction to the course and some higher level concepts of high-performance computing, systems, hardware, workflows, and facilities. These chapters provide a backdrop and motivation for the remainder of the course, and the ideas introduced in these chapters are designed to develop throughout the course. There is an emphasis on the kinds of problems that HPC helps to solve, how to design systems to solve these problems, and the subsequent challenges of these system designs.

*4.1.1 Chapter 1: Introduction to HPC.* This chapter provides the motivation for the rest of the course. While the course works by building up HPC systems knowledge from the ground up, the introduction takes a top-down approach to understanding HPC. In the introduction, we focus on the kinds of problems that scientists may need to solve. We then lay out how cluster computing designs provide an effective architecture for solving these problems. This helps to motivate the course by starting with a focus on research

| | Title | Purpose | Practical |
|---|---|---|---|
| Chapter 1 | Introduction to HPC | Overview of HPC systems, hardware, workflows, and facilities. | N/A |
| Chapter 2 | HPC Facilities | Space, power & cooling challenges for HPC. | Cable and label cluster racks |
| Chapter 3 | Exploring Linux | Basic Linux system operating system concepts, install, use, and administration. | Master node is installed with Linux. |
| Chapter 4 | Networks & Services | Basics of netoworking and common Linux services. | Master network, NAT, ssh configured. |
| Chapter 5 | Netboot provisioning | How to stateless netboot a node from scratch. | Some nodes provisioned, 1st pass. |
| Chapter 6 | HPC provisioning | Using HPC provisionig tools to provision the whole cluster. | All nodes provisioned, 2nd pass. |
| Chapter 7 | HPC tools | Overviews of common HPC tools for system management, scheduling & fabric management. | Clusters configured with workload management, high-speed network, power, and console control. First jobs run. |
| Chapter 8 | Version Control & Config Management | Learn version control and configuration management tools and motivations. | Clusters re-provisioned, configured with version controlled configuration management. |
| Chapter 9 | Monitoring & Benchmarking | Overview of tools used for benchmarking and active/passive monitoring clusters. | Monitoring and log analysis framework installed. Baseline benchmarks taken, system verified. |
| Chapter 10 | Parallel & Cluster Programming | Introduction to parallel programming concepts and challenges. Introduction to cluster programming with MPI. Visualization tools. | Job submissions and MPI functionality tests. First parallel runs. Real scientific application run & visualized. |
| Chapter 11 | Future technology | Discuss revolving topics of future interest to HPC. | N/A |

**Table 1: Syllabus for boot camp curriculum. The title, purpose of the chapter are given, and practical lessons are given for each chapter. Shading represents groupings used in section 4.**

problems, motivating the general cluster architecture, discussing some important particular details of that architecture, and then working on the tools to practically build a system with a clustered architecture.

The toy research problems that are used as motivation for the *Introduction to HPC* chapter reappear in later chapters as job and programming examples that can be practically run on the systems that the teams deploy throughout the boot camp. This aims at keeping the students focused on why the systems are being built while constructing them in stages from the ground up.

*4.1.2 Chapter 2: HPC facilities.* The general discussion on HPC system design in the previous chapter naturally segues into a discussion of the kinds of physical, power, and cooling concerns surrounding large clusters of computers. The second chapter overviews the HPC facilities topics.

The HPC facilities introduction also provides the first practical lesson for the students. As part of the facilities introduction, the students are introduced to particular racking, cabling, and data center organization techniques. With this introduction, the students are then guided through physically cabling and labeling their teams' clusters[2].

## 4.2 System Management

While we do require some Linux experience for admission to the program, the level of Linux experience has varied widely among the students. To achieve a baseline of knowledge in Linux, the chapters 3 and 4 cover some of the basic Linux skills required for HPC, ranging from basic commandline skills to basic network and system service configurations. Throughout the guides for these chapters, "challenge" questions are offered to students who finish early to start building shell scripting knowledge. Each question pushes the students to find a new way to explore the Linux system by writing a script.

*4.2.1 Chapter 3: Exploring Linux.* This chapter is longer than other chapters and spanned two days. We begin with a lecture on an overview of Linux. This lecture splits into three parts. The first part covers some history of Linux as well as Linux and open source community issues. It also touches on why we use Linux for HPC. The second part of is focused on the Linux kernel and operating system theory. The third part provides an overview of Linux distributions.

Discussion of distributions in the lecture leads to a lab where the students install CentOS Linux[3] on their cluster master nodes following a basic install guide. Students perform the rest of the work throughout the bootcamp on this system.

Following the install procedure, students work through two guided practica on using Linux. Students are instructed in the use of the tmux[13] tool to share terminal sessions across their individual workstations. The first guide covers a wide range of common Linux tools with an emphasis on tools of particular use in HPC environments. The second guide focuses on those tools dedicated to inspecting the Linux system status and health.

[2]For our boot camp, due to time and safety concerns, the clusters are pre–racked but un–cabled when the students arrive.

*4.2.2 Chapter 4: Networks & Services.* Chapter 4 continues the exploration of the Linux. A beginning lecture covers fundamentals of networking and Linux networks, as well as Linux network services.

The lecture is followed by a guide that explores setting up and using various network settings and services in Linux. An emphasis is put on verification steps as each configuration step is performed. This guide also includes an exploration of systemd and service unit files. At the end of this guide, the master nodes have a complete network configuration and NTP, SSH and nginx services have been configured.

## 4.3 Cluster Provisioning & HPC Tools

Chapters 5 through 8 center around cluster provisioning. This is done in three stages. The theory is to start by provisioning manually, and add useful layers of abstraction in stages. First, the system is provisioned by manually creating a stateless booting cluster using common services and a combination of provided and student-developed scripts. Next, the system is re-provisioned using a common cluster provisioning system (Warewulf[14]). In the third iteration, the systems are re-provisioned again using configuration management (Ansible[2]) in conjunction with cluster provisioning (Warewulf). Chapter 7 is injected in the middle of this sequence to introduce core HPC tools not related to provisioning, such as the workload manager, before moving on to the final stage of provisioning. At the end of Chapter 8 the teams should have fully-functional, useable compute clusters.

*4.3.1 Chapter 5: Netboot provisioning.* Chapter 5 consists of one long guide that steps the students through everything necessary to perform a stateless (diskless) network boot of a compute node. This follows directly on the discussion of network and services in the previous chapter, and configures the core services (DHCP and tftpd) required to perform a PXE boot. The students are also guided through the process of manually constructing a node image to be provided to the compute nodes. Finally, the students are given a base initramfs image[3] that they can use to construct the staged boot required by most stateless compute clusters. This simplified initramfs has been constructed with the intent of educating, so the provided init stage scripts choose simplicity and readability over features. By the end of this chapter, the teams' clusters have two nodes provisioned using this method, in addition to the manually installed master node.

*4.3.2 Chapter 6: HPC provisioning.* Chapter 6 builds on Chapter 5 by showing how HPC provisioning systems, in this case Warewulf[14] can be used to dramatically simplify the process that was worked through in Chapter 5. Because Warewulf simplifies the netbooting process, this also affords the opportunity for the teams' to build more configuration complete and feature rich images for their nodes. At the end of this section, the entire cluster has been provisioned with Warewulf. In order to simplify access to packages and HPC tools, the OpenHPC project[11, 17] is used for additional HPC software repositories.

[3]The initramfs source can be found under the "Supplements" folder in the curriculum materials repository at [9]

*4.3.3 Chapter 7: HPC tools.* Up until this point, the students have not been introduced to some of the fundamental tools for HPC. It is useful to pause to look at some of these tools before the final provisioning step in order to make them available in the final provisioning of the clusters.

Several tools are introduced in this section that provide console and power access to the nodes and InfiniBand fabric management. Particular attention is paid to workload management and scheduling. Using the existing Warewulf install, the Slurm workload manager is installed, configured, and tested.

A final section of this chapter provides a short guide for working with Slurm as a user. This includes various forms of job submission, job inspection, and batch job scripting.

*4.3.4 Chapter 8: Version Control & Config Management.* The final step the provisioning process involves moving all of the work that has been done into a version controlled repository containing a configuration management specification. Git is used for version control and Ansible is used for configuration management.

The chapter begins with a short lecture covering Git concepts, followed by a practical learning guide for basic Git usage. Next, a lecture is given on general configuration management concepts with an emphasis on Ansible. The Ansible practical guides are divided in two. The first guide teaches basic practical Ansible examples. The second of the Ansible guides takes the users through the process of re-provisioning their clusters with Ansible and Git. A base Ansible repository is provided to the teams, and they are left to integrate their cluster-specific changes as well as the examples worked out in the previous tutorial into this Ansible repository. Finally, the students are instructed to completely re-install their master nodes, and re-provision their entire clusters with this Ansible repository. Any further changes to the system are affected through version controlled Ansible. At this stage, the teams have fully functional compute clusters managed using modern configuration management techniques.

## 4.4 Monitoring & Benchmarking

Chapter 9 is focused on monitoring and benchmarking tools. These are related but disparate topics. They are presented separately but combined into the same chapter for brevity and to emphasize their common use to verify the state of the cluster.

*4.4.1 Chapter 9, Part 1: Monitoring.* We first start with a short lecture that covers the basic terminology used in HPC monitoring such as active versus passive monitoring, out-of-band monitoring, and a summary of HPC monitoring concerns.

The practical portion of the monitoring section focuses on passive log analysis. The students work through configuring rsyslog log aggregation from the compute nodes to the master node. They are then guided through setting up and using Splunk for log analysis, including setting up Splunk alerts.

*4.4.2 Chapter 9, Part 2: Benchmarking.* This section of the chapter also starts with a short lecture covering common terminology around benchmarking, and the role of benchmarking in typical HPC acceptance processes. Emphasis is placed on real versus synthetic and micro versus macro benchmarks. A brief survey of these different benchmarking methods is presented, including several

industry-standard benchmarks are introduced, including HPL and various synthetic micro-benchmarks (e.g. STREAM, iozone, and the MOFED IB benchmark tools).

In the practical guide for benchmarking, several of these tools are used to gather information about the teams' clusters. This also serves as an "acceptance" stage for the teams' clusters in which they can verify that performance of the systems is as expected and comparable to the performance of other teams. The students are also instructed to run some scaling benchmarks, where a benchmark starts at a single thread, and scales until it comprises the entire system. This gives an idea of how well the system will scale with idealized parallel applications.

As a final step in the benchmarking processes, the students were provided with a ready-to-run real applications. We chose a GROMACS[7, 16] molecular dynamics simulation, as it is a well supported, open source code, and was easy to force to run for predictable time periods. The students ran this code over night. This both provided a way to perform a "real" benchmark on the clusters, as well as data for visualization exercises in the next section.

### 4.5 Parallel & Cluster Programming and Visualization

Chapter 10 introduces parallel programming concepts. We should note that the focus of the boot camp is not on parallel programming - LANL offers the "Parallel Computing Summer Research Internship"[12] that focuses in this area for students who would like more emphasis on programming. Instead, the objective of this brief introduction to parallel programming concepts is two fold: 1) to introduce a basic common understanding of parallel programming techniques and pitfalls as may be relevant to their research projects; 2) to introduce cluster programming and message passing concepts through a simple introduction to MPI programming. These, together, help to give a more complete understanding of how the systems the students have assembled will be used in addition to providing more hands-on experience with the high-speed networking fabric.

The chapter is introduced with a lecture covering some basic theory and terminology of parallel programming, including message passing versus shared memory models. The practical guides are divided into two. The first guide walks the students through some simple threaded programming tasks in Python. The second guide extends these concepts to span multiple nodes using the Python mpi4py module.

Both the threaded and MPI guides follow examples of developing and enhancing code that illustrates two examples that were given in Chapter 1 as illustration of why we build HPC systems as clusters. The first example illustrates a simple parallel summing algorithm. The second example is more complex and implements different versions of a 3D box of colliding particles[4].

Finally, both the 3D box example and the results of the GROMACS sample simulation provide input data for some brief visualization experiments. For the 3D box example, students are guided through making a 3D rendered movie with ParaView. For the GROMACS example, the students are guided through a 3D visualization using VMD, the molecular dynamics visualization tool.

### 4.6 Future Technology

Chapter 11 concludes the boot camp with a discussion of upcoming HPC technologies. In the current year, chapter introduced ideas like Linux container and cloud computing. It is anticipated that this chapter would change substantially over time to match new upcoming technology trends. Due to time constraints for the boot camp, this chapter consisted solely of a lecture. Ideally, it might include some short, practical guides for working with some of the new technologies mentioned.

## 5 EVALUATION

The CSCNSI is a program that we have traditionally had difficulty evaluating. Unlike some summer programs, its value isn't as much in the results of the students' final products as it is in foundation we give them for understanding the fundamentals of high–performance computing. For many students, this means that we do not have a good way to evaluate the value of the program once they have left for the summer unless we make efforts to track them down and check on their progress in school and their careers. However, as we have already mentioned, this program is an important recruiting vehicle for LANL's HPC Division, meaning that we can put at least one numeric score on the success of each year. This, in conjunction with student surveys conducted throughout the program give us some qualitative and quantitative ideas of the success of the program.

### 5.1 Short Term: Qualitative Evaluation

Students who attend the CSCNSI are encouraged to give feedback, and they are given frequent opportunities to do so. This year, the instructor implemented a daily "sticky note" survey: each team was given a pad of sticky notes at the start of the day, and each member was asked to briefly summarize their feelings at the end of each day. These notes were treated anonymously and were used by the instructor to tailor the pace of the class and the topics being covered each day to fit the needs of the students.

At the end of the summer, the students and the mentors were asked to fill out a longer, more formal survey about their experiences that summer. Afterward, the results of these surveys were used to evaluate how the class went, decide which students should be followed up with by our recruiting team, and begin planning for the next year.

The surveys between the summers of 2018 and 2019 were also significantly updated, so it is difficult to directly, quantitatively compare the two results. Qualitatively, however, the student evaluation of the boot camp was significantly more positive than previous years[5]. Additionally, the overall approval rating of the program improved. Given that the boot camp was the most significant change between the years, it is reasonable to assume that the overall evaluation of the program benefited strongly from the updated boot camp curriculum. The curriculum changes introduced this year

[4]Source code for the 3D box simulation, called "gas", can be found in the curriculum Supplements at [9]

[5]The authors were unable to obtain releases for the survey data, so we are only able to speak subjectively and qualitatively about that data.

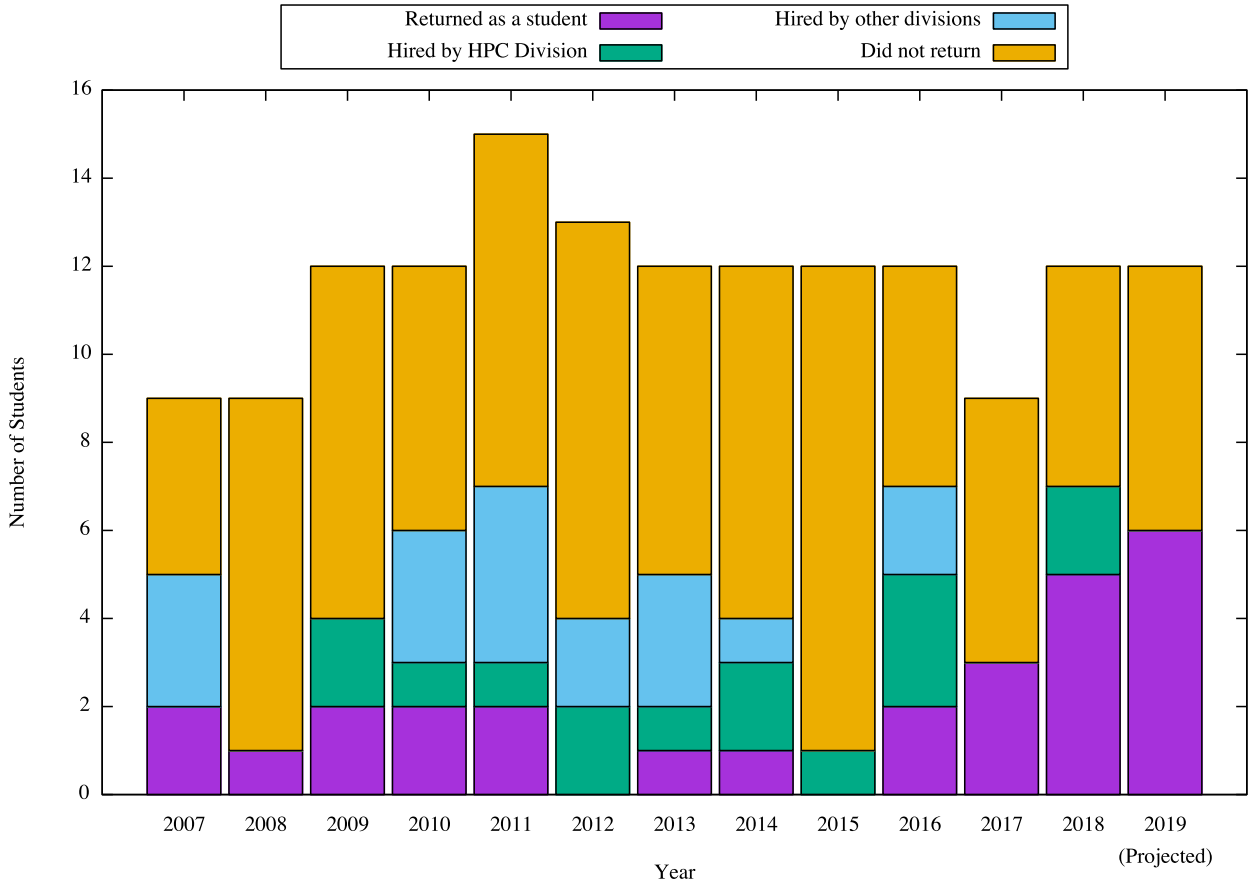



**Figure 1: Student outcome statistics for CSCNSI from 2007 to present.**

were significant enough that they offer us an opportunity to draw a strong distinction between the "old" and "new" curricula, which will give us a good place to do comparisons as the "new" curriculum ages and matures.

### 5.2 Long Term: Quantitative Evaluation

The CSCNSI has been an important recruiting tool for both HPC Division in specific and the laboratory as a whole. Since 2007, we have been maintaining records of the students who went through the program, which ones came back again as a student in our general student program, and which ones were hired as full time LANL staff in HPC Division or other divisions at the laboratory. Ignoring 2019, which is too early to count in the statistics, we have had 142 individual attendees in the program (with some years varying from the standard 12 attendees). Of these, about 15 returned to LANL as a student again, another 15 were hired as full time employees in HPC Division, and nearly 20 more were hired by other divisions at LANL. While we do not directly track statistics on students who end up at places other than LANL, we do know that several more have ended up at other laboratories and nearby businesses. The skills that the CSCSNI students learn during their summer are clearly applicable with HPC Division directly, but also with a variety of other scientific disciplines and industries. Figure 1 shows the outcomes, where known, of CSCNSI students from 2007 to present.

## 6 FUTURE WORK

We anticipate further developing the curriculum, the research, and the mentoring segments of the CSCNSI program going forward. Through the various sources of survey information, we will be making further minor curriculum adjustments, but overall feel that the new curriculum has provided a solid foundation to work on.

This years changes have emphasized the need for capturing better metrics on the performance of the program. Some of this may be achieved through ongoing improvements to the survey process. We are also examining the possibility of introducing entrance and exit exams to track student development.

Some students and mentors have pointed out that it would be desirable to spread out the boot camp curriculum through the program and to introduce the research components earlier. We are taking under consideration that the initial boot camp could be shortened

to include only include Chapters 1 through 8, with the remaining chapters taught in a more spread–out fashion throughout the remainder of the program.

Finally, we have opened the curriculum[9] to the broader community in hopes that it may both benefit the broader HPC educational community, as well as open a forum for community curriculum development. We have begun speaking with outside institutions that may be interested in helping to develop the curriculum for an academic course our workshop. The time frame of the CSCNSI limits the boot camp to an intensive two week period, but we believe this curriculum could be adapted and expanded to a semester course.

## 7 CONCLUSIONS

The CSCNSI program has a proven track record of demonstrating that a broad systems–based background in cluster computing can be a valuable background for students in a variety of HPC related fields. We have seen this qualitatively, through student and mentor surveys, and quantitatively, through the hiring pipeline it has provided. The changes to the CSCNSI program in the past year have marked a turning point for the program. We anticipate that the improved curriculum will further emphasize the benefits of a ground–up, systems based background in HPC. Though it is difficult to make broad conclusions given the limited sample size after one year, initial results are promising that this new mix of formal and informal learning will lead to an even stronger program going forward.

## REFERENCES


[1] 2019. ACM Richard Tapia Celebration of Diversity in Computing. (2019). Retrieved Sep 23, 2019 from http://tapiaconference.org/

[2] 2019. Ansible is a simple IT automation tool. (2019). Retrieved Sep 24, 2019 from https://www.ansible.com/

[3] 2019. CentOS Project. (2019). Retrieved Sep 24, 2019 from https://www.centos.org

[4] 2019. Cluster System, Cluster and Networking Summer Institute (CSCNSI). (2019). Retrieved Jul 30, 2019 from https://clustercomputing.lanl.gov

[5] 2019. CSCNSI: Past projects. (2019). https://www.lanl.gov/projects/national-security-education-center/information-science-technology/summer-schools/cscnsi/student-projects.php

[6] 2019. Grace Hopper Celebration. (2019). Retrieved Sep 23, 2019 from https://ghc.anitab.org

[7] 2019. GROMACS. (2019). http://gromacs.org

[8] 2019. International Conference for High Performance Computing, Networking, Storage, and Analysis. (2019). Retrieved Sep 23, 2019 from http://supercomputing.org/

[9] 2019. LANL Supercomputing Institute Curriculum. (2019). Retrieved Sep 24, 2019 from https://github.com/hpc/cluster-school

[10] 2019. Los Alamos National Laboratory, HPC Division. (2019). Retrieved Jul 30, 2019 from https://hpc.lanl.gov

[11] 2019. OpenHPC. (2019). Retrieved Sep 24, 2019 from https://openhpc.community

[12] 2019. Parallel Computing Summer Research Internship. (2019). https://www.lanl.gov/projects/national-security-education-center/information-science-technology/summer-schools/parallelcomputing/index.php

[13] 2019. tmux project. (2019). Retrieved Sep 24, 2019 from https://github.com/tmux/tmux

[14] 2019. Warewulf cluster provisioning. (2019). Retrieved Sep 24, 2019 from https://github.com/warewulf/warewulf3

[15] Prentice Bisbal. 2019. Training Computational Scientists to Build and Package Open-Source Software. *Journal of Computational Science Education* 10, 1 (2019), 74–80.

[16] R. van Drunen H.J.C Berendsen, D. van der Spoel. 1995. GROMACS: A message-passing parallel molecular dynamics implementation. *Computer Physics Communications* 91, 1-3 (September 1995), 43–56.

[17] David Brayford et al. Karl W. Schulz, C. Reese Baird. 2016. Cluster Computing with OpenHPC. *SC16: HPCSYSPROS Workshop* (2016).

[18] Kai Himstedt Nathanael Hübbe Sandra Schröer Michael Kuhn Matthias Riebisch Stephan Olbrich Thomas Ludwig Jean-Thomas Acquaviva Anja Gerbes Lev Lafayette Weronika Filinger, Julian Kunkel and Hinnerk Stüben. 2019. Towards an HPC Certification Program. *Journal of Computational Science Education* 10, 1 (2019), 88–89.

# Creating a Relevant, Application-Based Curriculum for High Performance Computing in High School

Vincent C. Betro, Ph.D.
Baylor School
171 Baylor School Road
Chattanooga, TN 37405
+1-423-313-7884
Vincent Betro
vincent.charles.betro@gmail.com

Mary E. Loveless, Ph.D.
Baylor School
171 Baylor School Road
Chattanooga, TN 37405
+1-615-260-2530
mloveless@baylorschool.org

## ABSTRACT

While strides have been made to improve science and math readiness at a college-preparatory level, some key fundamentals have been left unaddressed that can cause students to turn away from the STEM disciplines before they find their niche [10], [11], [12], [13]. Introducing collegiate level research and project-based, group-centered learning at a high school level has a multi-faceted effect; in addition to elevated learning outcomes in science and math, students exhibit improved critical thinking and communication skills, leading to improved preparedness for subsequent academic endeavors [1]. The work presented here outlines the development of a STEM ecosystem where both the science department and math department have implemented an interdisciplinary approach to introduce a spectrum of laboratory and computing research skills. This takes the form of both "in situ," micro-curricular elements and stand-alone research and computer science classes which integrate the language-independent concepts of abstraction and object-oriented programming, distributed and high-performance computing, and high and low-level language control applications. This pipeline has been an effective tool that has allowed several driven and interested students to participated in collegiate-level and joint-collegiate projects involving virtual reality, robotics and systems controls, and modeling. The willingness of the departments to cross-pollinate, hire faculty well-versed in research, and support students and faculty with the proper resources are critical factors in readying the next generation of computing leaders.

## Keywords

STEM; Education; Virtual Reality; Project-Based Learning; High School Research

## 1. INTRODUCTION

Even a cursory exploration of current educational research literature indicates that students' understanding of science, technology, engineering, and mathematics (STEM) topics is increasingly important to the well-being of our global economy [14], [15]. As noted in a U.S. Department of Labor study, "Long-term strategies to maintain and increase living standards and promote opportunity will require coordinated efforts among public, private, and not-for-profit entities to promote innovation and to prepare an adequate supply of qualified workers for employment in STEM fields. American pre-eminence in STEM will not be secured or extended without concerted effort and investment" [6].

The demand for scientists and engineers is expected to continue to increase at a significant rate, especially in computing-related fields. However, data from international studies, such as TIMSS [16], and national studies, such as the 2007 and 2009 National Assessment of Educational Progress Report [17], [18] indicate that both mathematics and science continue to be academic stumbling blocks for many students, that students are consistently not performing well in courses in these disciplines, and that, as a consequence, too few are pursuing degrees in technical fields.

Additionally, according to Ronald Barr of the American Society for Engineering Education, "When a national sample of adults was asked what kind of career they would recommend to young women, medicine was the top choice. A scant 3 percent suggested engineering" [13]. This perception of the stature and importance of qualified engineers and computer scientists must be changed, and the way we inform under-represented groups about career preparation must include heavy doses of STEM content. Also, as posited on a study from the University of Chicago, as early as elementary school, "teachers who are anxious about their own math abilities are translating some of that to their kids," and this has been found to be particularly true in reference to both female teachers and students, likely based on outdated social norms and lack of role models [19].

The low enrollment and engagement in STEM fields has become serious enough that the State of Tennessee has incorporated engineering standards into the integrated science and math curriculum. The state's goal is to expose students to the engineering design process and computational thinking strategies early with hopes of increasing the number of students that will become interested in pursuing a degree and career in a STEM field.

However, "developing a curriculum does not guarantee that engineering education in K–12 will be successful. A critical factor is whether teachers—from elementary generalists to middle school and high school specialists—understand basic engineering concepts and are comfortable engaging in, and teaching, engineering design. For this, teachers must either have appropriate background in mathematics, science, and technology, or they must collaborate with teachers who have this background" [5]. This statement from the National Academy of Engineering's study Engineering in K-12 Education is precisely why having both


https://doi.org/10.22369/issn.2153-4136/11/1/13

interdisciplinary collaboration and expert teachers is so essential to the success of a school's STEM ecosystem.

Even more specifically, high performance computing has become an indispensable technology in myriad fields of research. It has been noted by the National Science Foundation, as well as the U.S. House of Representatives, that computational simulation engineering (and thus engendering an understanding and interest amongst young people) is crucial to enhancing our national security and global competitiveness. The U.S. House of Representatives passed a resolution "recognizing modeling and simulation as a National Critical Technology important to the security and prosperity of the U.S." [7]. Moreover, in 2006, an NSF Blue Ribbon Panel noted that "computer simulation has become indispensable to the development of all other technologies" and "promises to revolutionize the way engineering and science are conducted in the twenty-first century" [8].

High performance computing (HPC) is an excellent vehicle to steer students back into the STEM pathway. Between the applications to such a broad spectrum of real-world problems and the creativity that is implicit in computer science, it draws students to want to better their math and science backgrounds and gives them a creative outlet at the same time. Moreover, by giving students access to faculty, resources, and time to explore computing and engineering, we are giving them tools to solve problems which have not yet been solved. This is quite different from the content and approach espoused in most classes in K-12 education, wherein students are simply trying to figure out what the question is asking so they can also solve a solved, often artificial, problem. It is this aspect of research, the unknown, that allows for both creativity and the understanding of how to approach a truly unknown problem and interpret results with sophistication beyond making numbers simply match the back of the textbook. This is what most excites students because it feels, and is, authentic; it is also the skill set that students most need to take to the university to succeed.

## 2. BUILDING THE ECOSYSTEM

Baylor School has developed a multi-grade, multi-entry point ecosystem to foster interest in STEM-related fields, particularly computer science-related topics and applications; the relevant HPC academic elements are represented in Figure 1. In order to accommodate as many students, as well as levels of interest, as possible, both curricular and extracurricular components have been established, bridging computer science, mathematics and science departments. Within the curriculum, two semester courses, Engineering Design and Independent Research, were introduced. Engineering Design teaches engineering-centric skills (programming, electromechanical systems, and modeling, for examples) in a project-based learning environment that utilizes a flipped classroom design. Both Introduction to Computer Science (Intro CS) and AP Computer Science Principles (AP CSP) were introduced and curriculum was based on the University of California at Berkley curriculum, The Beauty and Joy of Computing. Streamlined coding practices (same language, Snap! and NetsBlox [22, 23], and complementary curriculum pieces) were incorporated into Engineering Design.

Students who complete Engineering Design (fall semester) are eligible to take Independent Research I (spring semester); students who continue to meet grade and commitment requirements can then continue on to take Advanced Research: Engineering (1 year) and subsequently Thesis: Engineering (1 year), depending on their entry point. Projects specifically offered to foster opportunity in high performance computing include embedded systems and virtual reality (VR) application development.

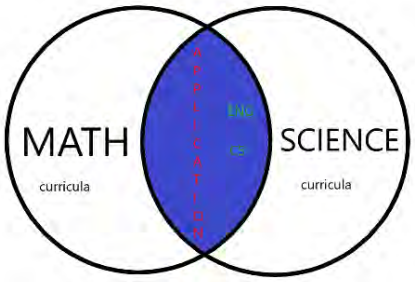



**Figure 1. Interaction between math and science departments is critical to fostering the interdisciplinary environment where students can work on real applications and become inspired to use HPC and other STEM tools.**

Extracurricular programming was introduced as well. Competitive afterschool Robotics teams were initiated in both the middle (MS) and upper school (US). The MS team competes in the First Lego League (FLL) while the US team competes in the First Tech Challenge (FTC). Core STEM content includes: design/design-thinking, modeling, mechanics, logic/code, physics, and prototyping/3D printing.

Another extracurricular offering is Creative Design, which allows students to participate in a range of exploratory and challenge-based STEM activities/competitions; each year, the extracurricular activity changes to meet the needs of the students' area of interest. For example, students have participated in NASA spin-off design competitions, VR projects with collegiate partnerships, as well as motion-tracking for the Baylor Dance Club using an XBOX 360 Kinect.

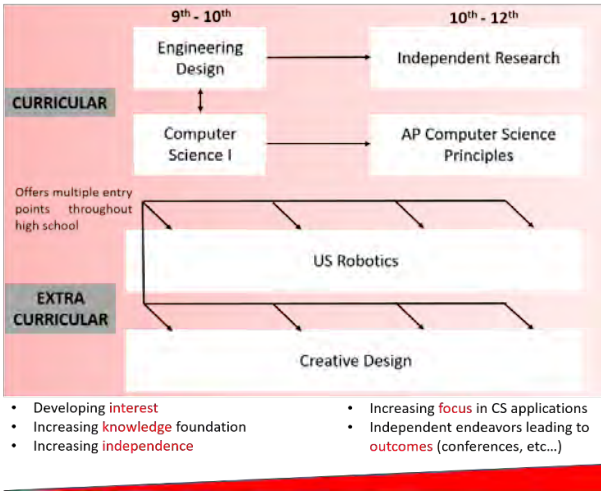



**Figure 2. Relevant HPC academic components. Both curricular and extracurricular option are available to students. Starting in 9th-10th grade, students can participate in *Engineering Design* and *Computer Science I*, which can be taken stand-alone or with the possible progression to *Independent Research* and *AP Computer Science Principles* courses. After school opportunities allow for entry at multiple points during the student's tenure. Progressing further through the pipeline, students deepen knowledge, interest, and independence in computer science-related project, with the eventual outcome of presentations and/or publications at collegiate-level conferences.**

# 3. PROJECT DESCRIPTIONS AND IMPACT

## 3.1 Within Curriculum

### *3.1.1 Introduction to Computer Science / AP Computer Science Principles*

In order to allow the most students access to computer science in their packed schedules, the decision was made to set up the introductory course and the AP course as semester offerings. This way, a student who already has some programming experience and who is willing to work through the first three units of the BJC course independently can join the AP course in the spring semester if that is the only space they have open. Otherwise, they can dive deeper into programming in the introductory course first and have more tools at their disposal when they take AP CSP. Despite the compressed schedule, our AP scores are consistently above the national and state averages.

The decision was made to offer *AP CSP* instead of *AP Computer Science A* since the language agnosticism of CSP allows more students to produce projects they are interested in. Also, it leaves room for discussing HPC and applications in more depth, since it is not primarily focused on *Java* syntax.

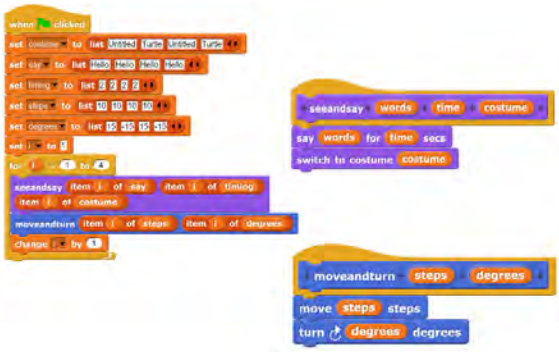



**Figure 3. The use of linked lists and standard lists, along with the use of "blocks" or functions, is exhibited here in Snap! Functions may be overloaded, as in C++, and if any Sprites are created as a copy of this Sprite, they will inherit all class functions.**

Also, students are introduced to programming using Snap! (University of California, Berkley, CA) in order to avoid the pitfalls of compilation and syntax errors as they are honing their logic and algorithmic skills. Snap! is preferred not only because it is visual but also because unlike Scratch! (Massachusetts Institute of Technology, Cambridge, MA) it is possible to create first class objects and utilize inheritance, making function creation extremely powerful and simple.

The natural direction to take students once they have mastered the basics of programming a serial application is to look at scalability and synchronization issues by having them create a distributed application using *NetsBlox* (Vanderbilt University, Nashville, TN) [23-24]. Students initially struggle with communication because they have never had to think about that level of granularity when it comes to things like order of execution and latency. Once they grasp the concept, they are much better computational thinkers, which is the overall goal of the curriculum. After all, given the numerous programming language a student may encounter over his/her career, being able to design based on the strengths of modern computing architectures and the principles of object-oriented development will serve them more appropriately.


**Figure 4. Each "role" in a *Netsblox* project may have its own code or it may simply mirror each other "role". Messages are passed within the "room" or to specific "roles", and instead of addressing to a specific "role", each role is set to only receive its type of envelope, with the contents passed in as the actual message that is sent.**

Once students have been exposed to these fundamental concepts, students can begin to explore and work on projects that interest them. For example, students have used XSEDE [24] resources to compute pi to millions of digits with OpenMP and C++; they have worked to develop MPI code that runs on our 8-node Raspberry Pi cluster and beyond; and they have even developed systems for various applications on the Arduino Uno. Currently, a student is working with the Microsoft Kinect 360 and Visual Studio to create an interactive backdrop for *Verve*, Baylor's dance troop.

Finally, it should be noted that this approach has allowed us to grow the numbers of students in our CS program every year, especially once we introduced Intro to CS in our second year, as can be seen in Table 1.

**Table 1. Enrollment in Intro and AP CSP over three years.**

| Year | Course | Enrollment |
|---|---|---|
| 2017-18 | Intro CS | -- |
| 2017-18 | AP CSP | 9 |
| 2018-19 | Intro CS | 15 |
| 2018-19 | AP CSP | 20 |
| 2019-20 | Intro CS | 18 |
| 2019-20* | AP CSP | 22 |
| *projected | | |

### *3.1.2 Engineering Design Coding Challenge and Introduction to Virtual Reality*

Engineering Design offers project-based curriculum components to convey engineering-centric skills. One of the most intrinsically important skills in engineering is programming. As such, a common programming language and technique were deliberately discussed and agreed upon across all three courses: *Introduction to Computer Science*, *AP Computer Science Principles* and *Engineering Design*. As mentioned previously, *Snap!* was chosen as a versatile, visual programming language that can run on any web browser (many of our students use iPads) such that students could focus understanding fundamental computational principles rather than the potential distraction of syntactic nuances of various languages such as *Java* or *C#*. To elevate curriculum elements and introduce an important yet complicated topic –

networking – students are introduced to *NetsBlox* to introduce concepts related to distributed programming.

Because *Intro CS* is not necessarily a pre-requisite for *Engineering Design*, students are offered an introduction to *Snap!* and *NetsBlox via* a video tutorial series as homework and a series of challenges increasing in complexity to be completed in class (flipped classroom paradigm). After an appropriate introduction to the environments, a challenge is presented to the students involving the gamification of a science/math concept that the students have encountered in the Baylor School curriculum. Using the Engineering Design Cycle as a guide for development, student partners work through studying previous approaches (*e.g.*, games readily available that are educational in context), identifying design requirements, and deciding on an appropriate game design to carry on to production (digitize using *NetsBlox*). The students are then given approximately two weeks to work in a collaborative *NetsBlox* environment (meaning both students can edit the same game space); this project culminates with a paper and presentation, including testing/feedback sessions with classmates. Games that were developed through this curriculum piece included complex control structures, definition of variables, and network elements such as message passing across a network and, in some cases, remote procedure calls (RPCs). Critical thinking / computational thinking skills, creativity, and presentation skills (text and oral) were some intangible skills exercised by the students.

In order to remain relevant, the *Engineering Design* curriculum changes depending on time-sensitive opportunities. For example, one team (four students, seniors and juniors) had the opportunity to work with a VR start-up company, founded at the Massachusetts Institute of Technology. The company's focus was VR experience for the elderly, particularly investigating the use of the immersive technology to combat neurodegenerative disease. The students were asked to design and created a VR environment using *Unity 3D* development software (Unity Technologies, San Francisco, CA) and C# scripting for this company, which they subsequently presented at MIT during the following semester.

### *3.1.3 Independent Research: High Performance Computing Applications*

Motivated students that finish *Engineering Design* with proficiency are eligible to take *Independent Research I*. This course allows students to delve deeper into a relevant engineering-centric research project, such as modeling and virtual reality application development. One student, continuing from the previous MIT collaboration in Engineering Design, took the application of VR in a very insightful direction. The student designed and developed a VR calm room application for children with autism. One challenge from which children on the autism suffer is sensory overload episodes; many places, such as amusement parks and stores have begun to offer physical spaces with dim lights, soothing music, and lowered sensory environments to offer families more flexibility on outings. While this is a step in the right direction, this Baylor student took it a step further, removing the limitations of needing a physical space by offering a virtual solution. The student designed and created a virtual calm room, mimicking those used in private therapy offices as well as those found in progressive places of business. The goal was to create something that could be carried with the family that offered a solution anytime, anywhere. The phone application (paired with something as simple as Google Cardboard / VR Cardboard viewers ($5-$15)) offers a flexible, inexpensive solution for families with autistic children. Professionals in occupational therapy have lauded this development. The student was also accepted to present the device at The American Occupational Therapy Association National Conference in Salt Lake City, UTC in 2018 [26]. While the application development was a step toward using advanced computing techniques for rendering and scaling, the goal of designing a VR experience on a mobile device also offered conversations and successful (and plenty of failed!) attempts at optimization; rendering optimization and reduced scene projection became a very advanced curriculum piece that was subsequently incorporated in off-shoot VR applications discussed below.

Following up on this work, another student expanded the application of the VR calm room to offer passive and functional VR de-escalation experiences for children with emotional troubles, such as those who may suffer from early childhood trauma. The VR application offers a way for teachers, professionals, and families to offer a safe virtual space for the child to be removed from triggering situations (passive VR calming space) as well as to learn techniques and exercises within the VR experience to help manage emotional outbursts, particularly anger and frustration (functional). This work was recently presented as a poster entitles "*A Virtual Reality Approach to Pediatric Conflict De-escalation and Anger Management*" at Practice and Experience in Advanced Research Computing (PEARC 19) in Chicago, IL in 2019 [21]. Currently, collaborations with the University of Tennessee at Chattanooga Occupational Therapy program are underway to test the efficacy of such experience in a real patient population.

Another relevant application of VR has been the use of immersive technology in education. One student in *Independent Research* developed the backbone for a virtual "hands-on" learning tool that works toward enabling a virtual dissection (and subsequent rebuilding) of a variety of models, ranging from components of human cell to the devices in a computer or car engine. In many classrooms, time, space, and personnel can limit the amount of hands-on learning as well as what type of hands-on learning can occur within a classroom. As educators are aware, hands-on learning has shown to be critical in the comprehension process [20]. The student developed multiple class structures, scripted in *C#* and implemented in *Unity* that established generic component and connections between game object (Figures 5 and 6); further, the student functionalized control of these objects using the *Unity Touch* system.

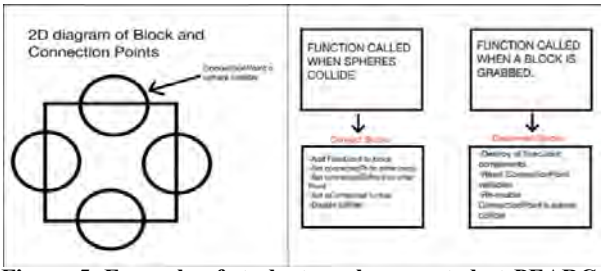



**Figure 5. Example of student work presented at PEARC. The student is defining generic gameObject handling for hands-on learning in in a VR environment. On the left a 2D depiction of block and connection points is represented while the right box is the description on connect and disconnect functions [22].**

With further work, game objects could be interchanged for relevant models and connection definitions (*i.e.*, this piece connects in this way to this other piece) to create virtual hands-on experiences that could be created with by educators with very little training necessary. This student's work was also *presented as a poster at* PEARC 19*;* the work was entitled **"***Virtual Reality Based Environment for Immersive, Hands-On Learning"* [22].


**Figure 6. Example of the student work presented at PEARC. The Point class definition for VR hands-on learning and the Connect function definition are above. Code is written in C# [22].**

## 3.2 Extracurricular Programming

Four years ago, in collaboration with a digital design teacher in the art department, a STEM after school offering was begun called *Creative Design*. The multi-disciplinary atmosphere brought aboard many students who had been looking for both a creative and technical outlet by which to join the new STEM ecosystem, not simply the latter.

At the end of the 2015-16 school year, we had our first STEM Symposium for the school and broader community. It was so successful in both recruiting students to the school/program and giving current students well-deserved exposure for their hard work that each year it has grown in attendance and has even begun to offer juried cash prizes for the top projects. This type of activity allowed us to increase our exposure to campus and begin to get students and teachers recommending these afterschool options to their advisees, which in turn led to them recommending students take more STEM electives, like *Intro CS*/ *AP CSP* and *Engineering Design*. This, in turn, allowed us to approach administration about expanding the teachers in our ecosystem and the courses that we offer in the school-day curriculum. As these courses have borne fruit (*e.g.*, student presentations, internships, AP credit, etc.), more students have become interested and begun to enroll, causing us to add more sections and serve more students' needs.

Another example of success from the *Creative Design* program would be the four students who participated in the NASA OPSPARC design challenge in 2017. Their design for a microshutter array fiber optic switch won the competition for best high school design in the country. Students were flown to NASA Goddard in Greenbelt, MD by the organizers in order to present their technology transfer idea to NASA engineers. These students became even more motivated to continue in computing, took more elective STEM courses than they had planned, and one even went on to create a poster for the PEARC 19 on yet another computer science topic: virtual reality [21]. Not only has he explored this space, but he is developing easy to understand tools for teachers to use to create their own VR units. Also, all four students who were on the winning OPSPARC team have gone to University of Alabama at Huntsville to major in aerospace and/or computer engineering.

These afterschool opportunities continue to offer another entry point for students to enter high performance computing applications, even when class schedules may not permit them to take specialized courses. *Creative Design* has also offered VR opportunities, which lend themselves to students creating and functionalizing 3D models as well as learning C#. Students enrolled in afterschool *Creative Design* had the option to develop an immersive curriculum complement for a science/math teacher at Baylor School. Teachers were asked if they had topics which might be better comprehended if complemented by an immersive VR experience; these educators emailed a list of topics (ranging from the solar system to evolutionary/developmental biology topics) and indicated their willingness to participate. Students were then asked to interview a teacher (who had previously indicated interest) to understand the design requirements outlined. Groups of students developed these modules. Two student groups produced experiences that gained the attention of the Tennessee Technological University (Cookeville, TN); these groups were invited to present and demo these VR experiences at the TTU *iCube* facility.

# 4. LESSONS LEARNED AND RECOMMENDATIONS

The foundational courses of computer science (*Intro CS* and *AP CSP*) as well as relevant application-based courses (*Engineering Design* and *Independent Research*) and extracurricular activities (*Creative Design* and *STEM*) offer a continuum of increasing knowledge and relevant application of high-performance computing techniques and skills. The program at Baylor School offers multiple points for students at any level to enter and progress with their own pace and tailored interest.

Early in the development of these academic elements, it was discovered that allowing students to pursue "anything related to STEM they wanted" was ineffective, as students often do not understand scope control as they have been fooled by the seamless nature of modern technology to think that their idea will require relatively little effort to implement. While the goal was for students to work harder or be more motivated if they controlled the selection of the project or topic, this open-ended project methodology did not manifest as intended. In some cases the students were too overwhelmed to even start a project; in other cases, they projects were so many and diverse that it was logistically unfeasible to mentor effectively. This resulted in students stagnating in projects, getting frustrated, and allowing distraction to take away potentially productive time if the mentor was working with another group. Thus, it is advantageous to both students and mentors to define constraints regarding applications. An example of this would be as follows:

> **VAGUE** Problem Definition: Explore Unity and come up with your own VR experience.
> **RECOMMENDED** Problem Definition: Using Unity, create VR curriculum complement pieces for a science class of your choice at Baylor School. Working with a science teacher, select a science topic in which your immersive experience will aid comprehension for the students in the class.

The vagueness of the first problem definition might paralyze students from even knowing where to start. They may try to do things outside of the scope of a beginning project and get frustrated by a steep learning curve. Additionally, every student will have a completely distinct experience in mind, stretching the mentor's time and breadth of knowledge. The specificity of the RECOMMENDED problem statement allows the students

autonomy (selecting the science course and/or topic) while having boundaries that allow them to focus on distinct aspects of development that the teacher has deemed approachable.

In classes such as *Intro CS* or *Engineering Design*, partner work can be very helpful. Collaboration and communication techniques are fostered in this model and confidence in the topic is developed. Partners should be changed regularly to allow students the opportunity to take on a different role. As motivated students delve deeper (*AP CSP* and *Independent Research*), they should progress toward independent contributions. With the growth of a program of this nature, managing individual projects can be time-consuming but, given proper foundational elements and class opportunities, motivated students can be trusted to take a more active role in problem solving at a more advanced level.

Offering both curricular and extracurricular opportunities not only allow for multiple entry points for busy students, it also allows for a range of risk/reward activities. As illustrated above, commitment levels and performance levels vary throughout the offerings, lending to student selection of projects based on their learning pace and abilities in areas of strength. As with any educational program, there needs to be room to fail and develop resiliency. Offering projects for the school, collaborations with external partners, and opportunities to contribute to the large scientific community helps a student develop confidence in lower risk options while highlighting the power of perseverance with high-reward opportunities.

## 5. FUTURE DIRECTIONS

The *AP Computer Science Principles* assessment will change after the 2019-2020 school year, requiring that more foundational knowledge be tested on the written exam. This will change the structure of the course somewhat, but it will make for a deeper dive into the hardware, as the "computing innovation" research project will be scrapped allowing for more time to be spent on how the computer works. This is an area of HPC that hasn't been explored outside of specific projects where the students worked hands on with an at times temperamental Raspberry Pi cluster.

Additionally, as we add more students to the pipeline, the projects will become more varied and there will be even more carry over allowing us to expand project scopes year after year, given that we set up a good "mentoring" program where students transfer their knowledge of the project and foundational issues so that each project does not die unfinished when a student graduates.

And finally, as Baylor School has a middle school ($6^{th}$-$8^{th}$ grades), creating components (in addition to the already offered *Robotics* extracurricular program), that help streamline computational thinking and programming practices/applications will allow students to achieve a high knowledge base and confidence level more rapidly. This extension would continue to allow an expanded project scope as well as the discussion of more high-caliber electives in the upper school.

## 6. ACKNOWLEDGMENTS

We would like to acknowledge the support of Baylor School and the Baylor Board of Trustees for allowing us the resources and the latitude to form the STEM ecosystem we now have. We would also like to thank the many faculty to have participated in these STEM-related endeavors, specifically Heath Montgomery. We thank the Weeks family, through the Harrison and Katherine Weeks endowment, which has been critical to the growth of programs presented here. Also, this work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562.

## 7. REFERENCES

[1] Angela Eeds, Chris Vanags, Jonathan Creamer, Mary Loveless, Amanda Dixon, Harvey Sperling, Glenn McCombs, Doug Robinson, and Virginia L. Shepherd (2014). The School for Science and Math at Vanderbilt: An Innovative Research-Based Program for High School Students. CBE—Life Sciences Education 2014 13:2, 297-310

[2] Bevan, Bronwyn & Gutwill, Joshua & Petrich, Mike & Wilkinson, Karen. (2015). Learning Through STEM-Rich Tinkering: Findings From a Jointly Negotiated Research Project Taken Up in Practice. Science Education. 99. 10.1002/sce.21151.

[3] Blikstein, Paulo & Krannich, Dennis. (2013). The makers' movement and FabLabs in education: experiences, technologies, and research. ACM International Conference Proceeding Series. 613-616. 10.1145/2485760.2485884.

[4] National Research Council. 2003. Evaluating and Improving Undergraduate Teaching in Science, Technology, Engineering, and Mathematics. Washington, DC: The National Academies Press. https://doi.org/10.17226/10024.

[5] Katehi, Linda, Greg Pearson, and Michael Feder (2009). Engineering in K-12 Education: Understanding the Status and Improving the Prospects. The National Academies Press, Washington, D.C. 2009.

[6] Jobs for the Future (2007). The STEM Workforce Challenge: the Role of the Public Workforce System in a National Solution for a Competitive Science, Technology, Engineering, and Mathematics (STEM) Workforce. U.S. Department of Labor, Employment and Training Administration. April 2007.

[7] House of Representatives (110th Congress, 2007). Recognizing the contribution of modeling and simulation technology to the security and prosperity of the United States, and recognizing modeling and simulation as a National Critical Technology. House Resolution 487. Passed July 16, 2007.

[8] National Science Foundation (2006). Simulation-Based Engineering Science – Revolutionizing Engineering Science through Simulation. Blue Ribbon Panel on Simulation-Based Engineering Science. 2006.

[9] Information Technology Association of America (2005). Innovation and a Competitive U.S. Economy: The Case for Doubling the Number of STEM Graduates. Washington: ITAA. 2005.

[10] National Science Foundation (2002). Characteristics of Scientists and Engineers in the United States: 1999. Division of Science Resources Statistics. Arlington, VA (SRS 03-407). November 2002. http://www.nsf.gov/statistics/us-workforce /1999/tables/TableB2.pdf . February 8, 2010.

[11] Higher Education Research Institute (2007). Survey of the American Freshman, special tabulations. University of California at Los Angeles. Los Angeles, CA, 2007. http://www.nsf.gov/statistics/wmpd/pdf/tabb-8.pdf . February 8, 2010.

[12] American Association of State Colleges and Universities (2005). Strengthening the Science and Mathematics Pipeline for a Better America. Policy Matters. Volume 2, Number 11. November/December 2005.

[13] Barr, Ronald (2005). U.S. Needs More Engineering Students. Miami Herald. August 11, 2005.

[14] Committee for Prospering in the Global Economy (2007). April 8, 2011.

[15] National Science Board (2006). Science and Engineering Indicators 2006, Volume 1. Washington, D.C.: National Academies Press. 2006.

[16] Michigan State University (2001). Third International Mathematics and Science Study. National Center for Education Statistics. April 4, 2001.

[17] National Center for Education Statistics (2007). National Assessment of Educational Progress Report. 2007.

[18] National Center for Education Statistics (2009). National Assessment of Educational Progress Report. 2009.

[19] Kaplan, Karen (2010). Female teachers may pass on math anxiety to girls, study finds. Los Angeles Times. Jan 26.

[20] Samantha Cleaver. 2018. Hands-On Is Minds-On. Retrieved April 9, 2019 from http://www.scholastic.com/browse/article.jsp?id=3751901.

[21] A. Mook and M. Loveless. Virtual Reality Based Environment for Immersive, Hands-On Learning. Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (Learning) (2019). ACM, New York, NY, USA.

[22] J. Liu and M. Loveless. A Virtual Reality Approach to Pediatric Conflict De-escalation and Anger Management. Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (Learning) (2019). ACM, New York, NY, USA.

[23] B Broll, A Lédeczi, P Volgyesi, J Sallai, M Maroti, A Carrillo. A visual programming environment for learning distributed programming (2017). Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science.

[24] B Broll, Á Lédeczi, H Zare, DN Do, J Sallai, P Völgyesi, M Maróti, L Brown. A visual programming environment for introducing distributed computing to secondary education. Journal of Parallel and Distributed Computing 118, 189-200.

[25] Victor Hazlewood, Scott Lathrop, Dave Lifka, Gregory D. Peterson, Ralph Roskies, J. Ray Scott, Nancy Wilkins-Diehr. XSEDE: Accelerating Scientific Discovery. Computing in Science & Engineering, vol.16, no. 5, pp. 62-74, Sept.-Oct. 2014, doi:10.1109/MCSE.2014.80.

[26] Harwood, H and M. Loveless. Virtual Reality-Based Calm Room for Individuals With Autism Spectrum Disorder (2018). Poster presented at The American Occupational Therapy Association National Conference. April 20, 2018 (Salt Lake City, UT).

# Introducing Novices to Scientific Parallel Computing

Stephen Lien Harrell
Purdue University
sharrell@purdue.edu

Betsy Hillery
Purdue University
eahillery@purdue.edu

Xiao Zhu
Purdue University
zhu472@purdue.edu

## ABSTRACT

HPC and Scientific Computing are integral tools for sustaining the growth of scientific research. Additionally, educating future domain scientists and research-focused IT staff about the use of computation to support research is as important as capital expenditures on new resources. The aim of this paper it to describe the parallel computing portion of Purdue University's HPC seminar series which is used as a tool to introduce students from many non-traditional disciplines to scientific, parallel and high-performance computing.

## KEYWORDS

JOCSE submissions, Undergraduate, Parallel Computing, Training, HPC

## 1 INTRODUCTION

Scientific computing supports a wide range of disciplines to enable new and exciting research topics and to create new opportunities for multidisciplinary collaborations, which are vital for cutting-edge research [13]. High performance computing (HPC) permits exploration of complex phenomena that cannot be observed or replicated by experiment. Recently, data-intensive science has emerged as, considered by many, the fourth paradigm of scientific discoveries [8]. Universities, research organizations, businesses and government entities are working to create the best possible environment for research and innovation to ensure the long-term success of computational scientific research. Educating future domain scientists and research-focused IT staff about the use of computation to support research is as important as the supercomputers themselves. A recent report by the NSF Cyberlearning Workforce Development (CLWD) Task Force states that "computational science must be introduced into the K-20 curriculum in ways that build deep understanding and stimulate further exploration. At the undergraduate level, interdisciplinary computational approaches have essential roles both as separate content areas and incorporated into existing math and science (including social and behavioral sciences) curriculum. These interdisciplinary computational approaches, including computer science, have to be presented as more than just programming" [14]. Similarly, NITRD's High End Computing Interagency Working Group suggests an approach that includes "Development of the next generation workforce in undergraduate and graduate university programs through collaborative curriculum development to establish base skills" [9]. Additionally, the necessary skills needed are varied depending on the specific research topics and typically require many fields of knowledge to be covered[12]. In this paper we will discuss Purdue Research Computing's approach to teaching novices (often in scientific undergraduate programs) and how to use parallel and data-intensive computing through a variety of lectures and exercises. By doing that, we aim to give undergraduate students an opportunity to explore the field of HPC and big data in a non-traditional computer science course setting and build a basic foundation of computational and data skills for their further education and research activities.

### 1.1 Inspiring Undergraduates

As part of Purdue University's Sesquicentennial anniversary campaign, students are asked "What if". What if we could control the brain for better health? What if we return to Pluto? What if the world ran on 100 percent renewable energy? At Purdue, most undergraduates are likely familiar with these lines from this "what if series". However, they may not realize that many researchers will rely on advanced computing and data solutions to enable them to answer these complex questions.

Through computational science, we inspire students to change the world. In this class, we pay special attention to hot topics, such as climate change and artificial intelligence, in order to give students an extra push to spend both extra time with their homework and exercises as well as consider graduate work and/or staff roles within advanced computing technologies.

### 1.2 Prior Work at Purdue

Purdue's Research Computing has had a history of mentoring, training, and educating students in HPC. Although we are not alone in these actions [3][4], our staff have had great success mentoring undergraduate students in Systems-Facing as well as Research-Facing roles [5]. The Student Cluster Competition [7] has also been a useful tool to inspire students to consider HPC as a career. In the classroom we have been developing techniques to explore the breadth of HPC [6] and broaden participation from under-served demographics [10].

## 2 HIGH PERFORMANCE COMPUTING SEMINAR

The primary goal of the class is to have undergraduates recognize that computing is an important creative vehicle for scientific discovery on a myriad subjects, ranging from physics to social studies. Aligning to this goal, we employed an integrated and informational approach in teaching computing for this course. Specifically, we integrated parallel computing instruction with different scientific domains. To do this we adapted a combination of lectures from domain science faculty and created labs where students led the


https://doi.org/10.22369/issn.2153-4136/11/1/14

**Figure 1: Entire Class with Instructors**

discussions on the tools and assignment. As the course designers, we began with these learning outcomes in our mind:

- A good understanding of scientific workflow
- Familiarity of building and using scientific applications
- Basics of parallel computing, such as difference between multi-node parallelism and node level parallelism
- Overview of state-of-art computing architectures (e.g. accelerators)
- Performance characteristics (strong and weak scaling) and their connection with the architecture choices
- Bottlenecks in HPC (e.g. communication and data movement) and strategies to minimize them

## 2.1 Approach

Our approach to this curriculum was twofold, we engaged students with hands-on exercises using a real-world scientific application and regularly lectured on more general parallel computing topics in the class. For their first assignment, we asked the students to follow a typical workflow of a weather forecast experiment and reproduce the results. Specifically, convert numerical weather prediction data from the National Weather Service into a full input grid for WRF, run WRF, and interpret the output results. During the lecture, the students were taught parallel computing concepts.

## 2.2 Broadening Participation

In order to communicate the availability of this newly created class, the instructor team was deliberate when it came to effectively advertising it campus wide and how we would engage the larger undergraduate community on campus. During the initial recruitments, the HPC Seminar leaders made a number of visits to clubs on campus, specifically clubs such as Women in Computer Science, Women in Engineering and the Women in HPC. Along with these recruiting opportunities, direct communication was sent to student peers of the previous all-female Student Cluster Challenge team [10] in an effort to further attract additional participants.

One goal of this experience was to create interest in the class from non-traditional computer science students âĂŞ specifically by taking advantage of our current Research Computing employees teaching across the university. Additionally, the faculty sponsor of the class made direct contact with the Data Mine [10], a large-scale living learning community for undergraduates from all majors, focused on Data Science for All, in an effort to recruit students that are not typically Computer Science students.

## 2.3 Syllabus

*2.3.1 Course Description.* This course introduces undergraduates to advanced topics in High Performance Computing clusters, operating systems, and the cluster batch-operating systems. Topics covered in this course focus on aspects of the design, implementation, and use of high performance computing systems at the system level. No previous experience with operating systems or programming is required.

*2.3.2 Course Objectives.* Students will be able to effectively communicate general High Performance Computing (HPC) concepts and knowledgeable on how scientific applications run on HPC resources. The specific learning objectives for this course are:

(1) Students will effectively communicate how to build and compile scientific applications
(2) Students will effectively understand the basics of the Linux Shell
(3) Students will effectively communicate how scientific related topics relate to high performance computing

## 2.4 Lectures

*2.4.1 Introduction to the Linux Shell.* Review the basic command line interface. Students receive a solid foundation in how to use the terminal and how to get a computer to do useful work. Some materials were adapted from Software Carpentry lessons. [2]

*2.4.2 Compiling and running HPL.* Introduce HPL as a measure of a computer system's floating point computing power, thus providing data for the Top500 list to rank against supercomputers worldwide. Students will understand and practice the usage of a benchmark program on a cutting edge HPC system.

*2.4.3 Installing Scientific Applications.* In this lecture students are familiarized with one of the most difficult tasks in the HPC world, installing new scientific software packages. Students are introduced to a few common used tools for managing the build process of packages, such as Automake and CMake.

*2.4.4 History of Weather and Computing - Guest Speaker.* A historical review of the connection between numerical weather prediction and high performance computing and how the advancement of computing technology changes research in the field. The lecture was given by Associate Professor Mike Baldwin from the Department of Earth, Atmospheric, and Planetary Sciences at Purdue.

*2.4.5 Weather Prediction - Guest Speaker.* Professor Baldwin presented a few weather case studies that require some of the most powerful supercomputers in the world. He also showcased his Purdue football game day weather forecast.

*2.4.6 Cluster Design.* Present the various factors in designing a computing cluster, such as performance, availability, scalability, cost, and the range of applications. Additionally, the impacts that applications can have on those factors.

*2.4.7 Big Data - Guest Speaker.* Introduce the basic ideas of big data to analyze and systematically extract information. Students were giving a tutorial on how to use R to manipulate data sets too large or complex to be dealt with by traditional data-processing tools. The lecture was given by Associate Professor Mark Ward of Purdue's Statistics Department.

*2.4.8 Science Writing and Presenting.* Students are provided an overview on effective scientific communication. Both presentation and scientific writing was covered. Some fundamental tips and techniques for effectively writing and presenting scientific information are given.

*2.4.9 Computational Fluid Dynamics - Guest Speaker.* Purdue Mechanical Engineering Professor Carlo Scalo illustrated the examples of using HPC and numerical analysis in solving problems that involve fluid flows. His primary example of fluid flows was aircraft design.

*2.4.10 Molecular Dynamics - Guest Speaker.* Purdue Material Scientist Alejandro Strachan presented predictive atomistic and molecular simulation to describe materials from first principles and their application to problems of technological importance. His presentation included shape memory and high-energy density materials. Dr. Schachan also demonstrated interactive simulation tools on nanoHUB.org [11], a community-contributed resource for nanotechnology.

*2.4.11 Astrophysics - Guest Speaker.* Research Computing Staff Matthew Route shared experiences in running a variety of current parallel codes in astrophysics. He presented examples on both large-scale simulations and big-data analysis of observational data sets.

*2.4.12 Introduction to Python.* Students learned the fundamentals of the Python programming language, along with some of the programming best practices. A few examples include using Python data types and variables to represent and store data, and using conditionals and loops to control the flow of your programs.

*2.4.13 Introduction to Juypter Hub and R Studio.* An introduction to two popular interactive and flexible computational environments for data analysis and graphics.

*2.4.14 Final Presentation.* Students presented in group about what they learned from this class and their suggestions for improvement. Excerpts from these presentations are available in section 4.1.

## 2.5 Assignments

*2.5.1 Student Biographies.* To get to know the students, we had each student responsible for writing biographies about themselves highlighting their area of study and special interests

*2.5.2 Fundamentals of Compiling Applications.* To warm the students up for the major assignments, each student had to learn how to compile HPL using Spack. Then each student did the same exercise compiling HPL by hand. The idea is to have them understand how to read logs and dependencies of compiling applications.

*2.5.3 Compiling and Running WRF and OpenFoam.* Students compiled and ran WRF and OpenFoam and did visualizations of their findings. These two applications make up the core assignments for the class and are described in detail in section 3.

*2.5.4 Guest Lecture Prompt Responses.* After each guest lecture the students received writing prompts about the science being discussed, how do we apply the lecture to the High Performance Community and how would they apply the science to solving a problem using a cluster.

*2.5.5 Final Team Project.* The students at the of the semester were asked to put together a power point presentation. They needed to answer 5 fundamental questions

- Describe your experience with building and compiling applications. Name the applications that you have built, what they were used for and what struggles you had building them.
- Discuss your understanding of Linux, what your experience was before attending the class and any new things you have learned.
- Describe how scientific related topics relate to high performance computing. Outline at least one presentation that you attended and how it relates to the class.
- Discuss your opinion on the scientist presentations that were held in class. Were they valuable? Why or Why not? Which presentations if any did you like? Why or Why not?
- Discuss your opinion of the pace and difficulty of the class. Be specific and describe the track you involved in.

They then presented their findings to the class and invited guests.

# 3 TEACHING SCIENTIFIC APPLICATIONS

To teach parallel computing principles, we chose WRF, widely used on various HPC systems, as an illustrative scientific application. WRF is related to the weather modeling history, and its background taught by Dr. Baldwin for this class. Additionally, WRF is a ubiquitous scientific code with layman-relatable input and output data. We designed the lecture and hands-on WRF excersizes to achieve the learning outcomes from Section 2. All assignments and hands-on exercises were performed using Purdue's teaching and learning cluster, Scholar [1].

Keeping the students' learning outcomes in mind, the second application chosen was OpenFOAM, a popular open source software for the solution of continuum mechanics problems, most prominently with computational fluid dynamics. The goal of the assignment was to have students further hone their skills learned in the class to solve a practical problem independently. In contrast to the WRF exercise, we decided not to reserve any class time for questions (discussions among students in person or on Slack are encouraged).

### 3.1 WRF

First, students learn the basics of parallel computation and MPI. Secondly, they learn the advantages and disadvantages of different parallel paradigms such as distributed memory parallelism vs. shared memory parallelism. In practice, students also learn the intricacy of properly setting up a parallel computing environment. Finally, students gain a good knowledge of the performance characteristics and how this will be impacted by the choice of hardware architectures. They are also asked to perform scaling studies (strong vs weak). Which illustrates the bottlenecks in parallel computation, such as network and I/O related overhead.

While WRF is a very common application among meterologists, when aimed at undergraduates it has some detriments. First, it can be very hard to build for a novice. For instance, at compile time you must know the differences between node-level and multi-node parallelism and the important frameworks for each. Additionally, some features are are hidden behind compile-time options in a non-obvious way. For instance the choice of NETCDF 3 vs 4 has consequences that may require one to recompile if the data set does not match. This is all but obvious for weather scientists, however, it requires a well-grounded understanding of Linux user-space environments as well as parallelism in standard HPC clusters today. Finally, the workflow for WRF is more complex than some other scientific applications, requiring the use of multiple executables from multiple packages in order to do a full weather simulation, which is well documented in the language of meteorologists, but not undergraduates.

### 3.2 OpenFoam

This assignment was given to the students after the Dr. Scalo's lecture on computational fluid dynamics. By this point the students were more comfortable with software dependencies and how to use tools like Spack, which was covered in a previous tutorial. Also, they had a basic idea of fluid dynamics and a few important terms after the guest lectures. Both would tremendously help them find a viable solution for the assignment.

The individual assignment for the student was to take the software OpenFOAM and manually compile the software or use Spack. Once the students had the application compiled, the next step was to run the simulations and visualize what they had obtained. Each student was provided an initial input file, but once they were comfortable visualizing the assignment, the next step was to change the input file to see the effects. The students used Paraview, which they also needed to familiarize themselves with to visualize the results. In this assignment, students were asked to solve a problem of simplified dam break in two dimensions, where a transient flow of water separated by a sharp interface. Knowing how to see the effect of water breaking over a dam, students were more engaged than in the WRF exercise, discussed in Section 4.1.1. We found that most students were able to successfully present demo simulation to the class.

**Figure 2: Students Presenting about Learning Outcomes**

## 4 OUTCOMES

### 4.1 Student Evaluation of the Class

In the final project, students were given prompts regarding the learning objectives of the class and if they were met, students presented their responses seen in Figure 2. Focusing on three of the prompts, the responses below are illustrative of the general outcomes of the class, the following information was conveyed:

*4.1.1 Describe your experience with building and compiling applications.* Students found that the directions for installing Spack and OpenFoam were generally straightforward and easy to follow, however, in practice it was very difficult to compile and run correctly. Additionally, the logic behind the steps was not explained thoroughly, so it was difficult to troubleshoot errors in the homework assignments. Through the exercises the understanding of Linux increased dramatically and although the students found the method of learning difficult, this learning outcome was met. Many students finished the class knowing how to navigate the shell and how it interacts with the applications.

However, there were also frustrations with this format. Such as the speakers didn't understand the class objective was to learn about HPC and it's different implementations, not just whatever the invited speaker science was. It would have been better if the speakers focused more on how they use computers and HPC to do their research and less on the details of what their research is for.

*4.1.2 Discuss your opinion of the pace and difficulty of the class.* One student said that having to learn the science and parallel computing at the same time was too much. Another said that the class in general was very fast paced.

*4.1.3 Discuss your understanding of Linux, what your experience was before attending the class.* Students stated they believed it was very important to have a base understanding of Linux before attending the class, as there was not enough of "getting to know Linux" done in class.

### 4.2 Lessons Learned

While considerable success was achieved, such as compiling WRF and getting correct simulation results, it was difficult for the students to fully appreciate the entire process due to their lack of meteorological knowledge. Additionally, the pace of the class had to slow down to accommodate the majority of the students, and consequently, instructors had no time for covering the visualization of the simulation results. This resulted in dampening the students' interest in the topic and thus the students became largely dependent on instructors for the finishing assignments for this portion of the class.

A better strategy would have been to have the students visualize an existing WRF output file, allowing the students to become familiar with meteorological visualizations before attempting the somewhat daunting exercise of compiling and running WRF for the first time. Although it does not follow the actual sequence of weather modeling, we propose that this would better keep students' attention and keep them motivated throughout the more esoteric work during the hands-on section. More importantly, such experience highlights the key difference in instructional design between a graduate course and an undergraduate one.

A second key lesson learned was that the Linux skills required for this type of class were a serious impediment to students interest and learning. A majority of the students started with little to no experience in Linux, even with a class period dedicated to command line basics, their understanding was insufficient. This put a lot of burden on the students to learn how to compile an application without knowing how to navigate the environment. In future classes, it is recommended that either Linux knowledge becomes a requirement for the class or a more significant amount time is dedicated to this topic.

## 5 FUTURE WORK

For the next HPC seminar we run we plan to vet prior Linux experience and split the first few classes between experienced Linux users and novices. This will allow novice users to get a more complete understanding of basic Linux skills. The other major change we will implement is switching visualization to be first when teach scientific applications, this approach allows the students to visually see the science in action.

## ACKNOWLEDGMENTS

We'd like to acknowledge Dr. Mark Ward, Dr. Mike Baldwin, Dr. Carlo Scalo, Dr. Alejandro Strachan and Dr. Matt Route for sharing their knowledge with our students. We'd also like to thank Christopher Phillips for his keen editorial skills.

## REFERENCES

[1] M. E. Baldwin, X. Zhu, P. M. Smith, Stephen Lien Harrell, R. Skeel, and A. Maji. 2016. Scholar: A Campus HPC Resource to Enable Computational Literacy. In *2016 Workshop on Education for High-Performance Computing (EduHPC)*. 25–31. https://doi.org/10.1109/EduHPC.2016.009

[2] Software Carpentry. [n. d.]. https://swcarpentry.github.io/shell-novice/

[3] Dirk Colbry. 2014. iCER Interns: Engaging Undergraduates in High Performance Computing. In *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment (XSEDE '14)*. ACM, New York, NY, USA, Article 71, 5 pages. https://doi.org/10.1145/2616498.2616573

[4] Andrew Fitz Gibbon, David A Joiner, Henry Neeman, Charles Peck, and Skylar Thompson. 2010. Teaching high performance computing to undergraduate faculty and undergraduate students. In *Proceedings of the 2010 TeraGrid Conference*. 1–7.

[5] Stephen Lien Harrell, Marisa Brazil, Alex Younts, Daniel T. Dietz, Preston Smith, Erik Gough, Xiao Zhu, and Gladys K. Andino. 2018. Mentoring Undergraduates into Cyber-Facilitator Roles. In *Proceedings of the Practice and Experience on Advanced Research Computing (PEARC '18)*. ACM, New York, NY, USA, Article 70, 7 pages. https://doi.org/10.1145/3219104.3219138

[6] Stephen lien Harrell, Benjamin Cotton, Michael Baldwin, and Andrew Howard. 2013. Developing a Scientific Computing Cluster Course for the Undergraduate Curriculum. In *Summit for Educators in System Administration 2013*. Washington D.C. http://funnelfiasco.com/research/sesa13.pdf

[7] Stephen Lien Harrell, Hai Ah Nam, Verónica G. Vergara Larrea, Kurt Keville, and Dan Kamalic. 2015. Student Cluster Competition: A Multi-disciplinary Undergraduate HPC Educational Tool. In *Proceedings of the Workshop on Education for High-Performance Computing (EduHPC '15)*. ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/2831425.2831428

[8] Tony Hey. 2012. The Fourth Paradigm–Data-Intensive Scientific Discovery. In *E-Science and Information Management: Third International Symposium on Information Management in a Changing World, IMCW 2012, Ankara, Turkey, September 19-21, 2012. Proceedings*, Vol. 317. Springer, 1.

[9] High End Computing Interagency Working Group. [n. d.]. *Education and Workforce Development in the High End Computing Community*. Technical Report. NITRD.

[10] Elizabett Hillery, Mark Daniel Ward, Jenna Rickus, Alex Younts, Preston Smith, and Eric Adams. 2019. Undergraduate Data Science and Diversity at Purdue University. In *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (Learning) (PEARC '19)*. Association for Computing Machinery, New York, NY, USA, Article Article 88, 6 pages. https://doi.org/10.1145/3332186.3332202

[11] Gerhard Klimeck, Michael McLennan, Sean P Brophy, George B Adams III, and Mark S Lundstrom. 2008. nanohub. org: Advancing education and research in nanotechnology. *Computing in Science & Engineering* 10, 5 (2008), 17.

[12] S. Lathrop. 2016. A Call to Action to Prepare the High-Performance Computing Workforce. *Computing in Science Engineering* 18, 6 (Nov 2016), 80–83. https://doi.org/10.1109/MCSE.2016.101

[13] National Academy of Sciences, National Academy of Engineering, and Institute of Medicine. 2005. *Facilitating Interdisciplinary Research*. The National Academies Press, Washington, DC. https://doi.org/10.17226/11153

[14] Task Force on Cyberlearning and Workforce Development. 2011. *A Report of the National Science Foundation Advisory Committee for Cyberinfrastructure*. Technical Report. National Science Foundation.

# Evaluating the Effectiveness of an Online Learning Platform in Transitioning Users from a High Performance Computing to a Commercial Cloud Computing Environment

Dhruva Chakravorty
High Performance Research Computing
Texas A&M University
College Station, TX, USA
chakravorty@tamu.edu

Minh Tri Pham
High Performance Research Computing
Texas A&M University
College Station, TX, USA
phamminhtris@tamu.edu

## ABSTRACT

Developments in large scale computing environments have led to design of workflows that rely on containers and analytics platform that are well supported by the commercial cloud. The National Science Foundation also envisions a future in science and engineering that includes commercial cloud service providers (CSPs) such as Amazon Web Services, Azure and Google Cloud. These twin forces have made researchers consider the commercial cloud as an alternative option to current high performance computing (HPC) environments. Training and knowledge on how to migrate workflows, cost control, data management, and system administration remain some of the commonly listed concerns with adoption of cloud computing. In an effort to ameliorate this situation, CSPs have developed online and in-person training platforms to help address this problem. Scalability, ability to impart knowledge, evaluating knowledge gain, and accreditation are the core concepts that have driven this approach. Here, we present a review of our experience using Google's Qwiklabs online platform for remote and in-person training from the perspective of a HPC user. For this study, we completed over 50 online courses, earned five badges and attended a one-day session. We identify the strengths of the approach, identify avenues to refine them, and consider means to further community engagement. We further evaluate the readiness of these resources for a cloud-curious researcher who is familiar with HPC. Finally, we present recommendations on how the large scale computing community can leverage these opportunities to work with CSPs to assist researchers nationally and at their home institutions.


DOI: https://doi.org/10.22369/issn.2153-4136/x/x/x

## CCS CONCEPTS

•**CS**→Computer Science; •**Cybertraining**→training on using cyberinfrastructure; •**HPC**→high performance computing

## Keywords

HPC training, cloud computing, assessment strategies, best practices. diversity

## 1. INTRODUCTION

The growing computing needs of researchers in data science and engineering have led to increasing use of cloud computing resources, coupled with innovative web-based platforms for data analysis. This is observable in National Science Foundation (NSF) investments in projects such as JetStream [1,2], Aristotle [3], and CloudLab [4]. Indeed, in recent years, the research and engineering (R&E) community has adopted commercially available cloud resources and services (CACRS, also known as commercial service providers) in research and education. This is particularly true for the computational biology community, led by the National Institutes of Health (NIH) Science and Technology Research Infrastructure for Discovery, Experimentation, and Sustainability (STRIDES) initiative [5]. The NIH STRIDES initiative explores new models of data stewardship as it provides access to cloud services such as compute, storage and training on Amazon Web Services (AWS) and Google Cloud Platform (GCP). Indeed, STRIDES is on its way to hosting the tens of petabytes of NCBI data set on the cloud. In recent times, the NSF has also taken steps to offer researchers cloud-based resources. The NSF Big Data program [6] partnered with IBM, GCP, Microsoft Azure and AWS to make a number of cloud-based awards as well. Similar partnerships led to awards in the decade-long NSF CC* program in 2019 [7]. In perhaps a sign of the growing role of CACRS in R&E, the NSF launched the NSF Cloud Access program [8] that created a CSP-based cloud-exchange for NSF researchers requiring CACRs for their NSF-funded projects. [NSF Cloud bank] The scientific community has relied on high performance computing (HPC) to meet its large-scale scientific computing needs. Owing to the rapid proliferation of cloud computing in science technology engineering and mathematics (STEM) disciplines, migration to

CACRS requires training on both, how to configure and use these resources.

Online training platforms offer opportunities to scale that are beyond the capabilities of instructor-led, in-person classrooms. To meet the growing demand for trained and certified cloud- and data-science engineers, a number of online training platforms have emerged to fill the void between existing curricula at institutions of higher education and the urgent needs of these sectors. As such, informal online education platforms have taken the lead in preparing aspiring engineers for certifications and careers in these fields. In addition to Coursera [9] and Linux Academy [10], Qwiklabs is an online training platform that offers hands-on, lab-learning environments on using cloud-based services. At the time of this study, Qwiklabs, provided temporary credentials to Google Cloud Platform (GCP) and Amazon Web Services (AWS). A study of recent awards made by the NSF Big Data, CC* and E-CAS programs (in collaboration with Internet2) [11] suggest that AWS and GCP remain the platforms of choice for the large scale computing community. Furthermore, researchers (and faculty) may readily access Qwiklabs via free credits from generous programs offered by GCP. Furthermore, instructors specializing in in-person training events could use Qwiklabs to develop, host and broadcast their hands-on training labs and lecture notes. Taken together, these factors make Qwiklabs a strong candidate for an online platform that can provide existing resources to a cloud-hungry group of researchers. It should be noted that while Qwiklabs started as an independent effort, it was acquired by Google in 2016. [12,13] While GCP-related courses continue to be added, Qwiklabs continues to offer AWS training courses as well. Support for instructor-led training activities continues on this platform. In this paper, we explore the readiness this learning platform in transitioning HPC users to the commercial cloud.

## 2. EXPERIMENTAL DESIGN

At the onset, we acknowledge that proficiency in computing widely differs from student to student. Furthermore, with a wide range of computing constructs available, there is a need to define a standard for introductory, intermediate, and advanced, activities proficiencies in training. Teaching computing practices to HPC users presents additional challenges. Researchers have a diverse set of skills and varying degrees of experience using computing in their research. The Qwiklabs training platform helps address some of these concerns by including scaffolded instruction methods that support learners with varied skill sets. Research find that active learning is more effective than procedural training. [14-25] This approach is adopted across the platform. Since Qwiklabs is owned by Google, here we evaluate the efficacy of its GCP-centric labs in helping HPC system administrators and users adopt to the GCP environment.

### 2.1 The Qwiklabs Approach

Qwiklabs, as the name suggests, is compilation of quick labs that give participants hands-on experience on the "real" cloud. Qwiklabs leverage community expertise to develop these tutorials. These labs are web-based. As such, they are platform agnostic and can be run on any computer! These tutorials/labs themselves have an easy-to-comprehend format. All labs are templated and follow a similar format. While some labs are free, others require credits. A collection of labs on a single topic are combined to create a quest. Depending on the learning levels, quests are identified as Introductory, Fundamental, Advanced and Expert. Community experts typically create these labs, and some labs have an associated tutorial entry on github. By employing community experts to create the labs, the curriculum on Qwiklabs can extend in complexity, while simultaneously supporting analysis across various STEM disciplines. While templating labs allows for a large number of contributors, the labs themselves may be of variable quality. Participants can grade a lab on completing it as a measure of feedback.

Informal effort that use well-reviewed pedagogical approaches to education have been found to encourage participation and adoption of computational thinking. [14-25] We have found that teaching students new computing concepts, such as navigating a Linux environment, using a command line, and writing code, is more productive when done through an interactive format. [26-29] With an interactive format, students are more motivated to follow along with the instructor and other students by participating in the activities. In agreement with existing literature, our data indicate that students with varying degrees of programing are best suited with scaffolded learning approaches like Jupyter notebooks for application specific training. [DKC citations] A problem-solving approach, though slower, encourages greater interactions and deeper learning of the subject matter. Including a scaffolded learning approach helps learners grasp complex concepts better, and helps reduce the barriers to computing enablement. This is particularly relevant for informal efforts supported by HPC centers that support users with a diverse range of research needs and computing prowess. The relatively inexpensive nature of these labs and the quests open possibilities for educators to couple activities with classroom and HPC training. Indeed, coupled with adequate assessment techniques, Qwiklabs can serve as a scalable platform on which students can develop prowess on GCP.

Here, we explore the usability of Qwiklabs, and the pedagogical approaches used to introduce users to cloud computing services and environments on GCP for R&E. In particular, we report on the use of employing scaffolded instruction, tiered instruction techniques and innovative active learning exercises in the context of teaching Cloud Computing via Quests and their associated labs. In this study, we do not expand on the challenges available in Qwiklabs. The programs goals were to assess the usability of these existing labs in (a) increasing participant engagement in cloud computing, (b) teaching relevant knowledge for current HPC users in order to help them migrate their workflows, and (c) providing participants with a learning environment that employed hands-on exercises.

### 2.2 Using Qwiklabs

The lead author on this publication (Chakravorty) has worked with HPC technologies since 2004 and did not use Qwiklabs prior to this exercise. Tri Pham has adopted HPC and cloud computing practices in recent years. The authors have completed over 50 Qwiklabs courses in 2019. These are listed in Table 1. The primary author was first introduced to the platform at the Google Next 2019 conference in San Francisco, where conference attendees were encouraged to explore Qwiklabs, with assistance provided by "Googlers." Since then, the authors have completed various GCP labs online. To understand the tiered training approach, we completed learning labs in quests at the Introductory, Fundamental, Advanced, and Expert levels. In this study we did not participate in Challenge labs. In addition to taking the online labs described in Table 1, the author attended an introductory level in-person Google Cloud Platform Fundamentals class offered by a third-party company on behalf of Google. This session provided an overview of Platform products and services, and demonstrated how to incorporate GCP solutions into business strategies. In addition to providing an introductory overview of the GCP suite of products, the course covered topics related to storage, virtual machines,

containers and applications in the cloud. The in-person section also covered topics in developing, deploying, and monitoring processes in the cloud, as well as the use of machine learning and Big Data technologies in the cloud. In addition, the author also attended a session taught by a Googler that demonstrated how APIs can be integrated with G Suite for Big Data analysis.

**Table 1. Completed courses and quests for GCP along with all labs associated with the quests and learning level are listed. Quests that are in progress are denoted with an asterisk (*). Only completed labs are reported for quests that are yet to be completed. A list of completed labs that are not associated with Quests are provided under the category of "Independent Labs Completed".**

| Quest | Courses |
|---|---|
| GCP Essentials *Level:* Introductory | A Tour of Qwiklabs and the GCP |
| | Creating a Virtual Machine |
| | Compute Engine: Qwik Start - Windows |
| | Getting Started with Cloud Shell & gcloud |
| | Kubernetes Engine: Qwik Start |
| | Set up Network and HTTP Load Balancers |
| Google Cloud Platform Fundamentals: Core Infrastructure *Level: Fundamental* | Getting Started With Cloud Marketplace |
| | Getting Started with Compute Engine |
| | Getting Started with Cloud Storage and Cloud SQL |
| | Getting Started with Kubernetes Engine |
| | Getting Started with App Engine |
| | Getting Started with Deployment Manager and Stackdriver |
| | Getting Started with Big Query |
| Baseline Infrastructure *Level: Introductory* | Cloud Storage: Qwik Start - Console |
| | Cloud Storage: Qwik Start – CLI/SDK |
| | Cloud IAM: Qwik Start |
| | Stackdriver: Qwik Start |
| | Cloud Functions: Qwik Start - Console |
| | Cloud Functions: Qwik Start – Command Line |
| | Google Cloud Pub/Sub: Qwik Start - Console |
| | Google Cloud Pub/Sub: Qwik Start – Command Line |
| | Google Cloud Pub/Sub: Qwik Start - Python |
| Security & Identity Fundamentals *Level: Fundamental* | Cloud IAM: Qwik Start |
| | IAM Custom Roles |
| | Service Accounts and Roles: Fundamentals |
| | Install a Forseti Server on GCP |
| | VPC Network Peering |
| | User Authentication: Identity Aware Proxy |
| | Getting Started with Cloud KMS |
| | Setting up a Private Kubernetes Cluster |
| Google Kubernetes Engine Best Practices *Level: Advanced* | GKE Migrating to Containers |
| | Monitoring with Stackdriver on Kubernetes Engine |
| | Tracing with Stackdriver on Kubernetes Engine |
| | Logging with Stackdriver on Kubernetes Engine |
| | Connect to Cloud SQL from an Application in Kubernetes Engine |
| | Quiz: Kubernetes Engine Best Practices Quiz |
| Baseline: Data, ML and AI *Level: Introductory* | Cloud ML Engine: Qwik Start |
| | Dataprep: Qwik Start |
| | Dataflow: Qwik Start- Templates |
| | Dataflow: Qwik Start - Python |
| | Dataproc: Qwik Start - Console |
| | Dataproc: Qwik Start – Command Line |
| | Cloud Natural Language API: Qwik Start |
| | Google Cloud Speech API: Qwik Start |
| | Video Intelligence: Qwik Start |
| Cloud Architecture *Level: Fundamental* | Orchestrating the Cloud with Kubernetes |
| | Deployment Manager- Full Production |
| Kubernetes in the Google Cloud* *Level:* Advanced | Introduction to Docker |
| | Kubernetes Engine: Qwik Start |
| | Orchestrating the Cloud with Kubernetes |
| | Managing Deployments Using Kubernetes Engine |
| Windows on GCP* *Level: Expert* | Compute Engine: Qwik Start - Windows |
| | Deploy Microsoft SQL Server to Compute Engine |
| | Running Windows Containers on Compute Engine |
| | Stackdriver: Qwik Start |
| G Suite: Integrations* *Level: Advanced* | Introduction to APIs in Google |
| | Creating a Gmail add on |
| Independent Labs Completed *Level: Various* | Hardening Default GKE Cluster Configurations |
| | Google Kubernetes Engine Security: Binary Authorization |
| | Provision Services with GCP Marketplace |
| | Using Role-based access controls in Kubernetes Engine |

## 3. RESULTS

An important consideration while evaluating this learning platform is that the author had self-selected himself to participate in these exercises, and has worked in the field of informal computing

education. Each approach and the associated exercises were appropriate for the author and were found to be equally engaging.

## 3.1 Pedagogical Approach

Complex computing projects can overwhelm the new learner. Qwiklabs facilitates in a tiered format where information is provided, comprehended, analyzed and employed before moving to the next step. As described above, quests may be at the as Introductory, Fundamental, Advanced and Expert levels. The exercises in these approaches build on each other. In a quest, labs typically begin with an introduction to basic concepts, ensuring that learners are familiar with common technologies and understand the language. This is especially important for users who have had little to no experience of using CACRS. These baseline activities also provide a useful review for students who had some experience with CACRS. Each topic in the lab contained a small activity to allow for immediate application of the respective topic. To synthesize knowledge gained in the lesson, learners were tasked with activities and performed small assignments. Each segment scaling up in difficulty and building on previously-learned concepts, as per a scaffolded instruction approach. The activities are designed to engage learners at varied skill levels. For example, in labs that required software installation or deployment of pods, the labs took steps to reduce the complications arising from having to installing software and associated libraries. Labs provide an installation script that automated most of the process. Furthermore, in Data Science labs, learners learn to how to call data sets and leverage existing modules to solve real world problems. At increased levels of complexity, activities that provided lesser scaffolding were harder to grasp than those that employed more scaffolding. The approach included structured components, and lecture notes (or tutorials) that can be accessed by the learner after completing the exercise.

## 3.2 The Learning Environment

Like many other online learning platforms, Qwiklabs quests and labs are timed and self-paced. QL labs can take anywhere from 30 minutes to 2 hours to complete, and a quest may contain between 5 to 7 labs. Rather than requiring a commitment of multiple hours to complete a module, the relatively shorter duration of these labs is a strength as learners are able to work them into their daily schedule with relative ease. For example, in order to maintain a continuous learning process, the authors strived to successfully complete one lab every day.

The labs rapidly introduce learners to the richness of the GCP environment. The menu is expansive and includes all attributes present in the GCP production environment. This includes the current analytics packages, orchestration environment and software stack. Mindful of traditional HPC users, the interface offers both command line and GUI-driven management options that may be used. While this rich environment remains a strength for experienced cloud users, we noted that this could be overwhelming for new users, particularly those who were not guided through the process. Users can select an option on the GUI and see the corresponding command on the shell. The advantage of this approach is that learners who are not familiar with the command line can get to work using the GUI instantly, while those more comfortable with using the command line and inbuilt technologies are accommodated as well. In addition to simultaneously learning how to use the GUI and command line to perform select functions, we found this two-pronged approach to be particularly useful when the instructions in the Qwiklabs tutorial did not map to the options on the GUI.

Cross-listed labs, i.e. labs associated with more than one quest, are used to create a cross-pollinating effect that encourages the learner to start an additional quest. For example, the lab, "Stackdriver: Qwik Start" is cross-listed on the "Stack Driver" and "Windows on GCP" quests. While there are significant gains associated with this approach, it can also cause learners to run into a cascading effect if a cross-listed Qwiklabs course is not well-developed or has issues. Such a situation could negate the learning environment for multiple quests. Though the author did run into less refined labs on the platform, he didn't encounter these issues with cross-listed labs. An additional issue is that Qwiklabs labs may go offline or be placed under maintenance. For example, the cross-listed course entitled "Continuous Delivery with Jenkins in Kubernetes Engine" has stopped progress on the "Cloud Architecture", and the "Kubernetes in the Google Cloud" quests.

## 3.3 Challenges

Technology plays an important role in the success of online learning environments. Glitches in an online training platform are likely to deter new users. The time limits, coupled with the for-credits (dollars) nature of these activities, makes the learner more sensitive to issues on these platforms. While the Qwiklabs platform is highly usable, here we describe some of the issues that we discovered in the process. These issues did not prevent us from completing a lab or exercise. Some of these issues were: (a) on starting a lab, the session would fail to launch; (b) the corresponding shell session would fail to launch for a session; (c) virtual machines took a large amount of time to spin up; (d) limits on resources supporting virtual machines (VMs) were not established in the exercise; (e) the automatic progress checks would not approve a step, while latter steps were successfully completed and authenticated by the system; and (f) labs previously completed would appear as incomplete in the learner's learning profile if restarted. A working solution for problems (a) and (b) was to refresh the web-browser, while for (c) we recommend choosing a smaller VM.

# 4. FUTURE DIRECTIONS

In this section we discuss ways in which Qwiklabs may be integrated into the HPC user-education, and staff professional-development environment.

## 4.1 Supporting a HPC Quest

Qwiklabs supports enterprise IT and web-based services. While these services overlap with the needs of the HPC community in areas such as cybersecurity and resource management, the specifics are different. In terms of research pathways, labs for Big Data analytics and artificial intelligence are perhaps best aligned to the needs of the HPC community. We once again find synergies in hosted data sets, use of shared server/notebooks (example Jupyter Notebooks, Codelab), and containers but the specifics such as the type of containers (docker on GCP, but Singularity in HPC workloads) may change. On our introduction to Qwiklabs, we tried to find an appropriate quest that focused on HPC technologies such as deploying clusters, job schedulers (SLURM and IBM Spectrum LSF), setting up SSH tunnels, how to enable cloud-burst and setting up virtual private networks. While Qwiklabs doesn't host a dedicated quest for HPC workflows, GCP has a collection of tutorials and resources that are available on Github. We direct the interested user to these resources listed in Table 2. Qwiklabs offers a number of training labs for installing and running AI-relevant software such as Tensorflow. This level of support extends to analytics such as Google BigQuery as well. To the best of our

knowledge, this resource-rich training ecosystem does not, however, extend to the use of some of the largest use-cases of HPC software stacks such as multi-physics codes (examples LAMMPS, GROMACS). A Qwiklabs quest that focuses on installing this software in a Singular container, followed by applications and best data- and computer-management practices would be useful for researchers in the HPC community. These modules could be extended to include ways to share data and results among researchers.

**Table 2. Resources for Cloud Bursting [30-35].**

| Resource | Courses |
|---|---|
| Cloud Burst Using Slurm *Workflow tutorial* | Deploy an Auto-Scaling HPC Cluster on GCP with Slurm |
| | Introduction to GCP Python Client for Compute Engine |
| | Setting up VPN tunnel between strongSwan VPN and GCP Cloud VPN |
| Documentation | GCP Compute Engine: Instance Template |
| | Slurm Elastic Computing (Cloud Bursting) |
| | GCP Cloud DNS: Overview |

## 4.2 Developing a Direct Pathway to GCP Certifications

Offering certifications to system administrators fills a key need in the cloud arena. Such certifications allow sites to help with employment and system production decisions. While Qwiklabs provides a quick way to jump onto GCP's platform, the correlation between completing quests and current GCP certifications such as the Google Cloud Associate Engineer is not clear. As discussed previously, a person may complete a lab by merely copying and pasting text from the instruction set to the lab, thus making it hard to assess the value of an individual completing a quest. A defined (or prescriptive) pathway toward achieving this certification would be meaningful. A casual look at sample questions for the Google Cloud Associate Engineer suggests that the certification exam is geared for folks who have months of experience on GCP supplementing the knowledge gained from Qwiklabs exercises. While this approach to training users is likely to work for researchers who are merely trying to find ways to migrate their workflows to GCP, it may create a "chicken or egg" situation for employers keen to adopt GCP. In the absence of lower-level certifications demonstrating an applicant's skill level, employers will be forced to make hiring decisions without knowing *a priori* whether their future employees will be able to be certified.

## 4.2 Surveys and Assessments

Qwiklabs relies on the honor system to report a learner's progress. The hand-holding (concierge service) nature of the learning labs is such that solutions to problems are visible to the users. Student success is generally evaluated based on their ability to complete the session's hands-on activities. In specific cases a student's progress is marked by an in-built checkpoint. While focusing on ease of use, the platform allows learners to copy paste the solution and complete a lab within minutes. Only a few labs have associated quizzes that test the knowledge gained by the exercise. This could be improved on as an extension of the work. There are opportunities to perform formative and summative assessments to gauge a learner's learning gains and the platform's overall delivery. Formative evaluation can be built into activities through short, informal student quizzes that evaluate a learners' understanding of the presented concepts. In turn, a summative assessment would evaluate a student's learning, and collect feedback on the utility of the labs to a user's learning. It is important to note that in contrast to the Learning labs discussed in this review, Challenge labs only provide objectives rather than step-by-step instructions. These labs test a participant's ability to achieve them. Challenge labs could be associated with the quests themselves, thus functioning as capstone projects that required the students to apply the knowledge gained from the session. Taken together this data could be used to build a quantitative model for evaluating course success and to develop a profile of the kind of students (or groups of students) that are most likely to benefit from these labs.

# 5. CONCLUSIONS AND LESSONS LEARNED

Student training in computing is a critical area where demand currently outweighs supply. By design, the scaffolded nature of the labs in quests provide few opportunities to solve problems by developing hypotheses and validating them. In contrast, to learning labs, Challenge labs present a general objective and require participants to complete a series of tasks. In such labs, participants may choose one of many approaches to solve the problem at hand. Our experience from this training program shows that intermediate-level coding can be effectively combined with a number of interesting activities. The largest challenges lies in that these activities are not connected directly to HPC users. An abundance of free-tutorials on Github, hands-on activities and scaffolding educational technique helped reduce *a priori* knowledge that a user is required to have in order to get started on traditional HPC resources. The strategies described in this work are likely to help specific aspects of undergraduate curricula. These approaches present exciting opportunities to engage HPC users in Cloud computing, a critical step, to get them to use CACRS effectively.

# 6. SUPPORTING INFORMATION

All training materials used in this study are available to the community at Qwiklabs (https://www.qwiklabs.com). The author's progress on the Qwiklabs platform can be followed on his Linked-in profile. Labs used in the in-person training exercise by a GCP external provider are the property of the instructor and may be accessed for reviewing purposes with explicit permission of the instructor. Surveys, and review exercises that will be developed as part of this longitudinal study may be requested from the author. Please send us feedback about your adoption experience via an email to help@hprc.tamu.edu.

# 7. ACKNOWLEDGEMENT

The authors thank staff at Texas A&M HPRC and Google Cloud Platform for assisting with the research related to this study. We thank the GCP training crew at Google Next 2019 for helping get started on Qwiklabs and offering introductory credits. Additional credits for Qwiklabs were kindly provided on request by GCP. We gratefully acknowledge support from the National Science Foundation (NSF). We thank the NSF for award #1649062, "*NSF Workshop: Broadening Participation in Chemical and Biological Computing at the Early Undergraduate Level"*, award #1730695, *"CyberTraining: CIP: CiSE-ProS: Cyberinfrastructure Security Education for Professionals and Students",* and award # 1925764, "*NSF CC* SWEETER: South West Expertise in Expanding Training Education and Research"*

# 8. REFERENCES

[1] C.A. Stewart, T.M. Cockerill, I. Foster, D. Hancock, N. Merchant, E. Skidmore, D. Stanzione, J. Taylor, S. Tuecke, G. Turner, M. Vaughn, and N.I. Gaffney, "Jetstream: a self-provisioned, scalable science and engineering cloud

environment," In Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure. St. Louis, Missouri. ACM: 2792774. p. 1-8. http://dx.doi.org/10.1145/2792745.2792774

[2] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G.D. Peterson, R. Roskies, J.R. Scott, N. Wilkins-Diehr, "XSEDE: Accelerating Scientific Discovery", Computing in Science & Engineering, vol.16, no. 5, pp. 62-74, Sept.-Oct. 2014, doi:10.1109/MCSE.2014.80

[3] Aristotle - https://federatedcloud.org/about/index.php

[4] CloudLab - https://cloudlab.us/

[5] NIH Strides - https://datascience.nih.gov/strides

[6] Critical Techniques, Technologies and Methodologies for Advancing Foundations and Applications of Big Data Sciences and Engineering (BIGDATA) - https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=504767

[7] NSF Campus Cyberinfrastructure (CC*) - https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=504748

[8] Enabling Access to Cloud Computing Resources for CISE Research and Education (Cloud Access) - https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=505591

[9] Coursera- https://www.coursera.org/

[10] Linux Academy - https://linuxacademy.com/

[11] NSF and Internet2 to explore cloud computing to accelerate science frontiers - https://nsf.gov/news/news_summ.jsp?cntn_id=297193

[12] Google acquires Qwiklabs to teach Developers cloud skills, November 21, 2016. This article may be accessed online at URL - https://techcrunch.com/2016/11/21/google-acquires-qwiklabs-to-teach-developers-cloud-skills/

[13] Google acquires Qwiklabs to teach Developers cloud skills, November 21, 2016. This article may be accessed online at URL - https://blog.qwiklabs.com/skill-up-the-world/

[14] Texas Regional Collaboratives, "Building the Texas Computer Science Pipeline Strategic Recommendations for Success | theTRC.org," 2014

[15] K. Saichaie, D. C. Brooks, P. Long, R. Smith, R. Holeton, C. Meyers, A. Finkelstein, S. Dugdale, "7 Things You Should Know About Research on Active Learning Classrooms," in ELI 7 Things You Should Know, Educause Learning Initiative (ELI), 2017

[16] P. Baepler, J. D. Walker, D. C. Brooks, K. Saichaie, C. I. Petersen, B. A. Cohen, "A Guide to Teaching in the Active Learning Classroom: History, Research, and Practice," Stylus Publishing, ISBN-13: 978-1620363003, 2016

[17] Student-Centered Active Learning Environment with Upside-down Pedagogies: http://scaleup.ncsu.edu/

[18] A. Yadav, N. Zhou, C. Mayfield, S. Hambrusch, and J. T. Korb, "Introducing computational thinking in education courses," in Proceedings of the 42nd ACM technical symposium on Computer science education, pp. 465–470, ACM, 2011.

[19] J. J. Lu and G. H. Fletcher, "Thinking about computational thinking," in Proceedings of the 40th ACM Technical Symposium on Computer Science Education, SIGCSE '09, (New York, NY, USA), pp. 260–264, ACM, 2009.

[20] K. Brennan and M. Resnick, "New frameworks for studying and assessing the development of computational thinking," in Annual American Educational Research Association meeting, (Vancouver, BC, Canada), 2012

[21] J. M. Wing, "Computational thinking and thinking about computing," Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences, vol. 366, no. 1881, pp. 3717–3725, 2008.

[22] M. Prince, "Does active learning work? a review of the research," Journal of engineering education, vol. 93, no. 3, pp. 223–231, 2004.

[23] J. Parsons and L. Taylor, "Improving student engagement," Current issues in education, vol. 14, no. 1, 2011.

[24] Larkin, M. (2002). Using scaffolded instruction to optimize learning. http://www.vtaide.com/png/ERIC/Scaffolding.htm

[25] C. Holdgraf, A. Culich, A. Rokem, F. Deniz, M. Alegro, D. Ushizima. "Portable Learning Environments for Hands-On Computational Instruction" in Proceedings of PEARC17, New Orleans, LA, USA, July 09-13, 2017, 9 pages. DOI: 10.1145/3093338.3093370

[26] D. K. Chakravorty, M. Pennings, H. Liu, Z. Wei, D. M. Rodriguez, Levi T. Jordan, D. F. McMullen, N. Ghaffari, and S. D. Le. "Effectively Extending Computational Training Using Informal Means at Larger Institutions," Journal of Computational Science Education 2018, 40-47 DOI 10.22369/issn.2153-4136/10/1/7.

[27] D. K. Chakravorty, M. Pennings, H. Liu, Z. Wei, D. M. Rodriguez, L. T. Jordan, D.F. McMullen, N. Ghaffari, S. D. Le, D. Rodriquez, C. Buchanan, and N. Gober. "Evaluating Active Learning Approaches for Teaching Intermediate Programming at an Early Undergraduate Level," Journal of Computational Science Education 2018, 61-66 DOI 10.22369/issn.2153-4136/10/1/10.

[28] D. K. Chakravorty, D. F. McMullen, N. Gober, J. H. Seo, M. Bruner, and A. Payne. "Using Virtual Reality to Enforce Principles of Cybersecurity," Journal of Computational Science Education 2018, 81-87 DOI 10.22369/issn.2153-4136/10/1/13.

[29] D. K. Chakravorty, M. Pennings, H. Liu, X. Thomas, D. M. Rodriguez, and L.M. Perez. "Incorporating Complexity in Computing Camps for High School Students – A Report on the Summer Computing Academy Program at Texas A&M University," Journal of Computational Science Education 2019. (Accepted).

[30] Slurm on Google Cloud Platform - https://github.com/SchedMD/slurm-gcp

[31] Using the Cloud Client Libraries for Python - https://cloud.google.com/compute/docs/tutorials/python-guide

[32] How to set up a VPN between strong Swan and Cloud VPN - https://cloud.google.com/community/tutorials/using-cloud-vpn-with-strongswan

[33] GCP Compute Engine: Instance Template - https://cloud.google.com/compute/docs/instance-templates/

[34] Slurm Elastic Computing (Cloud Bursting) - https://slurm.schedmd.com/elastic_computing.html

[35] GCP Clound DNS: Overview The URL may be found at - https://cloud.google.com/dns/docs/overview

## REPRODUCIBILITY APPENDIX

Computational results are not part of this paper. (Please refer tohttps://sc18.supercomputing.org/submit/sc-reproducibility-initiative/)

# Teaching HPC Systems Administrators

Alex Younts
ayounts@purdue.edu
Purdue University

Stephen Lien Harrell
sharrell@purdue.edu
Purdue Univeristy

## ABSTRACT

The ability to grow and teach systems professionals relies on having the capacity to let students interact with supercomputers at levels not given to normal users. In this paper we describe the teaching methods and hardware platforms used by Purdue Research Computing to train undergraduates for HPC systems-facing roles. From Raspberry Pi clusters to the LittleFe project, previous work has focused on providing miniature hardware platforms and developing curriculums for teaching. Recently, we have developed and employed a method using virtual machines to reach a wider audiences, created best practices, and removed barriers for approaching coursework. This paper outlines the system we have designed, expands on the benefits and drawbacks over hardware systems, and discusses the failures and successes we have had teaching HPC System Administrators.

## KEYWORDS

hpc, syspros, systems professionals

## 1 INTRODUCTION

As leadership computing facilities draw closer to exascale and academic research computing centers mature around the world, the need for competent HPC System Administrators is increasing. Similarly, the complexity of HPC systems is increasing with the slowing of the Moore's Law trend and node heterogeneity becoming all but a necessity. Gone are the days when commodity hardware connected with some cheap Ethernet switches were a viable solution to solving the world's science problems. Today, system administrators need to tackle accelerators, big data technologies, AI and ML frameworks, ever changing network fabrics, and a quickly changing ecosystem of core architectures. In the same way that this complexity has increased, HPC system administration training approaches must also mature in complexity and scope.

### 1.1 Roles of HPC System Administrators

Although system administration, as a professional practice, is well established, HPC adds a layer of complexity that requires it's own community, documentation, and training. While operating system skills are the same, the "High Performance" in HPC requires understanding of CPU architecture, exotic networks, computer architecture, and parallel technologies in a way that is foreign to most system administrators. As a background for teaching HPC System Administrators, it is important to set a baseline of roles and responsibilities needed to run an HPC machine.

There are four major roles that encompass the operation of HPC machines (clusters):

- Network Administration
- Parallel Storage Administration
- Data Center and Hardware Administration
- System Administration and Automation

Additionally, these roles often have different responsibilities within an HPC context than they would in a typical industry role. For instance, a traditional network system administrator may need to know *a*) routing protocols, *b*) network hardware administration, and *c*) TCP/IP, whereas an HPC network administrator would be expected to not only know these topics, but be expected to understand low latency interconnects, such as RDMA networks, as well as understanding how the network layer can impact parallel jobs running over it. Similarly, while storage administrators may need to know *a*) NFS, *b*) CIFS, and *c*) storage appliances in a traditional industry role, within HPC they would be required to know parallel file systems (Lustre, GPFS), file systems over RDMA and even tape archival systems that often play a vital role in research-focused computing. Data center focused administrators for large system installations largely have the same concerns as their HPC-focused colleagues. Lastly, systems administrators, who are responsible for the OS, create and submit hosts and nodes of a cluster, set up the scheduler, the general deployment of the entire cluster, and the on going configuration management, are expected to understand a completely different technology stack. While each of these roles requires considerable expertise, many time systems facing staff are expected to be experts in a few and knowledgeable in all of these roles. The more roles staff are knowledgeable about, the better understanding of each component (systems, network, storage) and how each component interacts with the whole system. This knowledge is crucial to understanding failures and tuning the system to be truly High Performance.

## 2 HPC SYSTEMS TRAINING APPROACHES

The traditional method for teaching system administration is the apprentice model [14]. Within this model, an "expert" slowly feeds tasks of increasing difficulty to the "apprentice", while at the same time being a resource for topics intrinsic to a specific task, but also as a guide to the self-learning process. While this is an important and time-tested method for training system administrators of every variety, this approach does not scale and is extremely high touch. It requires an "expert" to have plenty of time and the right demeanor for the apprenticeship to create a competent HPC System Administrator.


https://doi.org/10.22369/issn.2153-4136/11/1/16

## 2.1 Workshops

Currently there are two workshops that include HPC System Administration training and tutorials. The first and oldest, Linux Clusters Institute (LCI), "provides education and advanced technical training for IT professionals who deploy and support High Performance Computing (HPC) Linux clusters" [4]. LCI provides two separate workshops, one for Linux novices and one for Linux System Administrators that are trying to learn HPC. The second workshop, The TACC Institute Series Immersive Training in Advanced Computation: Designing and Administering Large-scale Systems, also provides a week-long workshop where students are "provisioning nodes, installing and configuring resource managers, maintaining a sane user environment, and addressing security concerns" [6]. Both of these workshop provide hands-on activities as well as lectures that provide context for the training.

## 2.2 Student Cluster Competition

The Student Cluster Competition (SCC) [9] is described as "a microcosm of a modern HPC center that teaches and inspires students to pursue careers in the field. It demonstrates the breadth of skills, technologies and science that it takes to build, maintain and utilize a supercomputer." [18]. This event, while not solely focused on HPC Systems Administration, includes opportunities for undergraduates to learn and practice HPC System Administration in the heart of the competition.

## 2.3 Undergraduate Training at Purdue

At Purdue University, as well as other institutions, the HPC systems staff provides job opportunities for undergraduates, as well as HPC and clustering classes on campus. The material for the classes grew out of the Purdue SCC program and have evolved into their own topics over the years.

# 3 PURDUE HPC SYSTEMS TRAINING THROUGH THE YEARS

Purdue University's Research Computing center started hiring undergraduates to do HPC systems work in 2003. This was the begining of our HPC systems administrator training which primarily used an apprentice model. Starting in 2007, Systems staff mentored students in the Student Cluster Competition series. Since then, staff have been iterating on how to best teach HPC concepts and system administration to undergraduates and have tried many technology platforms to provide consistent, affordable, and reliable platforms for teaching HPC.

## 3.1 Early Years

In 2007 and the following few years, training initiatives at Purdue were based around the Student Cluster Competition. [10] HPC systems staff partnered with faculty on campus to provide overviews of parallelism, however these early years of the competition were heavily geared toward HPC system administration, more so than later years. The classes were primarily an open lab format which could be categorized as a distributed apprentice model.

*3.1.1 Training Platform: SCC Competition Hardware.* The hardware platforms chosen for these competitions varied from year to year, and as the years progressed, the clusters became more complex. In 2007, the student cluster was architected after a traditional Beowulf cluster with simple servers and Ethernet networking. A year later, the Purdue team took an experimental SiCortex many-core cluster composed of hundreds of MIPS cores and a custom interconnect [2]. Later, the cluster concept turned from homogeneous compute nodes to hybrid nodes prominently featuring GPU's for compute acceleration.

**Figure 1: Purdue Student Cluster Competition Team**

What remained the same throughout the years of building the SCC clusters was that the hardware was always a short term loan from a sponsoring vendor. Students, especially those handling the team's system administrator needs, were always presented with the most recently released hardware and the time challenge of preparing it for the competition.

*3.1.2 Outcomes and Lessons Learned.* While being a naive implementation, this first method of teaching undergraduates HPC systems-facing topics was somewhat effective and an important first step. Those first classes, which were approximately ten students each, were responsible for three students becoming HPC system administrators and are currently working in the HPC community today. Additionally, students were introduced to the academic writing process and published two experience papers on the subject [12] [22].

We found that not having formal classes for something like the SCC was a detriment to student participation. Student attendance and commitment were sometimes low. Additionally, giving students complex HPC-centric hardware right away created a very steep ramp for students to overcome.

## 3.2 Formalized Classes

In 2011, to combat low participation in the SCC meetings and inconsistency in training, classes were formalized beyond an open lab. While the classes were still centered around the SCC, we worked to create a curriculum that could be reused and contained important parallel computing and HPC systems topics.

*3.2.1 Assignments.*

(1) Introduction to HPC and the SCC

(2) Usage of Supercomputers (login and compile HPCC)
(3) Introduction to Computer and HPC Architecture
(4) Linux Installation, Daemons, Configuration Files and Basic Networking
(5) Batch Schedulers, Advanced SSH
(6) Compiling with MPI and OpenMP
(7) Strong and Weak Scaling Studies and Bottleneck Identification
(8) Specific SCC Applications and Strategy for the Rest of the Course

*3.2.2 Training Platform: Recycled Desktops.* Our first attempts at providing a dedicated training platform over short term vendor loans was to re-purpose desktops [13] after their life in the student computer labs ended. Every student was given 4 desktops, cables, and an Ethernet switch. This was a fairly adequate solution when new parts were available yearly, but as budgets and technology have changed, computer labs changed their refresh cycles and are no longer on a predictable schedule so this effort was sustainable long term.

*3.2.3 Outcomes and Lessons Learned.* This era of classes was a marked improvement over previous years. Although the classes still included practical labs around the SCC, the instructors touched on more “theory” topics than before such as computer architecture.

Additionally as the desktops aged, replacement parts were necessary, especially after inexperienced students performed repairs. Plus, full sized desktops take up a lot of floor space. During an inspection of the space, a glib manager was heard saying “It would be terrible if a student got trapped under an avalanche of chalk-dust encrusted desktops."

## 3.3 Bare Hardware to Computational Visualization

In 2013, we went back to an open lab that revolved around the SCC and began a separate class that ran alongside the SCC open lab. This was open to anyone that was interested in HPC, not just SCC participants. It was also the time that we started to focus on inspiring undergraduates as well as teaching. A breadth-first approach was taken with the idea of having a final project where students could see the fruits of their labors in an accessible way. We chose to run weather code to forecast as weather maps from a visualization of a forecast is a common and accessible experience for almost everyone, regardless of background. [8]

*3.3.1 Assignments.*

(1) Introduction to HPC
(2) Basic Linux Installation
(3) Automating Installs
(4) Hardware Setup and Install [lab]
(5) Schedulers
(6) Interconnects and Storage
(7) DHCP and DNS for clusters [lab]
(8) Shared Storage for Clusters [lab]
(9) Scheduler Setup [lab]
(10) Installing MPICH
(11) Installing WRF
(12) Troubleshooting WRF and MPI

**Figure 2: Student with LittleFE cluster.**

**Figure 3: LittleFE Computer Lab.**

(13) Introduction to Python
(14) Visualizing wrfout
(15) Automating WRF Runs with Python

*3.3.2 Training Platform: LittleFE.* The "little iron" project brings together curriculum and a hardware plan that many schools have implemented to teach students high performance computing [16]. The platform itself is made of bare motherboards and metal rods to act as a mounting platform. A student or a small team of students get their own cluster. Each cluster costs a reasonable amount for the number of computers involved and students can get the visceral, hands on experience putting together their machine [17]. Also, it is possible to get low-end consumer video hardware build onto the node motherboards for lightweight accelerator work as well. While initial investment per seat is manageable, and cheaper than a half-rack of real cluster compute nodes, the costs are not insignificant.

After outgrowing the "pile of desktops" solution, we sought an internal grant and built 15 LittleFe clusters to continue as our teaching platform. Initially, it seemed like we could procure each seat for a reasonable sum of money, but after the build was over

and all receipts were collected, the price had risen to approximately 20% due to needing additional pieces. This solution provided many benefits that we were looking for but hit a wall when we needed to scale the number of students we reached. Not only did the per-seat cost end up being more expensive, but the desk space and off-semester storage of the hardware became a logistical problem.

*3.3.3 Lessons Learned.* The breadth approach, while well intentioned, was too much to include in one semester. Students were often faced with hardware or OS problems when they were supposed to be running WRF or writing python to visualize. Although this method may work if the topic list is paired down, one must be careful not to overwhelm the students. In the end, some students did not have the time to complete the visualizations, thus negating the point of this method.

The LittleFe hardware suffered a number of growing pains as we progressed through the build and the course. The materials were difficult to acquire through the University procurement system, a web tool designed to buy complete computers and not just a stack of parts. The mounting hardware shined a light on both cooling problems and static electricity issues, both causing general instability for students. Given the overall mounting solution, the clusters were essentially immobile and required us to open additional lab periods so students could complete their work. Although this hardware was not well suited for this task, it has been used quite successfully for single day hack-a-thons and is sufficiently stable for that use case. If we had the resources to develop a second generation of LittleFe instruments, we believe many of these issues could be mitigated.

# 4 CURRENT EFFORTS

As an amalgamation of our previous experiences, the courses that are instructed today [11] have two tracks. First a scientific computing track, which provides students with some basic Linux skills but focuses on running and visualizing scientific codes. Then an HPC Systems Track, which truly focus on important aspects of building systems. This was a hybrid approach of inspiring undergraduates but still focusing where the students interests lie in order to reduce the amount of topics from our previous efforts.

## 4.1 HPC Systems Track

The HPC Systems Track was integrated into the new course curriculum as a way to engage a more diverse set of students. Students in the sciences had a firm footing for many of the course activities, but students from the Polytechnic school and Computer Science and Engineering majors were provided this path to understand the technical work behind the scenes of supercomputing. The course was broken into three modules and this track was offered as an alternative to the second module. The first module covered introductory materials and labs and the third module was a crash course in simulating fluid dynamics problems using OpenFOAM.

The System Track included practical activities in the data center to work on the University's real resources but focused on providing a hands-on-keyboard experience to learn the guts of HPC clusters. The goal was for students to be able to explore a working system and replicate it themselves without copying the example. Each assignment had a final stretch goal that allowed us to judge whether students were simply copying configuration files around or actually exploring and learning the material.

*4.1.1 Assignments.*

(1) Introduction to HPC
(2) Tutorial on the Advanced Linux Shell [lab]
(3) Presentation on Cluster Architecture
(4) Data Center Tour and Hands-on Lab
(5) Basic Linux Virtual Machine Installation [lab]
(6) Master Node xCAT Installation [lab]
(7) Building Compute Node Images [lab]
(8) Installation of Slurm [lab]
(9) Running Sample Jobs [lab]

## 4.2 Virtual Labs

As we expanded our scale to dozens of students per semester and planned for even wider reach, it became clear the monetary investments in physical infrastructure and the time investment getting low grade hardware to cooperate were detracting from reaching our goals. We evaluated several commercial cloud-based offerings to host the lab environment but the options seemed geared towards traditional client-server IT teaching. We also researched using infrastructure as code tools, like Terraform [1], to automate lab environments in AWS, but found the variable costs very difficult to quantify and potentially quite large. We needed a new way forward that fit with the campus's available cyberinfrastructure and could be delivered remotely. We came up with the virtual HPC lab concept.

*4.2.1 Implementation.* The basics of the method was to enable our Scholar cluster [5], which is a supercomputer dedicated to computational research teaching, to run scripted virtualized clusters for students. We required the solution involve no privileged system access (e.g. sudo access) or access to the underlying network infrastructure [20] (e.g. Linux Ethernet bridges). We used the popular QEMU system emulator along with the Virtual Distributed Ethernet (VDE) userland networking stack [7].

Students had the ability to launch a script that brought up their virtual lab through our ThinLinc remote desktop in a web browser and get an empty, semi-configured, or completed configured cluster environment [21]. The script, just a bash script run by students at a terminal, lets students choose the lab to launch, handles creating copy on write snapshots from golden image masters, lets students continue progressing on current labs, and the ability to reset a lab back to a checkpoint if something goes wrong.

The lab environment spawns several windows, each representing the QEMU console to a running virtual machine or console access to a VDE network instance. Students are able to adjust the running parameters of the QEMU instances (e.g. inserting a boot disk) and, with some limitations, have essentially identical access to the lab as if it was running on real hardware.

*4.2.2 Lessons Learned.* Using previous courses as comparison, the first readily apparent success of the virtual lab concept was that students were learning valuable HPC skills in the first lab. Student frustration was also significantly down, as rolling back to a working check point or starting over did not take an hour waiting for the RedHat installer to run. Students also appreciated the ability to

work on the labs and assignments outside of class since the Scholar environment is available remotely any time of the day.

The class sessions themselves progressed fairly normally. The lecturer presented context and background information at the start of class. They could demonstrate any tips or tricks to the class using a snapshot of the environment at the same stage of progress as the students' copies. Teaching assistants were available to assist individual students on their laptops as problems arose or remotely by using ThinLinc session sharing.

While the experience was overall very positive, one speed bump did sneak up. Scholar, built as a platform for teaching science, had a scratch file system primarily tuned for standard HPC workloads. Having numerous QEMU virtual machines running with their disk images doing random small block I/O did take a toll on overall system responsiveness. The scratch file system was based on the ZFS file system and adding a small quantity of SSD disks resolved the issue.

### 4.3 Future work

After seeing a pair of courses successfully run using the virtual HPC lab environment, we are encouraged that our goal to some day offer our courses widely is possible. As we move forward to publish our curriculum and environments, we hope to build momentum to provide the academic HPC community the skill sets that are desperately required.

## 5 OTHER TRAINING PLATFORMS

Although these platforms have not been used in any HPC Systems Administration classes at Purdue, they have been evaluated and and may fit the needs of others depending on the availability of resources and time.

### 5.1 Raspberry Pi Clusters

A small stack of Raspberry Pi's are all the rage across the Internet. From business [15] to education to Department of Energy labs [19], everyone seems to be building tiny clusters [3]. These clusters do everything from compiling and testing pipelines, to simple MPI scalability testing, and to running production workloads through Kubernetes.

Aside from reliability issues with early Rasperry Pi's and clone boards, the best part about a stack of single board computers is the cost. A student can readily be provided a cluster for under $500. However, we believe that many of the drawbacks found in using dedicated hardware are still present in a cluster of Pi's.

### 5.2 Cloud Environments

Commercial cloud providers are a potential avenue to explore in greater depth. The core technology requirements *a*) isolated network segments, *b*) snapshots and rollback of instances, and *c*) the ability for remote student assistance already exist as features on all the various providers' platforms. Additionally, while the scripts written for Scholar could be portable to other institutions and system resources, some effort will be required. We hope to keep that effort to a bare minimum as we move towards publishing our work further, but we acknowledge the strong advantages of a universal platform with consistent lab materials and curriculum.

The largest drawbacks at the present time to the cloud are a lack of a cohesive interface for students and the variable costs an institution will incur depending on student usage of the environment.

## 6 CONCLUSIONS

Purdue Research Computing's training methods for HPC System Administrators and the hardware platforms have supported these efforts. We have found that running our own virtualized environment for teaching to be effective to meet our goals of low cost, low overhead, and low student frustrations. Additionally, we have split our class to have two separate tracks to focus on the HPC Systems topics while still maintaining our goal of inspiring undergraduates.

## ACKNOWLEDGMENTS

We thank our faculty partners, Dr. Evans, Dr. Baldwin and Dr. Ward, and all our colleagues and management who made this work possible. We'd also like to thank Christopher Phillips for his keen editorial skills.

## REFERENCES

[1] 2020. Terraform. https://www.terraform.io [Online; accessed 17. Jan. 2020].
[2] 2020. The SiCortex SC series | TOP500 Supercomputing Sites. https://web.archive.org/web/20090531211623/http://www.top500.org/2007_overview_recent_supercomputers/sicortex_sc_series [Online; accessed 17. Jan. 2020].
[3] Joel C Adams, Jacob Caswell, Suzanne J Matthews, Charles Peck, Elizabeth Shoop, and David Toth. 2015. Budget Beowulfs: A showcase of inexpensive clusters for teaching PDC. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. ACM, 344–345.
[4] David Akin, Mehmet Belgin, Timothy A. Bouvet, Neil C. Bright, Stephen Lien Harrell, Brian Haymore, Michael Jennings, Rich Knepper, Daniel LaPine, Fang Cherry Liu, Amiya Maji, Henry Neeman, Resa Reynolds, Andrew H. Sherman, Michael Showerman, Jenett Tillotson, John Towns, George Turner, and Brett Zimmerman. 2017. Linux Clusters Institute Workshops: Building the HPC and Research Computing Systems Professionals Workforce. In *Proceedings of the HPC Systems Professionals Workshop (HPCSYSPROS'17)*. ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3155105.3155108
[5] M. E. Baldwin, X. Zhu, P. M. Smith, S. L. Harrell, R. Skeel, and A. Maji. 2016. Scholar: A Campus HPC Resource to Enable Computational Literacy. In *2016 Workshop on Education for High-Performance Computing (EduHPC)*. 25–31. https://doi.org/10.1109/EduHPC.2016.009
[6] Texas Advanced Computing Center. 2019. TACC Institute Series Immersive Training in Advanced Computation. https://www.tacc.utexas.edu/education/institutes/designing-and-administering-large-scale-systems
[7] Renzo Davoli. 2005. Vde: Virtual distributed ethernet. In *First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*. IEEE, 213–220.
[8] Stephen Lien Harrell, Benjamin J Cotton, Michael E Baldwin, and Andrew L Howard. 2013. Developing a Scientific Computing Cluster Course for the Undergraduate Curriculum. (2013).
[9] Stephen Lien Harrell, Hai Ah Nam, Verónica G. Vergara Larrea, Kurt Keville, and Dan Kamalic. 2015. Student Cluster Competition: A Multi-disciplinary Undergraduate HPC Educational Tool. In *Proceedings of the Workshop on Education for High-Performance Computing (EduHPC '15)*. ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/2831425.2831428
[10] Stephen Lien Harrell, Preston M. Smith, Doug Smith, Torsten Hoefler, Anna A. Labutina, and Trinity Overmyer. 2011. Methods of Creating Student Cluster Competition Teams. In *Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery (TG '11)*. ACM, New York, NY, USA, Article 50, 6 pages. https://doi.org/10.1145/2016741.2016795
[11] Elizabett Hillery, Mark Daniel Ward, Jenna Rickus, Alex Younts, Preston Smith, and Eric Adams. 2019. Undergraduate Data Science and Diversity at Purdue University. In *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (Learning) (PEARC '19)*. ACM, New York, NY, USA, Article 88, 6 pages. https://doi.org/10.1145/3332186.3332202
[12] Andrew Howard, Alex Younts, Preston M. Smith, and Jeffery J. Evans. 2008. Undergraduate experience in clustering at the SC07 Cluster Challenge. In *In Proceedings of the 2008 Linux Clusters Institue.*

[13] Jason St John and Thomas J Hacker. 2017. A Small-Scale Testbed for Large-Scale Reliable Computing. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 1251–1258.
[14] David Jones. 2018. How do you teach Systems Administration?. In *SAGE-AU*.
[15] Patrick Kennedy. [n. d.]. Oracle Shows 1060 Raspberry Pi Supercomputer at Oracle OpenWorld 2019. https://www.servethehome.com/oracle-shows-1060-raspberry-pi-supercomputer-at-oow/
[16] Mobeen Ludin, Aaron Weeden, Jennifer Houchins, Skylar Thompson, Charles Peck, Ivan Babic, Kristin Muterspaw, and Elena Sergienko. 2013. LittleFe: The high performance computing education appliance. In *2013 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 1–1.
[17] Charles Peck. 2010. LittleFe: parallel and distributed computing education on the move. *Journal of Computing Sciences in Colleges* 26, 1 (2010), 16–22.
[18] SC17. 2017. Student Cluster Competition. https://sc17.supercomputing.org/studentssc/student-cluster-competition/index.html
[19] Adam Simpson, Anthony DiGirolamo, and Robert D. French. [n. d.]. Tiny Titan from Oak Ridge Leadership Computing Facility. https://tinytitan.github.io
[20] Julian Stecklina. [n. d.]. A Userspace Packet Switch for Virtual Machines. ([n. d.]).
[21] Abhinav Thota, Le Mai Weakley, Ben Fulton, HE Dennis, Laura Huber, Scott Michael, Winona Snapp-Childs, Stephen Lien Harrell, Alexander Younts, Daniel T Dietz, et al. 2019. Research Computing Desktops: Demystifying research computing for non-Linux users. In *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning)*. ACM, 54.
[22] Alex Younts, Andrew Howard, Preston M. Smith, and Jeffrey J. Evans. 2009. Bringing disruptive technology to competition. In *In Proceedings of the 10 th LCI International Conference on High-Performance Clustered Computing*.

# Contributing HPC Skills to the HPC Certification Forum

Julian Kunkel
University of Reading
Reading, United Kingdom
j.m.kunkel@reading.ac.uk

Kai Himstedt
Universität Hamburg
Hamburg, Germany

Weronika Filinger
EPCC, The University of Edinburgh
Edinburgh, United Kingdom

Jean-Thomas Acquaviva
DDN
Paris, France

Anja Gerbes
Goethe-Universität
Frankfurt am Main, Germany

Lev Lafayette
University of Melbourne
Melburne, Australia

## ABSTRACT

The International HPC Certification Program has been officially launched over a year ago at ISC'18 and since then made significant progress in categorising and defining the skills required to proficiently use a variety of HPC systems. The program reached the stage when the support and input from the HPC community is essential. For the certification to be recognised widely, it needs to capture skills required by majority of HPC users, regardless of their level. This cannot be achieved without contributions from the community. This extended abstract briefly presents the current state of the developed Skill Tree and explains how contributors can extend it. In the talk, we focus on the contribution aspects.

## 1 INTRODUCTION

Training was always important for the HPC community. However, creating and providing training for practitioners with diverse backgrounds and different levels of computer literacy is challenging. The continuous growth of the HPC community make the traditionally accepted training solutions insufficient. The multitude of paths leading into HPC means the training providers can hardly assume any previous knowledge or programming experience. There is no common base knowledge possessed by all new users. This makes the development and delivery of any training complicated. The main goal of the *International HPC Certification Forum (HPCCF)* is to ease the provision and uptake of training by clearly categorising, defining and eventually testing the skills required to efficiently use HPC resources. For this effort to be successful, the community needs to support and contribute to the process of defining the HPC Skill Tree. Input from members of different HPC branches is crucial. This extended abstract aims at presenting the current high-view state of the Skill Tree and describe the process of contributing.

## 2 SKILL TREE

The skills are organised in a tree structure from a coarse-grained to a fine-grained representation, allowing users to browse the skill based on the semantics. A skill is defined as a set of **learning outcomes** and relevant metadata. Within a single skill, there can also be multiple levels (basic, intermediate and expert level) building upon each other and further distinguishing the expertise. We expect the practitioners to acquire the lower levels before progressing to more complex levels. Currently, the Skill Tree[1] contains six branches: HPC Knowledge (**K**), Use of HPC Environment (**USE**), Performance Engineering (**PE**), Software Development (**SD**), System Administration(**ADM**), and Big Data Analytics (**BDA**). These are briefly described in the subsequent sections.

### 2.1 HPC Knowledge

This branch contains the basic information necessary to understand what supercomputers are, how they work and how to make use of them. It should give enough background to allow new HPC practitioners to understand different aspects of HPC environments and how to make use them. Its basic level sub-branches are: **K1-B** *Supercomputers*, **K2-B** *Performance modelling*, **K3-B** *Program Parallelisation*, **K4-B** *Job Scheduling*, and **K5-B** *Cost Modelling*.

**Learning outcomes**: A practitioner familiar with this branch should: understand different aspects of HPC hardware, software and operation of HPC systems; know how to use simple performance models for systems and applications; understand scaling and parallel efficiency; know different parallelisation paradigms; be familiar with using HPC systems, and understand job scheduling principles.

### 2.2 Use of HPC Environments

HPC environments are different from local systems and cloud environments, and different HPC systems may utilise specific solutions to setup and execute parallel applications. Although, knowing how to use a specific system is important, most users will eventually need to use more than one system. Therefore, understanding the underlying principles is equally important. This sub-tree covers skills for different user roles: users, testers and developers, allowing to efficiently develop, build, run and monitor parallel applications and automate repetitive tasks. The current basic level sub-branches include: **USE1-B** *Cluster Operating System*, **USE2-B** *Running of Parallel Programs*, **USE3-B** *Building of Parallel Programs*, **USE4-B** *Developing Parallel Programs*, **USE5-B** *Automating Common Tasks*, and **USE6-B** *Workflow Integration*.

A practitioner familiar with this branch should be able to: apply tools provided by the operating system to navigate and manage files and executables; select the software environment to effectively build and develop existing and novel applications; use a workload

[1] https://www.hpc-certification.org/skills/


https://doi.org/10.22369/issn.2153-4136/11/1/17

manager to allocate HPC resources; construct workflows that utilise remote (distributed) environments to execute parallel workflows; and design and deploy scripts that automate repetitive tasks.

### 2.3 Performance Engineering

Time to solution is one of the basic metrics in HPC - it's vital to obtaining the results in a timely manner and using the most optimal resources. Performance engineering gives a systematic approach to measuring and analyzing performance of systems and applications. This sub-tree should cover the performance of applications and systems, optimising of the runtime settings and applications and strategies for efficient use of HPC resources. It has five basic level sub-branches focusing on: **PE1-B** *Cost awareness*, **PE2-B** *Measuring System Performance*, **PE3-B** *Benchmarking*, **PE4-B** *Tuning*, and **PE5-B** *Optimisation Cycle*. A practitioner familiar with this branch should be able to: describe the optimisation cycle; estimate the cost a job on an HPC system; understand the typical performance pitfalls; know how to perform benchmarks and use their results as baseline; use profiling tools to analyse the performance and identify bottlenecks; understand how various system and application settings influence the performance; and finally be aware of optimised libraries and how to use them.

### 2.4 Software Development

Software engineering is often neglected in computational science. However, it can increase productivity by providing scaffolding for the collaborative programming, reducing the coding errors and increasing the manageability of software. This branch covers concepts, practices and methods from software engineering that should be applied in HPC environments. The current basic level sub-branches are: **SD1-B** *Programming Concepts for HPC*, **SD2-B** *Programming Best Practices*, **SD3-B** *Software Configuration Management*, **SD4-B** *Agile Methods*, **SD5-B** *Software Quality*, **SD6-B** *Software Design and Architecture*, and **SD7-B** *Software Documentation*. A practitioner familiar with this branch should be able to: apply software engineering methods and best practices when developing parallel applications; write a modular and reusable code by using software design principles; apply HPC design patterns; know how to configure and use integrated development environments (IDEs) to seamlessly perform a typical development cycle; use sophisticated debuggers for parallel programs; define and establish coding standards and conventions in a project; apply version and configuration management to establish and maintain consistency of a program or software system throughout its life; configure an environment for continuous integration with basic processing steps like compiling and automated testing; apply unit testing in a specific programming language using appropriate unit testing frameworks; and appropriately document the entire software ecosystem.

### 2.5 Big Data Analytics

The analysis of large volumes of data was traditionally performed in the cloud environment, utilising cheaper but less-reliable hardware. However, it's becoming more integral part of HPC workflows, utilising tools and methodology from Data Science (DS) and Artificial Intelligence (AI) to process data in order to obtain results quickly. AI can be used inside simulations or to steer workflows, while data science can be used to find interesting patterns inside the data. This branch should cover concepts and tools required for effective data analysis on HPC systems. The current basic level sub-branches include: **BDA1-B** *Theoretical principles of Big Data Analytics*, **BDA2-B** *Big Data Tools in HPC*, and **BDA3-B** *Integrating BDA with HPC workflows*. A practitioner familiar with this branch should be able to: describe and apply the concepts of artificial intelligence and data science; differentiate the various tools that could be used in an HPC environment effectively; and design a workflow consisting of HPC and BD tools to analyze the data.

### 2.6 System Administration

The administration of HPC systems requires integrating the state-of-the-art hardware and software at various stages of their life-cycle, to provide optimal environments for the users while managing them efficiently. This branch should cover the concepts and tools enabling efficient and cost-effective administration of HPC systems. The current basic level sub-branches include: **ADM1-B** *Cluster Infrastructure*, **ADM2-B** *Software Stack*, **ADM3-B** *Monitoring Tools*. The practitioners familiar with this branch should be able to: understand the differences between different hardware options; apply best practices for managing software and users; monitor the system and software usage; manage and maintain the optimal environment for users; and know how to establish the support structures.

## 3 HOW TO CONTRIBUTE

Ultimately, the chairs of the respective skill sub-trees will be in charge of curating suggestions and change requests. However, direct change requests are adopted for the phase of building the first release version of the Tree. The MindMap and Skill definitions are available in Markdown format[2] and a wiki is available to render them directly online. The skills are structured in directories according to the hierarchy in the skill tree. The MindMap structures are synchronised with the tree directory to test more invasive changes.

Contributions to the skill definitions can be made by 1) discussing them on Slack (it is a good idea to talk through non-trivial changes), 2) adjusting the skill-tree in the MindMap (editable via *FreePlane*), 3) editing the skill definitions on the Wiki, or 4) directly preparing a pull request that changes the Markdown files. As GitHub allows for commenting of individual lines, this provides means for rapid feedback as well.

## 4 CONCLUSIONS

The *HPC Certification Forum* is an effort to structure the HPC-related skills and to offer certification to users. The high-level descriptions of the sub-trees provide an overview and indicate our goal of creating a comprehensive tree. The tree itself will be released in stable versions (and version controlled) and updated periodically. We work towards the first release but need the input of the community to refine and complete, particularly the leaf-levels of the skill-tree.

## ACKNOWLEDGMENTS

We thank the contributors to the HPC Certification Forum.

[2] See our GitHub https://github.com/HPC-certification-forum

January 2020

Volume 11 Issue 1

ISSN 2153-4136 (online)