January 2019

Volume 10 Issue 1

JoCsE

Journal Of Computational Science Education

Promoting the Use of Computational Science Through Education

ISSN 2153-4136 (online)

Volume 10, Issue 1 **January 2019**

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 10 Issue 2: 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 disciplines as well as broader 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 Strategies for Enhancing HPC Education and Training (SEHET18), which was held in conjunction with PEARC18 in Pittsburgh, Pennsylvania, U.S.A., July 25, 2018 and the Fifth SC Workshop on Best Practices for HPC Training and Education (BPHTE18), which was held in conjunction with SC18 conference in Dallas, Texas, U.S.A., November 12, 2018.

This special issue begins with an article by Ponce et al. that presents tools, techniques and a methodology for developing curriculum courses aimed at graduate students in emerging computational fields, including biology and medical science. The teaching methodology used focuses on computational data analysis and statistical analysis, while at the same time teaching students' best practices in coding and software development. They found significantly good evaluation results especially in the Institutional composite questions. Those include items such as an intellectually stimulating course, provides a deeper understanding of the subject matter, learning atmosphere, and overall quality of the learning experience

The article by Stieber describes the curricula of a 10-week advanced computational inorganic chemistry course based on the course-based undergraduate research experiences (CUREs). The students used the computational resources enabled by the National Science Foundation's Extreme Science and Engineering Discovery Environment (NSF XSEDE) to conduct independent research projects following in-class lectures and tutorials. They conclude by highlighting the impact of the program on the students and its relevance to workforce training.

The article by Coulter et al. presents the XSEDE National Integration Toolkit that provides the software used on most XSEDE systems in an effort to enable easy knowledge and best-practice transfer from XSEDE service providers to campus cyberinfrastructure (CI) professionals. They describe the Use Case based approach for the creation of the XNIT Toolkit to provide campus cyberinfrastructure administrators a way to include software that is commonly found on XSEDE resources, without having to do a complete reinstallation of operating system that the XCBC cluster distribution would require. They conclude by discussing more flexible, extensible solutions for scientific software delivery for the XNIT toolkit to continue providing a useful solution to easing the pain of administrating a general HPC-style resource at the campus level.

The article by Fietkiewicz et al. presents a comparative study of student's experiences in a high performance computing course. They utilized a recursive matrix multiply as an exercise for designing a parallel program and to have students use an application that contrasts with a previous experience using the iterative, loop-based algorithm. They conclude by reporting on student outcomes with respect to their prior multithreaded programming experience and on the common errors that included recursively creating excessive threads, failing to parallelize all possible mathematical operations, and poor use of OpenMP directives.

The article by Ponce et al.. surveys the current academic and non-academic programs across the globe, and presents quantitative evidence demonstrating the need for programs in higher education in High-Performance Computing and Data Science. They focus on Canadian programs and specifically on the education program of the SciNet HPC Consortium, using its detailed enrollment and course statistics for the past six to seven years. They conclude by illustrating SciNet's path in developing and transgressing the usual role of training events for users, into full credited graduate courses recognized at the university level for masters and doctorates degrees.

The article by Destefano and Sung describes the external evaluation activities in the first three years of the Blue Waters (High Performance Computing) Community Engagement program for graduate fellows and undergraduate interns. The evaluation utilized the 'Educative, Value-Engaged Approach' and conducted formative and summative evaluations to improve the programs and activities based on continuous feedback, while collecting appropriate data and information to conduct a longitudinal analysis of the impact of the programs over the life of the project. They report that the evaluation plans, activities, findings significantly affected the fellowship and internship program implementation, and program impact.

The article by Chakravorty et al. describes an informal curricular model of short, intensive and applied micro-courses offered at Texas A&M University High Performance Research Computing (TAMU HPRC) that address generalizable competencies in computing as opposed to content expertise. The model used a number of interventions that have been systematically applied on a semester-by

semester basis for greater visibility for the courses, engaging students with active learning methods and retaining student interest via the HPRC seminar series. They found number of factors that amount to 300% growth in participation in HPRC short courses and provide a longitudinal report that assess the success of strategies implemented by TAMU HPRC to promote cybertraining efforts across campus.

The article by Nunez et al. presents different learning methods and tools employed by Pawsey Supercomputing Centre to address specific educational and training purposes. They used various techniques ranging from traditional on-site training, through self-guided training materials to online training and webinars and an open repository of materials, covering different aspects of HPC systems usage, parallel programming techniques as well as cloud and data resources usage. They found through numerous feedbacks that there is no universal learning solution; instead, various solutions and platforms need to be carefully selected for different groups of interest.

The article by Zon et al. describes the changes in the training and educational efforts the Canadian academic high performance computing center in the areas of scientific computing and high performance computing over the last six years. They used enrollment data, attendance and certificate numbers to report on trends on the growth, demand and breadth of the Scinet's training program. They conclude by highlighting the results of the assessment which is a steady increase in the demand for the data science training and wider participation of 'non-traditional' computing disciplines.

The second article by Chakravorty et al. describes a conceptual framework to evaluate and explore different pedagogical approaches that utilize technologies to introduce learners to complex programming scenarios suitable for the intermediate level. They provide quantitative and qualitative evaluation of three distinct models of programming education: (i) connect coding to hands-on "maker" activities using Raspberry Pi's; (ii) incremental learning of computational thinking elements through guided exercises that use Jupyter notebooks; and (iii) problem-based learning with stepwise code fragments leading to algorithmic implementation. They conclude by reporting on the student assessment outcomes that involve using the Jupyter notebooks to accelerate student learning with coding concepts.

The article by Mullen et al. describes the applicability of Massively Open Online Courses (MOOCs) for scaling High Performance Computing (HPC) training and education. They used two MOOC case studies, including the design decisions, pedagogy and delivery to outline how MOOC courses differ from face-to-face training, video-capturing of live events, webinars, and other established teaching methods with respect to pedagogical design, development issues and deployment concerns. They conclude by reporting on best practices for segmenting content into smaller concept sized chunks for addressing HPC specific technical needs and concerns.

The article by Bisbal et al. presents training guidelines for building open-source software to address common problems encountered in the scientific community members when developing their own codes and building codes written by other computational scientists. The article outlines topics that are needed to be taught to computational scientists in a logical order to train them to build open-source software. The article concludes by providing references to some of the topics that could be used to develop training materials or distributed directly to students as part of the training materials for bridging a major skills gap for the computational scientists.

The article by Seo et al. describes the design of the Cyberinfrastructure Security Education for Professionals and Students (CiSE-ProS) system. They used engaging approaches to evaluate the impact of learning environments produced by augmented reality (AR) and virtual reality (VR) technologies for teaching cybersecurity concepts. They conclude by reporting on the successes and feedback of CiSE-ProS virtual reality (VR) program using the pilot study with high school students at the Summer Computing Academy at Texas A&M University.

The article by Kunkel discusses the preliminary design of the HPC Certification Program and an independent body that curate the competencies and issue certificates for the users. The article presents two purposes of the program: defining and organizing the fine-grained skills and the establishment of the certificates and online exams that confirms that the user possess those skills. They conclude by reporting on the HPC certification forum which plays a virtual central authority to curate and maintain the skill tree and certificates.

The second article by Fietkiewicz discusses the experiential differences in student performance and perceptions in the undergraduate level high performance computing (HPC) at Case Western Reserve University. They used six main HPC techniques, which includes: batch job processing, general optimization for sequential programming, parallel programming using spawned (forked) processes, parallel programming using OpenMP and multithreading, parallel programming using OpenACC and GPUs, and parallel programming using message passing and MPI. They found that academic experience was correlated to performance, and technical experience may have no correlation at all, assuming adequate coverage in class is provided.

The article by Chen et al. describes an online education and training platform designed by NVIDIA Deep Learning Institute (DLI) that helps students, developers, and engineers solve real-world problems in a wide range of domains using deep learning and accelerated computing. The new educational platform uses a combination of current online learning andragogy along with a cloud computing platform consisting of a VM and Docker containers. The online platform enables students to use the latest AI frameworks, SDKs, and GPU-accelerated technologies on fully-configured GPU servers in the cloud so that the focus is more on learning and less on environment setup. They conclude by reporting on the project-based assessment and certification offered to the students at the end of some courses and the feedback from the DLI University Ambassador Program offered to educators to teach free DLI courses to university students, faculty, and researchers.

The article by Ngo and Denton describes the approaches in leveraging Cloud-Lab, a publicly available computing resource, as a platform to develop and host materials to support teaching topics in cluster computing. They used two approaches in using CloudLab to teach advanced concepts in cluster computing: direct deployment of virtual machines (VMs) on bare-metal nodes and indirect deployment of VMs inside a CloudLab-based cloud. They found

that the flexibility, availability, and scale of CloudLab bring significant applicability to other topics in computer science, including operating system, networking, and cyber-security.

The second article by Coulter discusses the elasticity and programmability of cloud resources to be used as a tool for educators who require access to a wide range of computing environments. The article presents the Jetstream cloud environment to provide training for both new HPC administrators and users, by showing a ground-up build of a simple HPC system that allows an educator to tackle everything from basic command-line concepts and scheduler use to advanced cluster-management concepts such as elasticity and management of scientific software.

The article by Marques et al. presents the best practices and experiences organizing the Best Practices for HPC Software Developers (HPC-BP) webinar series, an effort for the dissemination of software development methodologies, tools and experiences to improve developer productivity and software sustainability. They report on a model for a number of distinct roles and steps in the process of developing and delivering a specific webinar event. They conclude by reporting opportunities to further enrich the pool of topics related to software productivity and sustainability and further expand their outreach activities.

Bridging the Educational Gap between Emerging and Established Scientific Computing Disciplines

How to teach scientific computing to students in medical sciences and other emerging fields

Marcelo Ponce
Erik Spence
Ramses van Zon
Daniel Gruner
mponce@scinet.utoronto.ca
ejspence@scinet.utoronto.ca
rzon@scinet.utoronto.ca
dgruner@scinet.utoronto.ca
SciNet HPC Consortium, University of Toronto
Toronto, ON, Canada
Institute of Medical Science, Faculty of Medicine, University of Toronto

ABSTRACT

In this paper we describe our experience in developing curriculum courses aimed at graduate students in emerging computational fields, including biology and medical science. We focus primarily on computational data analysis and statistical analysis, while at the same time teaching students best practices in coding and software development. Our approach combines a theoretical background and practical applications of concepts. The outcomes and feedback we have obtained so far have revealed several issues: students in these particular areas lack instruction like this although they would tremendously benefit from it; we have detected several weaknesses in the formation of students, in particular in the statistical foundations but also in analytical thinking skills. We present here the tools, techniques and methodology we employ while teaching and developing this type of courses. We also show several outcomes from this initiative, including potential pathways for fruitful multidisciplinary collaborations.

CCS CONCEPTS

• **Social and professional topics → Computing education**; **Model curricula**; **Student assessment**; *Computational thinking*; *Computing education programs*; Accreditation;

KEYWORDS

Training and Education, Computational Statistics, graduate courses, curricula, student assessment

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

1 INTRODUCTION

In this paper, we present the methods and strategies we have used to expand our traditional scientific and high-performance computing programs into university-curriculum courses for disciplines ranging from physics and biology to medical sciences [5, 10]. We show our steady growth in these disciplines, demonstrating a clear need for approaches like ours, not only for the traditional high-performance computing sciences but also for the not-so-usually engaged disciplines, as shown in Figures 1 and 2. We discuss the methodology we use to teach these non-traditional students.

This paper is organized as follows: in Section 2 we explain the main motivations and goals that we target when designing a course like this; Section 3 describes the basic layout and main elements of the course; Section 4 describes the methodology we use to evaluate and transfer the knowledge to the students, including the different strategies have we used for offering the course; in Section 5 we evaluate the outcomes and discuss future directions and implementations for the course; in Section 6 we draw some final reflections about the approach presented in this paper.

2 MOTIVATION

Proper training in data analysis and statistical techniques is indispensable for modern scientific research. Recent times in particular have seen the adoption of computerized data analysis techniques in all fields (biology, human sciences, medicine, *etc.*). The last decade has also seen the advent of a new era of data availability and scale, with huge amounts of data easily collected and shared among scientific researchers, governments and businesses.

However, the skills and knowledge needed to seize the opportunities presented by these data have, in general, not been taught in university courses. Researchers, in particular graduate students and postdocs, are largely forced to learn these skills on their own, if they learn them at all. The effort to understand basic concepts and overcome the technical difficulties associated with data analysis tools diverts from and delays the main goal, research.

Expecting students to pick up this knowledge and skill by themselves is especially troublesome in fields that do not have a tradition

of doing computational research, such as biology, medical science, health science, humanities, *etc.* Students in those fields cannot turn to their senior colleagues for guidance, something that is common in traditionally computational disciplines like physics, chemistry, engineering and astronomy. We have offered scientific computing courses for students in the physical sciences for many years, but these courses focused on compiled languages, numerical libraries, solving differential equations, and parallel computing. They were found to be ill-suited for most students in biomedical fields both because of required prior knowledge and the mismatch of topics covered. Biomedical computation is more likely to involve statistics, data analysis and interpreted languages, not discretized partial differential equations.

Our recently developed courses aim to fill this gap by training students in the practical application of statistical data analysis, machine learning tools, and professional coding practices. It begins by offering an introduction to the R programming language [6], an open source tool for data analysis that is popular in the medical sciences. Basic concepts and elements of statistical analysis are presented, not only by reviewing theoretical foundations but also by examining examples and applications. Next, statistical methods such as hypothesis testing, parametric and non-parametric model creation, model diagnosis, clustering and decision trees algorithms are discussed and implemented using R, and applied to several real-world examples. This particular course is aimed to graduate students (master and doctorate degrees) from the Institute of Medical Science (IMS) at the University of Toronto.

In previous years, we offered a very successful subset of the current course, in a modular format. One important conclusion we drew from students' feedback and comments was that the students would definitely benefit from a more comprehensive and extensive program, incorporating more advanced topics and extending the duration of the course. Hence the latest iteration of the course spans a full semester. The course is in high demand: in the last year (fall 2017, winter 2018) we delivered this course in two consecutive terms and the registration was above the original number of spots reserved for the course, resulting in the creation of a long waiting list each time and an increase of 33% in the planned size of the class for the next year.

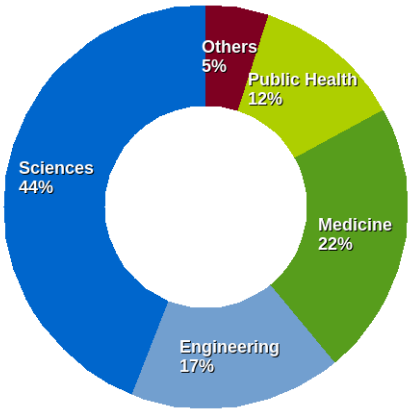


Figure 1: Distribution of student's departments/institutions in all SciNet courses, period 2013-2017. As can be seen traditional HPC disciplines, such as STEM, still constitutes the majority (roughly 60% of our trainees). However, biology and medical students are quickly catching up in numbers representing approximately 35% and with an increasing trend in the last years [13].

3 COURSE DESIGN

The goal of this class is to prepare graduate students to perform scientific data analysis. Successful students learn how to use statistical inference tools to gain insight into their data, and are exposed to cutting-edge techniques and best practices to store, manage and analyze data. We use the R Statistical Language [6] to teach the students the basics of programming and how to perform proper data and statistical analysis.

We focus on four main areas in the course: i) computer programming techniques, including: basics of programming, functions and arguments, documentation and well commented codes and scripts; ii) software development best practices, such as, modularity, version control and proper file IO operations; iii) implementation of statistical analysis techniques and pipelines employing the programming skills transferred in the previous points, *i.e.* computational statistics; and iv) advanced and cutting-edge statistical analysis, including machine learning and neural network implementations.

By the end of the course, the students should have developed basic programming skills and created a set of tools and scripts that help them analyze and tackle their own datasets and research problems.

3.1 Course Content

The course is given in twelve weeks, with two 1-hour lectures per week. Grading is based on 10 assignments set throughout the course, and further discussed below. The course is open to all graduate students at the university but there are a limited number of spots (currently 60), that were all filled in the past.

The list of topics covered includes:

- Introduction to the Linux Shell. File manipulation, regular expressions, bash scripting, automation of data-analysis pipelines.
- Introduction to programming with R. IDEs and R standard console, basic programming concepts: conditionals, loops, variable types.
- Introduction to programming best practices in R. Functions and scripts, interactive versus batch processing, variable scope, modular programming, defensive programming, comments and documentation.
- Introduction to version control. Motivation, implementation and use of version control, GIT, logs, rollback, branches.
- Binary file input/output. Accessing, reading and writing binary data, file input and output strategies, and best practices.

- Basic statistics using R. Review of the basic concepts of probability and statistics, probability distributions, descriptive and inference statistics. Statistical modelling implementations using R: linear models, quadratic models, generalized linear model. Testing models: correlation and covariance, Pearson coefficient. Hypothesis testing: examples of null hypothesis tests implementations, T-Student test, two sample T-test, matched pair experiments, independence tests, ANOVA-like based tests. Model Diagnostics: graphical tools, leverage, influential points, Cook's distance, residuals, validation of assumptions.
- Advanced statistical topics: Generalized linear models, power analysis, survival analysis, structural modelling equation, etc.
- Statistical discussion of some important "paradoxical" cases: Simpson's paradox, Anscombe's quartet, ...
- Introduction to machine learning. Regression, overfitting, bias-variance tradeoff, cross-validation, bootstrapping, LOESS, LOWESS.
- Advanced machine learning. Variable selection, dimensionality reduction, principal component analysis.
- Classification algorithms. Decision trees, confusion matrices, clustering, logistic regression, Naive Bayes.
- Introduction to Neural Networks. Motivation. Basic examples and implementations in R.
- Visualization of data. Publication-quality figures, basic plotting, 1D (curves), 2D (contour maps, heatmaps, dendograms, etc) and 3D plots, interactive visualization, animations.
- High-performance R. Memory management, in-core processing, byte-compiling, C++ interfaces, parallel techniques.

Examples and assignments, presented and discussed within the course, cover study cases based on clinical trials, drug tests, medical cases and hospital treatments, differential gene expression, bioinformatics and *omics techniques, etc.

3.2 Prerequisites

Students should ideally have some light programming experience in any language, and a bit of command-line experience is a plus. Students should have a laptop to bring to the lectures, with R installed, which is freely available for Linux, OS X and Microsoft Windows. We have noticed that due to the way we are able to deliver the course even students with no previous experience in coding are still able to follow and succeed in the course; however their dedication and time commitment might be a bit higher than for other students with a background in computing. Initially, we assumed that because this was aimed as a graduate course, students would have taken previous courses in statistics, however not all of them have a solid foundation or have even taken a recent course on basic statistics. Therefore, we decided to add as prerequisites for the course some basic knowledge on and exposure to at least one statistical course.

3.3 Passing Requirements & Grading Scheme

Most weeks, students are given a programming assignment, with a due date one week after. These assignments are designed to help students absorb the course material. There are 10 assignments in total. The average of the assignments makes up the final grade. To ensure a timely reporting of student grades, we adhere to the following policy: homework may be submitted up to one week after the due date, at a penalty of 0.5 point per day, out of the 10 points for each homework assignment. All sets of homework need to be handed in for a passing grade, although a make-up assignment can be given at the end of the course. Rather than focusing on the topic of a specific week, the make-up assignment may involve any of the material covered in the course.

Attendance is not mandatory for the course, but strongly recommended. This constitutes an important departure with respect to other courses offered at the university level, in particular for IMS students. Because the way we deliver our lectures and we offer the material to students (see next section) students have the flexibility of attending or watching our classes remotely.

4 METHODOLOGY

4.1 Strategies

One of the main challenges of formally implementing and offering this course was to make it available to students across the university as a listed graduate course. The difficulty originates from the fact that SciNet, the supercomputer department of the University of Toronto and the home department of the instructors, is not a teaching department.

One of the most efficient ways we found to overcome this was partnering with other departments or institutes at the university level (*e.g.* the Institute of Medical Science, Physics Department, Department of Physical and Environmental Sciences). By doing so, we provide a formal framework for the course, allowing students to enroll through the official university system, thus being recognized in their official transcripts, *i.e.* taking the course for actual university credit. At the same time, by having the course listed on the official calendar, the course is also visible to students from any other departments at the university, thus increasing its exposure to the graduate students.

With respect to strategies related to the consolidation of knowledge and application of concepts, an approach we usually employ in some of the assignments is to ask the students to *use their own data*, if it is the case that they have data which is suitable and available to be used in the assignment. This approach has several advantages, on the one hand, it allows the students to make direct contact with the techniques described in class and immediately apply them to their own research fields. It enables students to use the tools in a more friendly and familiar environment, as it is basically their own research questions and problems. It also demonstrates to the student the efficiency and capabilities of the techniques and tools we teach, and how they can be properly applied to their own research and problems. It has the tremendous advantage of allowing us to gain some insights of what the students struggle with in their day-to-day work, what type of questions they are trying to answer, in short what their research is about. Moreover, we may be able to help students develop actual tools that they can then use and bring into their corresponding labs and groups. The only downside to this sort of assignment, as these are quite open, is the grading itself. In this case we don't have tentative solutions that we can offer to the TAs (see below), but we provide very detailed guidelines. Similarly,

we also ask students to focus on particular techniques we discussed in class. Although they are welcome to use others not introduced in the course, they need to explain those in a brief report following a traditional paper structure, which also constitutes part of this type of assignment.

In a similar vein, we ask students to create scripts that allow them to produce professional/publication quality figures. At the same time that we evaluate students' understanding of the material, we empower them by creating tools that will be of utility when they need to generate plots for their own papers or research projects. To show off some of the remarkable results we have obtained, we have created a *Visualization Gallery* [14] where we post and display figures, graphics and animations created by the students of our courses.

Another crucial point that touches upon the partnership with other departments and institutes is the necessity of having *teaching assistants* (TAs) helping with the grading of the assignments. The importance of having TAs has been shown in many situations, *e.g.* [9]. In particular for courses with this number of students (approximately 60 per term) and 10 assignments per course, meaning around 600 assignments to grade. Furthermore, the way we grade the assignments is not automatically or in bulk, *i.e.* we do look at each assignment individually, look at the logic of the implementation, whether it works, if it's logically correct, and we provide individual feedback and detailed comments for each student. This is quite labour intensive, and hence not doable without the support of the TAs, which are the main graders of the course. The fact that the TAs are financially covered by the departments sponsoring the course, not only allows us to provide speedy turnover in the comments and feedback to the students, but also to substantially increase the number of students we can accept into our courses. For a course with 60 students we utilize 3 TAs, who are current MSc or PhD students. Ideally we expect the TAs to have experience with programming in general and specifically in R, Linux OS and command line terminal, version control systems (such as GIT), and being knowledgeable in statistics. Not surprisingly, excellent candidates for these TA positions are often students who have taken our course before, as they have being trained on the good practices we want to emphasize, have received the type of feedback we want them to provide to fellow students, and they have experienced first hand the course and the whole evaluation process.

The TAs' main duties are marking and grading 10 weekly assignments, i.e. evaluate assignments and provide comments and feedback. We also recommend that the TAs attend and review the content presented in the lectures (12 lectures at 2 hours per week) and spend some time in preparation for the grading (review topics covered in class and get familiar with the material). We usually host weekly meetings with instructors to discuss assignments and grades. In general, depending on the department a workload like this is given between 70 and 120 hours of work per TA.

As mentioned before, neither a fast feedback nor the size of the classes would have been possible without the support of the TAs, and this can be easily seen in Figure 2; each time a new partnership has emerged there is a bump in our enrollment numbers. However to make things work smoothly there is a lot of logistical work that has to be done, ranging from the usual paperwork for hiring and selecting candidates, up to the most important part related to the grading itself. In particular, we do provide the TAs explicit guidelines, grading schemes and tentative solutions for each assignment. We try to anticipate and cover all the basic mistakes and important concepts the students could face on the assignments. This is a lot of work upfront, but it is totally worth it for the number of students we have to handle. Of course, sometimes we have to look at a particular student's submission or assignments, as there are always corner cases, but this is still a manageable amount.

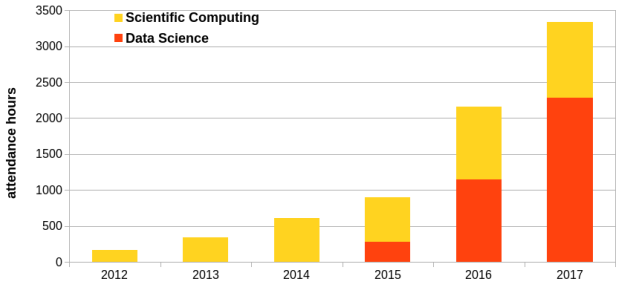


Figure 2: Aggregated attendance hours on "Data Science" and "Scientific Computing" graduate courses in the period 2012 to 2017. The effects of size increment, mostly due to the ability of being able to handle more students can be clearly seen in the years 2016 and 2017.

4.2 Lectures

The type of knowledge we teach in our classes is rarely found at university-level courses and is quite sought-after. In many cases and disciplines, students and even researchers spend a good amount of time self-learning many concepts we include in our "best practices" topics, such a modularity, testing and developing strategies, defensive programming, version control, etc. To instill such best practices in a graduate course setting, we have found that one of the most efficient ways to deliver our lectures is with a combination of theoretical concepts and practical examples and applications. Depending on the particular topic to be presented, we emphasize a more practical/hands-on approach during the class – for instance, when covering topics such as introduction to Linux Shell, exploring the basics of the R language, visualization, *etc.*. However, if the topic to be covered requires some theoretical background, we then deliver a more traditional lecture – *e.g.* reviews on probability and statistics, introduction to machine learning and neural networks, etc. In general, even when we are discussing and presenting theoretical concepts and examples, we still mix practical implementations of such concepts using the computer, so that the students can actively follow the computational implementation during the lectures.

We usually face the question whether we would prefer to deliver our lectures in a computer-lab or traditional classroom. Having a computer-lab based classroom offers the advantage of controlling the workstations and computers, as well as the installation of the software and libraries to use. However, we prefer that the students use their own laptops instead, so that they actually have to go through the process of installing, fixing and sometimes troubleshooting problems dealing with their own computers and installations. This comes back to the point that our courses are aimed

to deliver practical skills, so that the students sitting in our classes can help others in their labs and groups, after having transited that path before with our guidance and support. There are some features that are desirable in traditional lecture rooms: spacious locations are preferable, so that students can comfortably place their laptops and there is room for the instructors to help and move around, especially for the lectures when there is an important hands-on component. As students will be following along, features such as power bars and outlets are desirable, as well as a reliable wireless internet connection. Our lectures are delivered mostly using slides so having a projector is crucial, but often involve the blackboard as well to clarify or demonstrate some concepts or examples. For big rooms, having the support of AV systems, in particular microphones and speakers is important as well.

4.3 Tools

Online resources: All our material (slides, assignments, lecture recordings) is freely available online in our *education website* [12]. We keep all our past courses, lectures, slides, assignments and recordings available on their respective websites[1].

Our educational website [12] is developed based on the open source web-based learning content management system ATUTOR [1].

One resource that students appreciate very much is the fact that we *record* all our lectures and post them on the course website. This allows students to access the material even if they were not present in the class or if they want to review and revisit some of the topics covered in class. In some cases we also *live stream* the lectures if there is demand, with a live chat to answer remote questions as supported by the ATUTOR[1] framework. For recording and streaming the lectures we use a set of mostly open source tools[2] in combination with our own setup for streaming video [15].

Another useful and practical online resource, that we offer is an online *forum* system (also provided by ATUTOR), where the students can post questions, see questions posted by other students and even answer the questions, in other words, start a conversation among their own peers. We, the instructors and TAs, also keep an eye on it, and try to answer their questions there too. One nice feature that the system offers, is that the users can register to get email notifications when new posts are added in the forum.

Another important online contact resource we offer to the students is *email*, this is by far the online resource most used by the students. For instance, in this last year's edition alone, we have answered around 4000 emails. It is quite challenging to keep this under control and balance, but we know this way of interaction is greatly appreciated by the students. We personally, as instructors, like to see and answer every email, as that also provides us with important insights and diagnostic information about what topics the students are struggling with the most. In some cases, when we detect some particularly problematic issues, we can take quick measures to help alleviate the problems. For instance, we can clarify questions or address particular points during the classes, and even in some cases if it is more of a technical issue, we would create a post or discussion item in the course website.

We have also found that a crucial part of the learning process and something students truly appreciate are our weekly *office hours*, which students can attend to pose questions either on particular topics covered in class or to get help while they work on their assignments. Not surprisingly, we have found that just one hour is not enough and we usually find ourselves extending the period or even staying for around 30 minutes after class discussing and answering questions from students.

4.4 Evaluation Methods

Our main evaluation avenue in the course are quasi-weekly assignments. The reason we prefer assignments over mid-terms and/or finals is because we think that having almost one assignment per topic covered offers us the possibility of evaluating with much more fine granularity the knowledge gained by the students. It also offers the students the opportunity to practice the concepts discussed in class. Moreover, the type of knowledge we try to transmit in our lectures is applied/practical by its own nature. Thus having the students implement something by themselves is the best scenario we could think of to reinforce the learning and concepts presented in class. The assignments are designed with two major objectives: 1) to offer the student a chance to practice the most relevant parts of the techniques or concepts discussed in class, and 2) to challenge the students to digest and think beyond the material that was presented, presenting problems in which they need to join different techniques to arrive at new results. Usually we like to set up the assignments with a sort of "hidden message", a learning opportunity, something the students can discover by themselves by following specific guidelines and clues that we leave for them. In this case we believe this self-discovery process is much richer than the knowledge one can transfer in any sort of direct or explicit message delivered in lectures.

This way of evaluation posses quite significant challenges [4]: coming up with actual assignments that fulfill such a role, find the suitable sweet spot of being interesting but not too hard to overwhelm the students, and still be amenable to grading. However one of the most difficult challenges to take into consideration when having this type of homework is being vigilant of students sharing solutions or working on the same code/submission, in other words any sort of plagiarism. We encourage students to openly discuss with peers, but we strongly enforce individual work. No collaborative work or submission is allowed under any circumstances. We strictly follow the university's "Code of Student Conduct"[3] regarding plagiarism. In this particular case, this challenge is even harder [7, 8], as the students are submitting programs, scripts, in many cases just pure code. So in order to tackle this issue, we have in place a series of tools. When possible we actually run some scripts we developed with the goal of identifying substantial overlap in the submitted assignments (similar tools exist for students when writing essays or papers –see [2]–). If there is just one TA grading a whole batch, usually in smaller class sizes (upto 20-30 students), then we ask the TA to be vigilant about this type of situations and

[1] See, for instance, hyperlinks for our previous editions: Fall 2016, first module version part of the Translational Research program, Fall 2017, first full course edition, Winter 2017/2018, second full course edition.

[2] Open Broadcaster Software: https://obsproject.com, Camtasia: https://www.techsmith.com/camtasia.html, ActivePresenter: https://atomisystems.com/activepresenter.

[3] http://www.governingcouncil.utoronto.ca/Assets/Governing+Council+Digital+Assets/Policies/PDF/ppjun011995.pdf

can warn us about any suspicious submissions. In the case of larger courses, we can basically take two approaches: i) Have one TA doing the grading for one whole assignment; this has the benefit that this one TA can see all the submissions at once and identify any potential overlapping assignments. It also helps to normalize the grading criteria – even when we provide concrete and precise guidelines and grading schemes, each TA has also his own style, a fact reflected in the assignment feedback. However, this approach has its own disadvantages too, as it takes longer to give feedback to the students, and it is also more sensitive to grader bias. ii) A second approach, which is the one we follow for larger courses, is to equally divide the number of submissions among the TAs and have all of them working in parallel grading a subset of the assignments. This method allows us to achieve impressive turnaround times, of no more than 48 hours! *i.e.* on average two days after the students had submitted their assignments they got feedback, comments on what they did correctly and what to improve for future assignments. In order to minimize biases here, we randomize the list of students to be graded by the TAs. The disadvantages of this approach are: that the grading score is not quite normalized as there are different TAs grading at the same time (we believe that this is a really minor point, mostly because the precise grading instructions we provide to the TAs leaves little room for that, and if any particular issues are noted we are ready to intervene); and secondly, it is harder to catch situations as the ones mentioned before, however we have still been able to identify and detect cases of suspect plagiarism. After dealing with the incidents in question, the subsequent assignment submissions from those students are graded by the same TA to prevent recurrence.

In other courses we also use some online quizzes that allow us to quickly evaluate, with multiple choice questions, basic concepts the students should assimilate. The procedure is completely automated and included in the features of the ATUTOR [1] web-platform that we use in our education website, hence giving us rapid access to results and diagnostics. This particular technique, due to its automated nature, can be easily implemented in courses with large numbers of students. A variation of this technique had been employed in the shorter precursor of the current course, where it was used to take attendance live during class with minimal disruption, with approximately 100 students in class.

Finally in more-advanced courses, we also employ research based projects, which include having the students working on a particular project, submitting a preliminary report, a final report and a presentation describing the project to the rest of the class by the end of the course. This technique even while quite powerful and interesting, is more desirable and applicable to more mature students, having solid foundations and clear understanding of what the goals of the project should be. As the projects can grow in complexity and significantly change as the projects evolve, these are sometimes quite close to actual research explorations the students are pursuing (e.g. [3]), hence one needs to closely follow the evolution of the students and the projects. Because of this very same reason, this technique is probably not suitable for large classes, and if the class size is above the desirable number of projects/students to follow, or if the project appears to be too complex, partnering students in groups could be a good way to accommodate those situations.

5 DISCUSSION

5.1 Outcomes

There are several ways in which we can aggregate the outcomes from this type of courses. From observations during the course, assignment evaluations, and interacting with students during the office hours, we were able to detect a few weaknesses related to some particular areas. For instance, we noticed that in some cases, beyond the obvious differences in academic formations and backgrounds, there are some serious weaknesses that academic programs could and should target in incoming students. Among the most concerning are weak analytical and critical-thinking skills, and insufficient mathematical and statistical foundations and ability to understand concepts. This is of course worrisome but clearly provides some important information and indicators for academic program designers to take into consideration. From a more technical side, one of the most challenging topics for the students to assimilate is the concept of *functions*, arguments and return statements. Students were able to understand modularity, and even functions as elementary blocks in a modular framework, but the passing and receiving of arguments and/or returning information from functions into new parts of the code was probably the hardest concept many of the students dealt with. Of course, it is arguable that this could be one of the expected struggling points due to the abstract nature of the concept. However, this is also very important information for our future editions of the class.

Another interesting observation is the effect of "early dropouts". These are students that dropped out of the course very soon after it started, between the first and second week, sometimes even before the first assignment was posted. These early dropouts constitute not more than 10% of the class size, which in practice does not pose a problem (the enrollment was so high that we had a waiting list for students that did not initially get a spot). Interestingly, we speculate that this effect can be mostly due to a couple of causes: i) either the course was not what the student was expecting; ii) the workload demanded by the course might have produced a negative impression on the student (however as we argue later in this section it is quite the opposite); iii) the course structure was not appealing to the students.

One quantitative indicator of how well the course is perceived among students is course evaluations run at the university level. These surveys are optative, and the students can decide to complete them providing anonymous feedback about the course. The questionnaire has two major components: a series of standardized questions where students can pick numeric values ranging from 1 to 5, representing 1: "Poor", 2: "Fair", 3: "Good", 4: "Very Good", 5: "Excellent". The second part of the evaluation is composed by open-ended questions that allow students to provide more detailed feedback about the instructors, the level of assistance during the course, and overall quality of the course.

The first remarkable result is the level of participation in theses surveys, especially considering that these are optional. The percentage of participation in our courses' evaluations is usually between 57% and 72%.

Additionally for the so-called *Institutional composite* questions, which include items such as intellectually stimulating course, deeper

understanding of the subject matter, learning atmosphere and overall quality of the learning experience, the resulting mean from the evaluations is 4.5/5.0[4].

Another interesting and somehow surprising result is how the students perceive the workload of the course compared to other courses: 3.33/5.0[5]; which represents just a bit above the average workload. In principle one could think that having roughly weekly assignments for almost the whole duration of the course would be a stumbling block, however the students realize that overall, comparing the stress of just one or two instances of evaluations (*e.g.* mid-terms and/or finals) versus a more gradual evaluation, the latter is comparable or even preferable.

In addition to this, we could literally fill pages with testimonials about the level of instruction, support and professionalism, these courses offer, unfortunately we don't have a good way to measure this rather than just through anecdotal notes.

But perhaps the most outstanding quantitative result is that 97% of the students got an "A" (i.e. their average was above 85%). This again provides support for an evaluation approach based on assignments. As argued before, not only does this allow students to digest the material and implement it in a practical fashion, but also in the end helps them learn and assimilate this knowledge, which is a more "fair" and useful measure of success in the course.

One really interesting byproduct of these courses, is the potential for establishing research collaborations among different groups and labs in need of more robust scientific computing implementations. This is particularly true among non-traditionally computational scientific disciplines like the ones this course targeted. During the course, many students approached us with open problems and potential avenues for collaborations. One of the most recent demonstrations of this, is an ongoing research project where in collaboration with microbiologists and biochemists, we developed a bioinformatics pipeline employing traditional HPC resources and open source tools, which we expect will produce at least three publications, the first one being already published [11].

Another demonstration of the success of this course, is the remarkable interest shown by the students in becoming TAs for next year's course. By the end of every single term, we had students approaching us, asking about the possibility of TAing for the course in future editions. Somehow, students with a natural inclination discovered the enchanting realm of scientific computing and seek the opportunity of becoming active participants in the field. As educators, this kind of outcome is just priceless.

5.2 Future Directions

In order to further improve our teaching and student learning experience, we continue to develop new ideas and avenues to facilitate the transfer of knowledge and consolidate the assimilation of basic and foundational concepts. One way we think we can help students digest and familiarize themselves with new or difficult-to-assimilate concepts is the development of customized lecture notes for the course, in addition to the already available slide decks. We have noticed that it is quite challenging to find resources that we can refer the students to for further reading, as on the one hand it is difficult to find references that cover the variety of topics we tackle in this course and with the depth we try to achieve; and on the other hand the wide range and heterogeneity in the students' backgrounds make it even more challenging.

We are also considering implementing an additional evaluation requirement for passing the course, consisting of an online quiz to be carried out by the middle of the semester. The weekly assignments will continue being our main source of evaluation, however we have noticed that students either by a lack of comprehension or not performing the exercises in a mindful way, sometimes miss important concepts. The in-class multiple choice quiz will help us further diagnose difficulties in specific topics/areas and also make the students aware of their own weaknesses. We have implemented similar evaluation procedures for other courses, and we find them easy to implement and evaluate using the online platform [1] we use for our education website.

[4]The detailed statistical indicators are: mean=4.5, mode=5.0, standard deviation=0.22 over a period of 2 years –2017/2018–.

[5]In this case the numeric scale is interpreted as follows: 1: "Very Light", 2: "Light", 3: "Average", 4: "Heavy", 5: "Very Heavy".

6 CONCLUSIONS

In this manuscript we described the road we followed in order to create a graduate course aimed for non-traditional scientific computing students. We believe the strategies, partnerships and methodologies we present here can be useful for others to bridge the gap between traditionally computational disciplines and disciplines that are new to computing. Our approach is also different from the traditional standard university courses, but has proven to be successful in reaching and providing new and useful tools to students, scientists and researchers. Furthermore, having the chance to directly interact with students we were able to identify some important concepts that students were missing and diagnose some crucial weaknesses which graduate programs should tackle. Last but not least, providing this type of courses, not only offers benefits to the students learning new skills, but it is also a way to catalyze and push frontiers in new multidisciplinary research fields, instigating in this way collaborations that might not have been possible otherwise.

ACKNOWLEDGEMENTS

We want to thank all our colleagues at SciNet, and the many departments that have partnered with us at the University of Toronto: Department of Physics, Department of Physical and Environmental Sciences at UTSc, and the Institute of Medical Science (IMS). Special thanks to the IMS, for allowing us to offer this course, and in particular Prof. Howard Mount, Michelle Rosen, Sarah Topa and all the members in the IMS' Curriculum Committee, for their confidence and continuous support during these last 3 years of working together. Last but not least we want to recognize and thank our amazing TAs: Ricardo Harripaul, Sejal Patel, Parnian Pardis and Jonas Osmann, for their outstanding work and commitment to the course.

REFERENCES

[1] ATutor. atutor.ca. (????). Accessed: 2018-04-15.
[2] Turnitin. http://turnitin.com/. (????). Accessed: 2018-04-19.
[3] Philippe Berger and George Stein. 2018. A volumetric deep Convolutional Neural Network for simulation of dark matter halo catalogues. (2018). arXiv:astro-ph.CO/1805.04537

[4] Nick Parlante, Steven A Wolfman, Lester I McCann, Eric Roberts, Chris Nevison, John Motil, Jerry Cain, and Stuart Reges. 2006. Nifty assignments. In *ACM SIGCSE Bulletin*, Vol. 38. ACM, 562–563.
[5] Marcelo Ponce, Erik Spence, Daniel Gruner, and Ramses van Zon. 2016. Designing Advanced Research Computing Academic Programs. *CoRR* abs/1604.05676 (2016). arXiv:1604.05676v2 http://arxiv.org/abs/1604.05676v2
[6] R Core Team. 2017. R: A Language and Environment for Statistical Computing. (2017). https://www.R-project.org/
[7] Eric Roberts. 1998. Strategies for using technology in the teaching of ethics. In *ACM SIGCSE Bulletin*, Vol. 30. ACM, 209–212.
[8] Eric Roberts. 2002. Strategies for promoting academic integrity in CS courses. In *Frontiers in Education, 2002. FIE 2002. 32nd Annual*, Vol. 2. IEEE, F3G–F3G.
[9] Eric Roberts, John Lilly, and Bryan Rollins. 1995. Using undergraduates as teaching assistants in introductory programming courses: An update on the Stanford experience. *ACM SIGCSE Bulletin* 27, 1 (1995), 48–52.
[10] Eric S. Roberts. 2011. Meeting the Challenges of Rising Enrollments. *ACM Inroads* 2, 3 (Aug. 2011), 4–6. https://doi.org/10.1145/2003616.2003617
[11] Alejandro Saettone, Jyoti Garg, Jean-Philippe Lambert, Syed Nabeel-Shah, Marcelo Ponce, and et al. 2018. The bromodomain-containing protein Ibd1 links multiple chromatin-related protein complexes to highly expressed genes in Tetrahymena thermophila. *Epigenetics & Chromatin* 11, 1 (09 Mar 2018), 10. https://doi.org/10.1186/s13072-018-0180-6
[12] SciNet HPC Consortium. SciNet's Education website. https://support.scinet.utoronto.ca/education. (????). Accessed: 2018-04-15.
[13] SciNet HPC Consortium. SciNet's Training, Outreach and Education. http://www.scinethpc.ca/training-outreach-and-education. (????). Accessed: 2018-04-15.
[14] SciNet HPC Consortium. SciNet's Visualization Gallery. https://www.scinethpc.ca/visualization-gallery/. (????). Accessed: 2018-04-19.
[15] Ramses Van Zon and Marcelo Ponce. 2018. Software-Enhanced Teaching and Visualization Capabilities of an Ultra-High-Resolution Video Wall. *in prep.* (2018).

Student-led Computational Inorganic Chemistry Research in a Classroom Setting

Erica D. Hummel & S. Chantal E. Stieber*
California State Polytechnic University
Pomona, CA
*sestieber@cpp.edu

ABSTRACT

Advanced computational inorganic methods were introduced as course-based undergraduate research experiences (CUREs) through use of the National Science Foundation's Extreme Science and Engineering Discovery Environment (NSF XSEDE). The ORCA ab initio quantum chemistry program allowed students to conduct independent research projects following in-class lectures and tutorials. Students wrote publication-style papers and conducted peer review of classmates' papers to learn about the full scientific process.

Keywords

Computational chemistry, CURE, inorganic chemistry

1. INTRODUCTION

Undergraduate Research Experiences (UREs) have been shown to be beneficial for learning and retention [1], and science persistence rates for those who participate in UREs are 14-17% higher than for those who do not [2-4]. For small teaching institutions there are oftentimes not enough research positions for all students who are interested in UREs, so there is a need to provide other opportunities for students to have similar experiences. Course-based UREs (CUREs) have been proposed as an alternative, allowing research modules to be incorporated into a class to provide research experience to all students in the course [5]. Five specific components of a well-designed CURE include 1) Use of scientific practices, 2) Discovery, 3) Broadly relevant or important work, 4) Collaboration, and 5) Iteration. For a CURE to achieve similar benefits to learning and understanding as a URE, it should include student-led research modules and appropriate mentoring. A lack of mentoring and the short time-scale of coursework are the most common pitfalls in successful implementation of CUREs [1].

In the chemistry curriculum CUREs are oftentimes difficult to implement in a lecture-based course because laboratory experiments require specialized equipment, space, and have additional safety concerns. Computational chemistry offers a unique alternative because the safety concerns are mitigated, and computers can easily be implemented in a standard lecture class. Additionally, if computational chemistry modules are posted on the web, students can access content and conduct research from anywhere with an internet connection. Despite these advantages, there are few publically available computational chemistry CUREs. A semester-long course using WebMO is available [6], and two individual modules for organic and general chemistry have been published by Hope College. [7] Several modules focus on the introductory concept of valence shell electron pair repulsion (VSEPR) theory, so more advanced computational chemistry applications are of interest. Herein, a state-of-the-art quantum chemical computational CURE for an advanced undergraduate and Master's course is described.

DOI: https://doi.org/10.22369/issn.2153-4136/10/1/2

2. STRATEGY

2.1 Course structure

A newly developed CURE module was implemented in the course, Advances in Inorganic Chemistry (CHM 571 and 572), at Cal Poly Pomona (CPP) in Spring 2016 (6 students) and Fall 2017 (17 students). The course met twice a week for 1.5 hours during a 10-week quarter in a classroom with an individual PC computer workstation for each student. The general format of the course included a lecture the first day each week, and hands-on computational tutorials during the second class period. **Table 1** depicts the weekly schedule for the course. Although the course is offered as part of the Master's curriculum at CPP, advanced undergraduates also take the course and most students had not taken an inorganic chemistry course before.

Table 1: Weekly Schedule for Advances in Inorganic Chemistry, Fall 2017

Week	Type	Topic
1a	Lecture	Electron counting
1b	Lecture	Group theory
2a	Lecture	Molecular orbital theory
2b	Lecture	Density functional theory
3a	Lecture	Input files/geometry optimization
3b	Lab	Input files/geometry optimization
4a	Lab	Analysis of results
4b	Lecture	Crystallography
5a	Lecture	Infrared & Raman spectroscopy
5b	Lab	Vibrational frequency calcs.
6a	Lecture	Electron paramagnetic resonance (EPR) and magnetism
6b	Lab	EPR calculations
7a	Lecture	Moessbauer
7b	Lab	Moessbauer calcualtions
8a	Lecture	X-ray absorption spectroscopy (XAS) and time-dependent density functional theory (TD-DFT)
8b	Lab	TD-DFT and XAS calculations
9a	Lecture	X-ray emission spectroscopy (XES)
9b	Lab	XES calculations
10a	Lab	In-class peer review
10b	Lecture	Responding to peer review

The first two weeks comprised of a brief introduction to organometallic and inorganic chemistry to ensure that all students had a basic understanding of metals, d-orbitals, counting electrons, and symmetry. The computational component began in the fourth class meeting with a lecture introduction to density functional theory. For the rest of the quarter, a lecture was generally given the first day of the week to introduce a new spectroscopy and the computational method for calculating spectra.

2.2 Implementation of computational modules

2.2.1 Computational resources

Prior to the start of the course, 50,000 SUs of computational resources on the Gordon compute cluster and 500 GB of storage were requested and received from the National Science Foundation's Extreme Science and Engineering Discovery Environment (NSF XSEDE). On the first day of class, students were instructed to create user accounts as homework using a self-guided tutorial. The classroom contained 30 PC workstations for students to use. All students had personal laptops for conducting research projects from home, however students could also use the classroom computers outside of class.

2.2.2 Programs used

All programs used in the course are free for academic use. The ORCA ab initio quantum chemistry program [8] was used for all computational modules in this course. Additional support programs included WinSCP (PC) and Cyberduck (Mac) for file transferring, Avogadro for molecular visualization and centering, and Chimera [9] for visualization of orbitals. Excel was also used for plotting calculated spectra.

2.2.3 Tutorials

Step-by-step tutorials were developed as text documents with screen shot images to guide students through all of the computational techniques listed in the lab sections of **Table 1**. During class, students worked through the tutorials and the instructor was available to help. The tutorials were posted through the class Blackboard site to ensure that students could remotely access the tutorials as needed.

2.3 Independent research projects

2.3.1. Independent projects

Independent research projects were designed to incorporate all 5 areas of a well-designed CURE. In particular, a goal was to teach students the full scientific process including project proposal, project implementation, paper writing and peer-review. By week 3 of the course, students were required to submit a project proposal for a project that they would complete during the course of the 10-week quarter. In total, there were 7 project components (I-VII) that students turned in. This framework fulfilled CURE area 1) Use of scientific practices.

2.3.2 Proposal

The proposal (component I) included searching the inorganic and organometallic chemistry literature for complexes containing first row transition metals that had been crystallographically characterized. The complex should also have experimental data available for one of the types of spectroscopy addressed in class, but have no computational studies. Students each proposed four complexes, of which the instructor chose one for the student to study. Having the instructor choose the ultimate complex for study avoided problems resulting from systems that would be too complicated. Many students also had difficulty assessing whether or not calculations had been previously published, so the instructor could omit those. This project proposal fulfilled CURE area 2) Discovery.

2.3.3. Independent research

The independent research each student conducted included conducting a geometry optimization, generating a d-orbital splitting diagram (Fig. 1), and calculating one spectrum (Fig. 2). From the geometry optimization, students compared their calculated to experimental bond distances and angles to assess the validity of their computational model. The spectrum they calculated was compared to the experimental data. When students finished the in-class tutorials teaching a new computational method, they had time to work on their independent projects. The rest of the project was completed as homework. Although only a geometry optimization and one spectroscopic calculation were required for the final project, most students practiced each new method on their research complex. The research component fulfilled CURE areas 1) Use of scientific practices, 2) Discovery, and 3) Broadly relevant or important work.

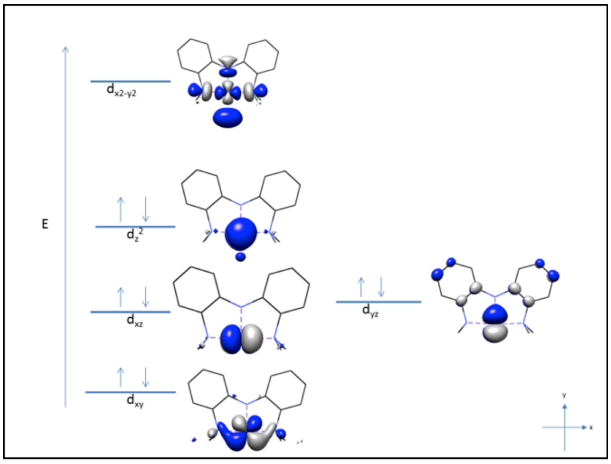


Figure 1: Qualitative *d*-orbital splitting diagram from a geometry optimization generated by a student in the course for ($^{Me}N_2N$)Ni-H [10].

2.3.4. Publication writing

In order to facilitate reasonable progress in the research project and writing, a series of due dates were set for specific project components I-VII. Students were also told that they could turn in assignments early for additional feedback. In week 5, students submitted a geometry optimization check (component II) with their input files and an expected d-orbital splitting diagram. This ensured that errors in coding could be addressed early on. In week 7 the paper introduction and results section (component III) for the geometry optimization were due. This included a comparison of experimental and calculated bond distances and angles, and a d-orbital splitting diagram. In week 8, the methods section and analysis of spectroscopic properties (component IV) were due. In week 9, a final draft (component V) was given to a peer reviewer. In week 10 the peer review (component VI) was due, and in finals week the final project (component VII) was due. This fulfilled CURE area 5) Iteration.

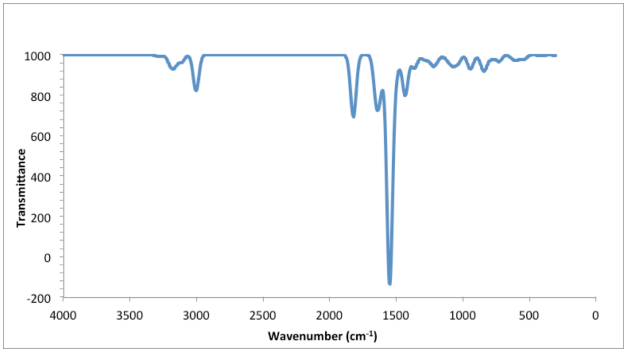


Figure 2: Calculated infrared spectrum for ($^{Me}N_2N$)Ni-H, generated by a student in the course.

2.3.5. Peer review

The final component of the project was to learn about the peer review process by conducting an in-class peer review. The instructor distributed student papers to other students in the course, making sure to omit names and assign papers to classmates who were not sitting near each other. The instructor gave a brief introduction to peer review and utilized a published paper with posted reviewer comments for students to see what real reviewer responses look like [11]. Most students did not actually know what peer review really meant, and how formalized of a process it was. At the end of the class period, students had the chance to ask the instructor clarifications about the paper. Students had one day to write their formal peer reviews so the instructor could return the reviews to the student authors. The following class period, students received their peer review comments and could ask the instructor for advice. The peer review fulfilled CURE area 4) Collaboration, although this CURE area was also achieved throughout the course because students oftentimes worked together on the tutorials.

3. ASSESSMENT

3.1 Assessment strategy

3.1.1. Course assessment

Students performance in the course was assessed through homework, in-class quizzes and the final research project. Since students had varying degrees of chemistry experience and most had not taken inorganic chemistry, the course was designed to be project-based.

3.1.2. Homework

There were eleven homework assignments to give students practice in basic inorganic chemistry. The areas of focus were electron counting and group theory. The group theory assignments were based on completion of self-guided programs using Alan Vincent's book, Molecular Symmetry and Group Theory: A Programmed Introduction to Chemical Applications.

3.1.3. Quizzes

Five in-class quizzes were distributed throughout the quarter. The first three tested basic inorganic chemistry concepts including electron counting and group theory. Because the group theory homework could only be assessed based on completion due to the nature of the programs, quizzes tested the students' understanding of group theory. The last two quizzes tested knowledge of the ORCA input and output files, respectively. The averages for quizzes were typically between 76 – 80% (n = 17).

3.2 Final project assessment

The final project comprised of 40% of the final grade. Of this, the grade distribution for each of the components was 40% for components I-V, 20% for component VI, and 40% for component VII. Components I-V were graded based on effort and completion to ensure that students were learning. This gave students the opportunity to work on their writing and receive detailed feedback from the instructor without worrying about whether the answer was "right" or not. In particular, since research does not always have a defined answer, this was a valuable approach for students to learn and gave them ownership and taught independence. Component VI, the peer review, was a significant component of the grade to ensure that students took the process seriously and wrote thoughtful reviews. Component VI was graded on completion, level of thought and thoroughness. The final project component VII was graded based on quality of research and writing. Most students made vast improvements in their writing over the course of the quarter and submitted research projects that were of very high quality. The average grade for this component was 91% (n = 17).

3.3 Course assessment

3.3.1. Evaluation of success

Students were highly engaged in the course and were very positive in both informal and formal evaluations. Formal university evaluations resulted in scores of "very good" and "good" (the highest possible) for all thirteen categories that were questioned in both years the course was taught. In particular, questions included "instructor presents material in an interesting manner," and what is your overall rating of the instructor in this course?" Informal interactions with students during office hours gave the impression that students had to work a lot, but they felt like they were learning so much that the time was worth it. Students reported being very excited to conduct their own research projects and do work that no one in the world had done before. They also remarked upon their own improvements in writing, and were excited by their progress. Perhaps the most indicative indicator of success of the course was that one out of six students in the first year of the course continued to use ORCA and XSEDE for his own computational research, taught his whole group how to use these, and applied for XSEDE resources with his PI. In the second year of the course, two out of seventeen students joined the instructor's research group and began computational research projects, and one additional student in a synthetic research laboratory used ORCA to calculate the compounds he was making.

3.3.2. Lessons learned

Students were very interested in conducting original research projects and doing calculations that had never been done before. As such, most students were highly motivated to learn and do a good job. It was impressive that students were able to learn such advanced computational techniques over a short period of time (10 weeks) and write such high quality research papers.

What was perhaps most surprising was the limited computer literacy of many students. In the first year of the course only one

student (n = 6) had used the command line, and in the second year none had (n = 17). As such, on the first computational lab day of class, a significant amount of time needed to be spent on basic command line and linux commands. Several students were also not familiar with computer file structures and did not know how to locate the C drive or search for files. Future iterations of the course will include additional command line tutorials to be completed as homework. With regards to writing, at least half of the students were not comfortable with basic formatting using Microsoft Word, including adding footnotes or references and figures. In the future, students will be directed to additional tutorials to learn formatting. The second year the course was taught, there was a TA who had taken the course in the previous year and was of significant help during the computational lab tutorials.

3.3.3. Reproducibility and recommendations for implementation

The course has been taught twice at CPP and student responses to the course were highly reproducible and favorable. The overall structure of the course and research project is a model that could be applied for a wide range of topics. Although computational inorganic chemistry is highly specialized, the general structure of the research project could be implemented for any type of computational project. For example, the research project proposal and writing components were implemented as a literature review project in an advanced Metals in Biology course in Spring 2017. A TA or student helper with command line experience is strongly recommended for classes with more than 10 students.

3.3.4. Relevance to workforce training

This course taught computational inorganic chemistry methods that are used by active research groups throughout the world and are at the forefront of the field. A course of this type is entirely unique and teaches students computational methods that they would normally only learn through a Ph.D. research program. Additionally, this gave students a research experience and confidence in using free software for conducting calculations of chemical systems, which they could apply to a variety of future careers. The strong focus on writing will benefit students in any future endeavor.

4. CONCLUSIONS

Computational inorganic chemistry techniques using density functional theory were taught in a 10-week lecture course through the use of a CURE. The course was taught in a lecture and tutorial lab format, and students conducted original self-designed research projects. The research projects included a proposal, a paper and a formal peer review process, allowing students to experience the full scientific process. Overall, student responses to the course were highly favorable, and several students continued to use the computational programs, methods and XSEDE resources after completion of the course.

5. ACKNOWLEDGMENTS

This work was partially supported by NSF XSEDE Educational Startup Allocations (CHE1160026 and CHE170071).

6. REFERENCES

[1] Marcia C. Linn, Erin Palmer, Anne Baranger, Elizabeth Gerard, Elisa. Stone. 2015. Undergraduate research experiences: Impacts and opportunities. *Science* 347, 6222 (Feb. 2015), 1261757. DOI: https://doi.org/10.1126/science.1261757

[2] M. Kevin Eagan Jr., Sylvia Hurtado, Mitchell J. Chang, Gina A. Garcia, Felisha A. Herrera, Juan C. Garibay.. 2013. Making a difference in science education: The impact of undergraduate research programs. *American Educational Research Journal*, 50, 4 (Aug. 2013), 683–713. DOI: https//doi.org/10.3102/0002831213482038

[3] Mitchell J. Chang, Jessica Sharkness, Sylvia Hurtado, Christopher B. Newman. 2014. What matters in college for retaining aspiring scientists and engineers from underrepresented racial groups. *Journal of Research in Science Teaching* 51, 5 (May 2014), 555–580. DOI: https://doi.org/10.1002/tea.21146

[4] Lorelle Espinosa. 2011. Pipelines and Pathways: Women of Color in Undergraduate STEM Majors and the College Experiences That Contribute to Persistence. *Harvard Educational Review*, 81, 2 (June 2011), 209. DOI: https://doi.org/10.17763/haer.81.2.92315ww157656k3u

[5] L. Corwin Auchincloss, S. L. Laursen, J. L. Branchaw, K. Eagan, M. Graham, D. I. Hanauer, G. Lawrie, C. M. McLinn, N. Pelaez, S. Rowland, M. Towns, N. M. Trautmann, P. Varma-Nelson, T. J. Weston, E. L. Dolan. 2014. Assessment of Course-Based Undergraduate Research Experiences: A Meeting Report. *CBE Life Science Education,* 13, 1 (Spring 2014), 29-40. DOI: https://doi.org/10.1187/cbe.14-01-0004

[6] WebMO. 2016. Retrieved Jun. 24, 2017 from https://www.webmo.net

[7] Herschel Hunt. 1936. Demonstrations as a substitute for laboratory practice in general chemistry. II. What, why, and to whom shall we demonstrate? *Journal of Chemical Education*, 13, 1 (Jan. 1936), 29-31. DOI: https://doi.org/10.1021/ed013p29

[8] Frank Neese, Ute Becker, Dmitry Ganyushin, Andreas Hansen, Dimitrios G. Liakos, Christian Kollmar, Simone Kossmann, Taras Petrenko, Christoph Reimann, Christoph Riplinger, Kantharuban Sivalingam, Edward Valeev, Boris Wezisla, Frank Wennmohs. *ORCA: An ab initio, DFT, and Semiempirical Electronic Structure Package, version 2.9.0*. Max Planck Institute for Bioinorganic Chemistry: Mülheim an der Ruhr, Germany

[9] Eric F. Pettersen, Thomas D. Goddard, Conrad C. Huang, Gregory S. Couch, Daniel M. Greenblatt, Elaine C. Meng, Thomas C. Ferrin. 2014. UCSF Chimera—A visualization system for exploratory researchand analysis. *Journal of Computational Chemistry,* 25, 13 (July 2004), 1605-1612. DOI: https://doi.org/10.1002/jcc.20084

[10] Jan Breitenfeld, Rosario Scopelliti, Xile. Hu. 2012 Synthesis, Reactivity, and Catalytic Application of a Nickel Pincer Hydride Complex. *Organometallics,* 31, 6 (Feb. 2012), 2128-2136DOI: https://doi.org/10.1021/om201279j

[11] Virtual Inorganic Pedagogical Electronic Resource. Retrieved Oct. 28, 2016 from https://www.ionicviper.org

Extending XSEDE Innovations to Campus Cyberinfrastructure - The XSEDE National Integration Toolkit

Eric Coulter
Indiana University
Bloomington, Indiana
jecoulte@iu.edu

Jodie Sprouse
Cornell University Center for Advanced Computing
Ithaca, NY
jhs43@cornell.edu

Resa Reynolds
Cornell University Center for Advanced Computing
Ithaca, NY
resa.reynolds@cornell.edu

Richard Knepper
Cornell University Center for Advanced Computing
Ithaca, NY
rich.knepper@cornell.edu

ABSTRACT

XSEDE Service Providers (SPs) and resources have the benefit of years of testing and implementation, tuning and configuration, and the development of specific tools to help users and systems make the best use of these resources. Cyberinfrastructure professionals at the campus level are often charged with building computer resources which are compared to these national-level resources. While organizations and companies exist that guide cyberinfrastructure configuration choices down certain paths, there is no easy way to distribute the long-term knowledge of the XSEDE project to campus CI professionals. The XSEDE Cyberinfrastructure Resource Integration team has created a variety of toolkits to enable easy knowledge and best-practice transfer from XESDE SPs to campus CI professionals.

The XSEDE National Integration Toolkit (XNIT) provides the software used on most XSEDE systems in an effort to propagate the best practices and knowledge of XSEDE resources. XNIT includes basic tools and configuration that make it simpler for a campus cluster to have the same software set and many of the advantages and XSEDE SP resource affords. In this paper, we will detail the steps taken to build such a library of software and discuss the challenges involved in disseminating awareness of toolkits among cyberinfrastructure professionals. We will also describe our experiences in updating the XNIT to be compatible with the OpenHPC project, which forms the basis of many new HPC systems, and appears situated to become the de-facto choice of management software provider for many HPC centers.

KEYWORDS

Clusters, Scientific Computing, System Administration, OpenHPC, XSEDE, XCRI

https://doi.org/10.22369/issn.2153-4136/10/1/3

1 INTRODUCTION

1.1 Overview of XCRI

The XSEDE Cyberinfrastructure Resource Integration (XCRI) team is charged with helping to provide resources that extend the impact of experienced campus support and computational capacity for support of research activities. XCRI is a sub-group of the XSEDE Cyberinfrastructure Integration (XCI) team within XSEDE. While many campuses provide substantial resources to their users, IT staff are performing a common set of tasks locally that are duplicated at each site. Furthermore, when users move from these local environments to XSEDE SP resources, the change can be disconcerting as environments and software implementations are different, despite being billed as similar systems. By providing a set of consistent instructions and software layouts, XCRI strives to improve the aggregate use of national infrastructure, save local IT staff time and frustration, and support environments that are more similar to those found in the national scale cyberinfrastructure, easing transitions for scientists moving from campus to national computing.

1.2 XCRI Toolkits

One of the core pieces of the XCRI mission is providing software toolkits to enable cyberinfrastructure providers to easily emulate eminent SPs within XSEDE. Most of the toolkits curated by XCRI involve helping ease some type of pain common to a provider of research computing services, such as building an HPC system (The XSEDE Compatible Basic Cluster - XCBC[4, 5], and the Jetstream Virtual Cluster), assisting users with data movement (Ansible scripts to build a local Globus Connect Server), or providing users with scientific software. The last case is the primary driver of XNIT.

1.3 Toolkit Development Process

The XCRI toolkits are not produced in a vacuum, of course. By interfacing with the Requirements Analysis and Capability Delivery (RACD) team in the XSEDE Cyberinfrastructure Integration (XCI) group, we are able to gain access to the needs and preferences of the large XSEDE SPs. RACD is concerned with identifying software or services that will help the XSEDE SPs operate in an integrated way, while XCRI aims to take the lessons learned from XSEDE SPs on a large scale, and make them available to smaller scale institutions. The XCRI toolkit plan is based on the initial set of use cases developed in collaboration between the XSEDE Campus Bridging

team (now known as XCRI) with the XSEDE Architecture team during the first iteration of the XSEDE project [14]. Campus Bridging use cases were largely based on the output of the NSF Advisory Committee for Cyberinfrastructure Task Force Report on Campus Bridging[6], which examined ways to best enable researchers at institutions on all levels of the funding spectrum to use advanced research computing infrastructure with minimal friction. Campus Bridging use cases were later augmented by a set of project-wide "canonical use cases" that cover activities for multiple facets of XSEDE[15].

In creating use cases, XCRI goes through a drafting and review process to ensure that the essential parts of the Use Case will serve its constituents without requiring a particular solution. The Use Case process involves review by multiple parties in XCI, who are familiar with the XSEDE SP network and aware of the many difficulties involved in sharing knowledge between educational institutions. While the Use Case describes an outcome, it does not *prescribe* the solution. At this point in the process, interaction with the larger XSEDE user community is necessary, to determine what technologies are commonly used or coming into use. XCRI also often undertakes surveys with community collaborators to determine their interest level in new toolkits, based on the Use Cases in question. Once a plan is in place, XCRI team members will divide the project, and prepare a delivery mechanism for interested parties to use the selected software. If necessary, "glue code" will be created to ease the installation process, and friendly users will be sought out to ensure the toolkit is easily used by an outside party. At this point, the toolkit is considered completed.

Once a toolkit is completed, the release and distribution process begins. This depends quite heavily on the form the toolkit takes. XCRI toolkits are always linked in the XSEDE Community Software Repository. Toolkits have taken the form of everything from iso files (the original XCBC toolkit, based on Rocks), to Ansible playbooks delivered via a git repository (the current XCBC toolkit). For more frequently updated toolkits, delivery via GitHub offers an easy form of distribution, which invites community feedback. In the case of XNIT, of course, updated packages are simply copied into the yum repository from build servers.

Many of the XCRI toolkits fall under use case CB-02, "Share the XSEDE Environment with Campus Uses"[14] as this is an area with significant challenges for resource-constrained institutions, but still remains amenable to primarily technical solutions without the need for massive institutional buy-in, as is necessary for say CB-06, "Sharing computational resources between campuses". Many problems faced by campuses attempting to expand research computing are in fact political and cross-departmental, which are not well addressed by the provision of toolkits. Something akin to the ACI-REF[1] model of research computing facilitator, or the XSEDE Campus Champion, is more appropriate in tackling such issues (for example, convincing the administration that sharing scientific data with the broader community will be severely hindered if all access to institutional data must be done through local affiliate accounts rather than utilizing the InCommon Federation tools[16].

2 DEVELOPMENT OF XNIT

The idea behind the creation of XNIT was to provide campus cyberinfrastructure adminstrators a way to include software that is commonly found on XSEDE resources, without having to do a complete re-installation of operating system that the XCBC cluster distribution would require. Campus cyberinfrastructure may be set up for any number of purposes and local administrators may have quite salient reasons for configuration choices within the local environment, based on the needs of campus faculty. To satisfy the needs of a wide range of users, XNIT had to provide a set of scientific software that can be installed without overly disturbing local environmental configurations. The XNIT was created as a straightforward yum repository, so that administrators could install scientific software as easily as installing standard linux tools, without forcing anyone to learn to use yet another piece of software.

The XNIT was intended to provide access to rpms of commonly used scientific software that future users of XSEDE software could expect to see on XSEDE resources. In order to facilitate free distribution and extension, packages were all based on open-source software. The earliest list of software included in XNIT was elicited from the XSEDE Campus Champions list, with some discussion and modification by the packaging team led by XSEDE staff at Cornell University. Package maintainers regularly update package files based on new versions as they are released by the developers (which can be few and far between), and add new packages requested by campus CI providers, XSEDE Campus Champions, or other members of the community. Initial testing of XNIT was undertaken on systems at Cornell and Indiana University, with some "friendly-user" testing courtesy of Notre Dame's Center for Research Computing.

3 SPREADING THE WORD

Developing a user community for XCRI offerings represents a unique set of challenges. While there are a number of venues for discussing research computing on campuses, many of these activities focus on the concerns of established centers, and there are few signposts for those looking to start and support a research computing effort, implement the first cluster on campus, or those who are directed to start supporting a new research effort. Campus CI providers that are already engaged with XSEDE are usually (though not always!) able to get information about XSEDE software and practices to incorporate into their own local offerings. Larger conferences such as Supercomputing (SC) are mostly directed at those who are already fairly established in providing HPC offerings to their user base. Previously one venue where the XCRI team has been able to find interested CI providers has been the Educause Regional conference series, although the stream of interactions which lead to XCRI engagements has slowed over time. While it is difficult to find people and institutions seeking the type of help that XCRI can provide, the rewards gained in terms of increasing engagement with national CI, and enabling students and researchers at resource-constrained institutions to work beyond their funding capacity, are incalculable.

It is surprisingly difficult to spread word about free resources in the research computing community, particularly resources aimed at helping people who are not already well-established members.

Site	Date	TeraFlops
Marshall University	April 2015	9
Souther Illinois University	September 2015	35
Bentley University	January 2016	2
University of Texas at El Paso	May 2016	43
Brandeis University	March 2017	200
South Dakota State University	June 2017	10
Slippery Rock University	October 2017	10

Table 1: Table of site visits in which XCRI Staff helped implement or upgrade a physical cluster using the XCBC toolkit. Remote consultations are not listed, but include a variety of activities, including support for existing clusters, XNIT implementation, Virtual Cluster builds, and pointing institutions to other resources if necessary.

While XCRI has access to a vast array of XSEDE resources, it is rare that an institution which is already aware of XSEDE or already providing on-campus computational resources is in need of aid from XCRI. It follows in the same vein that institutions who are sending staff to large HPC conferences such as Supercomputing are rarely (but not never!) looking for help implementing or improving their current resources. The XSEDE Campus Champions community has proven to be a most valuable resource when attempting to spread the word that institutions looking for help can come to XCRI. Having XCRI members involved in the mailing list and providing occasional highlights in the monthly Campus Champion calls are outreach strategies that have resulted in several useful interactions for XCRI. Connections for six of the site visits XCRI has conducted have been made through the Champions' mailing lists. These interactions have also included support of campuses applying for hardware awards from the NSF and other institutions, as well as institutions building or rebuilding hardware they already own. To date, XCRI has written letters of support for five different proposals for cluster acquisitions, pledging to provide an integration site visit for winning campuses. Working with Campus CI implementers to build a community of practice around XNIT and other XCRI toolkits takes time, useful sets of tools, and a critical mass of users. Regular showings at conferences is a primary method of fostering awareness of XCRI offerings in the research community. While conference papers are quite useful for disseminating technical information, we've found that running tutorials aimed at new-to-HPC CI professionals is an excellent way to both support the community and spread awareness of XCRI toolkits. In general, XCRI tutorials and informational webinars have generated the highest attendance, compared to more traditional presentations. Beyond conference gatherings, BoFs, and panels, XCRI has worked to provide permanent venues for discussion and capture of knowledge so that XNIT users can record and exchange information amongst each other. Uptake of these tools has been variable and mostly transient at best. CI providers are most engaged with tools that have low barriers to entry and a bit of current interest, notably Slack, in conversing with each other and sharing their experiences with other providers. In addition to an **#xcri** Slack channel, XCRI has provided forum access on the XSEDE Community Software Repository, an **xcbc-hosts** mailing list, and a wiki area hosted by XSEDE. Thus far the clear winner in terms of user engagement has been the Slack channel, whereas the wiki area, which requires more planning and regular engagement by users and XSEDE staff in order to be useful, has seen little-to-no engagement. The mailing list proved to be a difficult effort in light of the small population involved. Enabling a community of practice takes time and a population that is encouraged to engage by common goals, identity, and participatory spirit[2]. Given the relatively limited population of adopters of XCRI toolkits, considerable work could be put into creating frameworks for collaboration that still fail to reach critical mass. Adopters may feel that their local issues would not be interesting to CI providers at other campuses. XSEDE resources for XCRI are thinly-spread at best, some work to improve the offerings for community collaboration might be provided by a future Campus Champions fellow or other similiar effort.

3.1 Tracking Usage

During the early stages of XNIT, XCRI staff worked closely with two early-adopters at the University of Hawaii and Montana State University. In both cases, the campuses had existing HPC infrastructure which didn't require a full rebuild à la the XCBC, but needed the additional support provided by an easily-installed repository of scientific software. Beyond direct communication with end-users, tracking the uptake of XNIT is very difficult. The most information it's really possible to see is how often, and from where, rpms are downloaded. Over the lifetime of the project, XNIT has attracted roughly 90 regular users across the US, based on simple download numbers.

4 EVOLUTION OF XNIT

In the initial conception of XNIT, the plan was to provide a means by which local campus administrators could easily install the same open-source scientific software as that found on XSEDE resources, concentrated around packages specifically called out by the XSEDE community. In order to provide a widely applicable toolkit, which could work both for established and new systems, the packages were built to be fully relocatable, by providing a "prefix" option in the rpm configuration. While initially this seemed a promising method of satisfying the needs of existing systems, in practice it is actually fairly non-trivial to successfully install an rpm to a non-standard location while also providing users with an easy means of access, without conflicting with existing standard package installations. In addition to the challenges involved in providing relocatable rpms, which often involves tracking down source code and creating ".spec" files from scratch, the adoption of the OpenHPC[13] project for the foundation of the XCBC toolkit led to a fundamental incompatibility between the two toolkits.

Several factors in the ongoing development of the HPC community made it more apparent that XNIT needed to change. After a few site visits involving already established clusters, the team learned that many admins have a need to maintain multiple versions of scientific software, and that the dependency enforcement built into yum, in which only a single version of a package can be available, makes XNIT untenable in that situation. Many site administrators also seem to accept that the burden of building packages will fall on them, and simply use an array of tools such as EasyBuild[8],

Helmod[3], or Spack[7], along with a module environment[8] to manage their scientific software.

With the rise in popularity of container solutions for platform-independent application deployment, and the arrival of HPC-friendly container runtimes such as Singularity[10] and CharlieCloud[12], it became clear that there are now much more powerful means of providing general scientific software. While an immediate switch to containerized software is not in the works, the XNIT project is in the process of pivoting to providing a more focused set of OpenHPC-compatible packages alongside the current relocatable options. By offering packages built with the open-source compilers and MPI implementations available in OpenHPC, XNIT will continue to support existing systems while staying in step with an open-source cluster management system with active development that is often the choice for new deployments based on ease of use and management. In the near future, the XCRI team will also begin to offer containerized versions of some packages, selected based on usage numbers gleaned from the XSEDE-wide XDMoD[11] usage tracking site and other data sources from XSEDE, which will work within the Singularity container runtime. There is also collaboration in the works with one of the lead SingularityHub developers, to offer a container registry tool that will allow access to both local and non-local containers, and easy interoperability with the Slurm[9] resource manager, used in the XCBC and a large segment of HPC systems.

5 LESSONS LEARNED

Since the initial release of XNIT toolkit in 2012, the team has taken several lessons from working with the community. First and foremost, it is extremely hard to gain live feedback from a community without a dedicated engagement team. XSEDE External Relations and Broadening Participation teams have been vital in helping produce materials and identify venues for outreach in the areas where new offerings for campus cyberinfrastructure will be well-received. The Indiana University Center for Engagement and Support has been invaluable to the XCRI team in managing and collecting survey data from the relevant groups, despite the challenges in surveying such small numbers. It is equally hard to observe usage of a service when the main interface is a passive yum repository. While this is excellent for ease-of-use, it provides little visibility past a list of IP addresses accessing the repository.

Second, it is not always the case that soliciting user requests will result in a tool with long-term uptake. Even after designing the toolkit based on community input, it still required a great deal of outreach in order to keep the toolkit visible to the community. Requirements alone are not sufficient for generating a toolkit, as a number of potential users are interested and want to move into cyberinfrastructure projects, but have little to guide them as to what will support those activities. It is important for a potential toolkit developer to take requirements, direction of technological initiatives, and goals and needs of users into account. In light of this, taking advantage of tools like XDMoD[11], which gives detailed job information for the whole of the XSEDE SPs, is essential. Detailed reporting like this makes it possible to see what the community is *really* using without the necessity of collecting user responses in a survey format.

Third, it is absolutely necessary for software toolkits to be built in such a way that it is possible to evolve the delivery mechanism based on user needs and preferences. While the demand for scientific software remains, the desire for such software delivered in rpm form has greatly dwindled. Due to the relative inflexibility of yum, and lacking a module system such as is available in OpenHPC, actually using XNIT is much more cumbersome than initially planned. In an age where software build solutions run rampant (such as Spack[7],EasyBuild[8], and Helmod[3]), cyberinfrastructure administrators have a wide array of flexible options at their fingertips.

Moving to a more flexible, extensible solution for scientific software delivery is a must for XNIT to continue providing a useful solution to easing the pain of administrating a general HPC-style resource at the campus level.

6 CONCLUSION

THE XCRI team occupies a unique space in the national cyberinfrastructure environment. Neither SP nor software developers, we stand at the edges of the research computing world, trying to help new or under-resourced institutions find their way towards providing high-quality resources to their scientists. With a mandate to primarily provide software toolkits, it can be challenging to keep up efforts to make newcomers to the research computing community aware of our presence. These same challenges appear through the HPC education community. Through a combination of white papers, tutorials, and engagement with online communities, however, it *is* possible to reach the relevant population.

With the ever changing nature of computing, it is also necessary to continuously update the offerings that XCRI makes available. In the case of XNIT, it has been not only necessary to change the software we offer (by keeping the repository up-to-date and being open to adding new software as requested), but to completely change the model of *how* that software is delivered. By keeping extant packages viable, offering an installation method that stays in step with a modern HPC solution, and beginning to offer truly system-independent container software, the XSEDE National Integration Toolkit will continue to evolve, and broaden the reach of resources available to all institutions taking part in the national research computing arena.

REFERENCES

[1] ACI-REF. 2016. Research Computing - ACI-REF. https://rc.fas.harvard.edu/partnerships/aci-ref/

[2] Pete Bradshaw, Stephen Powell, and Ian Terrell. 2004. Building a community of practice: Technological and social implications for a distributed team. In *Knowledge networks: Innovation through communities of practice*. IGI Global, 184–201.

[3] Harvard Research Computing. 2018. HeLmod Package Manager. https://www.rc.fas.harvard.edu/helmod/

[4] J. Fischer, E. Coulter, R. Knepper, C. Peck, and C. A. Stewart. 2015. XCBC and XNIT - Tools for Cluster Implementation and Management in Research and Training. In *2015 IEEE International Conference on Cluster Computing*. 857–864. https://doi.org/10.1109/CLUSTER.2015.143

[5] Jeremy Fischer, Richard Knepper, Matthew Standish, Craig A. Stewart, Resa Alvord, David Lifka, Barbara Hallock, and Victor Hazlewood. 2014. Methods For Creating XSEDE Compatible Clusters. In *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment (XSEDE '14)*. ACM, New York, NY, USA, Article 74, 5 pages. https://doi.org/10.1145/2616498.2616578

[6] NSF Advisory Committee for Cyberinfrastructure Task Force on Campus Bridging. 2011. *Final Report*. NSF. http://www.nsf.gov/od/oci/taskforces/

TaskForceReport_CampusBridging.pdf

[7] Todd Gamblin, Matthew LeGendre, Michael R. Collette, Gregory L. Lee, Adam Moody, Bronis R. de Supinski, and Scott Futral. 2015. The Spack Package Manager: Bringing Order to HPC Software Chaos. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '15)*. ACM, New York, NY, USA, Article 40, 12 pages. https://doi.org/10.1145/2807591.2807623

[8] Markus Geimer, Kenneth Hoste, and Robert McLay. 2014. Modern Scientific Software Management Using EasyBuild and Lmod. In *Proceedings of the First International Workshop on HPC User Support Tools (HUST '14)*. IEEE Press, Piscataway, NJ, USA, 41–51. https://doi.org/10.1109/HUST.2014.8

[9] Morris A. Jette, Andy B. Yoo, and Mark Grondona. 2002. SLURM: Simple Linux Utility for Resource Management. In *In Lecture Notes in Computer Science: Proceedings of Job Scheduling Strategies for Parallel Processing (JSSPP) 2003*. Springer-Verlag, 44–60.

[10] Gregory M. Kurtzer, Vanessa Sochat, and Michael W. Bauer. 2017. Singularity: Scientific containers for mobility of compute. *PLOS ONE* 12, 5 (05 2017), 1–20. https://doi.org/10.1371/journal.pone.0177459

[11] Jeffrey T. Palmer, Steven M. Gallo, Thomas R. Furlani, Matthew D. Jones, Robert L. DeLeon, Joseph P. White, Nikolay Simakov, Abani K. Patra, Jeanette Sperhac, Thomas Yearke, Ryan Rathsam, Martins Innus, Cynthia D. Cornelius, James C. Browne, William L. Barth, and Richard T. Evans. 2015. Open XDMoD: A Tool for the Comprehensive Management of High-Performance Computing Resources. *Computing in Science & Engineering* 17, 4 (2015), 52–62. https://doi.org/10.1109/MCSE.2015.68 arXiv:https://aip.scitation.org/doi/pdf/10.1109/MCSE.2015.68

[12] Reid Priedhorsky and Tim Randles. 2017. Charliecloud: Unprivileged Containers for User-defined Software Stacks in HPC. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '17)*. ACM, New York, NY, USA, Article 36, 10 pages. https://doi.org/10.1145/3126908.3126925

[13] Karl W. Schulz, C. Reese Baird, David Brayford, Yiannis Georgiou, Gregory M. Kurtzer, Derek Simmel, Thomas Sterling, Nirmala Sundararajan, and Eric Van Hensbergen. 2016. Cluster Computing with OpenHPC. http://hdl.handle.net/2022/21082.

[14] Craig A. Stewart, Richard Knepper, Andrew Grimshaw, Ian Foster, Felix Bachmann, David Lifka, Morris Riedel, and Steven Tueke. 2012. *Xsede campus bridging use cases*. Technical Report. XSEDE, Tech. Rep.

[15] XCUC 2013. XSEDE Canonical Use Cases. https://www.ideals.illinois.edu/handle/2142/43877. Online; accessed 19-April-2018; University of Illinois Urbana-Champaign.

[16] XSEDE InCommon Page 2018. XSEDE InCommon Identity Provider. https://www.xsede.org/ecosystem/operations/security/incommon. [Online; accessed 24-March-2018; XSEDE].

Student Outcomes in Parallelizing Recursive Matrix Multiply

Chris Fietkiewicz
Electrical Engineering and Computer Science Department
Case Western Reserve University
Cleveland, OH 44106 USA
001-216-368-8829
chris.fietkiewicz@case.edu

ABSTRACT
Students in a course on high performance computing were assigned the task of parallelizing an algorithm for recursive matrix multiplication. The objectives of the assignment were to: (1) design a basic approach for incorporating parallel programming into a recursive algorithm, and (2) optimize the speedup. Pseudocode was provided for recursive matrix multiplication, and students were required to first implement a serial version before implementing a parallel version. The parallel version had the following requirements: (1) use OpenMP to perform multithreading, and (2) use exactly 4 threads, where each thread computes one quadrant of the array product. Using a class size of 23 students, including undergraduate and graduate, approximately 70% of the students designed valid parallel solutions, and 13% achieved the optimal speedup of 4×. Common errors included recursively creating excessive threads, failing to parallelize all possible mathematical operations, and poor use of compiler directives for OpenMP.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education - Computer Science Education, Curriculum

General Terms
Algorithms, Performance, Design.

Keywords
Parallel programming, OpenMP, recursion, matrix multiply.

1. INTRODUCTION
Matrix multiplication is a common application in high performance computing and is often used for benchmarking in distributed systems. The standard iterative, loop-based algorithm is easily parallelized, and we have previously used this as an elementary application of multithreaded programming in a course on high performance computing. Alternatively, a recursive approach serves as a basis for algorithms that are faster than the $O(N^3)$ time of the standard iterative algorithm [1]. We introduced recursive matrix multiply as an exercise for designing a parallel program and to have students use an application that contrasts with a previous experience using the standard iterative, loop-based algorithm. Certain constraints were given regarding the parallelization such that the solution would be different from common, public sources.

DOI: https://doi.org/10.22369/issn.2153-4136/10/1/4

2. METHODS
The class was taught during the Spring semester of 2018 at Case Western Reserve University in Cleveland, Ohio. The total enrollment was 23 students, including undergraduate, graduate, and non-degree students. Table 1 shows the distribution of students by level, including subcategories for undergraduate and graduate students. Graduate students include Ph.D. and Master's. Undergraduates include juniors (3rd year) and seniors (4th year). Results also include one student who was of non-degree status. Though the course had been offered twice before, this was the first time that the recursive algorithm was included in the content. Survey data was collected to determine whether students had prior experience with C programming and multithreading, and this data was considered with regard to student outcomes.

Table 1. Distribution of students by level.

Level	*Number*	*Portion*
Ph.D.	4	17%
Master's	6	26%
Senior	10	44%
Junior	2	9%
Non-degree	1	4%
Total	23	100%

Prior to the assignment used in the present study, students had already completed lectures and assignments on parallelizing the standard iterative matrix multiply using the C language. Coverage of the standard algorithm included techniques for cache optimization, multiprocess programming using *fork*(), and multithreading using OpenMP. To prepare students for programming the recursive algorithm, lecture coverage included three components: (1) a mathematical definition of the algorithm, (2) a pseudocode implementation, and (3) a primer on coding and debugging the primary recursive function.

Recursive matrix multiply is defined [1] as

$$\boldsymbol{C} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \boldsymbol{A} \times \boldsymbol{B} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$\boldsymbol{C} = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}$$

where C_{ij} is a subarray of $\boldsymbol{C}$ with quadrant indexes *i* and *j*. As part of the lecture, students were asked to perform a recursive multiplication by hand using arbitrary arrays of size 4×4. In a separate lecture, a pseudocode implementation was given for a function *multiply*(*A, B, C, size*), where *A, B,* and *C* are square arrays and *size* is the number of rows (and columns) of the arrays. The pseudocode for a serial algorithm was given as shown below:

1. Base case: if *size* = 1, then $C_{11} = A_{11}B_{11}$.
2. Otherwise: use a temporary array *T* and compute the following:
   - *multiply*(A_{11}, B_{11}, C_{11}, *size* / 2)
   - *multiply*(A_{11}, B_{12}, C_{12}, *size* / 2)
   - *multiply*(A_{21}, B_{11}, C_{21}, *size* / 2)
   - *multiply*(A_{21}, B_{12}, C_{22}, *size* / 2)
   - *multiply*(A_{12}, B_{21}, T_{11}, *size* / 2)
   - *multiply*(A_{12}, B_{22}, T_{12}, *size* / 2)
   - *multiply*(A_{22}, B_{21}, T_{21}, *size* / 2)
   - *multiply*(A_{22}, B_{22}, T_{22}, *size* / 2)
   - $C_{11} = C_{11} + T_{11}$
   - $C_{12} = C_{12} + T_{12}$
   - $C_{21} = C_{21} + T_{21}$
   - $C_{22} = C_{22} + T_{22}$

Due to the complexity of the algorithm, another lecture was given as a primer on coding and debugging the primary recursive function. Students were advised to use a reduced algorithm in order to test proper indexing and use of the temporary array *T*. As a test, it was recommended that students first compute only one of the four quadrants. An example was given in class for computing only C_{21} as follows:

$$\boldsymbol{C} = \begin{bmatrix} 0 & 0 \\ A_{21}B_{11} + A_{22}B_{21} & 0 \end{bmatrix}$$

An example of using this reduced algorithm was calculated by hand in class using the same 4×4 array that was previously demonstrated. Following the hand calculation, a discussion was provided regarding how to properly associate the quadrant indexes (*i,j* = [1, 2]) with array indexes in the range 0 to *size* – 1.

For the assignment, students were required to use the C language to implement separate serial and parallel versions. The parallel version had the following requirements: (1) use OpenMP to perform multithreading, and (2) use exactly 4 threads, where each thread computes one quadrant of the final array product. The first requirement was given because students had already completed a previous assignment in which the standard iterative algorithm was parallelized using OpenMP. The second requirement differs from other common approaches in which each call to *multiply*() is performed on a separate thread. We required exactly 4 threads, with one for each quadrant, for two reasons. First, this unusual approach was intended to discourage the use of publicly available source code. Second, it has the advantage of being appropriate for development on commonly available 4-core CPUs.

For the parallelized implementation, students were required to report the speedup which was defined as the ratio of the serial run time to parallel run time. A solution was considered valid if it achieved a speedup greater than 1×, up to 4×, as compared to the serial version. Students were not told in advance what speedup ratio could be expected.

3. RESULTS

All students achieved a successful serial implementation, suggesting that adequate coverage was provided in class regarding the algorithm and coding suggestions. Therefore, the outcomes were evaluated according to five categories of reported speedup values: (1) approximately 4×, (2) greater than 1× but less than 4×, (3) no speedup or approximately 1×, (4) decrease in speed (less than 1×), and (5) not applicable (N/A). The N/A category represents students who reported unreasonable speedup values above 4× that were due to errors. The results given in Table 2 show that 70% of the students successfully achieved a speedup in categories (1) or (2).

Table 2. Student outcomes categorized by speedup values.

Speedup	*Number*	*Portion*
4×	3	13%
Less than 4×	13	57%
1×	1	4%
Less than 1×	3	13%
N/A	3	13%
Total	23	100%

In analyzing the student outcomes, we considered the background experience of the students. In addition to the academic levels listed in Table 1, survey data showed that 57% of the class (*n* = 13) had prior experience with C and multithreaded programming prior to taking the course. Prior to the present assignment, however, all students had completed five previous programming assignments. All previous assignments required C programming, and two of the previous assignments required the use of multithreaded programming with OpenMP.

We determined that experience prior to the course was not a determining factor for success in the present assignment. With regard to academic experience, valid solutions were obtained by all students in the two lowest experience levels: undergraduate juniors and non-degree. Invalid solutions occurred for students in the three highest levels: Ph.D., Master's, and undergraduate seniors.

We also concluded that outcomes did not depend on prior experience with C and multithreaded programming. For students with valid solutions (*n* = 16), only 50% of those had prior experience (*n* = 8). For students who did not have valid solutions (*n* = 7), 71% of those had prior experience using C programming (*n* = 5). Furthermore, of the students that lacked prior experience (*n* = 10), 80% of them had valid solutions (*n* = 8), including one student with an optimal speedup of 4×. This high success rate among students without previous experience suggests that lectures and previous class assignments were appropriate in preparing students.

Among students with valid solutions, two particular factors affected the speedup value. One significant factor was the specific implementation for the temporary array *T* that was used for intermediate calculations. We observed three basic approaches to implementing the use of *T*: (1) allocating it privately within the *multiply*() function, (2) allocating it as a single, full-sized array that was shared among all threads, and (3) eliminating *T* altogether by performing addition in the recursion base case. Approach (1) was the most common and generally resulted in speedup values from

2× to 3×. All students who achieved optimal speedup of approximately 4× used approach (3).

A second factor affecting the speedup in valid solutions was whether the student actually parallelized the addition step $C = C + T$. Several students neglected to include this operation in the separate threads. While they still obtained a speedup, this error significantly reduced the speedup from what was otherwise possible. Interestingly, one student implemented a recursive version of the addition step, as opposed to the ubiquitous use of for-loops for this purpose. Unfortunately, this appears to have significantly limited the speedup value.

Among the 30% of students that did not have valid solutions (n = 7), there were many different types of errors. The expected solution, as explained in the assignment instructions, required the creation of 4 threads at the beginning of the program, with each thread making an initial call to *multiply*(). In some cases, students mistakenly created the 4 threads in the recursive function itself, resulting in an excessive number of threads. For large array sizes, the effect was a significant increase in run time. Another error that was observed was poor use of OpenMP compiler directives. In previous assignments using OpenMP, students were required to compare the clauses "parallel" and "parallel for". In the present assignment, some students attempted to use the "parallel for" clause with a for-loop to create the 4 threads. In all of these cases, the students introduced some type of error with regard to the number of threads that were created or the manner in which *multiply*() was called.

Some students (n = 3) reported unreasonably high speedup values greater than 4× and were categorized as N/A in Table 2. Each of these cases involved unique logic errors in the parallel implementation. We do not report the specific errors here because they were unique to each student. However, they all can be described as parallelization errors in which less than 100% of the required calculations were actually performed. Additionally, another common factor in these cases was that students apparently did not test their programs using appropriate array sizes. Their programs actually produced correct results for sizes up to N = 4, which happens to immediately lead to the recursion base case after the recursive subdivision into separate quadrants. However, their programs failed for N = 8 and higher.

In requiring students to first implement a serial version, we intentionally gave them complete freedom in how to associate the quadrant indexes (i,j = [1, 2]) with array indexes in the range 0 to $size$ – 1. An interesting observation is that several students included mechanisms to preserve the quadrant indexing from the original mathematical formulation. It is noteworthy that all of these attempts were successful in the serial version, but some students' designs involved inefficiencies that limited the parallel version for large arrays.

4. DISCUSSION

The assignment used in the present study was our first attempt at requiring students to parallelize recursive matrix multiply. Our intent was to provide experience in designing a parallel program and to use an application that contrasts with a previous experience using an iterative, loop-based algorithm. We consider the requirements for parallelization to have been straight forward. We conclude that the complexity of the algorithm posed the most significant challenge in understanding how to apply parallel programming techniques. It is important to consider the students' backgrounds, and our survey data suggests that appropriate preparation was given to the students prior to the assignment. This preparation included previous lectures and programming assignments on the topics of iterative matrix multiplication, C programming, and multithreaded programming with OpenMP. Additionally, the lecture on debugging may have also been important. Overall, we consider the assignment to have been successful in challenging students to work with an unusual application using familiar techniques.

In the future, we hope to expand the assignment to require a more detailed efficiency analysis. In the present study, we focused exclusively on the students' ability to implement valid parallel solutions with at least some amount of speedup. However, we did not require students to analyze and report on the efficiency of smaller components of their implementation. Many students casually accepted less than optimal speedup values, suggesting that it was due to unavoidable issues, such as unknown aspects to using recursion. In general, students did not analyze the efficiency of separate components of the algorithm. For example, the utilization of the T array involved a large amount of variability in students' results. Additionally, several students did not parallelize the summation of the C and T arrays. In future versions of the assignment, we will consider adding a requirement to use multiple approaches for comparison.

5. ACKNOWLEDGMENTS

This work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University.

6. REFERENCES

[1] Weiss, M. 2012. *Data Structures and Algorithm Analysis in Java (3rd Edition)*. Addison-Wesley, Boston, MA.

Scientific Computing, High-Performance Computing and Data Science in Higher Education

Lessons learned from the training program in a super-computer center in Canada

Marcelo Ponce
Erik Spence
Ramses van Zon
Daniel Gruner
mponce@scinet.utoronto.ca
ejspence@scinet.utoronto.ca
rzon@scinet.utoronto.ca
dgruner@scinet.utoronto.ca
SciNet HPC Consortium, University of Toronto
Toronto, ON, Canada

ABSTRACT

We present an overview of current academic curricula for Scientific Computing, High-Performance Computing and Data Science. After a survey of current academic and non-academic programs across the globe, we focus on Canadian programs and specifically on the education program of the SciNet HPC Consortium, using its detailed enrollment and course statistics for the past six to seven years. Not only do these data display a steady and rapid increase in the demand for research-computing instruction, they also show a clear shift from traditional (high performance) computing to data-oriented methods. It is argued that this growing demand warrants specialized research computing degrees.

CCS CONCEPTS

• **Social and professional topics** → **Model curricula**; **Computing education programs**; *Accreditation*;

KEYWORDS

Training and Education, Scientific Computing, High-Performance Computing, Data Science, Master's Program

1 INTRODUCTION

The computational resources available to scientists and engineers have never been greater. The ability to conduct simulations and analyses on thousands of low-latency-connected computer processors has opened up a world of computational research which was previously inaccessible. Researchers using these resources rely on scientific-computing and high-performance-computing techniques; a good understanding of computational science is no longer optional for researchers in a variety of fields, ranging from bioinformatics to astrophysics.

Similarly, the advent of the internet has resulted in a paradigm where information can be more easily captured, transmitted, stored, and accessed than ever before. Researchers, both in academia and industry [19], have been actively developing technologies and approaches for dealing with data of previously-unimaginable scale. Researchers' ability to analyze data has never been greater, and many branches of science are actively using these newly-developed techniques.

Unfortunately, the skills needed to harness these computational and data-empowered resources are not systematically taught in university courses [20]. Some researchers, postdocs and students may find non-academic programs to fill this void, but others either do not have access to these courses or cannot commit the time to follow them. These researchers typically end up learning by trial and error, or by self-teaching, which is rarely optimal.

A number of academic programs that aim to address this issue have emerged at universities across the world (a few examples are [11, 28]). Some of these grew out of the training efforts of High Performance Computing (HPC) centres and organizations (*e.g.* [27]). Recognizing the need for additional skills in their users, computing centres such as those in the XSEDE partnership [29] in the U.S., PRACE in Europe, and Compute/Calcul Canada have been providing local and online HPC training as part of their user support. Universities have also developed graduate programs in both Scientific and High-Performance Computing, to train scientists and engineers in the use of these computational resources.

A more-recent complement to these graduate programs is the development of the degree in Data Science (DS), that is, degrees which focus on the analysis of data, especially at scale. These degrees come in a variety of forms, from multi-year academic graduate programs to specialized private-sector training. These programs are in strong demand at present, as large companies have discovered the value in thoroughly analysing the vast quantities of customer data which they collect. It is expected that this field will continue to grow, and academic programs will continue to be introduced to meet this demand.

https://doi.org/10.22369/issn.2153-4136/10/1/5

The SciNet HPC Consortium [15, 26] is the high-performance-computing center at the University of Toronto. SciNet provides both computational resources and specialized user support for Canadian academic researchers, and as members of its support team, we are responsible for training researchers, postdocs and graduate students at the University of Toronto in HPC techniques. In this paper we provide a review of the current state of graduate-level Scientific Computing, High-Performance Computing and Data Science academic programs, and endeavour to share our training and teaching experiences as they might result useful for other super-computer centers. The paper is organized as follows. In Sec. 2 we discuss how computation has become an essential ingredient in many academic research endeavours; in Sec. 3 we review the current status of education in the areas of High-Performance and Scientific Computing. In Sec. 4 we present the Data Science education efforts at the academic and non-academic level. We conclude with final remarks and perspectives for the future in Sec. 5 and 6.

2 THE ROLE OF HPC IN CURRENT RESEARCH

It is essentially impossible to give an overview of *all* the uses of computational methods in current scientific and academic research. We will nonetheless attempt a review of at least some computational scientific research, since the way computers are used in research (and other realms of inquiry) influences what should be taught to students.

Astrophysical computational research inherently involves large scale computing, such as the simulation of gravitational systems with many particles, magnetohydrodynamic systems, and bodies involving general relativity. Atmospheric physics requires large weather and climate models with many components to be simulated in a variety of scenarios. High-energy particle physics projects, such as the ATLAS project at CERN, require the analysis of many recorded events from large experiments, while other high-energy physics projects have a need for large scale simulations (*e.g.* lattice QCD investigations). Condensed matter physics, quantum chemistry and materials science projects must often numerically solve quantum mechanical problems in one approximation or another; the approximations make the calculations feasible but still rely on large computing resources. Soft condensed matter and chemical biophysics research often involve molecular dynamics or Monte Carlo simulations, and frequently require sampling a large parameter space. Engineering projects can involve optimizing or analyzing complex airflow or combustion, leading to large fluid dynamics calculations. Bioinformatics often involves vast quantities of genomic input data to be compared or assembled, requiring many small computations. Research in other data-driven fields such as social science, humanities, health care and biomedical science [13], is also starting to outgrow the capacity of individual workstations and standard tools.

Examining these cases in more detail, one can distinguish different ways in which research relies on computational resources:

(1) Research that is inherently computational, *i.e.* it cannot reasonably be done without a computer, but which requires relatively minor resources (*e.g.* a single workstation).
(2) Research that investigates problems that do not fit on a single computer, and therefore rely on multiple computing nodes attached through a low-latency network.
(3) Research that requires many relatively small computations.
(4) Research that requires access to a large amount of storage, but not necessarily a lot of other resources.
(5) Research that requires access to a lot of storage, on which many relatively small calculations are performed.

The distinction between the various types of research determines the appropriate systems and tools to use. Graduate students that are just starting their research often do not have enough knowledge to make the distinction (as nobody has taught them about this), let alone select and ask for the resources that they will need [20].

Note that all five categories fall under "Advanced Research Computing" (ARC). The categories are not mutually exclusive, but research of the second and third kind are usually associated with HPC, while the fourth and fifth, and sometimes the first, are associated with Data Science (DS). Although there is a lot of overlap between HPC and DS, these fields require somewhat different techniques. For that reason, we will consider separate programs for HPC and DS.

3 PROGRAMS IN HIGH-PERFORMANCE COMPUTING

Much of the research presented in the previous section falls in the category of *Scientific Computing* (SC). The growth in the computational approach to research, both academic and industrial, has prompted some institutions to develop graduate-level programs crafted to teach the skills needed to design, program, debug and run such calculations. These programs, having been in development for more than two decades, are now fairly widespread and mature, and are known by the names of "Scientific Computing" or "Computational Science and Engineering". Scientific Computing graduate degrees are offered internationally in several graduate education hubs around the world (U.S., England, Germany, Switzerland, *etc.*, — lists of which can be found at the SIAM and HPC University [14] websites). Canada is no exception, with at least eight universities offering graduate-level programs in Computational Science. These programs include one-year and two-year Master's programs, as well as Ph.D. programs. Most of these programs (*e.g.* the ones shown in Tables 1 and 4) require a final thesis. The projects and theses are faculty-guided research projects and are usually one-term long, though, as with all research, these projects sometimes take longer.

A typical curriculum for a two-year Master's program in Scientific Computing (in this case from San Diego State University) is presented in Table 1. It clearly shows that Scientific Computing has its roots in research in the physical sciences; the program heavily emphasizes numerical analysis and scientific modelling. In some ways this is not surprising: computers are very apt at solving such problems, and the formalism of the physical sciences often lends itself easily to computer programming. Other topics of study which are also often encountered in these programs include finite element analysis, matrix computations, optimization, stochastic methods, differential equations and stability.

In contrast to Scientific Computing, HPC requires somewhat wider knowledge; its practitioners need to understand more than

Course Name	Type
Introduction to Computational Science	required
Computational Methods for Scientists	required
Computational Modelling for Scientists	required
Computational Imaging	required
Scientific Computing	required
Applied Mathematics for Computational Scientists	required
Seminar Problems in Computational Science	required
Computational and Applied Statistics	elective
Computational Database Fundamentals	elective
Research	required
Thesis	required

Table 1: The curriculum for the two-year Master's program at the Computational Science Research Center at San Diego State University [9]; this forms a good example of a typical Scientific Computing graduate program.

just the theoretical and numerical principles. They require skills such as serial and parallel programming (often in several languages, and on different platforms) and scripting, as well familiarity with numerics, data handling, statistics, and supercomputers and their technical bottlenecks. In addition, these practitioners are usually not computer science students, so they must cope without that background. This is somewhat unavoidable as they need to have sufficient domain knowledge as well. Much of the same holds true for Data Science.

3.1 Academic HPC Programs

There are not many academic programs that focus on HPC. Part of the reason may be that such programs require access to a high-performance-computing machine so that students can develop their skills on real hardware, in a real supercomputing environment. These machines require multiple computing nodes which are connected by a low-latency network. Fortunately, such systems do not need to be local: as long as the machine is accessible through the internet the machine could be used for teaching. Nonetheless, having the hardware local to the students lends advantages, since most of the administrators and analysts of the system are typically available to assist students with optimizing their codes and developing good computational strategies. Not surprisingly, the majority of the currently offered HPC graduate programs seem to have been developed by or in conjunction with supercomputer centres. Availability of HPC resources at the local institution or department level varies significantly by country or region. While most top-ranked institutions in the US have HPC facilities, the funding model in Canada [6] is such that all the HPC resources are concentrated in a few national centers (like SciNet).

As examples of High Performance Computing programs, the University of Edinburgh (UK) offers an MSc in High Performance Computing, the Universitat Politècnica de Catalunya/Barcelona Tech (Spain) offers a Master in High Performance Computing and a Master program in Data Mining and Business Intelligence, SISSA/ICTP in Italy offers a Master in High Performance Computing, while a collaboration between the University ITMO (Russia) and the University of Amsterdam (Netherlands) offers a Double-Degree Master Programs in Applied Mathematics and Informatics (Computational Science). Note that many of these programs emerged from locations with a very strong tradition and consolidated background in HPC.

3.2 SciNet's HPC Programs

Many HPC centers provide training for their users to fill the computational-skills gap for the wider scientific community, such as, SDSC, PSC, TACC, NCSA, BSC, EPCC, CSCS, SHARCNET, AceNet, Calcul Québec, among many others. In its capacity as an HPC and ARC centre based at the University of Toronto, SciNet has developed several education and training programs [24] aimed at helping students and users obtain the skills and knowledge required to get the most out of advanced-research-computing resources. SciNet's training events and courses are currently taken by researchers, postdocs, and graduate students across many different departments and even from outside of the University of Toronto (UofT). Some of these courses are considered part of the official curricula and count as graduate level courses within the Ph.D. programs at UofT.

Initially SciNet provided training specifically oriented toward Scientific Computing, with the purpose of maximizing user productivity. These early classes focused on parallel programming (MPI and OpenMP), best coding practices, debugging, and other scientific computing needs. Over the years the breadth of courses has grown, with classes offered in Linux shell programming, parallel input/output, advanced C++ and Fortran coding, accelerator programming, and visualization. This is in addition to the annual HPC Summer School which SciNet runs in collaboration with two other HPC centres within Compute Ontario[4], CAC[1] and SHARCNET[2]. The summer school is a week-long intensive workshop[1] on HPC topics, and more recently, also Data Science and Medical/Bio-Informatics topics. Table 2 shows the curriculum of our summer school for last year.

Table 3 shows the training events and courses that SciNet has already been teaching in the areas of HPC and Data Science. The number and types of classes which SciNet teaches have grown significantly [25]. This can be seen in Figure 1, which presents the total student class-hours taught by SciNet over the last six years. This remarkable growth is a testament to the latent need for this material to be taught. The need for this training is supported by the enrolment statistics: our students constitute 35% of SciNet's total users, clearly showing that even in a specialized audience this kind of training is still needed.

For several years the four-week graduate-style classes offered by SciNet have been accepted for *graduate* class credit by the departments of Physics, Chemistry and Astrophysics at UofT. This was possible by accepting the classes as "modular" (or "mini") courses, one-third semester long, and bundling three such classes into a full-semester course. This arrangement has been so popular with students and faculty that the Physics Department recently listed SciNet's winter twelve-week HPC class in the course calendar [5], allowing graduate students from other departments in the university to take the class for university credit. Following exactly the same path, a six-week module designed to teach students from Biological and Medical Sciences, basics on data analysis with emphasis

[1] Similar initiatives and trends are being carried on by the International HPC Summer School [3] within the theme of HPC Challenges in Computational Sciences.

HPC Stream	Data Science Stream	BioInformatics/Medical Stream
Introduction to HPC & SciNet	Introduction to Linux Shell	PLINK
Shared Memory Programming with OpenMP	Introduction to R	Next Generation Sequencing
Programming Clusters with MPI	Data Science with Python	RNASeq Analysis
Programming GPUs with CUDA	Parallel R for Data Science	Python for MRI Analysis
Debugging and Profiling	Python for HPC (Parallel Python)	Image Analysis at Scale
Bring your own code Labs	Visualization with Python	Machine Learning for NeuroImaging
	Scientific Visualization Suites	R for MRI Analysis
		Public Datasets for Neuroimaging
		HCP with HPC: Surface Based Neuroimaging Analysis

Table 2: Curriculum for the 2017 SciNet's summer school, showing three parallel streams: the traditional High-Performance Computing, Data Science and adding a new parallel stream on BioInformatics/Medical applications. This type of events not only benefit the students and participants of the summer school, but also enables collaborations between departments and consortia, as it was this particular case, where part of the training was delivered in partnership with colleagues from SHARCNET and the Center for Addiction and Mental Health (CAMH).

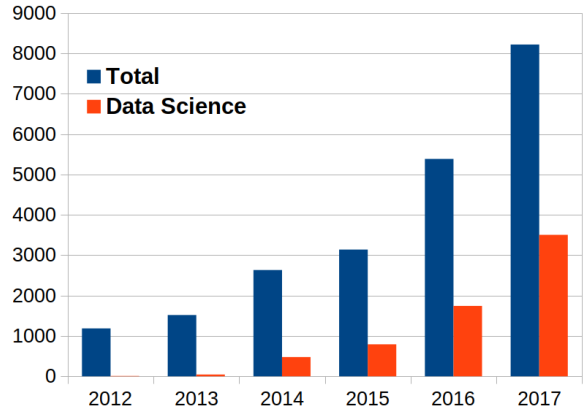


Figure 1: Attendance hours at SciNet training and education events, per year, for all SciNet classes and Data Science specific classes.

Course Name	Certificate	Credits
Quantitative Applications for Data Analysis‡	DS/SC	36
Introduction to Computational BioStatistics with R‡2	DS/SC	36
Introduction to Neural Network Programming	DS/SC	4
Neural Network Programming	DS/SC	16
Advanced Neural Networks	DS/SC	4
Intro to Apache Spark	DS	3
Machine Learning Workshop	DS/SC	6
Hadoop Workshop	DS	3
Scalable Data Analysis Workshop	DS	12
Relational Database Basics	DS/SC	6
Storage and Input/Output in Large Scale Scientific Projects	DS/SC	6
Workflow Optimization for Large Scale Bioinformatics	DS/HPC/CS	6
Python for High Performance Computing	DS/HPC/SC	12
Parallel R	DS/HPC/SC	3
Python GUIs with Tkinter	DS/SC	2
Scientific Visualization	DS/SC	6
Visualizing Data with Paraview	DS/SC	6
Scientific Computing for Physicists‡3	HPC/SC	36
Intro to Programming with Python‡	SC/DS	12
Intro to Research Computing with Python‡	SC	8
Intro to High Performance Computing	HPC	3
Advanced Parallel Scientific Computing	HPC	12
Intro to Scientific C++	HPC/SC	6
Intro to Scientific Programming with Modern FORTRAN	HPC/SC	7
Intro to Parallel Programming	HPC/SC	7
Programming Clusters with Message Passing Interface	HPC/SC	12
Programming Shared Memory Systems with OpenMP	HPC/SC	6
Practical Parallel Programming Intensive	HPC/SC	32
Intro to GPGPU with CUDA	HPC/SC	9
Programming GPUs with CUDA	HPC/SC	12
SciNet/CITA CUDA GPU Minicourse	HPC/SC	12
Coarray Fortran	HPC/SC	2
Parallel I/O	HPC/SC	6
Debugging, Optimization, Best Practices	HPC/SC	6
HPC Best Practices and Optimization	HPC/SC	3
HPC Debugging	HPC/SC	3
Intro to the Linux Shell	HPC/SC	2
Advanced Shell Programming	HPC/SC	3
Seminars in High Performance Computing	HPC/SC	4
Seminars in Scientific Computing	HPC/SC	4

Table 3: Courses taught by SciNet on *Data Science* (DS), *High-Performance Computing* (HPC), and *Scientific Computing* (SC). ‡ denotes courses already recognized at the University of Toronto in several departments, such as, Physics, Astrophysics, Chemistry, Ecology and Evolutionary Biology, Institute of Medical Science, Physical and Environmental Sciences, Engineering, as graduate level credits. We should also mention that students from other universities in the province of Ontario –e.g. Ryerson University– were allowed to enroll in some of these graduate courses for credit via a provincial academic transfer program.

in statistical analysis using the R Statistical Language [18], has now become one of the most popular courses at the Institute of Medical Science at UofT [7].

The skills that SciNet aims to transfer are rare and sought-after, and complement and enhance the skills students learn in regular curricula. That is why SciNet has developed a set of *Certificate Programs* [23], that users and students can pursue in *Scientific Computing*, *High Performance Computing*, and/or *Data Science*, once they have completed enough credit-hours. As a document that proves the holder has highly competitive skills, and in lieu of graduate credit for most SciNet courses, the certificates are in high demand. In a resounding endorsement of our teaching, thus far students have completed a total of 161 certificates (116 in Scientific Computing, 23 in High-Performance Computing and 22 in Data Science[4]). According to the current registration and trends, we are projecting to have over 250 certificates completed by mid-2018. Moreover, anecdotal feedback from some of our students suggests that the courses and their SciNet certificates were instrumental in successful job applications, in industry and in the financial sector.

4 PROGRAMS IN DATA SCIENCE

The wide adoption of the internet in the professional and the personal sphere ushered in the age of "Big Data". The ease of recording of people's online behaviour, and the ability to rapidly move data, lead to a large, diffuse, complex amount of data waiting to be mined for useful information. Because of the typically large size of the data special hardware and training are often needed. In contrast to Scientific Computing and HPC, there are many applications of Data Science in the private sector, e.g. in the medical, banking, retail, insurance, and internet industries.

Bioinformatics also has a large component in the academic world. Though a more-recent addition to the HPC disciplines, the bioinformatics field is well-populated with graduate programs, a testament to its rapid growth and latent demand. Its emergence as a major user of HPC systems has resulted in the development of "Master's of Bioinformatics", and related degrees. A typical Master's program is outlined in Table 4, this one from the Indiana-Purdue University at Indianapolis. While having many features in common with a more-standard SC Master's program, such as the study of programming and algorithms, it exhibits the particular needs of the bioinformatics community, stressing the importance of genetics and biological processes, and a lesser emphasis on mathematics and programming theory.

Degrees in Data Science are relatively new, with the first Master's program only being introduced in the U.S. (by North Carolina State University) in 2007. A sample of some of the classes offered in one such program is given in Table 5. As can be seen, these programs have a strong focus on data, with statistics, machine learning, and databases being their standard focus. Analyzing data that are too big to fit on a standard desktop computer requires specialized equipment; such training is also part of these graduate-level programs, as indicated by the presence of the "Cloud Computing" and "Distributed Systems" classes. Like typical graduate-level programs, these degrees usually require a final project or thesis to be presented by the student.

[4]This number has triplicated since the launch of the program in 2016.

Course Name	Type
Introduction to Bioinformatics	required
Seminar in Bioinformatics	required
Biological Database Management	required
Programming for Life Science	required
High Throughput Data in Biology	required
Machine Learning in Bioinformatics	elective
Computational System Biology	elective
Structural Bioinformatics	elective
Transitional Bioinformatics Applications	elective
Algorithms in Bioinformatics	elective
Statistical Methods in Bioinformatics	elective
Computational Methods for Bioinformatics	elective
Next Generation Genomic Data Analytics	elective
Next Generation Sequencing	elective
Bioinformatics Project	required

Table 4: The curriculum for the "Project Track" two-year Master's of Science in Bioinformatics at the Indiana University-Purdue University in Indianapolis; this forms a good example of a typical Bioinformatics graduate program.

One could argue that the novelty of methods in Data Science is due to its roots in Business Analytics (BA), where the objective is to make a decision. The field has certainly grown beyond that, and BA is now considered a sub-field of Data Science. Another more-recently developed sub-field is in the realm of health care ("Health Informatics"). Because these sub-fields are directly applicable to the private sector (and the associated revenue streams these present) these have become the most-commonly implemented post-graduate programs. The Business Analytics programs focus on using data to refine business administration, as well as develop marketing strategies. Health Informatics programs concentrate on using clinical data to optimize health care processes.

The practical focus of Data Science is reflected in the presence of an internship in the Data Science curriculum listed in Table 5.

Course Name	Type
Analysis of Algorithms	required
Machine Learning	required
Advanced Database Concepts	required
Distributed Systems	elective
Advanced Database Concepts	elective
Cloud Computing	elective
Information Retrieval	elective
Data Mining	elective
Web Mining	elective
Applied Machine Learning	elective
Complex Networks and Their Applications	elective
Relational Probabilistic Models	elective
Internship in Data Science	elective

Table 5: A selection of the courses available for the Master's of Data Science at the Indiana University.

Internships in such programs are similar to other co-op-type arrangements: the student works with an employer for a semester, allowing the student to gain hands-on experience applying the skills learnt during such period.

4.1 Academic Data Science Programs

Graduate level programs in Data Science are not difficult to find. For instance, programs in bioinformatics (a data-driven field), can be found on the web site of the International Society for Computational Biology. It speaks to the rapid rise of the field bioinformatics, that there are more bioinformatics programs available than Scientific Computing programs. Examples lists of other Data Science programs can be found at the NCSU analytics web site, the online business analytics programs site of `predictiveanalyticstoday.com` and at `online.coursereport.com`. Initially these programs were not as common as Scientific Computing, due to the fact that Data Science was a relatively new field of study. This situation has changed dramatically in the last few years. For instance in the United States alone there are hundred of Data Science degrees and certificates [8]. Among those programs about half are offered in the fields of Business Analytics and Health Informatics, with the other half being proper Data Science programs. There has also been a surge in Machine Learning and Artificial Intelligence programs, which span data science and scientific computing.

4.2 Non-academic Data Science training

The demand for Data Science skills (or "Data Analytics" skills as they are often called in the private sector) is so high [19] that the private sector has developed programs to meet the growing demand. See, for instance, on skilledup.com, which contains a list of data science boot-camps. The format of these classes is varied, though they are all oriented toward a "boot-camp" format: some are in person, some online; some are one-week long, others twelve weeks. These programs are very applied, often with one-on-one mentorship with a seasoned Data Analytics expert. They also include direct contact with possible future employers.

Moreover, a great number of these training programs are not focused on developing analytical thinking or problem-solving skills, [12] but rather are aimed at Ph.D.s and postdocs, whose problem-solving skills are assumed to have already developed. This allows them to focus on the technical training relevant to the job market. Some of these programs are free, some of them offer fellowships, and many of them charge on the order of 10-30 thousand US-dollars for a training period of, typically, three months. These programs have acquired such a level of popularity among young and recent graduates that the companies offering these programs have started to perform evaluation tests in order to assess which candidates are more suitable to be accepted to their programs. Perhaps the most appealing part for trainees is the networking platform offered by these programs, as in most cases they provide the opportunity to interact with actual companies looking for new talent and avoid recruitment layers.

Institutions in the non-profit arena are also starting to offer programs on Data Science. For instance the Fields Institute, a traditional institution for mathematical research, has offered several workshops and courses, and developed a thematic program on Big Data. Other examples include the International Centre for Theoretical Physics (ICTP) and the International School for Advanced Studies (SISSA), prestigious institutions with a well known tradition in theoretical physics, now offering training in "Research Data Science".

HPC centers are also venturing into Data Science training, offering workshops on R, Hadoop, machine learning, *etc.* SciNet started offering classes with greater data-oriented content (*cf.* Table 3) in 2013, with a four-week class in scientific analysis using Python. Having now finished its third year, the class remains popular, with about twenty students taking the class each year. The 2015 fall semester also inaugurated SciNet's first "Data Science with R" class, focusing on data analysis techniques using the R language. This class was very popular with over twenty-five students finishing the course, and most students requesting a second installment with more advanced material. Continuing its growth in the Data Science area, in the last year SciNet has held workshops in machine learning, scalable data analysis, and Apache Spark.

Comparing the student- and taught-hours per year shown in Fig. 2, one sees that the Data Science classes have been growing consistently, both in absolute as well as relative numbers (Data Science related courses roughly constituted less than 2% (2012), 4% (2013), 21% (2014), 22% (2015), 27% (2016) and 28% (2017) of the total classes taught respectively in each year. While attendance to Data Science courses, follows even a more significant trend: 1% (2012), 23% (2013 and 2014), 29% (2015), 40% (2016) and 41% (2017). We project for the current year a growth of at least 10%, positioning at equal levels the traditional Scientific Computing and Data Science trainings.

5 DISCUSSION

As mentioned above, scientific computing is used by scientists and engineers as never before, and graduate-level programs in Scientific Computing are numerous in Canada and around the world. In contrast, the development of HPC and Data Science programs is in its early stages, both in academia and the private sector. These programs are being developed to meet the continued shortfall in skill in these areas, with the McKinsey Global Institute estimating that the United States will be short 140,000 to 190,000 data analytics professionals by 2018 [16].

One may wonder whether online learning could not satisfy this need. A few examples of MOOCs (Massively Open Online Courses) in HPC and Data Science do exist[5]. However, seeing the growth in enrolment in SciNet's in-person courses and the summer school over the years (*cf.* Figs. 1, 2 and 3) shows that many students still prefer the face-to-face format.

Similarly, one may wonder why certificate programs do not suffice for HPC and DS education. As successful as these programs are, they have a few disadvantages. Firstly, they are mostly collections of fairly specific technical training: this leaves no room for more fundamental material. Secondly, it is also hard to encorporate an internship or thesis into such a certificate. Finally, certificates tend to carry less weight than degrees, and, in line with this, the demand for for-credit courses is larger than that for not-for-credit courses, as

[5] E.g. https://www.citutor.org/, http://www.hpc-training.org, https://www.futurelearn.com.

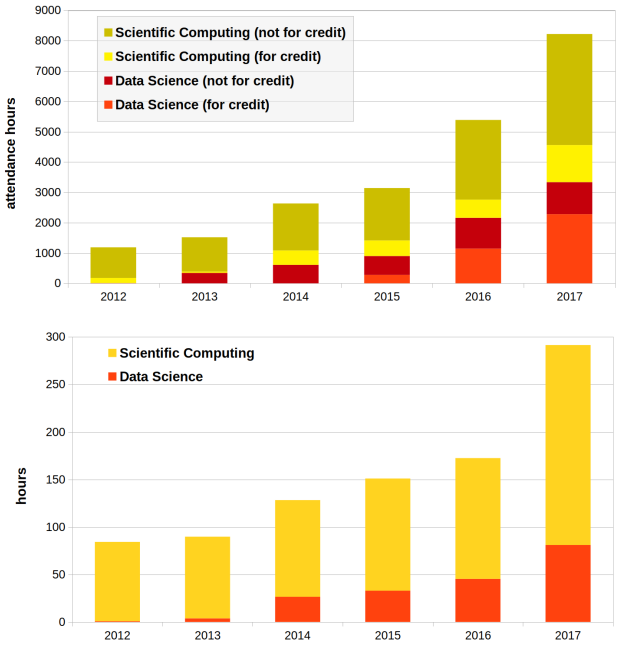


Figure 2: Total student hours (top) and taught hours (bottom) per year, for SciNet's Data Science and Scientific Computing related courses. The trends clearly show the need for training courses in both fronts, at the level of university recognized courses.

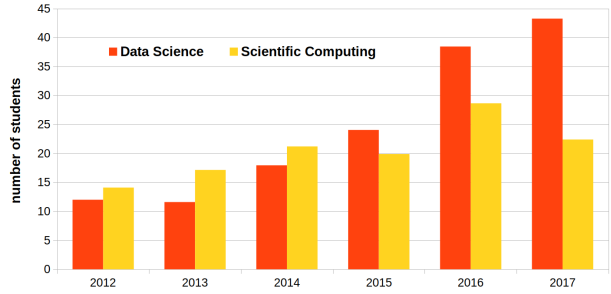


Figure 3: Average class size trend for Scientific Computing and Data Science courses at SciNet.

our experience with SciNet's Scientific Computing graduate course has shown.

A degree program in HPC or DS could offer more academic and fundamental education, which would leave the student with the analytical skills and high-level knowledge to stay on top of their field regardless of changes in computational technology.

6 CONCLUSION

We believe we have offered quantitative evidence demonstrating the need for programs in higher education in High-Performance Computing and Data Science, in particular in our own institution. If the qualitative evidence of this seems somewhat limited, it should be understood that existing HPC and DS programs (academic and non-academic) are still relatively new. While some such programs are already in existence, in many cases students must use non-academic options, or teach the material to themselves. Academic programs would offer the benefit of not just teaching specific technical skills, but an education in the fundamentals of HPC and DS and instilling the analytical skills needed to adopt to an ever-changing technological landscape.

We have reviewed existing academic and non-academic education programs, in both HPC and DS. In light of this review, we invite eager readers to look at a longer and complementary version of this manuscript [17] where we present the design for a tentative Master's programs in HPC and DS, based on the examples discussed here and drawing from the experience and enrollment statistics in not-for-credit training in HPC and DS by the SciNet HPC Consortium at the University of Toronto.

Getting well-founded graduate programs off the ground will not be without challenges. It will likely involve partnerships and discussions with other departments and institutes in order to offer a stronger and multi-disciplinary program. Existing HPC Centers, which already operate across multiple disciplines, can play a fundamental role in bringing together such programs. Thus, we have described SciNet's path in developing and transgressing the usual role of training events for users, into full credited graduate courses recognized at the university level for masters and doctorate degrees. Moreover, the creation of these graduate courses allowed us to leverage two key elements:

(1) Several of our analysts involved in the educational efforts got recognized by obtaining graduate teaching affiliations with the corresponding institutes/departments that were sponsoring the courses.
(2) Interact with students at a more professional level, collaborating and participating in research projects, allowing us to launch research initiatives [22] for which we already have successful cases [10, 21] and many more in the process of being developed. Perhaps the most important point here is to note that these collaborations were catalysed by the direct interaction with our students and researchers.

We hope this might help other super-computer centers transition a similar path and develop strategies to play a fundamental role in the multidisciplinary fronts that are HPC, SC and DS.

REFERENCES

[1] Centre for Advanced Computing (CAC) at Queen's University, Kingston, Ontario, Canada. https://cac.queens.ca/. (????). Accessed: 2016-06-14.
[2] 2001. Shared Hierarchical Academic Research Computing Network (SHARCNET). https://www.sharcnet.ca. (2001). Accessed: 2016-04-08.
[3] 2010. International HPC Summer School. http://www.ihpcss.org/. (2010). Accessed: 2016-04-08.
[4] 2014. Compute Ontario. http://computeontario.ca/. (2014). Accessed: 2016-04-08.
[5] 2016. Scientific Computing for Physicists (PHY1610H). http://www.physics.utoronto.ca/students/graduate-courses/current/phy1610h-f. (2016). Accessed: 2016-04-06.
[6] 2017. Canada's Fundamental Science Review. http://www.sciencereview.ca. (2017). Accessed: 2018-06-02.
[7] 2017. Introduction to Computational Biostatistics with R (MSC1090H). https://ims.utoronto.ca/wp-content/uploads/2014/04/IMS-syllabus-2017-REV-MSC1090H.pdf. (2017). Accessed: 2018-04-11.
[8] 2018. Masters's in Data Science. http://www.mastersindatasience.org/. (2018). Accessed: 2018-06-02.

[9] Computational Science Research Center at San Diego State University. 2011. Master's programs. http://www.csrc.sdsu.edu/masters.html. (2011). Accessed: 2016-04-06.
[10] Philippe Berger and George Stein. 2018. A volumetric deep Convolutional Neural Network for simulation of dark matter halo catalogues. (2018). arXiv:astro-ph.CO/1805.04537
[11] Helmar Burkhart, Danilo Guerrera, and Antonio Maffia. 2014. Trusted High-performance Computing in the Classroom. In *Proceedings of the Workshop on Education for High-Performance Computing (EduHPC '14)*. IEEE Press, Piscataway, NJ, USA, 27–33. https://doi.org/10.1109/EduHPC.2014.13
[12] Ed Lazowska. November 2015. Dear GeekWire: A coding bootcamp is not a replacement for a computer science degree. http://www.geekwire.com/2015/dear-geekwire-a-coding-bootcamp-is-not-a-replacement-for-a-computer-science-degree. (November 2015). Accessed: 2016-04-06.
[13] Petri Ihantola, Arto Vihavainen, Alireza Ahadi, Matthew Butler, Jürgen Börstler, Stephen H. Edwards, Essi Isohanni, Ari Korhonen, Andrew Petersen, Kelly Rivers, Miguel Ángel Rubio, Judy Sheard, Bronius Skupas, Jaime Spacco, Claudia Szabo, and Daniel Toll. 2015. Educational Data Mining and Learning Analytics in Programming: Literature Review and Case Studies. In *Proceedings of the 2015 ITiCSE on Working Group Reports (ITICSE-WGR '15)*. ACM, New York, NY, USA, 41–63. https://doi.org/10.1145/2858796.2858798
[14] Scott Lathrop, Ange Mason, Steven I. Gordon, and Marcio Faerman. 2013. HPC University: Getting Information About Computational Science Professional and Educational Resources and Opportunities for Engagement. In *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery (XSEDE '13)*. ACM, New York, NY, USA, Article 67, 4 pages. https://doi.org/10.1145/2484762.2484771
[15] Chris Loken, Daniel Gruner, Leslie Groer, Richard Peltier, Neil Bunn, Michael Craig, Teresa Henriques, Jillian Dempsey, Ching-Hsing Yu, Joseph Chen, L Jonathan Dursi, Jason Chong, Scott Northrup, Jaime Pinto, Neil Knecht, and Ramses Van Zon. 2010. SciNet: Lessons Learned from Building a Power-efficient Top-20 System and Data Centre. *Journal of Physics: Conference Series* 256, 1 (2010), 012026. http://stacks.iop.org/1742-6596/256/i=1/a=012026
[16] James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, and Angela Hung Byers. 2011. *Big data: The next frontier for innovation, competition, and productivity*. Technical Report. McKinsey Global Institute.
[17] Marcelo Ponce, Erik Spence, Daniel Gruner, and Ramses van Zon. 2016. Designing Advanced Research Computing Academic Programs. *CoRR* abs/1604.05676 (2016). arXiv:1604.05676v2 http://arxiv.org/abs/1604.05676v2
[18] R Core Team. 2017. R: A Language and Environment for Statistical Computing. (2017). https://www.R-project.org/
[19] Alex Radermacher and Gursimran Walia. 2013. Gaps Between Industry Expectations and the Abilities of Graduates. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*. ACM, New York, NY, USA, 525–530. https://doi.org/10.1145/2445196.2445351
[20] Mark Richards and Scott Lathrop. 2011. A Training Roadmap for New HPC Users. In *Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery (TG '11)*. ACM, New York, NY, USA, Article 56, 7 pages. https://doi.org/10.1145/2016741.2016801
[21] Alejandro Saettone, Jyoti Garg, Jean-Philippe Lambert, Syed Nabeel-Shah, Marcelo Ponce, Alyson Burtch, Cristina Thuppu Mudalige, Anne-Claude Gingras, Ronald E. Pearlman, and Jeffrey Fillingham. 2018. The bromodomain-containing protein Ibd1 links multiple chromatin-related protein complexes to highly expressed genes in Tetrahymena thermophila. *Epigenetics & Chromatin* 11, 1 (09 Mar 2018), 10. https://doi.org/10.1186/s13072-018-0180-6
[22] SciNet HPC Consortium. Research SciNet. https://www.scinethpc.ca/research-scinet/. (????). Accessed: 2018-04-13.
[23] SciNet HPC Consortium. SciNet's Certificate Program. http://www.scinethpc.ca/scinet-certificate-program. (????). Accessed: 2016-04-06.
[24] SciNet HPC Consortium. SciNet's Education website. https://support.scinet.utoronto.ca/education. (????). Accessed: 2016-04-06.
[25] SciNet HPC Consortium. SciNet's Training, Outreach and Education. http://www.scinethpc.ca/training-outreach-and-education. (????). Accessed: 2016-04-06.
[26] SciNet HPC Consortium. SciNet's website. http://www.scinet.utoronto.ca. (????). Accessed: 2016-04-06.
[27] Kris Stewart. 1995. HPC Undergraduate Curriculum Development at SDSU Using SDSC Resources. In *Proceedings of the 1995 ACM/IEEE Conference on Supercomputing (Supercomputing '95)*. ACM, New York, NY, USA, Article 19. https://doi.org/10.1145/224170.224209
[28] Guy Tel-Zur. 2014. PDC Education in the BGU ECE Department. In *Proceedings of the Workshop on Education for High-Performance Computing (EduHPC '14)*. IEEE Press, Piscataway, NJ, USA, 15–20. https://doi.org/10.1109/EduHPC.2014.9
[29] 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 (Sept 2014), 62–74. https://doi.org/10.1109/MCSE.2014.80

Initial impact of Evaluation in Blue Waters Community Engagement Program

Lizanne DeStefano
Georgia Institute of Technology
817 W. Peachtree Street, NW, Suite 300
Atlanta, GA 30308
1-404-894-0777
ldestefano6@gatech.edu

Jung Sun Sung
University of Illinois at Urbana-Champaign
615 East Peabody Drive
Champaign IL 61820
1-217-244-6385
jungsung@illinois.edu

Abstract
The external evaluation activities in the first three years of the Blue Waters Community Engagement program for graduate fellows and undergraduate interns are described in this study. Evaluators conducted formative and summative evaluations to acquire data from the participants at various stages during this period. Details regarding the evaluation methodology, implementation, results, information feedback process, and the overall program impact based on these evaluation findings are outlined here. Participants in both groups were selected from a variety of different scientific backgrounds and their high performance computing expertise also varied at the outset of the program. Implementation challenges stemming from these issues were identified through the evaluation, and accommodations were made in the initial phases of the program. As a result, both the graduate fellowship and undergraduate internship programs were able to successfully overcome many of the identified problems by the end of the third year. The evaluation results also show the significant impact the program was able to make on the future careers of the participants.

CCS CONCEPTS
Social and professional topics: Computational Science and Engineering Education

Keywords
Education, Program Evaluation, High Performance Computing, Blue Waters

1. Introduction
This paper describes the evaluation efforts regarding the Blue Waters Community Engagement program and relevant outcomes. The focus is on the first three years of the program, when challenges were identified and solutions implemented based on the evaluation results. Details regarding the evaluation methodology, implementation, results, information feedback process, and the overall program impact based on changes made in response to evaluation feedback are mainly outlined here.

The Blue Waters Community Engagement program is a high performance computing-based outreach program centered around the Blue Waters High Performance Computer at the University of Illinois Urbana-Champaign. Blue Waters was funded by the National Science Foundation, and managed by the National Center for Supercomputing Applications (NCSA). The system opened to the scientific community at large in March, 2013, and at the time of construction, was the fastest supercomputer on a University campus [1]. The Blue Waters Community Engagement program was created to support and educate computational science teams to make effective use of the unique and novel capabilities of Blue Waters. The community engagement program includes a graduate fellowship program, an internship program, webinars, workshops, annual symposiums, education allocation services, and community outreach efforts. The evaluation results presented here will focus only on the community engagement aspect involving Blue Waters graduate fellowship and internship programs.

The Blue Waters program is uniquely challenging, requiring a flexible and adaptive evaluation strategy to determine the effectiveness of both implementation and impact. The challenging aspects are that the program involved participants (both fellows and interns) who came from very different research backgrounds and are expected to interface with the same Blue waters supercomputing program structure from different scientific domains. Also, all participants had varying degrees of pre-knowledge regarding high performance computing. Based on these facts, a carefully planned initiation process and support was required in adjusting to the program. Additionally, this program also aims to achieve a goal of diversity and inclusion of various institutions with an emphasis on engaging women and minorities.

In this study, we implemented a flexible formative and summative evaluation strategy to capture the program implementation and effectiveness, as well as program impact and sustainability over the first 3 year period. A series of pre, mid, and post session surveys and focus groups were used

DOI: https://doi.org/10.22369/issn.2153-4136/10/1/6

for the formative evaluation during the program. Additionally, annual follow-up surveys and focus groups with program participants, mangers, and stakeholders were conducted for the summative evaluation and data collection. The results show that in such a program, providing detailed support plans and program expectations based on the entering knowledge level and background of the participants at the outset is important. Also, extended training and networking opportunities are critical in enhancing a positive learning experience and encourages pursuing further education and training required for a stronger next generation HPC community. We show that the evaluation feedback over the initial 3 years and subsequent changes have led to dramatic improvements in experience for most of the attendees.

2. Blue Waters and Program Outline

Blue Water is one of the most powerful supercomputers in the world and is also the fastest supercomputer on a university campus. The machine architecture balances processing speed with data storage, memory, and communication within itself and to the outside world in order to cater to a wide variety of scientific endeavors. It is supported by the National Science Foundation and the University of Illinois at Urbana-Champaign, and its projects are managed by the National Center for Supercomputing Applications. The NCSA provides expertise to help scientists and engineers take full advantage of the system for their research.

To achieve the vast potential of the Blue Waters system, well-educated and knowledgeable computational scientists and engineers are required. In an attempt to train and educate current and future generation of scientists and engineers who possess the extraordinary capabilities required at Blue Waters and other petascale computing systems, the Blue Waters established an expansive community engagement program engaging researchers, educators, HPC center staff, campus staff, and undergraduate and graduate students across all fields of study. As an effort to pursue growth and expertise in extreme scale computing for students, a graduate fellowship program and an internship program for undergraduate students were created as part of the a community engagement agenda. These initiatives were evaluated by an external evaluation team (Dr. Lizanne DeStefano, Executive director in CEISMC, Georgia Institute of Technology and Jung Sun Sung, Visiting Evaluator Specialist, University of Illinois at Urbana-Champaign). The community engagement program in total includes the graduate fellowship program, internship program, webinars, workshops, annual symposium, education allocation services and community outreach efforts. The evaluation results presented here will focus only on the community engagement aspect involving the Blue Waters graduate fellowship program and the undergraduate internship program.

2.1 Graduate fellowship Program

The Blue Waters Graduate fellowships provides PhD students with a year of full-time research support, including an annual stipend, an allocation of up to 50,000 node-hours on the powerful Blue Waters petascale computing system, and funds for traveling to the Blue Waters symposium to present research progress and results. The applicants are evaluated based on related experience and services, research plan in relation to Blue Waters, along with academic record. The fellows would work with assigned point of contacts at NCSA through regular meetings. The point of contacts are responsible for facilitating the fellows' access to Blue Waters, working with the fellow to solve computational problems, and helping fellows to connect with other sources of support. Six to ten fellows were accepted every year as Blue Waters fellows from 2014 to 2017, and a total of 26 fellows completed the program by the spring of 2018.

2.2 Undergraduate internship program

This program sponsors about 20 undergraduate research interns every year. A stipend, a two-week intensive Petascale Institute at the beginning, and an education allocation on Blue Waters are provided for each intern. Selected interns are able to travel to the Blue Waters symposium. Accepted students work with a faculty mentor either in their home campus or at another campus for one year. This program is also open to faculty who are willing to mentor undergraduate students in the internship program that involves teaching or research in the use of high performance computing. Faculty can participate in this program with assigned student(s), otherwise students and faculty mentors are matched by program managers. A total of 60 students completed the program between 2014 to spring 2017.

3. Evaluation Strategy

The external independent evaluation team conducted formative and summative evaluations to improve the programs and activities based on continuous feedback, while collecting appropriate data and information to conduct a longitudinal analysis of the impact of the programs over the life of the project. The ultimate goal of the evaluation is to validate and document the effectiveness regarding a new model of an education training program, and disseminate this model through publications and presentations. The evaluation utilized the 'Educative, Value-Engaged Approach' [2]. This approach, developed with NSF-EHR support, defines high quality STEM educational programs as that which effectively incorporates cutting edge scientific content, strong instructional pedagogy, and sensitivity to diversity and equity issues. In the Educative, Value-Engaged Approach, a key role of the evaluator is to work closely with program implementer to promote their understanding of program theory, implementation and impact.

The evaluation for Blue Waters community engagement program was specifically designed to answer four questions:

- Implementation: Is program being implemented on schedule and as planned?
- Effectiveness: Are key components of the program model operating effectively? How might they be improved?
- Impact: What outcomes (e.g. scientific knowledge, technical skills, and employment) are associated with participation in the program? How does impact vary across groups? What is the value-added from participation in the program?

- Sustainability: How and to what extent are elements of the program becoming institutionalized to ensure sustainability of program components? What opportunities and barriers exist?

3.1 Methods

In our approach, the external evaluators' functions were as educators as suggested in Lee J. Cronbach and associates Tower Reform of Program Evaluation [3]. The value of the evaluation would not be judged by accuracy of answers to the questions, but growth of understanding of others involved. As such, the evaluation results should be consumed in the progress of understanding the program and in discussions of alternative plans, not so much in determining if the current program is right or wrong. In acquiring a response to each question, the evaluation employed multiple methods.

- Implementation: Using a simple monthly reporting software and interviews with key implementers (at the University of Illinois and partner institutions), evaluators routinely monitored program implementation and reported on the progress, challenges, and slippage at each program staff meeting. When implementation problems or slippage occurred, the program staff determined strategies for overcoming barriers and keeping the program on track through communications with the Blue Waters Project Office.
- Effectiveness: All key components were routinely evaluated and continuously refined to improve the participants' experience. For example, to quantitatively assess the extent to which the program is creating training and education materials to address knowledge gaps among the HPC community, evaluators (1) conducted a formal review by stratified random sample of participants, oversampling under-served groups, (2) obtained expert endorsement of the quality of materials and services, (3) documented a reduction in training needs, and (4) assessed improved educational and research outcomes associated with training and education materials and activities, especially in under-served groups. For all new training and education materials and activities, evaluation included direct assessment of student knowledge and skills, observation of instruction, review of content, and measures of student and instructor satisfaction. All key components were routinely evaluated in this manner and evaluative information was used to continuously refine and improve program implementation.
- Impact: The external evaluation implemented a web-based survey system for carefully tracking all program participants over time. The system captured entry characteristics, program participation, subsequent use of materials and services; application of knowledge and skills gained, research, educational, and career outcomes. The value-added in program participation was evaluated by comparing key outcomes (e.g. diversity, research, publications, presentations, awards, retention, continued education, satisfaction, and within the timeline of funding time to degree and initial employment) where baseline data from each institution was identified. Impact data was reviewed by the Blue Waters User Advisory Committee to obtain an independent assessment of the quality and impact of the program.
- Sustainability: Through annual surveys, interviews with institutional leadership and key stakeholders, review of program requirements, and other means, the evaluation also examined the institutional changes that occurred as a result of the program including: changes in student knowledge and skills, increased diversity in targeted programs, institutionalization of elements of the program as routine university practices, etc. In addition to regular, informal reporting, the evaluation team produced an annual compilation of evaluation findings and work with the PIs to prepare the relevant sections of the annual report to NSF.

Table 1 and Table 2 show examples of the evaluation process for the graduate fellowship program and the internship program. Each activity was flexibly conducted depending on the year (e.g. the first year, the program managers, instructors were also invited to the interviews and surveys for initial implementation) and number of participants at each events.

Table 1. Example of overall evaluation process for graduate fellowship program

Time	Evaluation activities	Fellows	Faculty Advisors	Point of contacts
June	First Focus group at annual symposium	√		
September	Focus group at NCSA meeting		√	
February	Mid-Survey	√	√	
	Focus group			√
May	Final focus group	√		
	Post-survey		√	
After one year	Follow-up survey	√		

Table 2. Example of overall evaluation process for internship program

Time	Evaluation Activities	Interns	Faculty Advisors

May	Pre-Survey	√	√
May-June	Petascale Institute Daily Survey	√	
June	Focus group	√	
Aug-April	Monthly Report	√	
May	Final focus group at annual symposium	√	
August	Post-survey		√
After one year	Follow-up survey	√	

Informing and educating the program participants regarding the importance of their feedback, and goals of evaluation activates were found to be very effective. Participants who experienced immediate changes due to their feedback actively participated in the evaluation activities. Also, the formative evaluation results indicate the importance of maintaining flexibility in program implementation particularly in the early stages as the results show specific problems that can be changed. Evaluators also found the challenges of maintaining a longitudinal study and keeping perspective as seeking conclusive results regarding the impact of a year long program can be limiting and the response rate for the follow-up survey tends to decrease each year.

3.2 Findings

These findings are mainly focused on the challenges in the first year of the program revealing issues at the outset of program implementation.

3.2.1 Graduate fellowship program.

The first cadre of fellows were accepted in spring 2014 and the program period started in August for 1 year. These fellows were invited to the Blue Waters symposium in May, 2014 at the start of the program where they were introduced to the program model. The main context of the fellows were that each of them came from a very different science domain and their lab or faculty advisors may have had limited experience in HPC resources. Each fellow had an assigned point of contact to discuss technical difficulties and the general project progress. The initial process of starting the Blue Waters program and timelines of the research progress were the main focus of implementation at this stage.

Table 3. Findings from 2014 Blue Waters fellows at the initial phase of the program.

Fellows needs	Point of Contacts needs
• Keeping allocation on Blue Waters • One-On-One discussion with point of contacts • Basic tips and guidelines at the beginning	• Regular basis communication • Job description • Work plan from fellows
• Clear expectation for the symposium and conference	

Table 3 lists the needs from both the fellows and the point of contacts at the very early phase of the program. The main needs from fellows were connected to the context of the program. Since all fellows were from different scientific fields and generally new to the HPC domain, they required specific guidance to HPC conferences or symposia and in learning how to use the useful but enormous resources. Some of the faculty advisors had not used computational resources for their research data, and fellows definitely needed more technical support from their point of contact. Many of them desired one-on-one discussions along with regular group meetings to allow them to focus more on individual issues. Once they had experienced the power of Blue Waters, many of them wished to keep their data and work on Blue Waters to create further research outcomes beyond this program.

In the first year, the point of contacts were not very clear about how to provide specialized support for the fellows which were different from other allocation users. They expressed the need to have a clear research plan for the fellows at the beginning, and clear expectations for their roles as point of contact. The point of contacts also addressed the limits of their support, in that they lacked understanding of the scientific content in the research.

3.2.2 Internship program.

The main context of the internship program is to provide the undergraduate students with their first experience in the HPC field. As such, pursuing the goal of engaging and sustaining involvement by under-represented communities is also important. The participating students had very different levels of pre-knowledge on HPC, and some of them would have limited resource/programs to continue their education at their home campuses. There was an assumption that for the first time users to be successful, they would need (1) training, (2) practice, (3) user support, (4) extended collaborative support, (5) software tools and environment including science gateways to join the HPC community [4].

The pre-survey was conducted with students and faculty mentors a few weeks before the two-week workshop which started at the beginning of the program to find out the students' knowledge level. The daily session surveys had an important role in analyzing students learning experience from session to session during the institute. The interns were also invited to a face to face focus group at the end of the two-week institute to discuss their plans, needs, and concerns for the upcoming year-long internship. The evaluators also reached out to the faculty mentors, program managers, and instructors to find out the logistic challenges especially in the first year. The interns also responded with a monthly progress report and select interns participated in the focus group at the annual Blue Waters symposium to discuss the main impact of the program. Table 4 shows the main findings from interns each year.

Table 4. Findings from Blue Waters Interns for each year.

2014	2015	2016
• Over loaded daily schedule for two-week Institute • More resources for the session contents • More hands-on activities	• Detailed information before the start of the program • Improvement in evening lab • Insufficient program advertisement directly to the students	• Insufficient program advertisement directly to the students

In the first year, as interns had different levels of pre-knowledge in HPC, some found certain topics to be easy, while others struggled to keep up. Also, in daily session surveys and focus groups, many of the interns mentioned that the daily schedule at the institute was overloaded and covered too much content in too short a time. More handouts and hands-on activities were strongly suggested by the interns. In the second and third year, while content issues diminished, evaluators found out that faculty mentors heard about this program in a variety of ways, while only a few students were able to found out about this program on their own. As a result, it was became obvious that (1) if students could participate in this program only through their faculty mentors, this program would not be able to reach diverse institutions, and (2) the selection process is more likely to be strongly affected by the 'intern-faculty matching' process.

3.3 Program Adjustments

The findings from the preliminary evaluations were reported and the program managers and Blue Waters leadership were very flexible in adjusting the program implementations based on the feedback and discussing the future direction of the program. This part summarizes the main implementation changes based on the evaluation activities.

3.3.1 Fellowship program Adjustments.

These were the adjustments for the fellowship program

- Visiting NCSA: to overcome the challenges regarding lack of clear expectations, and starting/setting up close communication relationships, the fellows were invited to the NCSA in early fall to have a meeting with their assigned point of contact(s). The fellows are also were introduced to other resources and faculty at the University of Illinois campus.
- Connecting with other resources: to support more fellows' regional issues and context, either program managers or point of contacts started helping the fellows to connect with other possible technical staff at NCSA or other HPC student programs at their home campuses.
- Extended allocation: Blue Waters allowed fellows to extend the allocation period after the end of the program to continue conducting research.
- More face to face opportunities: More face to face meetings and activities were added to the beginning of the program to provide networking opportunities at the symposium.

3.3.2 Internship program Adjustments.

These are the adjustments for the internship program

- Fewer topics at the two-week institute: Based on the interns' feedback, the instructors narrowed the content topics for the two-week institute and researched the most/least desired topics every year through the post-survey.
- More hands-on activities: More hands-on activities were added into the less intensive schedule at the institute so that students had enough time to learn and attend the open-topic evening lab for catching up.
- More communication and workshops during the year: Interns expressed a desire to be connected after the institute and program. The webinars and workshops were provided to the interns during the year to share their experience and build a community.
- Inviting guest speakers from career development and HPC fields: The specialist, NCSA directors, HPC program directors, graduate program advisors were invited to the petascale institute sessions.

3.4 Impact of program

3.4.1 Impact of the fellowship program.

The impact of the program was mainly assessed from the focus groups, interviews, and post-surveys with fellows, point of contacts, and faculty mentors right before the completion of the program. We also utilized annual surveys with fellows a year or two after the end of the program.

Overall, the fellowship program enhanced the fellows' research progress by utilizing the unique power of Blue Waters. Fellows expressed that using Blue Waters allowed them to ask different types of questions with totally different physical scales in their research and bring about unforeseen results. After one year of the program, fellows pointed out that Blue Waters added a whole new dimension to their research and that it allowed them to make the best use of their resources. They emphasized not only the additional computational power and speed as a result of using Blue Waters, but also that the fellowship was a valuable learning experience. Fellows' comments regarding the program includes

– "For me, this fellowship is enabling a project that I just wouldn't be able to tackle without it. So getting started and beginning to move forward on Blue Waters itself, it's definitely letting me tackle science questions that I couldn't if I didn't have this fellowship…. Different types of questions and different scales of questions, so being able to resolve ends in my model with the biogeochemistry is not something that would be fiscal on the scale of my whole domain with the types of systems that I've been using up to this point."

Fellows also believed that the most prominent strengths of this program would be embracing all different fields of science and allowing the fellows to work on their research independently. They explained that one of the most powerful aspects of this program was learning the possibilities of multidisciplinary research (by using Blue Waters). They also expressed that the financial support helped them to conduct research independently for their degrees.

Hands-on experience on the powerful Blue Waters was emphasized as a benefit of this program by both faculty advisors and fellows. Attending the Blue Waters symposium provided the fellows with networking opportunities and helped them broaden their perspectives on HPC. The Fellows said that they had opportunities for interacting with other professionals, scientists, and also with other fellows through this program. Attending the Blue Waters Symposium is a good example of this. The networking opportunities helped them to broaden their field of research and expand their career choices as well.

The Fellows expressed their appreciation for the support from the point of contacts. The personalized technical assistance from point of contacts was greatly appreciated by the fellows. The fellows pointed out the importance of the communication with point of contacts, and how this help actually impacted their research progress. The meeting at NCSA enhanced the understanding of the research goal and detailed plan for both fellows and point of contacts.

On the post survey, faculty advisors reported that this program provided fellows with excellent computational resources along with personalized technical assistance, and a great opportunity for networking. They said their fellows were able to accomplish their research goals because of this program. At the follow-up survey, the fellows emphasized how this fellowship enhanced their skills in conducting research independently, and helped them to build a strong network with other scientists for their current and future careers. The comments from the previous fellows include

- "The BW fellowship was very important to my current and future professional endeavors. The fellowship allowed me the freedom and opportunity to propose and tackle my own research projects. Establishing this confidence and experience helped me obtain my faculty position without a postdoc. Moreover, the connections with NCSA staff and other BW fellows have been useful and will continue to be useful going forward. In particular the opportunity to collaborate with NCSA and other fellows our careers advance. Currently another BW fellow and myself are brainstorming and joint cross discipline NSF proposal coupling our work. We plan to write and submit once they complete their PhD and are either a postdoc or a junior faculty."

3.4.2 Impact of the internship program.

The impact of this program was gauged mainly from the pre, exit-surveys, focus groups, and annual follow-up surveys.

Overall, the internship program provided undergraduate interns with hands-on research experience which allowed them to have a strong HPC background and practical skills. Every year, more and more of the interns were planning to participate in the HPC-related programs/classes after this internship. In addition, a majority of the interns said that this internship program motivated them to pursue further research/career in this field.

The faculty mentors believed that this internship program was worthwhile in terms of providing interns with a very positive research experience and themselves with a professionally rewarding opportunity. The two-week institute at the beginning helped the interns in developing their technical skills and learning the overall concept of parallel computing through high-quality communication with the instructors. Table 5 shows that each item on the survey was highly rated by interns at the end of the two-week institute.

Table 5. 2016 Blue Waters internship program: Two-week Petascale Institute Exit- Survey.

Statements	Mean	N	SD
a. My goals for attending the 2-week training institute were achieved.	4.93	15	0.25
b. I am interested in attending similar programs as a result of this experience.	4.93	15	0.25
c. I am satisfied with the interaction and communication with other participants during the institute.	4. 87	15	0.34
d. I am satisfied with the interaction with instructors during this institute.	4.93	15	0.25
e. This institute helped me to develop my technical skills.	4. 73	15	0.44
f. I have the resources that I need in order to accomplish my goals during this program.	4. 73	15	0.44
g. I have a better understanding of the topics discussed as a result of this experience.	4. 87	15	0.34
h. I have a better understanding of Blue Waters as a result of this experience.	5.00	15	0

i. I have a better understanding of supercomputing as a result of this experience.	5.00	15	0
j. I have a better understanding of my future career as a result of this experience.	4. 00	15	0.82
k. The project presentation helped me understand my project better.	4.00	15	0.63
l. I know the next steps for me to proceed with my assigned project.	4. 47	15	0.50
m. I know the next steps for me to build on what I learned during this institute.	4. 53	15	0.50
n. Overall, I would rate this experience as successful.	5.00	15	0

(*5Rating scale: Strongly Disagree=1, Disagree=2, Neutral=3, Agree=4, Strongly Agree=5)

This program also includes a diverse group every year in an effort to reach out to underrepresented ethnic and gender group in STEM education. Figure 1 and Figure 2 show the underrepresented ethnic and gender group fractions of each year's participants.

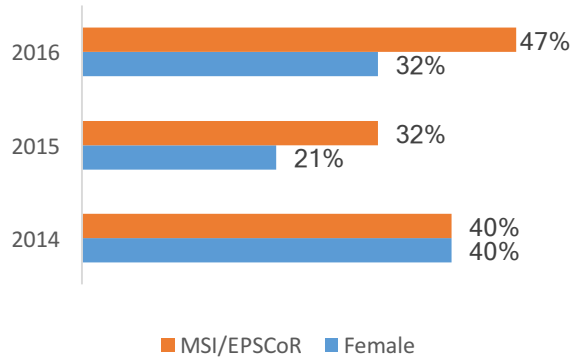


Figure 1. The ratio of female participants and MSI/EPSCoR Institution each year

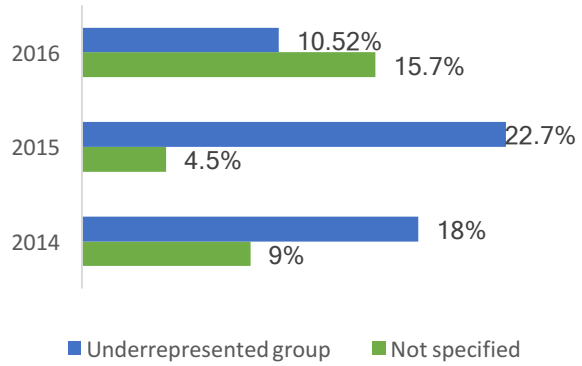


Figure 2. The ratio of participants from underrepresented group each year

(*Underrepresented ethnic group includes here African-American, American-Indian, Alaskan, and Hispanic)

3.5 Lesson learned

The evaluation process with Blue Waters community engagement program in the initial 3 years confirms some of the important program features which not only enhanced the fundamental goals of the program, but also led to critical adjustments at the early stage of the new program model. The evaluators tried to focus on analyzing data quickly to provide instant suggestions and feedback which directly affected the program directions.

- Complementary activities: In order to maximize the HPC education and training program, the importance of combining efforts with other complementary activates were emphasized by the results of the evaluation. Fellows expressed that attending the conferences, NCSA meeting, and symposium helped them to expand their networking, to expand their career spectrum, and how to cooperate with other scientists. For interns, adding professional development activities to the technical workshop allowed the interns to be encouraged to learn about the new career choices in HPC fields and advanced education potentials.
- Direct dissemination to students: to reach out to more diverse institutions, and underrepresented populations, it is important to have more direct information routes for the students who are in smaller colleges with limited HPC resources.
- User support: Providing close connection with point of contact for fellows were recognized as a huge success model for learning how to use the power of Blue Waters by providing a physical individualized support in addition to virtual resources.

4. Conclusion

The evaluation plans, activities, findings significantly affected the fellowship and internship program implementation, and program impact. The evaluation findings enhanced achieving the mission of educating a new and young generation for utilizing the powerful Blue Waters and other petascale computing systems in the future. Careful assessment of the program implementation and flexible adjustments can contribute to a successful outcomes in future HPC education and training programs.

5. Acknowledgements

This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications.

6. References

[1] https://bluewaters.ncsa.illinois.edu

[2] An educative, values-engaged approach to evaluating STEM educational programs," by J. Greene, L. DeStefano, H. Burgon, and J. Hall, 2006, New Directions for Evaluation, 109, p.53-72. Copyright 2006 by Wiley Periodicals, Inc.

[3] CRONBACK, L.J., AMBRON, S.R., DORNBUSCH, S.M., HESS, R.D., HORNIK, R.C., PHILLIPS, D.C., WALKER, D.F., AND WEINER, S.S. 1980. Toward Reform of Program Evaluation. Jossey-Bass, San Francisco, CA.

[4] Wilkins-Diehr, D. Gannon, G. Klimeck, S. Oster and S. Pamidighantam, "TeraGrid Science Gateways and Their Impact on Science," in Computer, vol. 41, no. 11, pp. 32-41, Nov. 2008. doi: 10.1109/MC.2008.470.

Effectively Extending Computational Training Using Informal Means at Larger Institutions

Dhruva K. Chakravorty
High Performance Research Computing
Texas A&M University
Texas, USA
chakravorty@tamu.edu

Marinus "Maikel" Pennings
High Performance Research Computing
Texas A&M University
Texas, USA
pennings@tamu.edu

Honggao Liu
High Performance Research Computing
Texas A&M University
Texas, USA
honggao@tamu.edu

Zengyu "Sheldon" Wei
High Performance Research Computing
Texas A&M University
Texas, USA
lele0027@tamu.edu

Dylan M. Rodriguez
High Performance Research Computing
Texas A&M University
Texas, USA
dylan@tamu.edu

Levi T. Jordan
High Performance Research Computing
Texas A&M University
Texas, USA
ljordan56@tamu.edu

Donald "Rick" McMullen
High Performance Research Computing
Texas A&M University
Texas, USA
mcmullen@tamu.edu

Noushin Ghaffari
High Performance Research Computing
Texas A&M University
Texas, USA
nghaffari@tamu.edu

Shaina D. Le
High Performance Research Computing
Texas A&M University
Texas, USA
sle1019@tamu.edu

ABSTRACT

Short courses offered by High Performance Computing (HPC) centers offer an avenue for aspiring Cyberinfrastructure (CI) professionals to learn much-needed skills in research computing. Such courses are a staple at universities and HPC sites around the country. These short courses offer an informal curricular model of short, intensive, and applied micro-courses that address generalizable competencies in computing as opposed to content expertise. The degree of knowledge sophistication is taught at the level of below a minor and the burden of application to domain content is on the learner. Since the Spring 2017 semester, Texas A&M University High Performance Research Computing (TAMU HPRC) has introduced a series of interventions in its short courses program that has led to a 300% growth in participation. Here, we present the strategies and best practices employed by TAMU HPRC in teaching short course modules. We present a longitudinal report that assesses the success of these strategies since the Spring semester of 2017. This data suggests that changes to student learning and a reimagination of the tiered instruction model widely adopted at institutions could be beneficial to student outcomes.

DOI: https://doi.org/10.22369/issn.2153-4136/10/1/7

CCS CONCEPTS

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

Keywords

HPC training, broadening participation, assessment strategies, best practices, diversity, computational thinking, tiered instruction

1. INTRODUCTION

Research efforts in STEAM (Science, Technology, Engineering, Art, and Mathematics) have significantly benefited from the rapid growth of computational capacity and the extensive use of data-analytics tools. The rapid proliferation of these methods has brought about an urgent need to train researchers who can effectively incorporate field-relevant computational tools and methods in their research workflows. In fact, developing computational and programming competency in the future science, technology, engineering, and mathematics workforce is a core component of the National Strategic Computing Initiative and the

National Science Foundation vision for Cyberinfrastructure for the 21st century [NSF NSCI]. In stark contrast to a global needs for a computationally-trained workforce, a vast majority of graduate students have limited exposure to computing. Indeed, during the NSCI presentations at SuperComputing 17 (SC17, Denver, CO) Irene Qualters (Director of the Office of Advanced Cyberinfrastructure, National Science Foundation NSCI Driver) discussed the need to train students to write coherent code [NSF HER]. There is a rapidly growing need to identify strategies to successfully introduce this population of students to computational methods and approaches [Lu 2009, NSF-Research, TRC 2014, Wing 2008, Yadav 2011].

Short courses and tutorials provided by HPC units remain stalwarts of informal education at the collegiate level. HPC-led short courses provide researchers with much needed technical information required for research and fill a void for student, staff, and faculty professional development that is not provided in a formal educational setting. Such courses further the institution's academic mission while simultaneously addressing the research computing needs of users who rely on these facilities. Unlike traditional credit-bearing courses that need to be approved at the department, college, and university level, an HPC unit can launch a short course in as little as two weeks. Furthermore, HPC units are constrained only by the expertise of their staff. While the importance of in-person training exercises cannot be stressed enough, "live" online training events organized at the regional or national levels are also effective. Events such as the XSEDE Big Data workshops and the Peta Scale Institute allow HPC units across the country to provide training on specialized topics that may go beyond local expertise at any specific site.

HPC-led courses are dynamic in nature. Owing to variations in the availability of expertise and researcher needs, HPC units have adopted different models of user training. At a rudimentary level, HPC short course offerings traditionally include courses that provide an introduction to operating systems on the cluster (Linux), cluster utilization (schedulers and file structures), and interpreted languages (Perl or Python). Larger HPC centers offer courses that included parallel programming paradigms (Open MP and MPI), rudimentary bioinformatics job-submission interfaces (Galaxy), and perhaps software applications as well (Abaqus or AMBER). As many branches of science have adopted large-scale computing, the HPC user profile has changed in recent years. We now offer additional courses that cover the use of data analysis toolkits (MATLAB, SAS and R), interfacing interpreted languages with data analysis packages (MySQL for Python users), and machine learning frameworks (TensorFlow and Caffe). This change in course offerings has been complemented by the gradual adoption of HPC course materials in into the formal classroom space. For example, TAMU HPRC does not offer the standard course that introduced Galaxy to our users. This material is now covered in the BIOL 647, "Digital Biology", a credit-bearing course taught by Prof. Rudolfo Aramayo with assistance from TAMU HPRC. Conversely, we have witnessed a significant growth in interest in our Python offerings because the traditional Computer Science course on Python is no longer approved for a graduate student's degree plan.

Researcher participation in HPC-led courses can be remarkably different at various institutions. This is surprising, considering that the topics covered in HPC courses and the computing needs of researchers largely remain the same across institutions of a similar size. Furthermore, the best practices in informal education are also well documented. The typical factors attributed to such variations are the instruction models used in teaching HPC courses, the amount of technology used in training, the length of the courses, the frequency of course offerings, the location of course offerings, the composition of research projects at the university, advertising information about the courses to the research community, student preparation at the undergraduate level, formal courses offered at the institution, and involving faculty in HPC instruction. While a number of interventions are possible to address these factors, there remains a dearth of quantitative data about the effects of these interventions on researcher participation in literature.

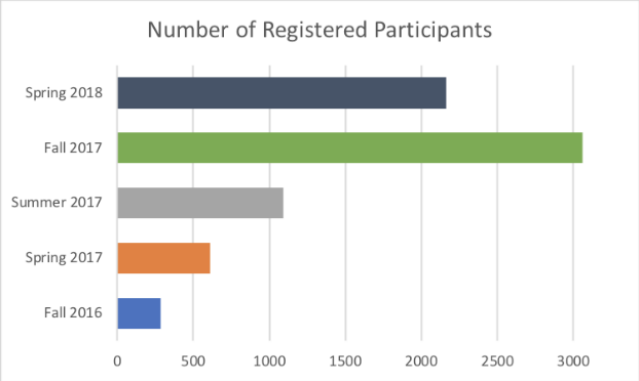


Figure 1. Number of registered participants in TAMU HPRC short courses from Fall 2016 through Spring 2018. The number of registered participants is not adjusted for hours of instruction, or the number of courses offered in each semester.

In this paper, we present a report on the effects of introducing curricular interventions on researcher participation in TAMU HPRC short courses program since Fall 2016. In the subsequent sections of this paper, we present quantitative data from our short course program along with details of our current offerings. We next describe a list of interventions that were introduced to our short courses program over the last four semesters. The paper next describes our efforts toward assessing the success of these courses on student learning using evaluations. We finally discuss the lessons learned over the previous year and summarize our findings in conclusion.

2. TAMU HPRC SHORT COURSES

TAMU HPRC short courses use active learning techniques and rely on HPRC staff expertise for content development. The courses are traditionally structured on a tiered instruction model (TIM) [Adams 2003, Tomlinson 1999, OME 2005, and OME 2013]. The tiered instruction approach provides lessons at different ability levels or areas of interest for a diverse learning community. Students and researchers using HPC resources come from varied academic and research backgrounds. In addition, these researchers may have different levels of exposure to computing and will require diverse computing skill sets to meet their research needs. In a typical TIM approach, the vast majority of learners would first participate in foundational courses, with a gradual drop-off as medium to higher level topics are approached. The TIM, however, faces challenges from advances in technology that eliminate the need for certain foundational courses and popular advanced offerings, such as CUDA or Machine Learning, that appeal to a wide range of researchers. These short courses are offered free of charge in an in-person format at the TAMU main campus in College Station, TX. These courses are also offered "live" via WebEx to an online audience that includes participants from a number of universities in the United States (including Puerto Rico). Figure 1 shows the number of registered participants in TAMU HPRC courses since Fall 2016. For the purposes of this paper, we use participation data

from Fall 2016 to describe a baseline that is compared to subsequent semesters where curricular and technical interventions were implemented.

3. IMPLEMENTED INTERVENTIONS

A number of technical, curricular, and engagement interventions have been systematically applied on a semester-wise basis. These include greater visibility for the courses, engaging students with active learning methods, better advertising, and retaining student interest via our HPRC seminar series. These steps have been simultaneously complemented by improved documentation on our website and wiki. Table 1 describes a series of interventions that were implemented in the TAMU HPRC short course offerings starting in Spring 2017. Taking a semester-by-semester approach as opposed to implementing all interventions in one semester, allowed us to quantify the effects of each semester. This approach also lets one to refine each intervention individually while allowing a short-staffed operation to adjust to the changes in schedules. While these interventions have overall contributed to making our courses more accessible, we have seen that our evolving online platform has had the largest effect on participation in our courses.

Table 1. Interventions introduced to TAMU HPRC short courses since Fall 2016. Interventions were carried into following semesters unless otherwise noted.

Semester	Interventions
Fall 2016	2-hour long lecture format courses. Hands-on exercises were not included.
	Certificates of attendance provided to attendees.
Spring 2017	2-hour long lecture format courses.
	Printed flyers distributed across campus.
	All courses slides were made available online in a standard format post-course.
	Handouts offered to students.
	Surveys collected via email.
	Seminal course on databases offered.
Summer 2017	WebEx introduced and online registration systems tested.
	Multiple courses offered on the same day
	Courses co-located and advertised with research computing event and open to REU students visiting TAMU
Fall 2017	Registrations standardized via Google forms interface.
	Offered seminal course on data management practices new to HPC training nationwide.
	Courses are first advertised to TAMU HPRC users and then to the entire TAMU community via campus email
	Introduced a new course, titled "Introduction to R". Python offerings increased to include w courses.
	Handouts and PowerPoint presentations offered pre-class online.
	To avoid issues with user registrations on HPRC systems, virtual machines were used for short course support.
	All courses were broadcast via WebEx.
	Certificates were restricted to in-person attendees alone.
Fall 2017 continued	Courses offered three days a week at two different locations near Engineering departments and biology/life sciences departments. These two locations were selected to ensure convenient commute for the participants.
	Partnered with the Laboratory for Molecular Simulation to offer new courses.
	Typical course length was 1.5 hours.
	Interactive exercises introduced.
Spring 2018	Surveys collected in-person on paper on conclusion of the courses.
	Classes offered on an all-day Friday setting at a single location near engineering and science departments.
	All courses offered in 3-hour format with a 10-minute break.
	All courses use active learning methods.
	Courses were recorded for future ADA compliant online courses.
	Introduction to Galaxy HPRC course discontinued. Supported BIOL647 "Digital Biology" a credit-bearing course.
	Offered training support to formal courses at TAMU.
	Standardized format to support XSEDE online workshops/courses.
	Open-on-demand shell access used in lieu of Moba-X-term and Putty during training.
	Offsite in-person training offered at other universities.
	Reports and analytics on short courses were prepared.
	Employed analytics to make decisions
	Machine Learning/Artificial course bouquet was offered to complement AI/ML support push by HPRC.

4. GROWTH IN PARTICIPATION

As described above Texas A&M HPRC offered a number of courses in Spring 2018. These courses and workshops were offered on a Friday morning and afternoon schedule to maximize the opportunities for researchers to attend these courses. A complete listing of short courses offered in Spring 2018 along with the number of registered participants in each course are provided in Table 2.

Table 2. List of TAMU HPRC short courses offered in Spring 2018. The courses are listed in the order in which they were

offered. The number of registered participants includes both online and in-person attendees.

Course Name	Registered Participants
Introduction to Linux	118
Data Management Practices	76
Workshop - Introduction to Linux	65
Introduction to HPRC Clusters	98
TAMU Open on Demand Portal	17
Deep Learning with TensorFlow	226
Molecular Modeling Workshop	30
Introduction to Python	217
Introduction to Scientific Python	190
Introduction to MATLAB	103
Python for MATLAB Users	89
Introduction to Perl	89
Introduction to Databases	108
Modern Computational Physics	11
Introduction to CUDA	43
Introduction to MATLAB Parallel Toolbox	10
Software Carpentry - Git, Shell & R	40
Using LS Dyna	18
Code Parallelization Using OpenMP	35
Code Parallelization Using MPI	30
Introduction to NGS	47
Introduction to NGS Assembly	45
Linux and Cluster Usage (TAMU Galveston)	18
Introduction to NGS Metagenomics	22
Introduction to NGS RADSeq/GBS	23
Introduction to the R programming language	136
Introduction to Fortran	57
Machine Learning and Deep Learning with MATLAB	183
XSEDE Big Data Workshop	40

4.1 Impact of the Academic Year

While we observed significant growth in participant registrations since Fall 2016, we recorded the highest number of participants in Fall 2017, when three HPRC courses were offered each week on a Tuesday, Wednesday, Friday schedule. Each course was taught for 90 minutes. This matches traditional expectations of the academic year, as new graduate students traditionally enroll at the university in the Fall semester. In Spring 2018, a number of these courses were consolidated to offer two courses each week that were taught for three hours each on a Friday morning and afternoon schedule. It is important to note that while the number of offered courses was reduced, the increase in instruction hours more than compensated for this effect. Furthermore, in Spring 2018, TAMU HPRC taught portions of formal graduate level courses that relied on the use of HPRC resources. These co-taught training models helped us strengthen ties with faculty and freed up time on our training program allowing us to offer new courses, a Software Carpentry series and support XSEDE workshops. Consequently, the total number of participants when adjusted for each hour of instruction represented a slight increase in Spring 2018 as compared to Fall 2017. This is surprising, as there were fewer new students to the Texas A&M campus in Spring 2018 as compared to the Fall 2017 semester. An additional compensating factor could be that existing graduate students at TAMU who had not previously enrolled in the HPRC short course program registered for the offerings in Spring 2018. We anticipate that participation data from Fall 2018 will bring clarity to this discussion.

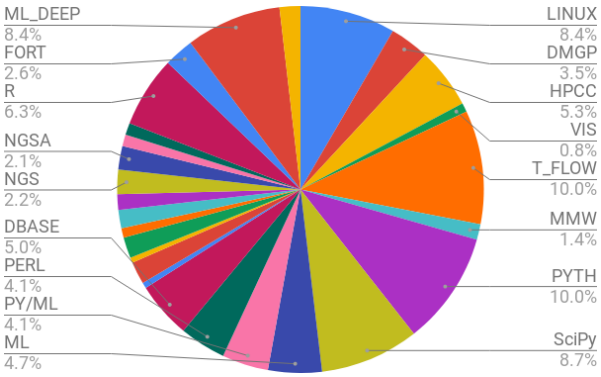


Figure 2. Distribution of participant registrations in HPRC short courses during Spring 2018. The figure displays combined attendance for both the in-person and WebEx sessions for each course. Topics include Fortran (FORT), R programming language (R), MATLAB (ML), Machine Learning and Deep Learning with MATLAB (ML_DEEP), Python programming language (PYTH), Linux classes and workshops, (LINUX), Databases (DBASE), Molecular Modeling Workshop (MMW) Scientific Python (SciPy), Python for MATLAB users (PY/ML), Perl (PERL), Next Generation Sequencing (NGS), NGS Assembly (NGSA), HPC Cluster usage (HPCC), Visualization portal (VIS), Data Management Practices (DMGP), TensorFlow (T_FLOW) and other topics.

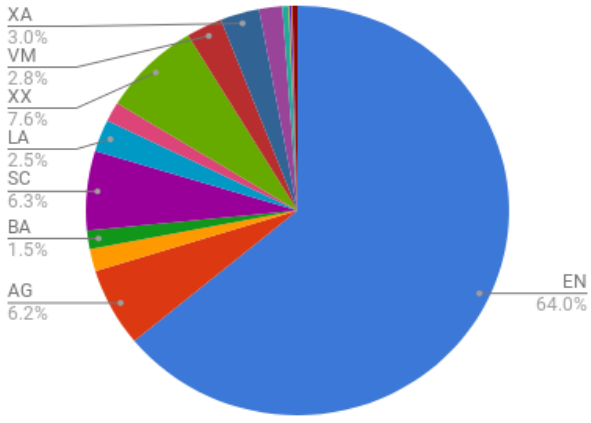


Figure 3. Distribution of registered participants in Spring 2018 across various TAMU colleges. The abbreviations used for the various colleges are Engineering (EN), Agriculture (AG), Business (BA), Science (SC), Liberal Arts (LA), Veterinary Medicine (VM), Other Entities including Industry (XX), and Other Academic Institutions (XA).

4.2 Developing Online Efforts

In Summer 2017, we observed that our data analysis bouquet of classes (Python and MATLAB) was routinely over-subscribed and the classrooms could no longer accommodate all interested participants. As we were on track to offer three courses in the following semester (Fall 2017), offering repeat courses for popular topics was not a viable option. To ensure that all researchers interested in taking these in-demand courses had an opportunity to benefit from them, we started offering HPRC short courses over WebEx in Fall 2017. WebEx participants participate in the same hands-on exercises as in-person attendees. To achieve this, we include a monitored discussion channels for online participants and have opened usage by migrating the training platform from HPRC

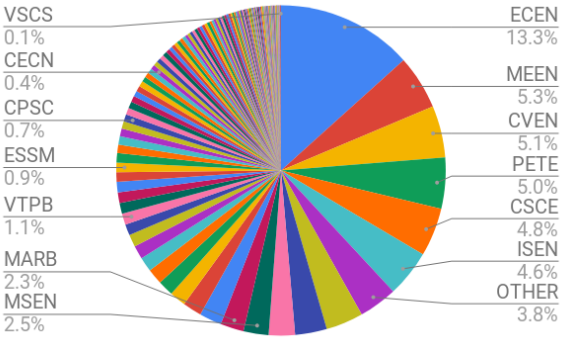


Figure 4. Distribution of registered participants across TAMU departments. Only departments with significant participation numbers are shown. Over 100 departments and institutions were served during this time frame. Percentage registrations from Electrical and Computer Engineering (ECEN), Mechanical Engineering (MEEN), Civil Engineering (CVEN), Petrochemical Engineering (PETE), Computer Engineering (CSCE), Industrial Science and Engineering (ISEN), Veterinary Small Animal Clinical Sciences (VSCS), Computer Science (CPSC), Ecosystem Science and Soil Management (ESSM), Veterinary Pathobiology (VTPB), Marine Biology (MARB), Materials Science and Engineering (MSEN) and non-TAMU units (Other) are shown.

clusters to Jupyter notebooks hosted on virtual machines. The online class platforms were further standardized in Spring 2018. We noticed that these sessions had an equal number of participants register as the in-person sessions. Providing these courses online via WebEx helped us reach out to a number of non-TAMU participants across the nation. While the majority of non-TAMU attendees participate using WebEx, individuals from local universities have also attended in-person sessions in College Station.

4.3 Location and Participation

Broadening participation in computing is a core tenet of the HPRC training program. We have taken a number of steps to assist users from non-traditional fields of computing. Figure 3 describes the distribution of registered participants for Spring 2018 across TAMU colleges, other universities and industry. As TAMU is a predominantly engineering university, it is not surprising to note that the majority of participants in the TAMU HPRC short course program (64%) belong to the College of Engineering. While it is heartening to note the participation from the department of education, the limited participation from biology users is noticeably low for an agriculture-focused school. This "anomaly" in national trends is because a number of HPRC-themed bioinformatics courses are now taught by the Biology department in the "Digital Biology" course. An accompanying distribution of TAMU departments with the most registered researchers is provided in Figure 4. Unlike Figure 3, Figure 4 does indicate four biology departments with significant student participation.

TAMU HPRC short courses have been traditionally taught at a location that is close to most departments in the Colleges of Science and Engineering. TAMU main campus is divided into East and West Campus with buildings being miles apart. While the East campus has a stronger Engineering focus, the West Campus houses a number of the biology disciplines. It is possible to hypothesize that the location of our short courses may be a deterrent to participation from non-engineering disciplines. In an effort to rule out "location" as being a factor in the lack of short course

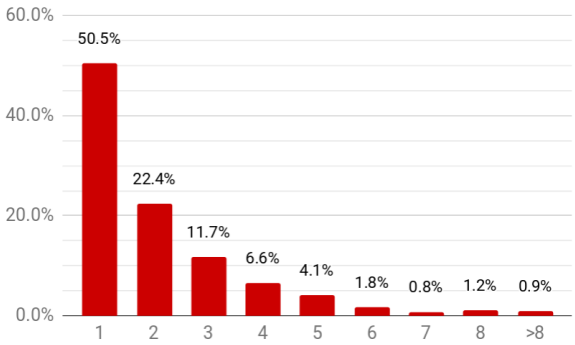


Figure 5. Student persistence profile for TAMU HPRC short courses in Spring 2018. Almost 50% of students enrolling in a HPRC short course returned for other courses. Participants enrolled in up to 14 short courses during the semester.

attendance from, we hosted a number of short courses in the West Campus library that is located close to the Mays Business School, the College of Agriculture and the College of Veterinary Medicine in Fall 2017. While we observed a slight dip in participation from engineering departments, no similar uptick was observed from the departments located on West Campus. With the expanding online training platform further reducing the impact of location, all in-person courses were returned to their original locations. Indeed, overall that the trends observed in Figures 3 and 4 have been consistent over the last few semesters regardless of location, suggesting that location of these courses is not critical to their success. To assist our biology-oriented users, we moved our short course schedule from a mid-week schedule to an all-day Friday schedule so that our users would find it easier to park and would not have to travel back and forth across campus on multiple days. A longitudinal study is planned to test the hypothesis that the availability of courses over WebEx has reduced the importance of the location of our courses.

4.4 TIM and Student Persistence

In an effort to refine our offerings, we have investigated the participation profiles of our course attendees over Spring 2018. Figure 5 presents a student persistence profile that describes how many HPRC courses each participant registered for in Spring 2018. It is heartening to note that over 49% of participants re-registered for two or more courses. A significant number of these participants returned to take 4 or more courses over the Spring 2018 semester, with single participants registering for 13 and 14 courses as well. Though there was limited participation from students belonging to non-traditional fields of computational enquiry, those who did attend were most likely to register for a large number of all HPRC courses. This is not surprising considering that programming and HPC training is not commonly offered in these departments. These

data present an emerging paradigm that students in disciplines that less computationally inclined are most likely to learn about computing from informal HPC short courses as opposed to formal courses. A similar longitudinal study about student attendance across multiple semesters is in the works. This data presents interesting insights into the design of a formal data sciences-oriented minor around these short courses.

The persistence information data indicates that a significant number of one-time participants register for some of the more popular HPRC courses (Machine Learning, CUDA, R, Python and MATLAB). As described above, TIM anticipates continued participation of students in foundational courses with a drop-off in participation as more advanced topics are covered. On the surface, the observed participation profile appears to violate the anticipated TIM learner progression, suggesting a reimagining of the instruction model. It is unclear whether technological interventions implemented by TAMU HPRC have allowed researchers to skip the foundation courses, or that immediate applicability of these materials to research forces students to cover the foundational materials on their own time. These data suggest that there is a need for the HPC-education community to develop TIM approaches for topics like Python and MATLAB. The TAMU HPRC short courses program is experimenting with a TIM model for Python. In Spring 2018, we offered a number of courses that introduced complexity in Python programming - Introductory Python, Applications of Scientific Python, TensorFlow, and Python for MATLAB users. In future iterations, we will include courses that cover topics in Parallel Applications of Python and the use of Pandas and Forecast Libraries.

It is interesting that the TAMU HPRC courses on R, MATLAB and Python continue to draw significant interest despite being supported by a number of formal efforts. In sharp contrast, our once popular "Introduction to CUDA" has since waned in popularity. This may be because, much like the case of Digital Biology formal classes, CUDA too is now being adopted in the formal Computer Science classroom. In contrast to CUDA, our classes with Artificial Intelligence or Machine Learning themes have been filled to capacity with students from Engineering. These participants represent a population of HPC users with different needs and are different from our typical user base. This became evident in our more traditional user-oriented courses. For example, attendees who took our "Scientific Python" course didn't appreciate the examples from Machine Learning training sets that were used as examples!

5. SUSTAINABILITY

The demonstrable need for TAMU HPRC short courses makes them inherently sustainable. They are offered free-of-charge to all participants on free-to-use software and machines. All course materials and notebooks are available for download free-of-charge from the TAMU HPRC website and we intend to release course recordings in the near future. The material from our short courses has been incorporated by courses currently taught at TAMU is currently being adopted by TAMU Galveston and Prairie View A&M University as well. The equipment for online WebEx broadcasts and video recordings is commercially available and may also be checked out from the TAMU libraries free-of-charge. Our National Science Foundation funded Cybertraining grant has provided us with the opportunity to develop a minor with a field of concentration in HPC as well. In addition, TAMU has submitted proposals that leverage the strengths of these short courses in search of federal dollars. As such, the approach toward institutionalizing the TAMU HPRC short course is likely to further strengthen the sustainability aspects.

6. EVALUATIONS & ASSESSMENTS

TAMU HPRC has worked with faculty in developing of "phase-gapped" evaluation strategies that help assess these programs. We are currently evaluating our initial designs in terms of their (1) connection to delivering key chemical and STEM concepts, (2) engagement and accessibility for students in a cyber learning context, and (3) support for instructors/peer leaders [Prince 2004, Parsons 2011]. TAMU HPRC currently collects data from two forms of evaluation: (1) a formative evaluation to assess the quality of project components, monitor project implementation, and provide ongoing feedback to the leadership team, and (2) a summative evaluation to examine the benefits to instructors and assess the impact of the project in reaching its stated goals. Both types of evaluation use a mixed method approach of qualitative and quantitative indicators. In the near future, we will also evaluate progress on the decided learning objectives, including our effectiveness in student-teacher engagement and learning.

We have traditionally relied on in-person interviews for feedback from the community. Registration and attendance data further help identify the effectiveness of our short courses. We rapidly came to the realization, however, that while these data demonstrated the demand for our courses on campus they didn't inform us about the quality of our courses. We experimented with collecting short surveys about the courses in Fall 2017. While we initially followed a model of mailing surveys electronically, the returns were extremely limited. Physical post-class surveys that required attendees to complete a questionnaire were implemented in the tail end of Fall 2017. Spring 2018 represents the first semester when evaluations were standard to each HPRC short courses. We currently follow a post-training evaluation model that includes in-person interviews and a free-format survey questionnaire. The survey focuses on course content and the participant's objectives. A free-format survey with open-ended questions was chosen over a Likert-scale style survey to ask open-ended questions and not constrain our participant's choices. Our typical surveys, while anonymous, provide participants with the opportunity to provide their email addresses if they wish to be contacted. Since TAMU HPRC courses are taught by CI professionals who volunteer to teach these topics, questions about the quality of the instructor that are typical in surveys on formal courses are not included. The Spring 2018 survey was a one-page document that includes questions such as [i] Did you attend this course for research, personal or class needs? [ii] What did you expect to learn from this course? [iii] Did the course meet your objectives? [iv] What did you like about the course? [v] What would you like us to do differently? [vi] What other courses would you like HPRC to offer? [vii] If you would like to subscribe to HPRC announcements please provide us with your email address. [viii] Please provide any additional comments below.

While the feedback from the surveys has helped refine our program, we face challenges in efficiently quantifying the collected data. Responses to the free-format surveys have provided us with data points for course success that we had not considered. Analyzing these surveys tends to be laborious and leaves room for ambiguity and personal interpretation. For these reasons, we will transition to a Likert-scale style survey approach in Fall 2018. This survey has been developed with feedback from TAMU faculty who focus on education. Some key points that we will be addressing in our future surveys are:

The open-ended questions in our survey will be:

1. How did you learn about this course?
2. Why did you register for this course?

3. Is this course related to your research or degree plan?
4. What are the difficulties that you faced in this course?
5. What do you think are the strengths of this course?
6. What specific content/concepts in the course were particularly challenging for you?
7. What specific content/concepts in the course were particularly easy for you?
8. How do you plan on using what you learn in this course?
9. Who would you recommend the course to?
10. Please add any additional comments below.

The Likert Scale (1-5 scale) questions will be:

1. How easy was the course for you?
2. How satisfied are you with the course?
3. How likely are you to take the course again?
4. How likely are you to recommend the course to others?

Our future surveys will provide us a rationale for why people are attending our classes. We will use these data to build a quantitative model for evaluating course success that goes beyond repeat attendance, and finally develop a profile of the kind of students or groups that are most likely to take our courses. In the future, we will partner with research groups on campus for an Internal Review Board (IRB) approved study to investigate whether attendees at our courses are meeting their desired learning objectives. Over the next few semesters, we will correlate data from assessments with course registration profiles to develop a teaching model for specific short courses. These steps are critical to shape the design of future HPC short courses type of efforts for users in non-traditional fields of computing.

7. CONCLUSIONS & LESSONS LEARNED

TAMU HPRC has implemented a number of interventions to drive a 300% growth in participation in HPRC short courses. In addition to the general need for data analytics in the scientific workflow, this growth may largely be attributed to our social and curricular interventions. Our data suggests that the influencing factors include offering courses on exciting topics, making the community aware of these courses, better student engagement by using active learning methods, avoiding policy bottle necks that curbed user participation, and finally by supporting our users with better documentation and support. The data from the TAMU HPRC short courses program supplies interesting insights on widely accepted models of the "tiered instruction" approach such as TIM. A longitudinal analysis of this data is further required to entirely understand these effects. While our current assessments have allowed us to refine our courses significantly, we will be utilizing stronger research-based methods in the near future. We will further standardize the active-learning segments of our courses so that all participants are guaranteed a similar experience in all of our courses. Over the coming semester we anticipate that developing ADA-compliant online courses will be our single-largest legal and administrative challenge. In our previous curriculum revisions, we have found that initial student resistance to new approaches can be overcome by good communication and persistence. Once students get accustomed to new approaches and expectations, they quickly regain their comfort level. We have utilized our student workers to serve as tutors and act as liaisons in resolving issues that arise. These steps have placed us on a firmer footing to develop this home-grown HPC effort into a certificate program.

8. SUPPORTING INFORMATION

All training materials developed by TAMU HPRC are available for download free-of-charge on the TAMU HPRC website. Please access the material at https://hprc.tamu.edu/training and send feedback about your adoption experience to help@hprc.tamu.edu.

9. ACKNOWLEDGMENTS

The authors would like to thank staff, student workers and researchers at Texas A&M HPRC, the Laboratory for Molecular Simulation, TexGen, Division of Research and Provost IT for supporting the HPRC short course program at Texas A&M University. Portions of this research were conducted on the Ada and Terra clusters, and virtual machines provided by TAMU HPRC. We gratefully acknowledge support from the NSF Abstract #1730695, CyberTraining: CIP: CiSE-ProS: Cyberinfrastructure Security Education for Professionals and Students. Special thanks to the instructors of each course: Dylan Rodriguez, Michael Dickens, Donald Rick McMullen, Lisa Perez, Mark Huang, Ping Luo, Jian Tao, Yang Liu, Marinus Pennings, Keith Jackson, Noushin Ghaffari, and Shichen Wang. We also gratefully acknowledge support from the Francis Dang, Mark Huang and Jack Perdue for maintaining the clusters and virtual machines used in these efforts.

10. REFERENCES

[1] Adams, C.M., and Pierce, R.L. " Teaching by tiering, " in Science and Children, vol. 41, no. 3, pp. 30-34, 2003

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

[3] Lu, J. J. and Fletcher, G. H., 2009. 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.

[4] Parsons, J. and Taylor, L., 2011. Improving student engagement. Current issues in education, vol. 14, no. 1.

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

[6] National Science Foundation website describing the National Strategic Computing Initiative. Access at - https://www.nsf.gov/cise/nsci/

[7] Ontario Ministry of Education. (2005). *Education for All: The report of the expert panel on literacy and numeracy instruction for students with special education needs, Kindergarten to Grade 6.* Toronto, Ontario: Queen's Printer for Ontario. Access at: http://www.oafccd.com/documents/educationforall.pdf

[8] Ontario Ministry of Education. (2013). *Learning for all: A guide to effective assessment and instruction for all students, Kindergarten to Grade 12*. Toronto, ON: Queen's Printer for Ontario. Access at: http://www.edu.gov.on.ca/eng/general/elemsec/speced/LearningforAll2013.pdf

[9] Research on Learning in Formal and Informal Settings, National Science Foundation.

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

[11] Tomlinson, C.A, 1999. The Differentiated Classroom: Responding to the Needs of All Learners. Alexandria, Va.

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

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

HPC Education and Training: an Australian Perspective

Maciej Cytowski
Supercomputing Specialist
Pawsey Supercomputing Centre
Kensington WA Australia
maciej.cytowski@pawsey.org.au

Luke Edwards
Data Specialist
Pawsey Supercomputing Centre
Kensington WA Australia
luke.edwards@pawsey.org.au

Mark Gray
Cloud Lead
Pawsey Supercomputing Centre
Kensington WA Australia
mark.gray@pawsey.org.au

Christopher Harris
Senior Supercomputing Specialist
Pawsey Supercomputing Centre
Kensington WA Australia
chris.harris@pawsey.org.au

Karina Nunez
Marketing and Event Manager
Pawsey Supercomputing Centre
Kensington WA Australia
karina.nunez@pawsey.org.au

Aditi Subramanya
Marketing & Events Officer
Pawsey Supercomputing Centre
Kensington WA Australia
aditi.subramanya@pawsey.org.au

ABSTRACT

The Pawsey Supercomputing Centre has been running a variety of education, training and outreach activities addressed to all Australian researchers for a number of years. Based on experience and user feedback we have developed a mix of on-site and online training, roadshows, user forums and hackathon-type events. We have also developed an open repository of materials covering different aspects of HPC systems usage, parallel programming techniques as well as cloud and data resources usage. In this paper, we will share our experience in using different learning methods and tools to address specific educational and training purposes. The overall goal is to emphasise that there is no universal learning solution, instead, various solutions and platforms need to be carefully selected for different groups of interest.

Keywords

Online training, Self-guided learning, HPC training and outreach

1. INTRODUCTION

The Pawsey Supercomputing Centre is constantly evolving its training programs to build a critical mass of advanced computing knowledge in the research community. It will continue to engage in a broad range of activities to further grow the expertise of the next generation of supercomputing specialists and create a skilled workforce in Australia. The success of our HPC training program is driven by its diversity. From basic computer science training for non-experienced users, through introductory and intermediate supercomputing and cloud, to parallel programming courses, GPU hackathons and customised training for research groups.

DOI: https://doi.org/10.22369/issn.2153-4136/10/1/8

In this paper, we will address some of the most recent challenges in delivering HPC training and describe our ideas and experiences in facing them. We will share our experience in using different learning methods and tools to address specific educational and training purposes. Starting from traditional on-site training, through self-guided training materials to online training and webinars. The overall goal of this paper is to emphasise that there is no universal learning solution; instead, various solutions and platforms need to be carefully selected for different groups of interest.

2. PAWSEY TRAINING PROGRAMME

The Pawsey Supercomputing Centre has been developing and offering its training program for over ten years. It consists of various training modules related to HPC, Data, Cloud and Visualisation which are being offered to Australian researchers within the National Training Programme. The list of the training modules together with a short description of the content is presented in Table 1.

National Training is traditionally delivered on-site at universities and research institutions across Australia. Currently, it is composed of 5 training modules (first five modules listed in Table 1) presented during two days. The target audience of the training is researchers (students, PhD students, postdocs) who are willing to use infrastructure services offered by HPC centres for their research. It is required that the participants have a basic understanding of computer science. After two intensive days of training the participants should be able to (among others):

- understand HPC, Data and Cloud services offered in computing centres,
- understand what a shell program is and use basic Unix shell commands,
- setup, manage and use VMs in the Cloud environment,
- submit various types of jobs on HPC systems with the use of schedulers,
- use compilation environment on HPC systems,
- understand the concept of parallel file systems and their efficient use.

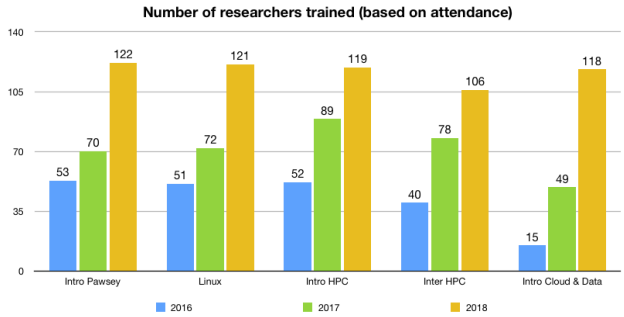


Figure 1. Attendance for the 5 core training modules.

We have gathered the attendance statistics for all training events provided over the years. Figure 1 presents attendance numbers for 5 core training modules of the National Training Programme over the past three years. Significant growth in attendance can be observed and we believe that this trend will continue in next years with some of the new online training offerings.

Table 1. Pawsey Training Programme

(as of September 2018)

Training Module	**Description of content and format**
Introduction to Pawsey	Pawsey HPC, Data and Cloud resources. Usage scenarios. 30min presentation
Introduction to Unix	Intro to shell, navigating, pipes and filters, shell scripts, finding things. 1.5hr self-guided hands-on training
Introductory Supercomputing	Basic supercomputing concepts, supercomputing architectures, use of queueing systems. 3hr session: presentation + hands-on
Intermediate Supercomputing	Compiling codes on HPC systems, advanced workflows and queueing scripts, parallel file systems. 3hr session: presentation + hands-on
Using Nimbus: Cloud computing at Pawsey (Intro to Cloud)	Intro to cloud computing, How to create and launch a Nimbus VM (including making keypairs, security, attaching storage, managing instance). 3hr hands-on session
Introduction to Data Services	Intro to Pawsey Data services and good data management practices. 30min presentation
Introduction to pshell	Intro to command-line access to long-term data storage. 30min hands-on session
Introduction to Supercomputing Technology	Supercomputing building blocks: CPU architectures, memory hierarchy, nodes, interconnect, network topologies. 2hr session: presentation + hands-on
Remote Visualisation	Intro to remote visualisation tools offered at Pawsey. 3hr session: presentation + hands-on
Containerising Workflows	Intro to Docker, Discipline specific examples. 2-3hr session hands-on session
Optimising Serial Code	Programming languages, profiling codes, cache usage optimisations, basic loop transformations, vectorisation. 3hr session: presentation + hands-on
Parallelising your code with MPI	Distributed memory parallelisation concepts. Basic MPI: point-to-point communication, non-blocking communication, collectives. 3hr session: presentation + hands-on
Parallelising your code with OpenMP	Shared memory parallelisation concepts. OpenMP directives, library routines and environment variables. 3hr session: presentation + hands-on
GPU Programming Essentials	GPU parallelisation concepts. GPU architecture, developing codes with CUDA and OpenACC, GPU libraries. 2x 3hr sessions: presentation + hands-on

All hands-on training materials are available on Github:

http://github.com/PawseySC

All presentations are available on Pawsey Centre user support pages:

https://support.pawsey.org.au

Pawsey Supercomputing Centre has more than 1500 registered users affiliated with different research institutions across Australia. For that reason, Pawsey's National Training sessions are being offered in Australia's largest cities: Sydney, Melbourne, Perth, Brisbane and Hobart. As the travel costs associated with the organisation of those training programs are usually substantial, we try to accommodate as many modules as possible within a single training. However, we have found that it is very hard to address different skill sets and interest of attendees with an increasing variety of topics presented during National Training. As a result, the most advanced training modules were usually less attended.

Over the years we have also struggled to increase the turnout of our training events. Statistics for previous years are presented in Figure 2. Introducing small nominal registration fees was one of the considered solutions. This might potentially increase turnout, however, more organisation overhead is required and there is always a risk that the training might become less accessible, especially for students.

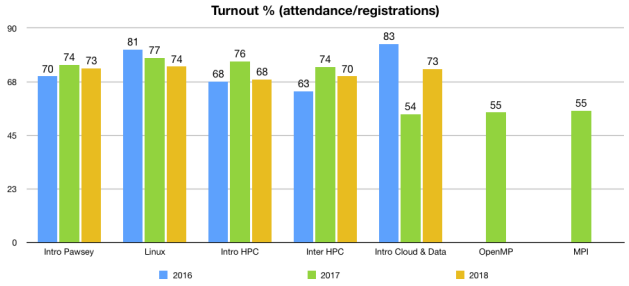


Figure 2. Turnout % (percentage calculated from actual attendance versus registration) for various training modules.

3. IN SEARCH OF THE NEW LEARNING FORMATS...

In the past, HPC developer courses were part of the National Training program. Training modules covering MPI, OpenMP and serial code optimisations were offered during the third day of the training. Unfortunately, we have found that with the proposed format it was particularly challenging to address different learners experience. Also, the number of registrations, as well as turnout of those training sessions, were often below expectations (Figure 2). To address those particular issues we have decided to experiment with a new online courses format designed especially for advanced HPC topics.

In the new format, parallel programming courses are divided into short webinar sessions (max. 90 minutes) organised throughout the week. Exercises are introduced to participants at the end of each session. Participants work on the solutions offline as access to HPC systems is granted to all participants throughout the whole week.

Interaction and communication are some of the most important aspects of training and education activities, which should be carefully addressed, especially in the case of online activities. Participants of the Pawsey online courses are encouraged to use chat room messaging during webinar sessions and a dedicated Slack channel for communication with Pawsey staff (or between themselves) while working offline on the solution of the exercises. This can be easily handled by two trainers, one presenting the material and the other keeping track and responding to chat messages.

We have found it very interesting that participants use both communication channels for exchanging their own experience and comments as well as helping themselves while working on the exercises. Participants can easily share their solutions by forking training materials repositories available on Github.

Table 2. Content and format of the Developing with MPI online course.

Online Course: Developing with MPI
Webinar 1 Introduction and Point-to-Point Communication **Exercise 1** Ping-Pong
Webinar 2 Non-blocking Communication and Communicators **Exercise 2** Message in a Ring
Webinar 3 Collectives, overview of other topics and next steps **Exercise 3** Game of Life

Table 3. Content and format of the GPU Programming Essentials online course.

Online Course: GPU Programming Essentials
Webinar 1 GPU Computing at Pawsey **Exercise 1** Compile and run a GPU program
Webinar 2 Introduction to CUDA **Exercise 2** Host-device transfer and vector addition
Webinar 3 Programming with OpenACC **Exercise 3** Accelerating a Jacobi simulation
Webinar 4 GPU Libraries **Exercise 4** Using the cuBlas library

We have run two HPC developer online courses as a pilot in June 2018. Contents and schedules of those courses are presented in Table 2 and Table 3. Both courses were very successful in terms of the number of registrations and participants. Although the turnout (actual webinar attendance versus registrations) was on the similar level as for the on-site training (around 50%), the registration numbers where much higher. We've got 54 registrations for the MPI course and 60 registrations for the GPU one. All registered participants received links to the webinar recordings. Therefore, potentially all registered participants could follow the course even if, for any reason, they were not able to connect to the live webinar session.

The pilot run of two described online courses was followed by the "Overview of Containers in HPC" webinar which covered a brief introduction to containers, usage of containers on Pawsey systems as well as example workflows and benchmarks using containers. It was a huge success, with 87 overall registrations and more than 50 actual attendees.

4. SELF-GUIDED LEARNING

Basic computational science skills should be developed as part of a University's curriculum, especially in computational science areas where computer simulations are a basic scientific tool. In practice, HPC centres still struggle to educate non-experienced computer users to use high-end computing infrastructure. We have identified self-guided learning platforms as particularly useful tools to address that problem. Specifically, our basic Unix training, which is based on Software Carpentry material [2] and uses the JupyterHub platform [3]. We deploy JupyterHub on our Pawsey cloud infrastructure (Nimbus), which provides us with the flexibility to give all users the same command-line environment to undertake hands-on exercises. This assists new users as they just need a browser and don't need to be familiar with a shell / command-line environment. Limiting the diversity of command-line clients also reduces the troubleshooting issues faced by trainers and helps training stay on schedule.

Our Nimbus Cloud and Container training are also structured in the 'Software Carpentry style' which enables instructors to easily undertake hands-on and interactive training but also enables users to undertake training in their own time. We have also adopted the "One-Up, One-Down" Software Carpentry feedback approach [5] at the end of training each day. This approach involves the instructor asking the learners to alternately give one positive and one negative point about the day, without repeating anything that has already been said. We write the responses on a whiteboard and we have found this 'encourages' people to say things they otherwise might not, compare to post-training surveys. This has given us valuable feedback to improve our training.

5. COLLABORATIVE COURSES

As opportunities have arisen, Pawsey staff have collaborated with researchers and software developers to run various HPC and domain-specific courses, including areas such as quantum mechanics, radio astronomy, and fluid mechanics. The format of these courses has varied, from workshops over a couple of days to semester-long university courses.

Introductory HPC material is provided and delivered by Pawsey staff, and it is important that this occurs at the start of the course before the participants commence more specific learning that uses the systems. Our staff also manage processes for reserving resources and setting up accounts to support the coursework on our systems.

A critical factor in the success of these courses is the knowledge and expertise provided by the collaborating researcher or developer, and their willingness to provide effort to develop and deliver material.

6. BRIDGING GAPS

Carefully planned outreach activities can be extremely useful in bridging gaps between education programs and HPC training. The Pawsey Supercomputing Centre has not only continued to deliver training sessions to maximum capacity but has introduced a variety of complementary outreach activities addressed to different communities, students and research groups. These activities include roadshows, internship programs, careers nights, open days and community data centre tours [4]. During the talk, we will mention a few of those activities and share our experiences with running them.

Pawsey Roadshows are information sessions where researchers showcase their science and research to university students across Australia with focusing on how their outcomes have been positively impacted by high-compute power and staff expertise. This encourages the future scientists of the Nation to keep supercomputing at the forefront of their mind when the time comes for them to begin their research.

Though in previous years Pawsey Roadshows saw high attendance, 2018 numbers have been minimal to date. Although attendees found benefit in learning about supercomputing, it did not muster enough interest to increase attendance. Due to this, the Pawsey Uptake Group (PUG) and training committee trialled hosting roadshows within university events and research open days rather than stand-alone activities. This decision has proven to be successful with Roadshow attendance and engagement with Pawsey nearly doubling.

The Pawsey Supercomputing Centre Summer Internships is a good example of bridging gaps and developing skills for a new generation of HPC-ready researchers. The internship program runs for 10 weeks during the period November/December through to February. The internships are open to 3rd year, higher undergraduate students (including honours) or Masters students at Australian higher education institutions looking to complement their discipline knowledge with hands-on HPC experience. Pawsey staff run an intensive week of hands-on training based on the existing Pawsey training program expanded to include other topics which would assist students in their projects, such as an introduction to Git and Python.

As part of our outreach activities, Pawsey hosted its first Careers Night in July in an endeavour to encourage year 10 students to pursue Science, Technology, Engineering and Mathematics (STEM) subjects in university. The careers night saw presentations from Pawsey staff, researchers and industry representatives - all with a story to tell and how studying a STEM subject have influenced their lives. After the presentations, the students spent their evening networking with the presenters and additional Pawsey staff to answer their curious questions.

Table 4. Pawsey outreach activities to bridge gaps.

	2016	2017	2018
Pawsey Roadshow	56	139	26*
Student Summer Internship	16	14	12*
Career Night	-	-	52
Pawsey Open Day	300	-	500
Community Tours	28	66	43[1]

[1] as of October 2018

Events such as the Careers Night reinforce Pawsey's focus on STEM and its importance in growing the minds of future scientists. It gives them insight into potential career paths and the ability of HPC to be part of any domain.

Similar to the Careers Night, every two years, Pawsey opens its doors to host a *Pawsey Open Day* as part of Australia's National Science Week. This day targets Perth's families of all ages and local government to raise awareness amongst the community about the benefits of supercomputers and the ground-breaking science Pawsey enables in Australia.

The Open Day includes back-to-back tours of the facility, 'designing a supercomputer' competition, researcher presentations and science activities (including child-friendly games that demonstrate parallelisation) to highlight the opportunities and encourage the ambition of young minds to work, research or be involved in HPC.

As a consequence of the first Open Day, Pawsey Community Tours are run every month and is an opportunity for the Perth residents to explore the Tier-1 facility that is sitting in their backyard. Any person over the age of 12 can tour through the Centre, which is predominantly guided by Pawsey's Head of Supercomputing. Tour groups are briefed with a *Science Showcase* of the work undertaken at the Centre, followed by a walkthrough of the supercomputing cell, I/O cell and tape cell.

Table 4 outlines the outreach activities discussed here and their corresponding attendance for the year.

7. SUMMARY AND FUTURE WORK

Pawsey has seen a marked change in the nature and size of its training and education program to better reflect the requirements of potential and existing Pawsey users. Through various feedback mechanisms, we seek to continue this change. In the future, we expect to increase the range of self-guided domain-specific training offerings, such as the recent bioinformatic training workshop we hosted on the 7th September 2018. This included specific material on running bioinformatics software from containers using our Nimbus cloud.

Similarly, we are planning to significantly increase the number and scope of training offered as online courses. Pawsey Supercomputing Centre experts have found the organisation of those training very effective, both in terms of the attendance and the overall cost. We also recognise a large potential of those training activities. Although there is a number of similar webinars and online pieces of training available worldwide, most of them are hardly accessible due to the time difference.

We are also planning to continue the National Training Programme as this creates unique possibilities to reach out to research groups and to understand their particular interests and needs.

8. ACKNOWLEDGMENTS

We would like to acknowledge all past and present Pawsey Supercomputing Centre staff who actively contributed to the development and running of our training and outreach activities over the years.

9. REFERENCES

[1] Pawsey Supercomputing Centre Training Github Repository, https://github.com/PawseySC

[2] Software Carpentry, https://software-carpentry.org

[3] JupyterHub from Project Jupyter, http://jupyter.org/hub

[4] Pawsey Supercomputing Centre Annual Report 2016-2017

[5] Software Carpentry - Instructor training workshop https://carpentries.github.io/instructor-training/06-feedback/index.html

Trends in Demand, Growth, and Breadth in Scientific Computing Training Delivered by a High-Performance Computing Center

Ramses van Zon
Marcelo Ponce
Erik Spence
Daniel Gruner
rzon@scinet.utoronto.ca
mponce@scinet.utoronto.ca
ejspence@scinet.utoronto.ca
dgruner@scinet.utoronto.ca
SciNet HPC Consortium, University of Toronto
Toronto, Ontario, Canada

ABSTRACT

We analyze the changes in the training and educational efforts of the SciNet HPC Consortium, a Canadian academic High Performance Computing center, in the areas of Scientific Computing and High-Performance Computing, over the last six years. Initially, SciNet offered isolated training events on how to use HPC systems and write parallel code, but the training program now consists of a broad range of workshops and courses that users can take toward certificates in scientific computing, data science, or high-performance computing. Using data on enrollment, attendence, and certificate numbers from SciNet's education website, used by almost 1800 users so far, we extract trends on the growth, demand, and breadth of SciNet's training program. Among the results are a steady overall growth, a sharp and steady increase in the demand for data science training, and a wider participation of 'non-traditional' computing disciplines, which has motivated an increasingly broad spectrum of training offerings. Of interest is also that many of the training initiatives have evolved into courses that can be taken as part of the graduate curriculum at the University of Toronto.

CCS CONCEPTS

• **Social and professional topics** → **Computing education**; **Model curricula**; **Student assessment**; *Computational thinking*; *Computing education programs*; Accreditation;

KEYWORDS

Scientific Computing, Training and Education, Graduate Courses, Certificates, Curricula

https://doi.org/10.22369/issn.2153-4136/10/1/9

1 MOTIVATION

Not that long ago, many larger scientific computations for academic research were performed on clusters built by researchers using commodity components strung together by a commodity network, i.e. Beowulf clusters. The research groups building these clusters were also the ones using it. The knowledge of how to build, program and use these systems were transmitted from one graduate student to the next, with perhaps the briefest of instructions posted on a website. Every system was a bit different, which did not matter too much, as the knowledge to use the system was already in-house, and the systems were maintained by a member of the research group that happened to have affinity with the technology.

This era of self-built clusters was driven by the need for more computational resources and faster computations than a single workstation could provide, which is reflected in the name of the field, High Performance Computing (HPC). It was made possible by the availability of relatively cheap commodity hardware and technical knowledge in the research groups. This practice of self-built, maintained, and documented systems often took place in traditionally highly technical areas such as engineering and the physical sciences (physics, astronomy, chemistry).

However, the demand for computational resources in academic research has not stopped increasing, and is not exclusive to the physical sciences. Part of this demand is now driven by the increase of data availability in many disciplines and industries: bioinformatics, health sciences, social science, digital humanities, commerce, astronomy, high energy physics, etc.

With the increased demand for scientific computational resources build-your-own clusters were no longer sufficient. A select few researchers in some countries had access to large national systems, but most researchers did not. To solve this issue, many researchers started using shared clusters, first within their department, then their own university or HPC consortia in which several universities collaborated, and finally nationally and cross-nationally available shared computing resources (XSEDE [17], Compute Canada [4], PRACE [11]). But this brought about a second issue, that the knowledge on how to use these systems no longer resided within the research groups. This especially put 'non-traditional' fields at a

disadvantage: while they now had access to great computational resources, they lacked the institutional knowledge required to use these as effectively as possible and missed the opportunity to develop this knowledge within their own research groups.

This computational knowledge gap is the motivation for training provided by experts that are situated at the centres that provide the HPC resources. Larger computing centres [2, 5, 8, 15] have usually engaged in these kinds of training events. But such endeavours do not automatically tie in with universities' educational systems. This implies that participants of such training events do not get formal recognition of the skills and knowledge they learned, unless an explicit mechanism for this is put in place, such as badges and certificates.

This paper focuses on SciNet's training efforts. SciNet is the HPC centre at the University of Toronto. It hosts some of the largest academic supercomputers in Canada. Since SciNet's operations started in 2009 [7], courses have been taught on scientific technical computing, high performance computing, and data analysis for the Toronto-area research community.

As will be detailed below, what started as small-scale training sessions on parallel programming has grown into a large, successful program consisting of seminars, workshops, courses, and summer schools, delivered by computational science experts. Graduate courses are given in partnership with other departments, and all events are part of a program in which participants work towards certificates in scientific computing, high-performance computing, or data science. In this paper, we use the (anonymized) enrollment data of SciNet's online learning system, which includes attendance records for almost 1800 students over the last seven years, augmented by information on field of study and gender, to get a picture of the growth in amount and depth of training in general scientific computing, as well as trends in training in data science and high performance computing[1].

2 TRAINING FORMATS

The training at SciNet takes place in various different formats. These will be briefly describe here, including their intended usage and their advantages and disadvantages.

Common to all training events is a substantial on-line presence which supports the learning and administration of the program. SciNet's training and education site[2] contains lecture videos, slides, links, forums and other electronic material, freely publicly available and organized by course [14].

On the site, users of the SciNet facilities can log in with their SciNet account, while students that are not users must be assigned a temporary account. Logging in is not required to access the content, but is required to enroll in courses, to take tests, to submit assignments (for the graduate-style courses), and to track progress towards earning certificates.

All of SciNet's training is free for anyone working in academia.

[1]The curated and anonymized data can be requested from the authors for academic research purposes.
[2]courses.scinet.utoronto.ca

2.1 Seminars

Seminars are short, one-hour sessions. Sometimes they are about a technical topic, sometimes they are a research presentation. The format of seminars may not be ideal for knowledge transfer and training, but it is a good way learn about something new. Furthermore, many of these seminars happen at the monthly SciNet User Group ("SNUG") meetings, which are an opportunity for SciNet users to come together and exchange experiences.

2.2 Workshops

Workshops are usually half a day to one day long, and focus on a very specific topic. Examples of topics are "Parallel I/O", "Relational Database Basics", "Intro to the Linux Shell", "Intro to HPC", and "Intro to Neural Networks". Such workshops are given a few times throughout the year, and typically have a hands-on component.

The annual summer school (further described below) consists of a carefully selected collection of these kinds of workshops.

SciNet also provides occasional workshops for other organizations, such as the Fields Institute, Creative Destruction Lab, and the Chemical BioPhysics symposium in Toronto.

2.3 Graduate-style courses

The graduate-style courses share a common approach which is focussed on the practical application of presented materials. These courses typically have two lectures per week of one hour each. In addition, each week, students are given a programming assignment, with a due date one week after, and feedback is given to the students in the following week. These assignments are designed to help absorb the course material. The average of the assignments also make up the final grade. To further support the students' learning, there are office hours, online forums, and instructor email support.

Initially, these courses ran for four weeks at a time, a format that coincided with the "mini" or "modular" courses given by the Physics Department and the Astrophysics and Astronomy Department of the University of Toronto. Topics for these mini-courses included "Scientific Software Development and Design", "Numerical Tools for Physical Scientists", "High Performance Scientific Computing", "Introduction to Programming with Python", "Numerical Computing with Python", "Advanced Parallel Scientific Computing", "Introduction to Machine Learning", and "Introduction to Neural Networks".

Some of these graduate-style mini-courses have grown into full-fledged, term-long graduate courses. The process of creating these recognized graduate courses is described in more detail in section 3.2 below.

2.4 Summer schools

The annual one-week long summer school is a flagship training event for graduate students, undergraduate students, postdocs and researchers who are engaged in compute intensive research. SciNet's first summer school was given in 2009 and was called a "Parallel Scientific Computing" workshop. As the program in table 1 shows, it was heavily focussed on HPC, parallel programming, and applications in astrophysics.

These days, SciNet's summer school is part of the Compute Ontario Summer School on Scientific and High Performance Computing. Held geographically in the west, centre and east of the

First day
Welcome and Introduction to Parallel Scientific Computing
Introduction to OpenMP with brief C tutorial
Second day
Introduction to OpenMP, continued
Introduction to MPI
Third day
Map Making
Compressible Hydrodynamics
Fourth day
OpenMP N-Body
MPI N-Body
Fifth day
CUDA N-Body
C-Blocks and Erlang
Advanced Topics: Additional Resources,
Performance tools and exotic architectures

Table 1: SciNet's first summer school in 2009 focussed on Parallel Scientific Computing and placed emphasis on scientific applications such as in astrophysics.

province of Ontario, the summer school provides attendees with the opportunity to learn and share knowledge and experience in high performance and technical computing on modern HPC platforms. The central edition is the continuation of the SciNet summer school. Not only is the school organized in a wider context, its program has expanded as well. As table 2 shows, in 2018 there were three streams in the Toronto edition, and a wide variety of topics, from shell programming to data science, machine learning and neural networks, biomedical computing, and, still, parallel programming.

This type of event not only benefits the students and participants of the summer school, but also enables collaborations between departments and consortia, as part of the training was delivered in partnership with colleagues from SHARCNET [16] and the Centre for Addiction and Mental Health [3].

SciNet participates also in the International HPC Summer School [6], sending a few instructors and 10 students to this competitive one-week program every year.

2.5 Guest Instructors

In the capacity of "guest instructors", SciNet also delivers a 7-week module in an undergraduate "Research Projects Course" from the Department of Physics at the University of Toronto. Topics include an introduction to High Performance and Advanced Research Computing, Data Science, and Scientific Visualization.

SciNet also occasionally provides guest lectures in other courses.

3 CERTIFICATES AND CREDITS

3.1 Certificate Programs

Since December 2012, SciNet has offered recognition to attendees of its training events in the form of SciNet Certificates [13]. Requirements for these certificates are based on the number of credit-hours of SciNet courses a student has successfully completed. For a short course (typically a day long or shorter, without homework), a lecture hour counts as one credit hour; for a long course with homework due between sessions, a lecture hour counts as 1.5 credit hours.

There are currently three certificate programs, with the following descriptions:

Certificate in Scientific Computing: Scientific computing is now an integral part of the scientific endeavour. It is an interdisciplinary field that combines computer science, software development, physical sciences and numerical mathematics. This certificate indicates that the holder has successfully completed at least 36 credit-hours worth of SciNet courses in general scientific computing topics.

Certificate in High Performance Computing: High Performance Computing, or supercomputing, is using the largest available computers to tackle big problems that would otherwise be intractable. Such computational power is needed in a wide range of fields, from bioinformatics to astronomy, and big data analytics. Since the largest available computers have a parallel architecture, using and programming high performance computing applications requires a specialized skill set. Those earning this certificate have completed at least 36 credit-hours of SciNet courses in high performance computing topics.

Certificate in Data Science: The SciNet Certificate in Data Science attests that the holder has successfully taken at least 36 credit-hours of data science-related SciNet courses. The latest certificate launched, it is indeed one of the fastest growing in popularity, clearly displaying the growing interest in data-science-related computational fields, such as artificial intelligence and deep learning.

3.2 For-Credit Graduate Courses

By partnering with different institutions at the University of Toronto, many SciNet courses have been consolidated into recognized graduate courses that students enrolled in Masters and Doctorate programs can take as part of their graduate curriculum. So far, SciNet has started three graduate courses recognized at the University of Toronto: PHY1610 "Scientific Computing for Physicists" in partnership with the Department of Physics, MSC1090 "Introduction to Computational Biostatistics with R" in partnership with the Institute of Medical Science, and EES1137 "Quantitative Applications for Data Analysis" in partnership with the Department of Physical and Environmental Sciences at the University of Toronto at Scarborough. These courses are in principle open to students of other departments as well, and indeed attract students from Physics, Chemistry, Astrophysics, Ecology and Evolutionary Biology, Engineering, Computer Science, and others.

Some of the shorter graduate-style courses are still taught as well, and are recognized by a subset of the departments at the university as "mini" or "modular" courses.

In fact, the physics graduate course started out as a collection of three such modular courses, on "Scientific Software Development and Design", "Numerical Tools for Physical Scientists", and "High Performance Scientific Computing", respectively. These three modules were recognized by the Physics, Astrophysics, and Chemistry Departments, and were subsequently merged into a single, one-term course with a Physics designation. This meant the course

HPC Stream	Data Science Stream	BioInformatics/Medical Stream
First day		
Welcome and Introduction to HPC and SciNet		
Programming Clusters with MPI	Introduction to Linux Shell	Python for MRI Analysis
Second day		
Programming Clusters with MPI (cont.)	Introduction to R Introduction to Python	Image Analysis at Scale HCP with HPC: Surface Based Neuroimaging Analysis
Third day		
Programming GPUs with CUDA	Parallel Python Machine Learning with Python	PLINK Next Generation Sequencing
Fourth day		
Programming GPUs with CUDA (cont.)	Neural Networks with Python Scientific Visualization Suites	RNASeq Analysis R for MRI Analysis
Fifth day		
Shared Memory Parallel Programming with OpenMP	Debugging, Profiling Bring-Your-Own-Code Lab	Public Datasets for Neuroscience Biomedical Hacking

Table 2: SciNet's latest (and largest) summer school, held in June 2018. This summer school had three parallel streams: the traditional High-Performance Computing, one on Data Science and a stream on BioInformatics/Medical applications, which was added in 2017. Details of the courses covered in the school can be found in the Summer School website [12].

was now listed in the graduate curriculum and drew a larger audience. Similar tracks were followed to establish the other full-term graduate courses, but under different partner departments.

To avoid a growing teaching burden, teaching assistant support is provided by the partner department. The course instructors are still SciNet analysts, now hired as sessional lecturers for the purpose of these courses.

These for-credit courses follow the same format as our other graduate-style courses, with assignment-based learning and evaluation, with on-line support in terms of forums, email, and availability of materials including lecture recordings. There is no final exam for these courses, although for some courses a mid-term exam is set.

It should be noted that designing a course in scientific computing for students in a non-traditional field such as medicine and biology poses its own unique problems, which are discussed in more detail elsewhere [9].

4 ENROLLMENT DATA

The start of the certificate program in 2012 required a more comprehensive online registration and learning management system than SciNet's previous Drupal-based site could provide. The replacement system is based on ATutor [1], an open-source web-based learning management system. This system was augmented with a few in-house-developed modules for event management, certificate programs, and integration with the LDAP authentication server used for SciNet's computing resources. In addition to the LDAP authentication for users with computing accounts, there are temporary accounts, which are authenticated separately through a local database.

The site keeps track of all courses in which users are enrolled, and which of those courses have been completed. Every course is also categorized. For clarity, in this paper, a restricted set of four categories is used: "High Performance Computing", "Scientific Computing", "Data Science" and "Seminar".

Each course consists of a set of 'events', e.g. lectures, meetings, or workshops, which in total determine the length of the course and when it was given. Attendance of events by people without an account on the site is allowed, and is tracked by entering the number of 'anonymous attendees' for each event. The system also records whether users have earned a certificate in Scientific Computing, High Performance Computing or Data Science, and the date when they obtained it.

Unfortunately, the system was not setup to gather all the information needed to, for instance, investigate the distribution of training demand over different genders and different fields. To be able to investigate this, it was necessary to assign genders and fields to users of the site. For anonymous attendees, this assignment is impossible, but for the users with accounts on the system, these attributes were reconstructed as well as possible given the information that was in the system.

The gender assignment for accounts whose gender was not known was performed by checking first names against an online database that returns the most likely gender (https://genderize.io). In the end, the data set of 1776 users was determined to contain 1047 males and 567 females, leaving 162 unknown.

The assignment of fields of study was done in a variety of ways. For users with accounts on computing resources, the research group is known and for most groups their field of study is known. Temporary users were often asked for a description of what they do, from which the research field could be deduced. For the graduate courses, the field of the user's 'host' department was used. Sometimes the email address of a user revealed his or her field. The assignment of the field of study was the most laborious part of the data analysis, only made possible by using a restricted set of categories of fields of study: 'engineering', 'physics', 'chemistry', 'earth science', 'computer science', 'mathematics', 'medicine', 'biology', 'economics', 'humanities' and 'social science'. In the end, 1577 of the 1776 user accounts on the site could be assigned a field of study.

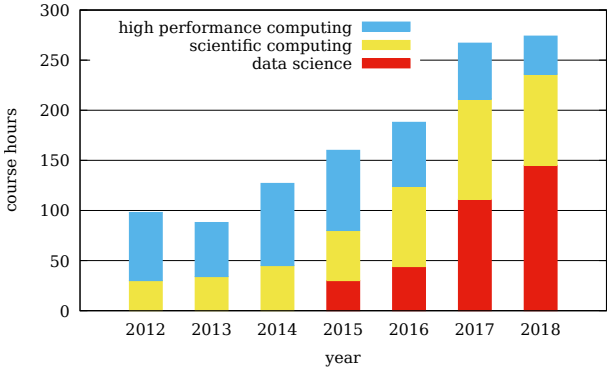


Figure 1: Growth in total course hours delivered by SciNet. The shaded regions in each bar show the course hours in the topics scientific computing, high performance computing, and data science.

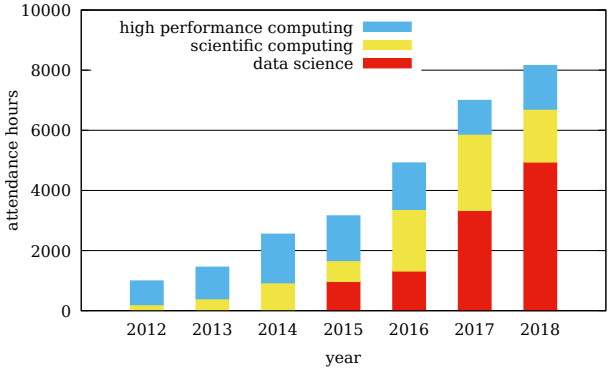


Figure 2: Break-down of the attendance for the years 2012-2018 by the three high-level categories of SciNet courses: scientific computing, high performance computing, and data science.

The data covers the period from 2012 to July 2018. Thus, the statistics for 2018 do not constitute a full year. Furthermore, the 2012 data was imported from the older courses website, with attendance numbers added by hand. Virtually all attendees in 2012 were therefore recorded as anonymous attendees, for which neither gender or field of study is known.

While these assignments of gender and field of study used some of the user's personal attributes in the system, once the assignment was done the personal information was no longer needed. The subsequent analysis of trends was performed having removed names, emails, institutions, supervisor information, and any other identifying information, using only anonymous data.

5 RESULTS

Before presenting the results, two central notions must be introduced: the first is a course hour, which is an hour in which an event was held, be it a lecture, seminar, presentation, meeting, or some other type of event. The number of course hours is a measure of the teaching effort. The second notion is that of an attendance hour, which is one person attending one hour of training. For example, if a training event of two hours has ten attendees, that training event counts as twenty attendance hours. Attendance hours are a measure of the effect of the teaching and to some extent an indicator of the demand for that training.

5.1 Overall Growth and Distribution by Topic

Figure 1 shows the overall growth of the number of hours of training events delivered by SciNet over the years. One sees a general increasing trend from about 100 course hours in 2012 to over 250 course hours in 2018. One can also see a levelling off of this trend in the last two years. This can be attributed mostly to limits in available human resources.

The same figure also shows how many of these training hours were devoted to the three main categories of topics that are taught: data science, high performance computing, and scientific computing. It may seem that no data science was taught before 2015, but actually the distinction between data science and scientific computing was not yet made at that time. One sees a strong surge of data science training, which now comprises more that 50% of SciNet's training effort.

Figure 2 shows the overall growth of the number of attendance hours over the years. Once more, one sees a general increasing trend from about 1000 attendance hours in 2012 to over 8000 in 2018. The latter number is an underestimate, as the attendance of our fall classes has not yet been counted. A rough estimate based on the enrollment numbers suggests that the final number of attendance hours in 2018 will be closer to 10,000. There is no sign of attendance levelling off, from which we may conclude that the demand for training continues to increase.

As in the previous figure, the breakdown by high-level category (data science, high performance computing, and scientific computing) is also shown. Overall, all topics see increasing attendance numbers, but data science is the fastest growing category,

5.2 Trends in Participation by Gender and Scientific Field

Whereas the previous section looked at what is taught, we are also interested in who is taking SciNet's courses.

Of particular interest is the gender balance. Figure 3 shows the percentages of different genders. One must keep in mind that the gender was in many cases inferred rather than collected. Because of that, it was not possible to go beyond a basic binary division of genders. The data nonetheless shows a trend from very little female participation in 2012 to about 40% female participation in 2018.

The disciplines that require or use scientific computation are changing as well, and the training data supports this. Table 3 shows the relative participation of students subdivided into 11 groups corresponding to their field of study. The largest and second largest groups are highlighted in orange and yellow in the table. A very striking trend is apparent here. Whereas the majority of students previously came from engineering and physical sciences, in recent years students in life sciences (biology and medicine) have become

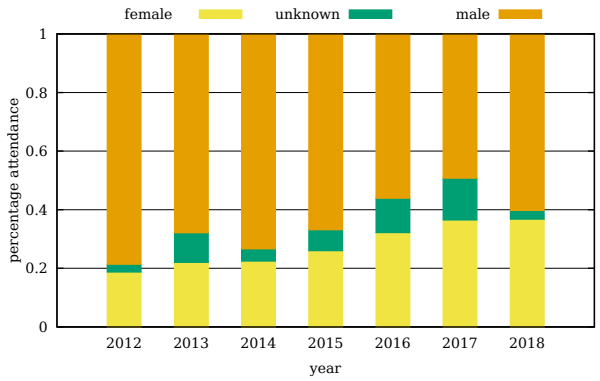


Figure 3: Division of attendance hours in SciNet training by gender (based on an approximate gender determination – see text).

	data science	HPC	scientific computing
engineering	20%	46%	34%
physics	15%	34%	51%
earth science	18%	41%	41%
chemistry	25%	41%	34%
computer science	12%	72%	16%
mathematics	21%	49%	30%
medicine	65%	17%	18%
biology	58%	16%	26%
economics	26%	56%	18%

Table 4: Relative percentage of participation in training in scientific computing (SC), data science (DS) and high performance computing (HPC) for different fields of study. There were too few data points for sensible results for social science and humanities.

the majority of participants in SciNet's training. It is worth mentioning that in absolute numbers, the engineering and physical science fields have not decreased, but the life sciences have simply contributed more to the growth in demand for training.

One might expect that the increase in attendance in data science training is correlated with the increase in attendance from biology and medicine. Table 4 shows the division among scientific computing, data science and high performance computing for each of the fields of study in the previous table. This table confirms what one might already expect: the life sciences are more concerned with data science, while the physical sciences and engineering have a greater interest in scientific computing and high performance computing.

5.3 Summer School Statistics

The annual summer school has been highly successful and has been growing in number of sessions (compare table 1 and table 2) and in attendance. From 2012 to 2018, the attendance has grown from 35 people to 215.

	2013	2014	2015	2016	2017	2018
engineering	**34%**	14%	15%	20%	12%	17%
physics	29%	**36%**	**30%**	24%	18%	12%
earth science	10%	8.1%	8.8%	6.0%	3.4%	2.3%
chemistry	6.7%	16%	2.7%	4.4%	3.4%	4.4%
computer science	1.2%	1.4%	0.3%	1.6%	1.7%	0.8%
mathematics	0%	0%	0%	2.1%	0.8%	2.5%
medicine	10%	17%	20%	**25%**	**35%**	**40%**
biology	8.0%	4.8%	22%	16%	24%	18%
economics	0%	0.9%	0%	0.4%	0.4%	2.2%
social science	0%	0%	0%	0%	0.2%	0.2%
humanities	0%	0.1%	0%	0%	0%	0%

Table 3: Relative percentage of the attendance in SciNet training by researchers from different fields, separated by year. The two fields with most participation in a given year are highlighted. There was not enough statistics to be able to include 2012 in this table.

After completing at least 3 days of the summer school, participants in the summer school receive a certificate of attendance. These are special summer school certificates that are separate from the SciNet certificates. In 2012, 20 certificates were awarded, but by 2018, this number was 135. This growth is partly due to the inclusion of a data science stream and a biomedical stream.

The school is offered for free, but without support for travel, lodging or meals. It is therefore not surprising that most participants are from the Toronto area, although there are always some who travel to attend the event. In 2018, there was a sizable number of attendants from outside Toronto (60), from outside of Ontario (15) and even from outside Canada (5).

This event is in high demand: in 2018, within one day of opening the registration, there were over 100 registrations, and just one week later, the 200 registration mark was crossed. In the end, 304 people signed up. The fact that there is no charge for this event means that not everyone attends; attendance rates of 70% are typical. This helped the people on the waiting list, most of whom could be accommodated in the end.

5.4 SciNet Certificates Statistics

As was mentioned above, SciNet issues certificates in scientific computing, data science, and high performance computing to students who have taken at least 36 hours of courses in the respective category. As figure 4 shows, over 200 certificates have been issued so far. The data science certificate still has the lowest number (it was only started in 2015), but is the fastest growing category. The graph shows a stagnation in the number of high performance computing certificates. It is possible that this is due to a shift in demand from HPC to data science, but it could also be related to the decrease in the number of HPC courses that are on offer; as figure 1 shows, the number of HPC training hours has decreased in the last two years.

5.5 Retention

The statistics of the number of certificates give an indication of the retention of students taking SciNet courses, as one course is

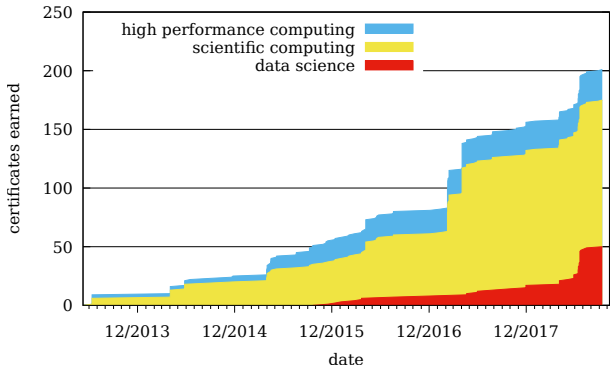


Figure 4: Certificates earned in the last six years (cumulative) in each of the categories scientific computing, data science and high performance computing.

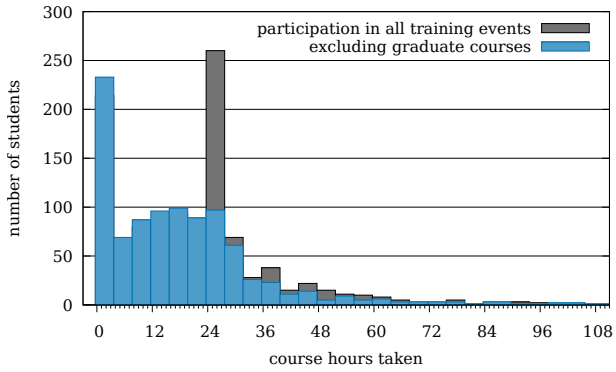


Figure 5: Histogram of the number of course hours taken by students. The bin width is 4 hours. Gray bars include all training activities, while blue bars exclude for-credit courses.

not enough to earn a certificate. We can get more insight into the retention rates by considering the number of course-hours taken by individuals. Figure 5 shows the number of students that spent a given number of hours in SciNet training, counted in bins of 4 hours wide (i.e., the first bin counts students that spent between 1 and 3 hours in SciNet courses, the second counts students that took between 4 and 7 hours, etc.). It also shows those same statistics when the graduate courses are taken out.

A few things can be observed. One is that a substantial number of attendees (about 230/1800) come to only one to three hours of training. Since there are very few individual training events with a duration of more than three hours, the remaining attendees are returning students. The second observation is that there is a large peak at 24 hours of attendence. This peak disappears when the graduate courses are taken out, which makes sense as all our graduate courses have 24 hours of lectures. What then remains is a broad distribution, centered around 16 hours. There is also a noticeably long tail, which shows that a number of students attend a lot of training.

6 DISCUSSION

In this paper, we have shown the growth and demand-driven change in focus of SciNet's training program, from the year 2012 to 2018. In the analysis of the enrollment data, we had to infer gender and field of study (we are exploring ways of implementing ways for users to self-identified their gender and field of study). The data shows a large increase in life science participation as well as in female participation. There has been an overall growth trend both in the amount of training offered and in the attendance. In particular, training in the field of data science has shown remarkable growth. The tremendous increase in participation shows the large demand for this kind of training.

One might wonder whether or not this kind of training should be part of the graduate curriculum [10], or whether perhaps it should be provided by the Computer Science department. While parallel computing and concurrency are active lines of research in computer science, users are often interested in practical techniques and desire training in efficient ways to enable larger scientific computations. Other departments might want to teach courses on data science and scientific computation, but may not have the required knowledge. Computing centers usually have the required expertise, and can provide much of the necessary training, but may not be in a position to give or create for-credit university courses. At least in the case of SciNet, through partnering with other departments, several graduate courses in Scientific Computing have been developed.

Overall, this has been a successful program which continues to grow.

ACKNOWLEDGEMENTS

SciNet is funded by the Canada Foundation for Innovation, the Government of Ontario, and the University of Toronto.

REFERENCES

[1] ATutor. 2018. ATutor Learning Management System. https://atutor.github.io. (2018). Accessed: 2018-09-24.
[2] Barcelona Supercomputing Center. 2018. BSC-CNS Education. https://www.bsc.es/education. (2018). Accessed: 2018-09-28.
[3] CAMH. 2018. The Centre for Addiction and Mental Health | CAMH. http://www.camh.ca/. (2018). Accessed: 2018-09-28.
[4] Compute Canada. 2018. Compute Canada | Calcul Canada. https://www.computecanada.ca/. (2018). Accessed: 2018-09-28.
[5] EPCC. 2018. EPCC Education & Training. https://www.epcc.ed.ac.uk/education-training. (2018). Accessed: 2018-09-28.
[6] IHPCSS. 2018. International High Performance Supercomputing Summerschool. https://www.ihpcss.org/. (2018). Accessed: 2018-09-28.
[7] Chris Loken, Daniel Gruner, Leslie Groer, Richard Peltier, Neil Bunn, Michael Craig, Teresa Henriques, Jillian Dempsey, Ching-Hsing Yu, Joseph Chen, L Jonathan Dursi, Jason Chong, Scott Northrup, Jaime Pinto, Neil Knecht, and Ramses Van Zon. 2010. SciNet: Lessons Learned from Building a Power-efficient Top-20 System and Data Centre. *Journal of Physics: Conference Series* 256, 1 (2010), 012026. http://stacks.iop.org/1742-6596/256/i=1/a=012026
[8] Pittsburg Supercomputer Center. 2018. PSC Training & Education. https://www.psc.edu/training-education. (2018). Accessed: 2018-09-28.
[9] Marcelo Ponce, Erik Spence, Daniel Gruner, and Ramses van Zon. 2018. Bridging the Educational Gap between Emerging and Established Scientific Computing Disciplines. *JoCSE, in press* (2018).
[10] Marcelo Ponce, Erik Spence, Daniel Gruner, and Ramses van Zon. 2018. Scientific Computing, High-Performance Computing and Data Science in Higher Education. *JoCSE, in press* (2018).
[11] PRACE. 2018. Partnership for Advanced Computing in Europe. https://www.prace-ri.eu/prace-in-a-few-words/. (2018). Accessed: 2018-09-28.
[12] SciNet HPC Consortium. 2018. CO Summer Schoool Central. https://courses.scinet.utoronto.ca/368. (2018). Accessed: 2018-09-28.
[13] SciNet HPC Consortium. 2018. SciNet's Certificate Program. https://www.scinethpc.ca/scinet-certificate-program/. (2018). Accessed: 2018-07-01.

[14] SciNet HPC Consortium. 2018. SciNet's Education website. https://courses.scinet.utoronto.ca. (2018). Accessed: 2018-07-01.
[15] SciNet HPC Consortium. 2018. SciNet's Training, Outreach and Education. http://www.scinethpc.ca/training-outreach-and-education. (2018). Accessed: 2018-07-01.
[16] SHARCNET. 2018. SHARCNET website. https://www.sharcnet.ca/. (2018). Accessed: 2018-09-28.
[17] XSEDE. 2018. XSEDE website. https://www.xsede.org/for-users/getting-started. (2018). Accessed: 2018-09-28.

Evaluating Active Learning Approaches for Teaching Intermediate Programming at an Early Undergraduate Level

Dhruva K. Chakravorty
High Performance Research Computing (HPRC) at Texas A&M University
College Station, Texas, USA

Marinus "Maikel" Pennings
HPRC at Texas A&M University
College Station, Texas, USA

Honggao Liu
HPRC at Texas A&M University
College Station, Texas, USA

Zengyu "Sheldon" Wei
HPRC at Texas A&M University
College Station, Texas, USA

Dylan M. Rodriguez
HPRC at Texas A&M University
College Station, Texas, USA

Levi T. Jordan
HPRC at Texas A&M University
College Station, Texas, USA

Donald "Rick" McMullen
HPRC at Texas A&M University
College Station, Texas, USA

Noushin Ghaffari
HPRC at Texas A&M University
College Station, Texas, USA

Shaina D. Le
HPRC at Texas A&M University
College Station, Texas, USA

Derek Rodriguez
HPRC at Texas A&M University
College Station, Texas, USA

Crystal Buchanan
HPRC at Texas A&M University
College Station, Texas, USA

Nathan Gober
HPRC at Texas A&M University
College Station, Texas, USA

ABSTRACT

There is a growing need to provide intermediate programming classes to STEM students early in their undergraduate careers. These efforts face significant challenges due to the varied computing skill-sets of learners, requirements of degree programs, and the absence of a common programming standard. Instructional scaffolding and active learning methods that use Python offer avenues to support students with varied learning needs. Here, we report on quantitative and qualitative outcomes from three distinct models of programming education that (i) connect coding to hands-on "maker" activities; (ii) incremental learning of computational thinking elements through guided exercises that use Jupyter Notebooks; and (iii) problem-based learning with step-wise code fragments leading to algorithmic implementation. Performance in class activities, capstone projects, in-person interviews, and participant surveys informed us about the effectiveness of these approaches on student learning. We find that students with previous coding experience were able to rely on broader skills and grasp concepts faster than students who recently attended an introductory programming session. We find that, while makerspace activities were engaging and explained basic programming concepts, they lost their appeal in complex programming scenarios. Students grasped coding concepts fastest using the Jupyter notebooks, while the problem-based learning approach was best at having students understand the core problem and create inventive means to address them.

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

CCS CONCEPTS

• Social and professional topics → Computing education programs; Computing literacy; Computational thinking; Informal education; K-12 education; • Applied computing → Interactive learning environments;

Keywords

Undergraduate education, informal education, computing literacy, Python, Jupyter notebooks, active learning, problem-based learning, scaffolding strategies, Raspberry Pi

1. INTRODUCTION

Offering incoming (freshman) undergraduates opportunities to learn basic computing skills is an idea that is rapidly gaining popularity in higher education. Due to the rapid proliferation of computing in science, technology, engineering, and mathematics (STEM) disciplines, these efforts need to be supplemented with intermediate level student training programs as well [6, 7]. Such intermediate programming experiences need to provide a solid skill foundation in order to help early undergraduates develop complexity in programming skills. Proficiency in computer science widely differs from student to student, depending on their previous experiences with computing. In this paper, we investigate how important previous exposure to computing concepts is to computing education. While a wide range of programming constructs are considered introductory activities, a defined standard

for intermediate- level proficiency is currently lacking. The Python programming language helps address some of these concerns. The language has an easy-to-comprehend syntax and can grow in complexity to support analysis in numerous STEM disciplines. In addition, inexpensive computers, like the Raspberry Pi, open possibilities for educators to couple makerspace activities with Python programming in the classroom to serve as a scalable platform on which students can develop an intermediate-level skill set in computing. Informal efforts that use well-reviewed pedagogical approaches to education have been found to encourage participation in and adoption of computational thinking [3, 13, 23, 24]. Previous studies have shown that projects connected to greater societal impact or include elements of physical creativity are more likely to appeal to a broader audience [14, 16]. Using participatory technologies such as visualization, modeling, and robotics provide a number of opportunities in this space. By combining increasingly easy-to-access computing resources with a scaffolded teaching approach [10], the barrier to entry can be reduced [4, 20]. Reducing this barrier to entry is particularly relevant for informal efforts hosted by high performance computing (HPC) centers that support users who have a diverse range of research needs and computing prowess.

In this study, we explore different pedagogical approaches that utilize these technologies to introduce learners to complex programming scenarios suitable for the intermediate level [2, 17, 19]. In the subsequent sections of this paper, we present the conceptual framework of our program, and describe the methods used to evaluate three learning approaches. The paper next describes our efforts toward assessing the success and pitfalls of these approaches using capstone exercises, interviews and evaluations. We finally discuss the lessons learned over the previous year and summarize our findings in conclusion.

2. EXPERIMENTAL DESIGN

Instructing groups of students on coding practices who have a diverse range of skill sets and educational backgrounds remains a challenge in computing education. Scaffolded instruction methods offer avenues to support students with varied skill sets [10]. Active learning has been shown among high-ability trainees to produce significantly higher levels of metacognitive activity than procedural training. Here, we compare and contrast the benefits of well-reviewed approaches to scaffolded instruction and innovative active learning exercises in the context of Python programming over a week-long training session. The program's goals are to (a) increase participant engagement (b) develop a participant's understanding of complex computing concepts, and (c) pro- vide participants with a learning environment that employs hands-on exercises.

Complex coding projects can overwhelm the new or intermediate learner. Such learning is best facilitated in a tiered format where information is provided, comprehended, analyzed and employed before moving to the next step. Traditional approaches utilize handouts for students, presenting the code on the screen, and perhaps provide prepared versions of the code. Jupyter Notebooks provide interesting avenues in this space, because they provide a number of useful features [8]. Unlike traditional applications, these notebooks run as interactive web-browser applications that allows users to write Python code in cells. The output from the cells is easy to access and visualize because of its closeness to the Python code. Students can see the output from each portion of code as they are writing it. This allows the instructor to keep students engaged through exercises that focus on smaller pieces of a complex code and demonstrate how the output from each cell combines to form the larger program structure. The last advantage of Jupyter Notebooks is that, as a web-based effort, these notebooks are platform agnostic and can be run on any computer! This attribute makes them tremendously portable. A notebook that runs on a Raspberry Pi can be ported to run on a supercomputing cluster with graphical processing units [15].

There is a significant body of work describing the importance of makerspace activities and problem-solving approaches in informal education [22]. Specifically, here we explore three different approaches that use Raspberry Pi microcontrollers. We specifically report on the following qualitative and quantitative outcomes, (1) connecting coding to sensing and control of the real world through hands-on maker activities [14, 16]; (2) incremental learning of computational thinking elements [3, 13, 17, 23, 24] through guided exercises using Jupyter Notebooks [8]; and (3) problem-based learning with step-wise code fragments leading to a complete implementation of an algorithm in which students are presented a "narrative" program and goal, and work through converting specific objectives into code to write the program and achieve the goal.

To help ensure student engagement, these approaches used exercises in game design using the Raspberry Pi Hardware Attached on Top (HAT) sensor platform, image recognition using machine learning (ML) [11], and sharing secret messages using cryptography respectively. All three approaches included structured and unstructured components, handouts for students, and a number of advisors (1:4 ratio) available to assist if needed. In order to effectively judge the efficacy of each approach, we ensured that the exercises in these approaches did not build on each other. To avoid biases due to familiarity with an instructor, each approach was taught on a separate day by a different instructor (2 males and 1 female). Participants were informed of the nature of activities and approaches only on the day of the activity. This ensured that the participants were exposed to these learning activities for the first time on the day of the camp, and helped reduce artifacts arising from the participants having previous knowledge of these activities.

In order to create an early undergraduate environment that comprises of intermediate-level learners, we recruited a cohort of 23 participants who had recently graduated from high school or would do so in the near future. 7 of these 23 students were female. In all, 13 of these 23 students be- longed to groups that are traditionally underrepresented in computing. Recruiting was performed using social media, emails to listservers, our website and contacts at local school districts. Participant applications were managed via our website. As part of their applications, applicants were asked to completed a pre-training evaluation in the form of Likert Scale and open-ended questions. These questions helped establish the participant's familiarity with Python, programming, Linux and Raspberry Pi computers. Thanks to the exciting nature of offered projects, we received a number of applications from well-qualified participants. Selection to the training program was merit-based. All participants had basic Python programming skills defined by the ability to write scripts that employed loop constructs, had earned a GPA of above 3.5, and were interested in attending college. The participants also had some experience with the Linux operating system and text editors. The students belonged to two distinct learning groups. The first group of 12 students were self-identified intermediate-level Python learners. The second group included 11 learners who had participated in our introductory Python programming course 2 months prior to this exercise. For the purpose of brevity, these cohorts of participants are henceforth referred to as Group 1 and Group 2, respectively, in the remainder of this manuscript.

3. PEDAGOGICAL APPROACHES

Each participant received a Raspberry Pi that was preloaded with the Raspbian operating system, a Debian-based distribution of Linux. To ensure that all participants had a set of common Python programming skills, they were provided with the training material from our introductory programming camp. These materials contained information about introductory Python programming practices, the Linux operating system, GitHub, and the Gedit text editor. Trainees who had previously participated in our introductory camp program were taught using this material. Introductory computing skills were reviewed on the first day of the camp. Each slide in the lesson contained a small activity to allow for immediate application of the respective topic. To synthesize knowledge gained in the lesson, the students were tasked with their first maker activity; they performed variable assignments, basic arithmetic operations, and console output. The students created both string and integer variables to store numerical values. The purpose of these tasks was to teach students to explore through individual trial and error what actions could be executed on which data type. On completion of the first day, all students were able to successfully demonstrate the use of algorithms and loops in a review exercise. Surveys and in-person interviews further supported that all students were at a similar learning level after the first day's reviews. A brief description of the activities during the training exercises are provided in the Reproducibility Appendix in the order in which they were taught. Owing to the complexity of the topics covered, measures were taken to reduce the complexity of the problem set. Sessions included the use of PowerPoint slides for instruction and students received handouts containing pertinent information and review exercises. Instructors and their assistants were provided with handouts that included more details about the exercises along with possible solution sets.

3.1 Maker Spaces With Raspberry Pi HATs

Raspberry Pi computers have been successfully coupled with sensors to create a wide range of makerspace activities. While a wide variety of inexpensive discrete sensors and actuators, such as sound-buzzers, LEDs, and touch sensors are available, the integrated Raspberry Pi HAT sensor platform was used for these activities. The Pi HAT combines discrete components nicely into a portable and usable package. This platform provided participants with numerous common utilities including an LED display. The maker activities in this section had the trainees explore cybersecurity and gaming scenarios that used the Pi HAT to present a visual output on the LED display. The lesson began with an introduction to basic programming concepts, such as variable nomenclature, basic operators, and basic data types. The intention was to develop an intuition for translating common language commands into Python. The next segment began with an introduction to the Python list, list operations, comparison operators, user input, classes, and scripts. Students employed Git repositories to exercise version control during these activities. It followed the structure of the previous segment, with miniature activities on each slide and a maker activity recursively consolidating the segment information. Students were provided with a slide showing parts of the program. New concepts were highlighted as they were discussed during the session. The lesson continued as such, with each segment scaling up in difficulty and building on previously-learned concepts, as per a scaffolded instruction approach.

The students completed two key activities during this session. In the first exercise, they sequentially controlled the light pat- tern on an RGB LED. For the second exercise, the students created a Pi-stacking game on the Pi HAT that required players to stack colored tiles on top of each other. Student's engagement with maker activities throughout the lesson pro- vided opportunities for the instructor to manage the varied skill levels with extra focus given when needed. Successful completion of the coding exercises was used as a metric of student success.

3.2 Jupyter Notebooks in Machine Learning

During this session, we described a series of hands-on activities that introduced the participants to aspects of machine learning on the widely-used Keras and TensorFlow software [1, 5]. Participants were provided with an install script that automated most of the process to reduce the complications arising from having to install these software and their associated libraries. Students were first introduced to concepts in machine learning. They were taught how machine learning uses labels to categorize the subjects in images and how it predicts the subject in an image file based on what it learns from a training data set. Topics covered during this session included the need for training using established and respected data sets, emphasizing the need for higher levels of accuracy in terms of training and complexity of models, and finally envisioning scenarios where machine learning will make incorrect predictions. The introductory session was followed by interactive hands-on activities that introduced these learners to various aspects of ML. These included (i) using a training database to teach ML systems to recognize hand-drawn numbers in an image, (ii) accurately predicting the kind of flower seen in an image, (iii) improving the predictive ability of exercise by using convoluted neural networks, and (iv) identifying the objects in a given image downloaded from the internet [9]. As students worked through these exercises, they employed the popular MNIST and ImageNet data sets to explore logical regression models, transfer learning, Python imaging libraries and the scikit learning libraries [12, 18]. Hand-drawn images for these activities were created using the GIMP software. On completion of these activities, students were able to leverage existing modules to solve a real-world machine learning problem with a small training data set and computing constraints. Once again, student success was evaluated based on their ability to complete the session's hands-on activities. As a capstone, each student created two trained platforms. The first platform identified hand-drawn numbers in the range of 0 to 9, and the second platform classified images. Since Jupyter Notebooks were used, additional, more periodic metrics could be used to evaluate student performance. These metrics include qualitative evaluations of the student's use of programming concepts to manipulate training data sets and images. In our exercises, students came up with different ways to confuse the algorithm. One case was of particular interest: a trainee from Group 2 drew upon concepts from programming sessions to create an image distorted with static to understand how the ML programs would react to this data.

3.3 Real World Problem Solving Exercises in Cryptography

During the cryptography session, the instructor first presented the importance of cryptography in computer security. During this session, the emphasis was placed on discussing a relevant real-world problem and identifying approaches to solve it. Unlike the previous activities, these problems could be solved by implementing a number of different approaches. Much time was spent discussing the fundamentals of modern cryptography: modular arithmetic and one-way functions. Computationally difficult functions such as discrete logarithms and the prime factorization problem are not typically covered in introductory courses, so these concepts are worth spending more time on to ensure the students have a strong understanding of the underlying

complexity of cryptography. Additionally, the session on cryptography provides an opportunity to talk about how brute force solutions to problems are often computation- ally infeasible and require the problem set be compressed, a common issue in high performance computing. To apply the material, the instructor presented the Diffie-Hellman Key Exchange algorithm, an accessible example of cryptographic concepts. As part of evaluating the effectiveness of this approach, three exercises were completed during this activity. Each progressive exercise would have more possible solutions. The trainees perform an encryption exercise followed by a decryption exercise.

The session was followed by a capstone exercise where students were challenged to decrypt the communications of others in the class. To demonstrate how it is critical to encrypt secure communications over non-secure lines, all messages, including key exchange negotiations, were broadcasted to the entire room. The students were given examples of the Diffie-Hellman algorithm implemented in Python. The intended recipient should be able to successfully decrypt the message, while unintended recipients will find it computation- ally infeasible. After this exercise, the students were able to describe the need for cryptography and apply their knowledge of modular arithmetic to computing problems.

4. EVALUATIONS, SURVEYS AND ASSESSMENTS

The program and the activities in the program were evaluated using a variety of approaches. Data collected as part of the registration process helped identify the participant's base line skills. As described in previous sections, development of competencies is gauged by the student's ability to complete in-class exercises and activities. To test competency on a topic, students were asked to complete in-class exercises. As described in the previous section, each of the three Python sessions included capstone projects that required the students to apply the knowledge gained from the session. Finally, surveys were completed by students at the end of the camp. These surveys asked questions about the activities taught during the camp. The survey consisted of questions that asked students to rate activities based on a Likert scale. The survey also included a number of open- ended questions that gave participants an opportunity to include additional details about the exercises. To compensate for the limited size of the sample group, surveys were augmented with data from in-person interviews and student performance in activities. Some of the open-ended questions in our post-program survey were:

1. How did you learn about this camp?
2. Why did you register for this camp?
3. What are the difficulties that you faced in this camp?
4. What do you think are the strengths of this camp?
5. What specific content/concepts in the camp were particularly challenging for you?
6. What specific content/concepts in the camp were particularly easy for you?
7. How do you plan on using what you learn in this camp?
8. Who would you recommend the camp to?

Essay-style questions that allowed campers to explain their perceptions of the camp sessions were:

Please add any additional comments and details about topics that you felt were easy or difficult to learn. What was the most enjoyable aspect of a given session? What else you would like to have learned during these sessions?

(a) Linux review session

(b) Python review session

(c) Coding Games

(d) Artificial Intelligence activities

(e) Secret sharing activities

(f) Virtual Reality activities

The Likert Scale (1-5 scale) questions were:

1. How easy was the course for you?
2. How satisfied are you with the camp?
3. Please rate your proficiency in Python before attending the Intermediate camp.
4. Please rate your proficiency in Python after the Intermediate camp.
5. How likely are you to recommend the camp to others?
6. Please rate how likely you are to tell your teachers about this camp.
7. Please rate how likely you are to participate in conferences, science fairs, or other STEM programs in the coming school year?

These surveys provide us a rationale for why people are attending our classes and help with the recruiting efforts. Post-program surveys will be administered at the 6-month and 1-year anniversaries of the program. We will use these data to build a quantitative model that includes a longitudinal aspect. We are particularly interested in understanding how the participants used these skills in future efforts. We hope to develop a profile of the kind of students or groups that are most likely to participate in intermediate programing efforts. In the future, we will partner with research groups on campus for an Internal Review Board (IRB) approved study to investigate whether participants in these programs are able to meet the stated learning objectives as well.

5. RESULTS AND CONCLUSIONS

All participants completed the week-long program. An important consideration while evaluating these data is that these students had self-selected themselves to participate in intermediate computing exercises. Our data indicates that the choice of instructors did not impact student learning in the three models. Each approach and the associated exercises were appropriate for an intermediate program (90%), and were found to be equally engaging (80% or higher). Based on the performance of capstone (and review) exercises, we find that students who had been exposed to computing a year or more ago (Group 1) were better prepared for the program as compared to students who had recently participated in an introductory-level programming camp (Group 2). There was no discernible difference in the academic prowess of these two groups of participants. We assume that students who have had previous exposure to computational thinking would have had more time to synthesize new ideas and develop an understanding of the concepts. Another possible rationale for this observation is that, as self-described learners, Group 1 participants have learnt additional coding and computing concepts that were not taught to Group 2 participants during the introductory camp. Some of the major finding from these the approaches are:

Maker Spaces With Raspberry Pi HATs. While these activities scored highly in terms of processing ideas and taking ownership of ideas, the activities required the students to demonstrate Python programming skills. Students who had not previously developed these skills struggled in exercises that applied them to specific problems (30%).

Jupyter Notebooks in Machine Learning. Participating students were successfully able to manipulate the Jupyter Note- books to change training sets (100%), images (100%), and the number of epochs (100%). While all students could per- form the exercises, the conceptual underpinnings on ML were unclear to some (40%). Understanding of ML concepts was demonstrated by students working to confuse the model by finding images outside of the training data set (30%).

Problem Solving Exercises in Cryptography. Students appreciated that this approach connected to real world problems. The interactive nature of this session provided students opportunities to develop hypotheses to solve problems, and validate them (100%). In these exercises, students had the freedom to select one of many approaches to solve the problem at hand. Instructors were available to provide assistance, but did not direct the students. The data shows that the trainees found this approach to be more challenging in scenarios where there were a large number of possible solutions to solving a problem (80%). As such, this approach is perhaps better suited where students can directly apply the gained knowledge to solving a problem. Throughout this experience, we have found that teaching students new computing concepts, such as navigating a Linux environment, using the 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. The various ways in which the problem could be solved increased the complexity of this approach for a number of students (80%). In agreement with existing literature, our data indicate that students with varying degrees of programming are best suited with scaffolded learning approaches like Jupyter Notebooks for application specific training. A problem-solving approach, though slower, encourages greater interactions and deeper learning of the subject matter. The makerspace activities provided the least amount of scaffolding and were less successful than their counterparts at incorporating increased levels of complexity in programming. Throughout the exercise, we find that, while the scaffolded elements allowed students to complete exercises, the lower learning gains indicate that it is best suited for complex topics where a single approach is attempted. The strategies described in this work were found to be effective for a group of intermediate learners and can be adopted in undergraduate curricula. Taken together, these strategies can help attract students to become the next generation of computer scientists, especially from groups that are currently underrepresented in the field.

6. SUSTAINABILITY & LESSONS LEARNED

The training program was designed with sustainability in mind. Student training in computing is a critical area where demand currently outweighs supply. Post-training surveys indicate that the program's format was well received by the community of students. Data from this training pro- gram shows that intermediate-level coding can be effectively combined with a number of fun activities that engage early undergraduate students and encourage them to participate in computer science. The largest challenges lie in presenting programming concepts at a level that can be comprehended and learnt by a diverse set of students. While certificates of attendance were provided to the participants, we are looking into making these efforts "transcriptable" so that employers can recognize them. While pre-training assessments were per- formed, in future iterations, we hope to assess student skills and competencies both pre- and post-program using evaluation scheme in the style of Student Assessment of Learning Gains (SALG) [21]. 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 computing, STEM, and cybersecurity and ways to refine the offerings. The seemingly vast availability of Python teaching tools and the low cost of computing plat- forms makes the effort inherently sustainable. An abundance of free tutorials, educational makerspace coding activities, and intuitive Python interpreters have helped reduce the amount of programming that a user had to know prior to using large-scale computing or HPC resources. These approaches present exciting opportunities to engage students in programming, a critical step, to get them to learn and contribute to computing efforts.

7. SUPPORTING INFORMATION

All training materials developed by Texas A&M High Performance Research Computing (HPRC) are available for download free-of-charge on the Texas A&M HPRC website. Future adoptees can access the material at https://hprc.tamu.edu/training. Surveys, machine learning installation scripts, and a Linux review exercise are included as part of the Reproducibility Appendix. Agendas, registrations forms, sample announcements, templates to track participants, Trello event-boards and other such materials will be made available by the authors upon request. Please send us feedback about your adoption experience to help@hprc.tamu.edu.

8. ACKNOWLEDGMENTS

The authors thank staff, student workers and researchers at Texas A&M HPRC, the Laboratory for Molecular Simulation, TexGen, Division of Research, the Texas A&M Engineering Experiment Station (TEES) and Texas A&M Provost Information Technology for supporting the program. We gratefully acknowledge support from the National Science Foundation for award number 1649062, "NSF Workshop: Broadening Participation in Chemical and Biological Computing at the Early Undergraduate Level", and award number 1730695, "CyberTraining: CIP: CiSE-ProS: Cyberinfrastructure Security Education for Professionals and Students." We acknowledge the instructors and support staff of these courses: Narendra Chaudhary, Michael Dickens, Lisa Perez, Yang Liu, Keith Jackson, and Lan Li.

9. AUTHORS EMAIL ADDRESSES

The authors email addresses are – Dhruva Chakravorty, chakravorty@tamu.edu; Donald McMullen, mcmullen@tamu.edu; Honggao Liu, honggao@tamu.edu; Noushin Ghaffari nghaffari@tamu.edu; Dylan Rodriguez, dylan@tamu.edu; Shain Le, sle@tamu.edu; Nathan Gober, ngober@tamu.edu; Zengyu Wei, lele0027@tamu.edu; Levi Jordan, ljordan56@tamu.edu; Marinus Pennings, pennings@tamu.edu; Derek Rodriguez, whomps@tamu.edu; and Crystal Buchanan, crysb818@tamu.edu

10. REFERENCES

[1] Mart´ın Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A System for Large- scale Machine Learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16), Kimberly Keeton and Timothy Roscoe (Eds.), Vol. 16. USENIX Association, Savannah, Georgia, USA, 265–283

[2] Paul Michel Baepler, J.D. Walker, D. Christopher Brooks, Kem Saichaie, and Christina I. Petersen. 2016. A Guide to Teaching in the Active Learning Classroom: History,

Research, and Practice (1st ed.). Stylus Publishing, LLC, Sterling, Virginia, USA.

[3] Karen Brennan and Mitchel Resnick. 2012. New Frameworks for Studying and Assessing the Development of Computational Think- ing. In Proceedings of the 2012 Annual Meeting of the American Educational Research Association. American Educational Re- search Association, Vancouver, British Columbia, Canada, 25.

[4] Dhruva K. Chakravorty, Marinus “Maikel” Pennings, Honggao Liu, Zengyu “Sheldon” Wei, Dylan Rodriguez, Levi T. Jordan, Donald “Rick” McMullen, Noushin Ghaffarri, and Shaina D. Le. 2018. Effectively Extending Computational Training Using Infor- mal Means at Larger Institutions. Practice and Experience in Advanced Research Computing (2018).

[5] Franc ̧ois Chollet et al. 2015. Keras. https://keras.io

[6] Carol L. Fletcher. 2014. Building the Texas Com- puter Science Pipeline: Strategic Recommendations for Success. Technical Report. Texas Regional Collabo- ratives. https://www.thetrc.org/web/assets/files/pdfs/ Building-the-Texas-CS-Pipeline Fletcher.pdf

[7] National Science Foundation. 2018. Program Solicitation NSF 18-537: Computer Science for All. https://www.nsf.gov/pubs/ 2018/nsf18537/nsf18537.pdf

[8] Chris Holdgraf, Aaron Culich, Ariel Rokem, Fatma Deniz, Maryana Alegro, and Dani Ushizima. 2017. Portable Learning Environments for Hands-On Computational Instruction: Using Container- and Cloud-Based Technology to Teach Data Science. In Proceedings of the Practice and Experience in Advanced Re- search Computing 2017 on Sustainability, Success and Impact, David Hart and Maytal Dahan (Eds.). ACM, New York, NY, USA, 32.

[9] Andrej Karpathy. 2017. Stanford University CS231n: Convolu- tional Neural Networks for Visual Recognition. http://cs231n. stanford.edu/syllabus.html

[10] Martha Larkin. 2002. Using Scaffolded Instruction To Optimize Learning. ERIC Digest. (Dec 2002).

[11] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep Learning. Nature 521, 7553 (2015), 436.

[12] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. [n. d.]. The MNIST Database of Handwritten Digits. http://yann. lecun.com/exdb/mnist/

[13] James J. Lu and George H.L. Fletcher. 2009. Thinking About Computational Thinking. ACM SIGCSE Bulletin 41, 1 (Mar 2009), 260–264. https://doi.org/10.1145/1539024.1508959

[14] Robin R. Murphy. 2016. Emergency Informatics: Using Computing to Improve Disaster Management. Computer 49, 5 (May 2016), 19–27. https://doi.org/10.1109/MC.2016.135

[15] Hyesung Park, Kihyun Kim, and Cindy Robertson. 2018. The Impact of Active Learning with Adaptive Learning Systems in Gen- eral Education Information Technology Courses. In Proceedings of the Southern Association for Information Systems Confer- ence (SAIS 2018). Association for Information Systems, Atlanta, Georgia, USA.

[16] Jim Parsons and Leah Taylor. 2011. Improving Student Engage- ment. Current Issues in Education 14, 1 (2011).

[17] Michael Prince. 2004. Does Active Learning Work? A Review of the Research. Journal of Engineering Education 93, 3 (2004), 223–231.

[18] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. International Journal of Computer Vision 115, 3 (2015), 211–252.

[19] Kem Saichaie, D. Christopher Brooks, Phillip Long, Robert Smith, Richard Holeton, Carole Meyers, Adam Finkelstein, and Shirley Dugdale. 2017. 7 Things You Should Know About Research on Active Learning Classrooms. In ELI 7 Things You Should Know. EDUCAUSE, Washington, DC, USA.

[20] Jinsil Hwaryoung Seo, Michael Bruner, Austin Payne, Nathan Gober, Donald “Rick” McMullen, and Dhruva K. Chakravorty. 2018. Using Virtual Reality to Enforce Principles of Cybersecurity. Journal of Computational Science Education (Nov 2018). (to appear).

[21] Elaine Seymour, Douglas J. Wiese, Anne-Barrie Hunter, and Susan M. Daffinrud. 2000. Creating a Better Mousetrap: On- line Student Assessment of their Learning Gains. In Proceedings of the National Meetings of the American Chemical Society Symposium. American Chemical Society, San Francisco, California, USA.

[22] Kimberly Sheridan, Erica Rosenfeld Halverson, Breanne Litts, Lisa Brahms, Lynette Jacobs-Priebe, and Trevor Owens. 2014. Learning in the Making: A Comparative Case Study of Three Makerspaces. Harvard Educational Review 84, 4 (2014), 505–531.

[23] Jeannette M Wing. 2008. Computational Thinking and Thinking About Computing. Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 366, 1881 (2008), 3717–3725. https://doi.org/10.1098/ rsta.2008.0118

[24] Aman Yadav, Ninger Zhou, Chris Mayfield, Susanne Hambrusch, and John T Korb. 2011. Introducing Computational Thinking in Education Courses. In Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, Thomas J. Cortina, Ellen L. Walker, Laurie Smith King, and David R. Musicant (Eds.). ACM, Dallas, TX, USA, 465–470. 457110.

The impact of MOOC methodology on the scalability, accessibility and development of HPC education and training

Julia Mullen
MIT Lincoln Laboratory
Lexington, MA
jsm@ll.mit.edu

Weronika Filinger
EPCC, The University of Edinburgh
Edinburgh, United Kingdom
w.filinger@epcc.ed.ac.uk

Lauren Milechin
Massachusetts Institute of Technology
Cambridge, MA
lauren.milechin@mit.edu

David Henty
EPCC, The University of Edinburgh
Edinburgh, United Kingdom
d.henty@epcc.ed.ac.uk

ABSTRACT

This work explores the applicability of Massively Open Online Courses (MOOCs) for scaling High Performance Computing (HPC) training and education. Most HPC centers recognize the need to provide their users with HPC training; however, the current educational structure and accessibility prevents many scientists and engineers who need HPC knowledge and skills from becoming HPC practitioners. To provide more accessible and scalable learning paths toward HPC expertise, the authors explore MOOCs and their related technologies and teaching approaches. In this paper the authors outline how MOOC courses differ from face-to-face training, video-capturing of live events, webinars, and other established teaching methods with respect to pedagogical design, development issues and deployment concerns. The work proceeds to explore two MOOC case studies, including the design decisions, pedagogy and delivery. The MOOC development methods discussed are universal and easily replicated by educators and trainers in any field; however, HPC has specific technical needs and concerns not encountered in other online courses. Strategies for addressing these HPC concerns are discussed throughout the work.

KEYWORDS

Supercomputing, HPC, MOOCs, HPC Education, HPC Training

1 INTRODUCTION

Traditionally, HPC concepts have been taught in formal academic settings as part of an undergraduate or graduate degree, or as highly condensed one-off on-site training courses lasting anywhere between a half-day to a few days. For students pursing research requiring significant computing resources, HPC education may be integrated into courses in a student's subject domain, e.g. computational engineering or physics; however, the number of institutions offering HPC coursework is low. The challenges associated with creating effective and personalized educational experiences for HPC fall into two primary areas: a limited pool of trainers leading to a limited number of workshop offerings over a calendar year and the diversity of the subdomains of interest within the HPC community.

Consider the diversity of subdomains. While there is some overlap of content and skills that HPC practitioners must learn, each subdomain within the HPC ecosystem has a different focus. Additionally, as research applications in medicine, social science and biology become more complex and require more extensive computing power, a new cohort of students with limited computer literacy are searching for a pathway to HPC expertise. This variety in focus leads to a vibrant community, but it requires that HPC educators create effective educational materials capable of speaking to a range of disciplines and computer literacy levels. Unlike past decades when HPC educators and trainers could assume that new graduates had basic computer literacy upon which to build HPC expertise, educators need to add basics to the training program, which in turn affects the number and type of workshops that can be held within a given year. Furthermore, for professionals in need of training, workshops appropriate to their learning needs may not be available in a time frame appropriate to their project and needs. To address these challenges and diversify, the community requires developing scalable, personalized and accessible training to provide multiple learning paths to build HPC expertise.

2 MOOC OVERVIEW

HPC has always been about scaling, developing new technologies, and pushing the boundaries of existing systems. One educational approach that combines these traits is Massively Open Online Courses, or MOOCs. At the end of 2017 there were 81 million MOOC learners spread across the major MOOC providers [15] and another 13 million across 1500 individual sites running their own Open edX platforms [14] in support of their own organizations [8]. Clearly MOOCs offer a means of reaching diverse communities beyond the traditional university cohort of the developed world. However, unlike standard university courses that are part of a program, MOOC courses are open to everyone, generally without pre-requisites, are asynchronous, to support "Just In Time" learning, and compete with work and life responsibilities for learner attention without the promise of a degree or credential.

These characteristics make MOOCs notably different from other more traditional teaching methods, and those differences require

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

specific approaches in the creation and delivery of teaching materials. For example, the open nature of the courses and lack of pre-requisites means that educators need to predict and design for the diverse background knowledge and potential knowledge gaps of their learners. The absence of credentialing equates to a range of learner motivations for joining and completing courses. Adult learners, in particular, bring a broad range of experience to their learning, need to be self-directed and are drawn to learning experiences that address a given problem or question that they are working to understand [10]. Some MOOC learners may be interested in only the initial overview material in order to get a glimpse of a domain; others may want to work with a few units in order to answer their questions or developed deeper understanding of a specific problem, while others may desire a whole course. In this learning environment, a learner that fully is engaged with only a fraction of course material may deem it more useful than a learner who completed the whole course. It is possible that traditional university students have a similar response to full courses, but until recently, when MOOCs began "unpacking" courses into modules that allow learners to determine when and how they they engage with educational material, this aspect of learner behavior has not been studied. As part of the effort to design engaging learning environments, this aspect of learner behavior is getting more attention.

With respect to educational design, MOOC courses build on pedagogical research demonstrating the value of segmenting content into concise and easily digestible chunks and providing frequent assessments to build mastery learning [5]. The fragmentation of content allows students to access the material in asynchronous, interruptive and often non-linear ways. Each content step should be self-contained, to the extent possible, fit in the overall narrative and bring something new, such as content or perspective, to the narrative. By design, learners can easily navigate between content units, revisiting material as necessary, eliminating the need for repeated content. While each step needs to be informative, it doesn't have to be comprehensive. Assessment questions are generally auto-graded and interspersed between units of video and text content. The interleaving and auto-grading provides students with immediate feedback to quickly resolve misconceptions and reaffirm learning. Finally, to support social learning, most MOOC platforms include a discussion forum. Many individual courses provide synchronous times for course interaction via video-conference tools and some even support smaller local gathering in a study group format, complete with questions to prompt discussion.

To better understand how MOOCs can be used to scale HPC education we consider two efforts. Section 3 describes a MOOC that has been offered three times over the past few years while Section 4 focuses on a Small Private Online Course (SPOC) that is being converted to a MOOC. While the two studies use different MOOC platforms, the designs rely on similar pedagogy, and the challenges listed in Section 1 and Section 2 are common.

3 SUPERCOMPUTING MOOC

3.1 Design

A course called "Supercomputing" [3], was developed on behalf of the Partnership for Advanced Computing in Europe (PRACE) by EPCC at The University of Edinburgh in collaboration with SURFsara from the Netherlands. Similar to many HPC centers, EPCC, as a part of the University of Edinburgh, the UK national HPC service provider and one of the PRACE Advanced Training Centers, offers a series of highly successful workshops and short courses to teach parallel and distributed computing concepts to students, faculty and researchers across the UK. However, despite years of increasing the number of training opportunities, meeting the training demands of the user community remains difficult. Considering these restrictions and the existing gaps in the available HPC training materials, a decision was made to create a course that would serve as an introduction to supercomputing, answering the what, why and how questions for a target audience of newcomers up through beginners in the field. The goal was to create an interesting course with an accessible introduction for newcomers while also building firmer foundations and fill-in the missing links for those with some degree of supercomputing knowledge. Concurrently, PRACE was exploring the MOOC approach to increase training accessibility and funded this project as one of two pilot MOOCs ("Supercomputing" and "Managing Big Data with R and Hadoop" [1]). PRACE selected the FutureLearn [9] platform as the host for the two courses. This discussion focuses on the former course, "Supercomputing".

The design of the FutureLearn platform is based around a pedagogical concept of social learning and follows three basic principles:

- telling stories,
- provoking conversation and
- celebrating progress. [2]

The cohesive 'story-like' narration is maintained by segmenting the course content into individually themed weeks. The FutureLearn platform follows the pedagogy described in Section 2 so that the content for a week is further segmented into a relatively large number of bite-size steps and delivered using a variety of content delivery modes, e.g. a mixture of videos, articles, discussions, exercises, quizzes and tests. Furthermore, the learning material for a week supports the learning outcomes through a set of activities with clearly defined learning goals associated with each unit or step. To help learners track their progress, work toward their goals and stay motivated, the FutureLearn platform presents clear progress updates in the student dashboard.

One of the key differentiators of the FutureLearn Platform is the emphasis on learning through social interactions and discussion. To encourage discussion, each step (video, text, exercise) includes space for learners to post comments and most steps include explicit calls to action to encourage conversations among learners and with educators. These discussion components allow learners to exchange opinions and verify their understanding of the covered material by providing learners the opportunity to reflect on their learning and share their insights with others. The importance and value of learners' contributions cannot be over-emphasized as they provide perspectives and motivation for other learners, prompting other students to join the discussion and share their own opinions on each topic. Equally, if not more important, are contributions from the educators. Once a course has begun, the instructor roles include replying to comments and questions and prompting discussions that share experiences, perspective and knowledge among the entire

cohort. The impact of this design element in the “Supercomputing" MOOC is presented in Section 3.3.

3.2 Conversion Challenges

Conversion from traditional course designs to a MOOC model generally requires a major effort to segment the material into bite-size steps that when combined provide a clear narrative. Similarly, the structure of the material typically covered during a two-day course was not easily aligned with any MOOC model. One key takeaway from this re-design effort was the amount of content from previous courses and workshops that could be trimmed because it didn’t add new insight or deeper exploration into the concepts being presented within a section. Significant attention is paid to the idea that variety in delivery modes makes the content more accessible and easier for learners to absorb, but another takeaway is recognizing that content shouldn’t be made to fit a specific delivery type but rather be presented in the simplest way that convenes the greatest meaning. In reaching a balance, experience suggests variety is good but simplicity is even better.

3.3 Results and Lessons Learned

One of the benefits of the segmented model used by MOOC platforms is that small content chunks are easy to modify and new content can be inserted into the narrative seamlessly. This course used this feature, and after each run of the course a number of small changes were made. Most were motivated by either the need to clarify or explain specific topics, or in response to suggestions made by learners or fellow educators. For example, we were surprised by the number of learners who confused supercomputing with Artificial Intelligence, AI. This lead to a new step titled “What supercomputing is not" introduced early in the course to clarify this misconception. Similarly, between runs of the course additional formative exercises, auxiliary content and “useful links", a collection of all the links referenced in discussions, were added to provide deeper experiences for the learners.

The discussions in which educators participate tend to be more animated than others and give learners certainty of being on-track with their understanding of the subject. The instructor team was pleasantly surprised by how well the discussions played out during the various course runs. The diverse background and different levels of familiarity with the topic led to many thought-provoking conversations. One compelling observation was that steps not containing any explicit questions or discussion topics had a much smaller number of comments. Figure 1 shows a number of discussion comments from the first run of the course for each step within week 3. The steps 3.1, 3.10 and 3.23, shown along the x axis in Figure 1 did not include any calls to action, which directly affected the number of learners who engaged in discussions. Including additional discussion prompts was one of the first modifications made between the first and second run of the course.

Table 1 presents the enrollment and engagement statistics collected by FutureLearn for the three runs of the Supercomputing MOOC. The term joiners is used to describe anyone who signed up for the course, learners are those people who actually accessed the content, active learners are those who track their progress through the content (i.e. mark steps are completed) and finally, social learners are those who are active in comment sections. Although, the numbers from the first two runs were relatively low by the FutureLearn and MOOC standards, they were not a source of concern for the educator team because the classes were significantly larger than a similar workshop. Additionally, lower numbers facilitated direct engagement between educators and students, increasing the opportunity for social learning. The third run showed that reaching out and attracting the right audience was one of the biggest challenges of the course. The enrollment numbers a week prior to the start of the fourth run, were even smaller. It seems that the active FutureLearn participants interested in the course have already taken it and advertising the conceptual no-programming introductory course within the HPC community has not brought the desired effects.

From the very beginning the number of students participating in the Supercomputing course was low by MOOC standards. However, it never was the primary indicator of course success. It has been shown that on average only about 5 percent of joiners actually complete massive open online courses [11]. Most of the time this is not a reflection of the course quality or learners’ satisfaction with it. Due to the nature of adult online learning, taking place after hours and competing with life obligations, and the wide spectrum of learner motivations, it is hard to evaluate the success of an online course offering. Just because a learner did not go through the entire course does not mean they did not meet their learning objective, find the information they needed or fully engage with the content they completed. Also, not every learner likes or feels the need to engage socially so the number of comments is not a clear indicator of success. Although MOOC platforms collect data on student performance, engagement and demographics, this is only part of the impact story. Being able to reach learners from over 140 countries is a fantastic achievement in itself and being able to hear or meet someone who was inspired by it is even better. One success story was a Nepalese woman whose participation in the Supercomputing course motivated her to attend ISC’18.

4 UNDERSTANDING HPC WORKFLOWS AND HOW TO EXPLOIT THEM

4.1 Design

The Lincoln Laboratory Supercomputing Center (LLSC) Team pioneered an on-boarding process for professional engineers and scientists that flattens the HPC learning curve. As part of this on-boarding process, a Computational Science and Engineering (CSE) Team member meets with each new user to provide a targeted introduction to HPC and discussion of strategies for scaling the user’s workflow. While this form of consulting is highly effective, it is not scalable, especially as support needs expand to include members of the MIT campus community. To scale the on-boarding process, the team leveraged the Open edX platform for the delivery of a MOOC course. The Open edX platform was selected because as an open source product it affords

- the ability to modify and extend the platform to provide tools that support the our teaching
- access to all of the student data which is used for
  - continuous course improvements

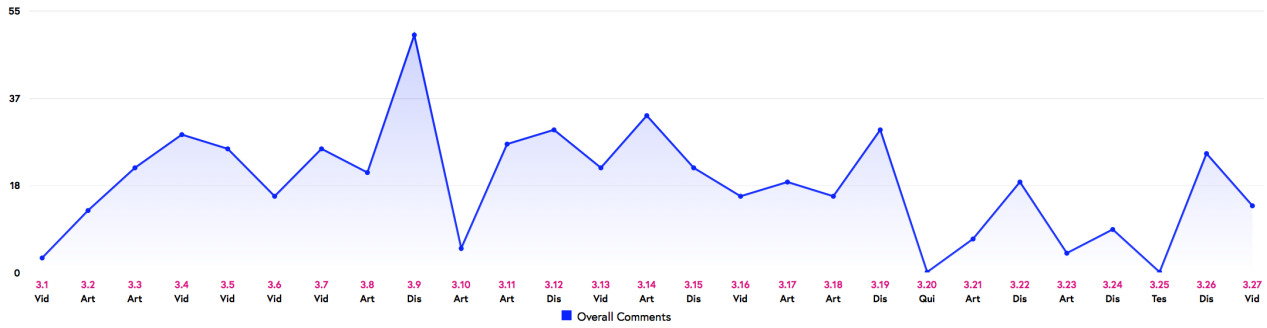


Figure 1: Comments for steps within Week 3 of the first run of Supercomputing.

Table 1: Enrollment and engagement statistics for the first three instances of the Supercomputing MOOC. The percentage of leaners is calculated using joiners as a basis, active learners using learners as a basis, and social learners and step completion using active learners.

Run	Joiners	Learners	Active Learners	Social Learners	Learners completing ≥ 50%	Learners completing ≥ 90%
1 (6 Mar 2017)	3,263	1,722 (52.8 %)	1,138 (66.1 %)	379 (22 %)	337 (19.6 %)	211 (12.3 %)
2 (28 Aug 2017)	3,100	1,910 (61.6 %)	1,285 (67.3 %)	403 (21.1%)	362 (19 %)	186 (9.7 %)
3 (15 Jan 2018)	1,825	1,218 (66.7 %)	761 (62.5 %)	219 (18 %)	207 (17 %)	115 (9.4 %)

– learning analytics

The initial pilot included in the on-boarding process was intended to teach users how to map their serial or small scale application onto a set of common HPC workflows, e.g. high throughput, MapReduce, Leader-Worker and full MPI based parallel applications. Once students understand where their applications fit in a set of canonical workflows, they can focus on techniques to effectively refactor or extend their application to make it suitable for an HPC system [12, 13].

Within the user population the domains of interest ranged from those traditionally associated with computing and HPC, e.g. physics, engineering, math and computer science, to more recent adopters in the biological and social sciences. One consequence of this domain range is a significant range of computational experience, from researchers with minimal experience developing computational workflows to researchers running commercial parallel codes to those comfortable developing new parallel solutions. Reviewing the on-boarding requirements for these professionals and campus researchers, the instructor team recognized the common concerns of both populations, namely:

- deep domain knowledge but lack of HPC specific training
- computational experience ranging from novice to advanced
- need for strategies rather than HPC tools

The first design challenge was to create a course that addressed the three concerns listed above by providing enough theory and practice to enable users to accomplish their HPC goals: faster time to solution and more research turns in a day. Considering the range of computational experience, the expectation was that advanced users needed a simple refresher on how to use the Laboratory HPC system combined with an understanding of updated best practices and techniques. Novice users needed a bootcamp whose learning outcomes included understanding their application type in the context of HPC and how to effectively use the HPC system for their application. Intermediate users fell between these two, and their needs, while more fluid, were covered by the material created for the other two user cohorts. The design challenge centered on creating multiple learning paths through the course, including sufficient content and guidance so that each user could "build their own adventure".

The second design challenge was mapping a decade of experience providing individual consultation to users, tutorials on pMatlab at numerous conferences, and presentations about interactive supercomputing, to the Open edX platform. The pedagogy of the Open edX platform centers on mastery learning through the joining of theory and practice. The theory is provided in small content chunks followed by practice in the form of quizzes, problems, essays or discussions. Aligning the existing material to address both design challenges resulted in a complete redesign of existing tutorials and presentations. The refactoring was accomplished through the use of the Cmap [6] concept mapping tool, to create a map of the concepts necessary to understanding and executing successful HPC strategies for a given application type. The concept map for the full course is shown in Figure 2. Concept maps, also known as knowledge graphs, are driven by a major question, and the concepts required to answer the question form a hierarchy of material that a student needs to understand in order to answer the major question. The major question for the pilot Introduction to HPC course was

"How do I convert my application to an HPC workflow and exploit concurrency?" The map in Figure 2 was designed to include all of the concepts that a novice user would need to understand in order to answer the question. The map is developed by considering questions that follow from the major question, e.g. “What is Scientific Computing?", “What is HPC?", “How do I get started using the supercomputing system?” and “How do I parallelize my application code?”. Each of these questions corresponds to a topic area in a more traditional syllabus and yields additional concepts as illustrated by the hierarchy of concepts associated with “What is High Performance Scientific Computing?" The authors note that each of the other major topics has a similar set of hierarchical concepts that have been minimized in Figure 2 to provide clarity.

The creation of concept maps to visualize related topics offers advantages not seen in traditional linear syllabi

- links between concepts highlight interconnections that novices often miss but are usually important to the discipline
- concepts that do not link to other concepts highlight material that is not necessary for understanding
- the knowledge graph provides a starting point for crafting personalized learning paths for students
- the creation of the concept map automatically segments the material into bite-sized chunks

For the instructor team, the creation of the concept map for the HPC course addressed both the second design challenge, segmenting existing material in a coherent manner and the first design challenge, building a course appropriate to a range of experience levels and goals. As an example, consider the educational needs for a user with HPC experience who has not used the supercomputing system in years and brings an application that is new to him or her. This student needs to learn the strategies associated with parallelizing and running the new application on the HPC System. Reviewing the full concept map, it is clear that this student only needs two topic areas as illustrated in Figure 3.

4.2 Lessons Learned

Among the key benefits of MOOCs are

- the ability to track student activities
- insight into problem areas of the course
- ease of updating, modifying and extending content

These benefits allow instructor teams to better understand levels of student engagement with the material and to gain insight into potential content gaps or problems that need clarification. The combination of the platform design and the segmented content make it easy to close content gaps by adding and extending material. With an eye toward redesigning the pilot SPOC and creating a MOOC, there are three primary lessons that are being used to inform the new design; tracking student learning paths, modifying hands-on exercises to support open courses and building exercises for a student population that prefers web-based portals.

While designing a course where students can build their own path has great educational value, the basic premise renders it difficult to apply traditional academic metrics when attempting to determine the success of the course. By design each student should touch on the material that is appropriate to their situation. Even novice students aren’t encouraged to consider all of the use cases, but rather to focus on the use case that best fits the application they are working with at the present time. The result is limited data on completion rates leaving the team to develop other approaches to evaluating success and recognizing gaps. Based on anecdotal data, the sections providing an overview of scientific computing, HPC and interactive computing are successful but student’s understanding of the role and importance of the scheduler are less well understood. Students recognize the components of the supercomputing system but seem to have difficulty recognizing how different they are from the system on their desks, or why these differences are important. Finally, crafting personalized learning paths within a course offering is not fully supported within the standard Open edX platform. The pilot incorporated hands-on experience through programming assignments that could be developed and run on the LLSC and MIT supercomputing systems. While this is possible with a SPOC, providing access to supercomputing resources renders hands-on HPC experience difficult to include at the MOOC scale. Additionally, the hands-on components included instruction on how to run the applications on the local systems, using specific scheduler, file system and user account specifications. While videos demonstrating correct responses to job submissions and explanations of scheduler behavior are helpful for students, example application snippets that are too tightly coupled to a given system not only suffer from lack of portability but they quickly become outdated.

Finally, while the LLSC and Supercloud have pioneered interactive supercomputing and emphasize alternatives to batch processing, new users have a distinct preference for web-based portals to compute systems and application codes. The trend towards web-based access to systems is so ubiquitous that a Jupyter Notebook viewer and grader have been built for the Open edX platform. [4] As these students join the HPC environment, the LLSC and Supercloud CSE Teams have begun leveraging Jupyter Notebooks for both teaching and application development and recognize the need to integrate the notebooks into the next generation of the HPC MOOC.

4.3 New Design

Building on the lessons learned and the changes in student preparation described in Section 4.2, the MIT Supercloud HPC SPOC is being redesigned to separate the understanding of HPC from the job submission and monitoring concerns. Additionally, because the current state of the Open edX platform does not fully support personalized learning, the new design focuses on a set of “short courses". Short courses have the added benefit that they more closely align with the needs and time constraints of adult learners, described in Section 2, by providing the student with the material they need in a small bundle. Furthermore, the development of short courses follows a trend within edX, where there has been increased focus on micro-masters programs and an increase in the number of courses meant to run for three to four weeks rather than the ten to twelve weeks of earlier offerings [7]. The MIT Supercloud team split the course into two short courses “Understanding HPC Workflows and How to Exploit Them" and “Using the MIT Supercloud". The former covers the introduction to HPC concepts, and overview of the canonical workflows and short hands-on examples to explore

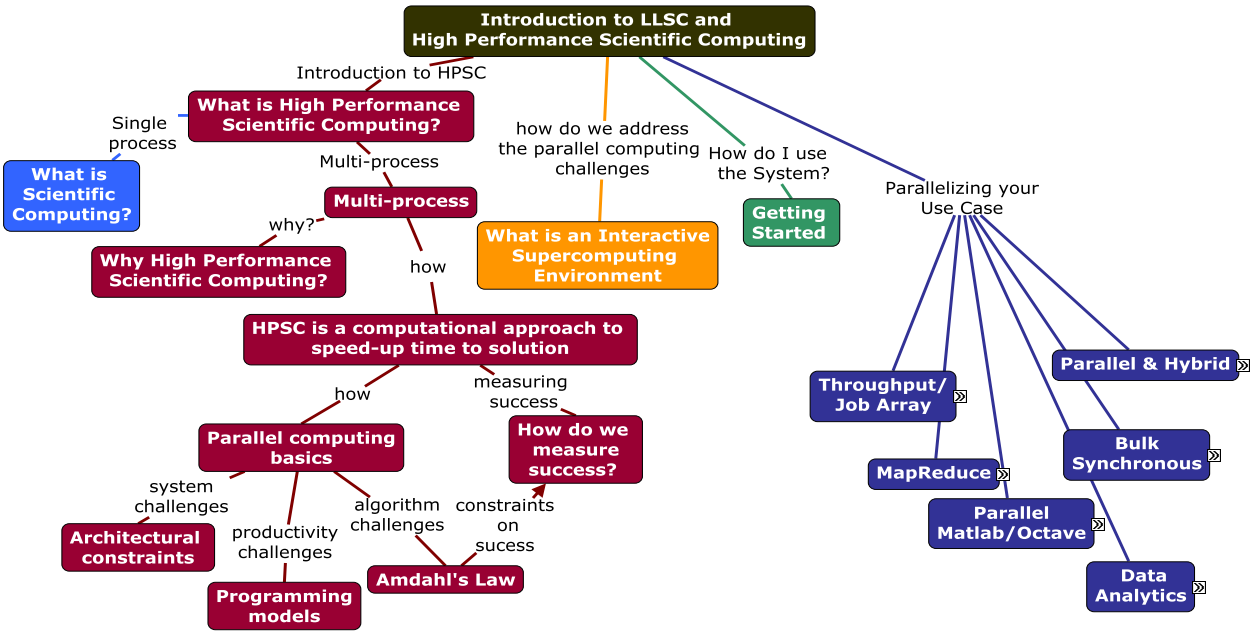


Figure 2: Concept Map for Introduction to HPC Pilot MOOC.

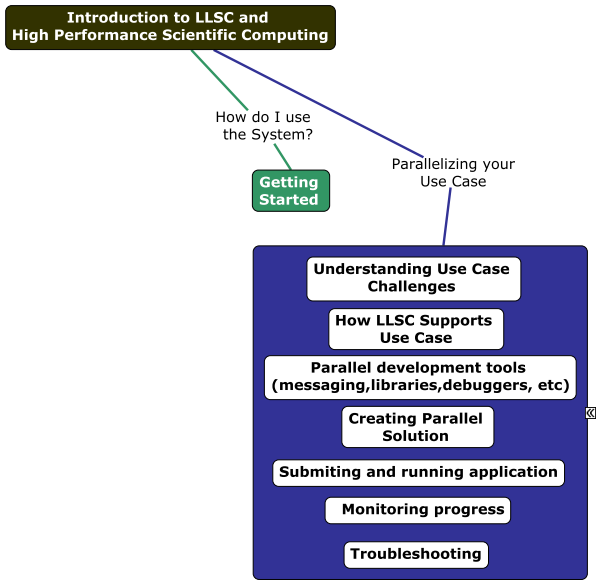


Figure 3: Concept Map for Expert Student in Introduction to HPC Pilot MOOC.

HPC workflows. The hands-ons examples include thought exercises and small programming exercises contained in Jupyter Notebooks that can be run on any system with multiple cores. The latter mini-course, “Using the MIT Supercloud" provides training on how to use the HPC system, including how to launch and monitor jobs, and Best Practices for using each system. The hands-on exercises using the MIT Supercloud provide students a chance to develop their competencies. This modular development maximizes re-use for both the LLSC team and the larger HPC education and training community.

5 CONCLUSIONS

It is clear that though MOOCs provide a means for scaling and expanding the accessibility of HPC education and training, there are significant challenges that need to be addressed. MOOCs cannot be viewed as an inexpensive alternative to face-to-face training, because of the time and effort required to develop and facilitate a course. There are best practices for segmenting content into small concept sized chunks which can reduce the effort and the long range benefit is a collection of small teaching units that can be recombined and reused for a range in a range of educational contexts..

In addition, HPC has unique requirements for access to parallel and distributed systems for courses where hands-on activities are to be included. The case studies presented here addressed this challenge by re-factoring the hands-on components to engage learners in thought experiments. In each case the expectation is that the student will be better prepared to utilize site and system specific training to develop and execute their applications. The benefit of creating a general course or series of mini-courses is that the material is accessible to a wider audience, e.g. a group of researchers who are beginner programmers, pre-university students, teachers and corporate and government decision makers. An additional benefit of segmenting the theory and thought exercises from the system details is that the resulting course components are general and can easily be reused by other centers, universities and laboratories. For example, new learners could take the Supercomputing MOOC described in Section 3 and follow that up with the MOOC described in Section 4 so that when they have access to an HPC system they are familiar with the basics of supercomputing, understand where their application fits in the larger HPC application landscape and know what questions to ask in order to get an efficient solution.

A REPRODUCIBILTY

This paper has examined methods of designing and developing MOOC courses for HPC education and training. All of these methods are reproducible by using standard instructional design methods for designing courseware and can easily be implemented on any online course platform. The key discussion here is re-use and each of the MOOC courses is modular such that it can be re-used either in full or part.

ACKNOWLEDGMENTS

The authors acknowledge funding from PRACE via the EU’s Horizon 2020 Research and Innovation Programme (2014-2020) under grant agreements 653838 and 730913 for the Supercomputing MOOC. The authors would also like to thank the members of the MIT Lincoln Laboratory Supercomputing Center and MIT Supercloud Teams who support the deployment of a unified educational environment, including an Open edX instance and the supercomputing system used for HPC practice for the MIT MOOC.

REFERENCES

[1] [n. d.]. Managing Big Data with R and Hadoop. Retrieved August 2, 2018 from https://www.futurelearn.com/courses/big-data-r-hadoop
[2] [n. d.]. The Pedagogy of FutureLearn, How our learners learn. Retrieved September 22, 2018 from https://ugc-about.futurelearn.com/wp-content/uploads/FL-pedagogy-RGB.pdf
[3] [n. d.]. Supercomputing. Retrieved August 2, 2018 from https://www.futurelearn.com/courses/supercomputing
[4] Lorena Barba and Miguel Amigot. 2018. Jupyter-based courses in Open edX - Authoring and grading with notebooks. Video. Retrieved August 3, 2018 from https://www.youtube.com/watch?v=Qml1x_SbVzc
[5] R.C. Clark and R. Mayer. 2011. *E-Learning and the Science of Instruction, Third Edition.* Pfeiffer, A Wiley Imprint.
[6] Cmap [n. d.]. Retrieved September 27, 2018 from https://cmap.ihmc.us/
[7] edX [n. d.]. Retrieved September 27, 2018 from edx.org/course
[8] Open edX. 2018. Open edX Conference 2018 The State of Open edX. Video. Retrieved August 3, 2018 from https://www.youtube.com/watch?v=HyK2YoKKaGQ
[9] FutureLearn [n. d.]. Retrieved August 3, 2018 from https://www.futurelearn.com/
[10] M.S. Knowles. 1980. *The Modern Practice of Adult Education, From Pedagogy to Andragogy, 2nd Ed.* Cambridge Books.
[11] Daphne Koller, Andrew Ng, and Zhenghao Chen. 2013. Rentention and Intention in Massive Open Online Courses: In Depth. Retrieved September 27, 2018 from https://er.educause.edu/articles/2013/6/retention-and-intention-in-massive-open-online-courses-in-depth
[12] J. Mullen, Chansup Byun, Vijay Gadepally, Siddharth Samsi, Albert Reuther, and Jeremy Kepner. 2017. Learning by Doing, High Performance Computing education in the MOOC era. *J. Parallel and Distrib. Comput.* 105 (July 2017), 105–115. https://doi.org/10.1016/j.jpdc.2017.01.015
[13] J. Mullen and et. al. 2016. Designing a New High Performance Computing Education Strategy for Professional Scientists and Engineers. http://ieee-hpec.org/2016/techprog2016/index_htm_files/R-Mullen-HPEC-16.pdf
[14] Open edX [n. d.]. Retrieved August 2, 2018 from openedx.org
[15] Dhawal Shah. 2018. By The Numbers: MOOCs in 2017. Retrieved August 2, 2018 from https://www.class-central.com/report/mooc-stats-2017/

Training Computational Scientists to Build and Package Open-Source Software

Prentice Bisbal
Princeton Plasma Physics Laboratory
Princeton, NJ
pbisbal@pppl.gov

ABSTRACT

High performance computing training and education typically emphasizes the first-principles of scientific programming, such as numerical algorithms and parallel programming techniques. However, many computational scientists need to know how to compile and link to applications built by others. Likewise, those who create the libraries and applications need to understand how to organize their code to make it as portable as possible and package it so that it is straightforward for others to use. These topics are not currently addressed by the current HPC education or training curriculum and users are typically left to develop their own approaches. This work will discuss observations made by the author over the last 20 years regarding the common problems encountered in the scientific community when developing their own codes and building codes written by other computational scientists. Recommendations will be provided for a training curriculum to address these shortcomings

KEYWORDS

Open-source Software, Computational Science, Training

1 INTRODUCTION

It has been observed that physical scientists and engineers often do not have the basic computing skills necessary to use computers effectively. This even includes computational scientists engaged in parallel computing.[15] This observation has lead to training efforts such as Software Carpentry[13], Data Carpentry[12], and HPC Carpentry[8]. These "carpentry" programs, collectively known as "The Carpentries," are workshops designed to provide training for physical scientists and engineers, data scientists, and computational scientists, respectively, in practical computer skills relevant to these groups of scientists to help them use computers effectively in their work. The purpose of these workshops is not to make the participants experts on any of the topics covered in these workshops, but to give them "good enough" skills.[14]

In the spirit of this observation and these workshops (and others like them), the author recommends training programs for computational scientists consider including training for two skills that, based on the author's experience, would be valuable to computational scientists:

(1) How to compile open-source software packages
(2) How to package open-source software for others to use

Not all computational scientists need to be proficient in both skills. All should be proficient in (1), but only those computational scientists developing codes to be used by others need to know (2).

The author basis these recommendations on his experience. The author has been a Linux system administrator and high-performance computing (HPC) specialist for the past 20 years. During this time, he has supported computational scientists in a variety of fields including plasma physics, engineering, chemistry, computational biology, astrophysics, particle physics, and weather modeling.

Most of this experience has been in smaller departments or institutions where there author was the sole HPC specialist, responsible for every aspect of HPC operations, including installing the computational software needed by the users, and providing technical support to those users. As a result, he has built and installed open-source scientific packages thousands of times, and has helped numerous users trying to compile software themselves.

For example, in the summer of 2009, the author provided HPC support for a two-week long summer program for graduate students and postdoctoral fellows in Astrophysics[9]. The instructors were accomplished computational astrophysicists from the United States and Europe. Despite the author setting up a parallel computing cluster specifically for this for this program, and installing all the computational software that the students would need for this program, which was all open-source, the instructors perferred that the students try to install all the need software themselves on their own laptops. They felt the ability for these aspiring computational scientists to install opens-source tools like this was an essential skill for their careers as computational scientists.

In order to facilitate discussion, the organizers of the program set up a group mailing list where the students could discuss the course content and ask each other for help with their homework assignments. The discussion on this list centered almost exclusively on how to compile and install the open-source tools needed for the for the program. The author was part of this mailing list, and estimates he sent over 200 hundred emails to the list during the 2-week program providing the students with assistance debugging their installations issues, and explaining the configuration and build process.

This intensive experience provided considerable insight into what practical computer skills and knowledge computational scientists are lacking at the graduate and postdoctoral levels, as well as the areas of building open-source software where most errors occur. Many of the recommendations how to teach building open-source software are based on this experience in particular.

https://doi.org/10.22369/issn.2153-4136/10/1/12

Even most though computational scientists can rely on the IT or HPC support staff at their home institution to provide most of the software they need, there are a number of reasions why computational scientists should know how to build open-source software themselves:

- Their IT or HPC department may be short-staffed, leading to delays in installing the software they need when they need it.
- To keep workloads reasonable, some HPC centers limit the software that they will support to key libraries applicable to most users or the most performance-critical libraries, leaving it to the scientists to manage the software specific to their research themselves.
- As a policy, some HPC centers will only install final release versions of a software package, so if a user needs a new feature or bug fix in a pre-release version, they may have to install it themselves.
- As laptops have become smaller and more powerful, the author has noticed more computational scientists using their laptops for small computational workloads, and wanting all the computational packages they use on their home institution's cluster on their laptop so they can work at home or when travelling to conferences.

For computational scientists who develop software, it is in their best interest to learn how to package and distribute it in a way that makes it as easy as possible for others to install and use it. The academic research culture is often described as "publish or perish". That is to succeed you need to publish your research often to survive and get ahead. Related to this, many researchers who publish are also judged by how often their research is cited. For computational software developers, their work is often cited when other researchers use their software in their own research. These users are more likely to use a software packages that is easy for them to install, so the easier a package is to install, the more likely it is to be used, and the more it is used, the more likely it is to be cited.

Many computational software development projects are publicly funded through grants from agencies like the National Institute of Health (NIH), or the National Science Foundation (NSF), or they are developed a national lab, where they have to compete for budgeting with other projects. To continue to be funded, these projects often have to periodically demonstrate to the funding agency that the projects are worthwhile. "Worthwhile" in this case usually means that other researchers see value in this software and use it. A metric commonly used to demonstrate this is how many times that software has been cited. Making a software package easy to install can help increate this citation count and help justify continued funding of its development.

In the author's experience, open-source computational software packages are, on average, much harder to install than more general-purpose open-source packages. These difficulties are generally due to several issues with the software:

- Poor or non-existent documentation on how to build the software, either on the software's web page, or included in the source code.
- No automated configuration and/or build system is provided, leaving the user to determine the proper configuration and/or manually run the commands to compile and install the code.
- The software does not follow follow existing standards or current practices, requiring the builder to modify their procedures to accommodate these differences.

Any one of these deficiencies can create significant obstacles for experienced software builders. For computational scientists with limited experience, any one of these deficiencies can be insurmountable.

In the next two sections, the author will outline a curriculum for training computational scientists in each of these skills, including topics that should be included, and why, as well as point out topics that should be omitted to prevent overwhelming the student. Resources will be provided that provide more detail on each topic. These outlines, combined with the recommended resources, can then be used by the reader to create a complete training class.

In the spirit of Wilson, et. al.[14] the goal of this training should not be to make the trainees experts in either of these skills, but to give them skills that are "good enough" for them to to be more productive and successful.

2 HOW TO BUILD OPEN-SOURCE SOFTWARE

In the author's experience most open-source software, provides a `configure` script created with GNU Autoconf[1]. The author estimates 90% or more of the software he has installed falls into this category, so he recommends that training focus exclusively on building software that uses this tool, since this is most relevant tool due to market share, and introducing multiple configuration tools at once may take up too much time, or overwhelm the student.

Most of the errors encountered by running this script, or actually compiling the software, are the result of a file not being found. This is usually corrected by specifying the correct location of the file by defining an environment variable, or specifying the proper path with a command-line option to the configure script. Since the main goal of the build process is to compile source code, the following topics should be covered before explaining to use the configure script:

(1) The Filesystem Hierarchy Standard
(2) Environment Variables
(3) The Compiling Process

2.1 Filesystem Hierarchy Standard

The Filesystem Hierarchy Standardard[11], or FHS, is standard that specifies where certain types of files should go on a Linux filesystem. Sofware builders do not need to know the entire standard, but they should know about the directories listed below, which are relevant to the configuration and build process. When creating class materials, always consult the standard itself, and not the descriptions provided below. The descriptions below explain the directories as the author would describe them to students, and may not be 100% in agreement with the language of the standard.

/usr A major section of the filesystem. It is read-only, and holds most of the programs, libraries, and other files used by the users. The only time this section of the filesystem should

be written to is when packages provided by the operating system are added or removed by the system administrator. 3 subdirectories, important to this lesson, /usr/bin, /usr/lib, and /usr/include, are located in this directory.

/usr/bin Contains binaries and other executable commands (shell scripts, Python scripts, etc.) that any user can run (no administrator privileges necessary). In general, directories named 'bin' anywhere in the filesystem will usually contain programs to be run by non-root users.

/usr/include Contains header files for libraries stored in /usr/lib.

/usr/lib Contains shared and static library files. After the introduction of 64-bit x86 processors, it became common to use this location to store 32-bit libraries, and store 64-bit libraries in /usr/lib64

/usr/lib64 Similar to /usr/lib, but contains 64-bit libraries.

/usr/share This part of the /usr filesystem contains architecture-independent files that can be shared between systems with different processor architectures. In practice this is where a lot of non-executable text files are stored, including software documentation (/usr/share/doc) and man pages (/usr/share/man)

/usr/local This directory is similar in purpose and organization to /usr, and is meant as a place where the system administrator can install additional software on the system without interfering with the software provided by the operating system.

/home This directories contains subdirectories named after each user account. These subdirectories are known as *home directories*. They are owned by the user they are named after, and the user has full read-write-execute privileges over the entire contents of this directory. Personal settings are stored here, and users can save their files here, and install and run software here, too.

The main items to emphasize here are the following:

(1) Header files for libraries included with the operating system are located in /usr/include, and others may be installed in /usr/local/include, or possibly other places, depending on where the system administrator decided to install additional software. Knowing the possible location of headers is essential to building software.
(2) Library files included with the operating system are located in /usr/lib and /lib. For 64-bit systems, libraries may also be located in /usr/lib64 and /lib64. They may be located /usr/local/lib or /usr/local/lib64 or other locations depending on where the system administrator has installed additional software. Knowing the location of libraries is essential to building software.
(3) /usr/local is a traditional location for installing additional software that is not included with the operating system. Unless another location is specified, many open-source packages will try to install in /usr/local. This means header files will be installed in /usr/local/include, library files in /usr/local/lib or /usr/local/lib64, executables in /usr/local/bin, and man pages and documentation in /usr/share.
(4) Users may install and run software from their home directory. The author often gets inquiries from users asking if it's okay or safe for them to install software they need in their own home directories. For most users, if they are installing software themselves on a shared system, like their departmental or campus cluster, they will want to install the software into their home directory.

2.2 Environment Variables

When providing this training the topic of environment variables should be discussed: what they are, and how to set and unset them. While this may seem like a very basic topic to experience Linux users, the author has seen even experienced users who have misunderstandings about how environment variables work. Some misconceptions he has encountered:

- Once an environment variable is set in one shell, it affects all other shells
- Once an environment variable is set, it is persistent and doesn't need to be set again

Whether or not this topic needs to be included depends on the knowledge level of students in the class, and me be skipped if the instructor doesn't feel this needs to be covered.

Setting certain environment variables will be helpful to the build process, and it's necessary to set environment variables when the software installation is complete in order to update PATH, MANPATH, LD_LIBRARY_PATH and other environment variables to use the new software.

As stated in the previous section, knowing the location of header files and libraries is important to the build process. If a header is located in a standard location such as /usr/include or /usr/local/include, or a library is located in /usr/lib, /usr/lib64, /lib or /lib64, they should be correctly detected during the build process and no further steps are required.

However, when software is installed in another location, builders should know that they can set environment variables to ensure they are detected properly. For GCC, the following environment variables can be used to specify the location to include directories in non-standard locations:

CPATH A list of directories to be searched for header files, independent of the programming language

C_INCLUDE_PATH A list of directories to be searched for header files, but only when compiling files written in C.

CPLUS_INCLUDE_PATH A list of directories to be searched for header files, but only when compiling files written in C++.

For library files there is only one variable to set, LIBRARY_PATH.

The values these environment variables are set to are lists of directories, using a colon (:) as the separator, and the directories are searched in order from left to right. For example, to look for C include files in $HOME/include and /opt/include before the standard locations, you can set C_INCLUDE_PATH with the following command (bash shell syntax shown).

```
$ export C_INCLUDE_PATH:$HOME/include:/opt/include
```

It's important to note that these environment variables add to the default search locations - they do not overwrite them, but they do supercede them.

The export command above is needed to make that variable available to the commands called by that shell. This is needed for

the compiler, which will be called later, to access the values of these variables.

2.3 The Compiling Process

The final prerequisite before discussing the configuration and build process is to briefly explain the compiling process. Technically, the compiling process has 4 steps:

(1) The preprocessor step
(2) The compiling step
(3) The assembly step
(4) The linking step

However, for the purpose of this instruction it might be better to simplify to 3 steps:

(1) The processor step
(2) The compiling step
(3) The linking step

The author feels this simplication is justified, since to the user, the assembly step is not normally visible, and any errors during the build process typically do not occur during the compiling or assembly steps, so no information is lost that the students would need.

In the author's experience most build errors occur for one of the following reasons:

(1) A header file cannot be found by the preprocessor
(2) A library file cannot be found by the linker
(3) The linking stage results in unresolved symbol errors, caused by libraries listed in the wrong order or a necessary library not specified on the command-line.

Errors do not normally occur during the compiling step. And when they do, they are normally harder to solve: the code is using a version of the syntax that is either too new or too old for the compiler to understand, or the code is using language extensions supported by a compiler other than the one being used. Not only are errors at this step relatively rare, they require knowledge of the programming language used, and teaching programming languages is beyond the scope of this training.

There are numerous resources that cover the compiling process, so the details of the different steps will not be discussed here, instead let's focus on how to address those 3 common errors mentioned above:

If a header file cannot be found, determine the correct location of the header file, then specify it's location on the compiler command line with the -I switch. For example, if the correct header is in $HOME/include you would specify that like this:

```
$ gcc -I $HOME/include ... ...
```

This additional directory can also be added to the preprocessor environment variables (CPATH, C_INCLUDE_PATH, etc.) as described in the previous section. However, the author recommends using the command-line whenever possible, since that consolidates all your settings in a single command. Since environment variables only affect the shell they're issued in, if a user has a lot of terminal windows open, it's very easy to type the command in a terminal window where the environment variables are not set correctly leading to unexpected errors.

For a library that cannot be found by the linker, the solution is almost the same as for a missing header file: determine the correct location of the library file using the -L switch. Assuming the library file is located in $HOME/lib, that would look like this:

```
$ gcc -L $HOME/lib ... ...
```

The LIBRARY_PATH environment variable can also be set as decribed in the previous section.

The linker can also report unresolved symbol errors. This means that either the file being compiled makes references to a function that is not provided by one of the libaries, or one of the libraries being linked to relies on a function provided by another library that is not included. There are two possible causes for this:

The first is that the libraries are not listed in the compiler command with the -l switch in the correct order. The libraries must be listed in the correct order for the linker to resolve the symbols correctly. The libraries are searched in order from left to right as they appear on the command-line specified with the -l switch. The library needing the function needs to be listed before the library providing the function. This issue can be tested easily be changed the order of the -l options and seeing if that eliminates the error. Sometime you can do an Internet search of the unresolvable symbols to determine which library provides it, and use that information to correct the order of the libraries.

Unresolvable symbol errors can also occur during linking if the library providing a symbol is omitted from the list of libraries to link. If you're not sure what library needs to be added to the command, an Internet search can often provide useful clues.

For instructional purposes, it is helpful to create simple examples to illustrate these errors, and show the student how to fix them in real time.

2.4 The Configure Script

Now that we have covered some important prerequisites to help the students understand the build process, we can take a look at how to use the `configure` script to configure the software with the correct settings for building.

As stated earlier, this `configure` script is created by the software's developer using GNU Autoconf[1]. When the source archive is uncompressed, this script will usually be located at the top level of that resulting directory.

When executed, this script will run a number of small tests to probe the environment the software is being built it to determine if the prerequisites for mandatory features and optional features are present, as well as determine how to optimize the code for the environment. For example, if it detects the processor supports FMA4 or AVX2 instructions, it may enable optimization to take advantage of those processor features.

An example of an optional feature would be a command-line program that has an optional GUI interface that uses X11. The program can still perform all of it's functions from the command-line, without the GUI. If configure can't detect the X11 headers or libraries on the system, it will print out a brief message stating that, and continue on. On the other hand, if the user specified the option to build the GUI using the appropriate switch to the `configure` script, and the X11 libraries were not found, configure would halt with an error, since `configure` would cosider this an error.

When `configure` runs, it executes a number of small, simple tests and uses the results of those tests to determine if certain features are present on the operating system. These tests include attempting to execute a command to see if it's present, trying to compile a simple program to see if the header file in the program is found by the processor, and so on.

The details of how configure works under the hood aren't critical to computational scientists at this level. What is important is how to invoke it with the correct options to build the application to meet their needs, so most of the instructional time should be spent focusing on this.

Every `configure` script has a number of options. These options will be unique to the package being built. The best way to see what configuration options are available is to run the configure script with the `--help` switch, like this:

```
$ ./configure --help | less
```

In the above example, the output of the command is piped into *less* in order to facilitate scrolling through the output, since most configure scripts will print out a lot of options. In general there are three types of configuration options available:

(1) Options to specify where the software is installed. Where library and header files will be installed, for example.
(2) Options that specify how the software is built and what features are enabled or disabled, like whether to build shared libraries or not.
(3) Environment variables that control the behavior of the configure script. These variables can be used to specify what compiler to use, or what flags should be passed to the preprocessor.

In a classroom environment, this would be a good time to display the output of a `configure --help` command for some open-source package, and discuss what some of them mean. Due to the amount of output, it's not really possible to show an example of this output in this article.

There are some configuration options that are common to all configure scripts. The most important of these is the `--prefix` option. This option tells configure in what directory the software should be installed. If this is not specified, the default is used, which is usually /usr/local. All other directories and files are then installed under here. For example if the default is used, all header files will be installed in /usr/local/include, all libraries will be installed in /usr/local/lib, and all executables will be installed in /usr/local/bin.

This is typically not what you want, since this will make it harder to keep track of what files in those directories belong to which application, and prevents having multiple versions of an application installed, since the files from whatever version is installed last will overwrite the versions installed earlier.

For software that's installed manually like this, it's much easier to put each application in it's own directory, in a path that makes it easy to understand what application is installed where. For example, if you want to install versions 1.1 and 2.2 of an application named "example", you might install them in /usr/local/example-1.1 and /usr/local/example-2.2, respectively, or /usr/local/example/1.1 and /usr/local/example/2.2, respectively.

For users installing software in their home directory, it is recommended they create a directory named 'apps' or 'software' in their home directory, and then install everything under that. For example, using the previous example but installing it in $HOME, those versions could be installed in $HOME/apps/example-1.1 or $HOME/apps/example/2.2.

Some other common options that the author likes to set are:

–disable-silent-rules This enables verbose output from the make process, which makes debugging problems much easier
–enable-shared Build shared libraries. This is usually the default, but not always, so it's easier to be explicit every time.
–enable-static Build static libraries. This is usually not the default. Since the author often build libraries for a number of users, some of whom may need/prefer static libraries, he always specifies this.

It's not possible or practical to discuss all the options specific to any software package, such as what features to enable or disable, but it would be good in a classroom environment. For example here's some of the options from the configure script for FFTW 3.3.8[6]

–enable-single compile fftw in single precision
–enable-float synonym for –enable-single
–enable-long-double compile fftw in long-double precision
–enable-quad-precision compile fftw in quadruple precision if available
–enable-sse enable SSE optimizations
–enable-sse2 enable SSE/SSE2 optimizations
–enable-avx enable AVX optimizations
–enable-avx2 enable AVX2 optimizations
–enable-avx512 enable AVX512 optimizations
–enable-avx-128-fma enable AVX128/FMA optimizations
–enable-kcvi enable Knights Corner vector instructions optimizations
–enable-altivec enable Altivec optimizations
–enable-vsx enable IBM VSX optimizations
–enable-neon enable ARM NEON optimizations

At the end of the `--help` output environment variables will be listed that can be used to influence the behavior of `configure`. This list will be different for every package, but ther are some that are common to just about every package, such as these:

CC C compiler command
CFLAGS C compiler flags
LDFLAGS linker flags, e.g. -L<lib dir> if you have libraries in a nonstandard directory <lib dir>
LIBS libraries to pass to the linker, e.g. -l<library>
CPP C preprocessor
CPPFLAGS (Objective) C/C++ preprocessor flags, e.g. -I<include dir> if you have headers in a nonstandard directory <include dir>

It is always a good idea to use these environment variables to specify which compilers you want to use, such as a C compiler with CC. This makes sure you are using the desired compiler. This is especially critical in environments where you have more than one compiler installed (Intel and GCC, for example). If you have different versions of the same compiler, specify the full path to the correct version in CC to make sure you are using the correct version. To install version 2.2 of package "example" in $HOME/apps/example/2.2,

using the gcc compiler in /usr/local/bin, and enabling some of the common options the author recommends, the configure command would look like this:

```
./configure \
--prefix=$HOME/apps/example/2.2 \
--disable-silent-rules \
--enable-shared \
--enable-static \
CC=/usr/local/bin/gcc
```

Please note in the above example the backslashes at the end of each line are to escape the newline character at the end of each line. This enables the shell to treat those multiple lines as if they are one line. There can be nothing after those backslashes other than the newline character for this to work. The author prefers this syntax, since it allows long configure lines with many options to be easier to read.

CC can be defined as an environment variable before running configure, or put on the command-line before the configure command instead of after it, but the style shown above, where CC (and other environment variables) are defined on the command-line after the configure command, is actually recommended in Section 7.1 of the GNU Coding Standards[4]

Actually running configure can take several minutes, depending on how large and complicated the package being configured is. When configure completes, it will create a number of makefiles which will the guide the actual compiling and installation of all the files with the correct settings as determined by configure.

2.5 Make

The actual command that compiles and installs the software is make[3]. Make is tool that automates the compiling of software based instructions provided to it in *makefiles*. Make is a very powerful tool that deserves it's own training session. Knowing how it works is not really relevant to students in this training, but they should have a basic understanding of what it does at a high-level, so they have an idea of what they are doing when they run the make commands below.

To actually build the software, at this point, simply run the `make` command:

```
$ make
```

When the above command completes, the next step is to actually install the software, which means to copy the files to the correct locations and make sure ownership and permissions are set correctly. That is done with the `make install` command, like this:

```
$ make install
```

2.6 Post-install tasks

`make install` is the final step in *installing* an open-source software package, but there are couple steps that need to be completed before this software can actually be used. Environment variables such as PATH and LD_LIBRARY_PATH may need to be updated to include the installation locations of the the executables and libraries. Other variables thay may need to be updated include CPATH, C_INCLUDE_PATH, and MANPATH

2.7 Resources

The Filesystem Hierarchy Standard[11] is the definitive document for where files and directories should be located on any Linux operating system. As far as standards go, it's relatively brief, and easy to understand. Since it's freely available in PDF from from the FHS website, it's recommended to distribute it to students when this topic is covered in training as part of the instructional materials.

Environment variables are a feature of the shell. The bash is the most commonly used shell on Linux. There are numerous resources online and in print covering the bash shell, but the author is not familiar enough with any of them to recommend them as a resource.

Brian Gough's *Introduction to GCC*[7] provides an excellent overview of how to use the GCC compilers with many easy to understand examples, including how to use -I, -L and -l flags as mentioned above. For an instructor preparing a course on building open source software, this is a good resource for refreshing their knowledge of GCC if necessary. It is also suitable for distributing to the students as an instructional material.

Chapter 7 of the GNU Coding Standards[4] includes a section "How Configuration Should Work" which supplements the information provided here. It provides more detail than what is provided here, which could be useful to an instructor preparing to teach this topic. This same chapter includes information on make, which may be useful if an instructor would like to cover `make` as part of this curriculum.

GNU's Autoconf Manual[2] provides some introductory material that an instructor might find helpful when preparing their curriculum. Since GNU Autoconf is a tool for developers more than users, there is no need to be an expert on using GNU Autoconf to teach this material.

While the author chose not to discuss make in this curriculum, other instructors may feel differently, GNU's online documenation for `make`[3] is an excellent source for information about make.

3 HOW TO PACKAGE OPEN SOURCE SOFTWARE

The author is admittedly not an expert on packaging open-source software himself. In fact, he's never done it, but as someone who has built many open-source packages over the years, he has seen what makes one package easier or harder to install than another. In the remainder of this section, he recommends some best practices that should be employed by computational scientists when developing software to make it as easier as possible for other to use their software to make its use as widespread as possible.

All packages should have a clearly defined version number. This version information should be easily identifiable on the website for the software, in the source code archive, and after installation by running some command with the `--version` option. It is very difficult for users to know if they're using the latest version or not without this information. This information is often necessary when reporting a bug or determining if the current version in use has certain features. There are different version numbering conventions in use, and pros and cons of each convention could be a topic of discussion in the training.

Software developers should make sure the files from their software packages are put in locations that are consistent with with

current standards and conventions. The FHS[11], discussed in the previous section, is the definitive source for this information.

How to properly build the software package should be well documented both on the website for the software, as well as in a README or INSTALL file included in the source code archive. It should never be assumed that it is obvious how a package should be built.

Adhering to standards or commonly used conventions can make maintaining code easier, and make collaboration easier. If responsibility for maintaining the code is ever transferred to someone else, adhering to well-known conventions or standards will make that transition easier. The GNU Coding Standards[4] is one set of standards that could be used for this.

Automatic configuration tools like GNU Autoconf[1] should employed to make it easier for users to build the software correctly and as easily as possible. The previous section of this paper focused on building code with an GNU Autoconf-generated configure script because it is by far the most common tool for doing this. However, there are other tools out there that serve the same purpose, such as CMake[10] and Scons[5]. These tools can be discussed, but the author recommends focusing on GNU Autoconf, since that is currently the *de facto* standard for this.

4 CONCLUSION

This paper has identified building open-source software as a skills gap for computational scienists. It has provided an outline of what topics need to be taught to computational scientists in a logical order to train them to do this. The author has provided references to some of the topics discussed that could be used to develop training materials, or distributed directly to students as part of the training materials.

For computational scientists developing software, packaging this software in a way that makes it easier for users to build that software has also been identified as skills gap. Although the author is not an experienced software developer himself, he provided several best practices that should be taught to make it easier for others to build and use their software.

The author has successfully used the outline presented here to teach building of open-source software to junior coworkers. He hopes to continue refining this instruction and eventually include it HPC training workshops, such as the Software or HPC Carpentry programs mentioned here.

REFERENCES

[1] Free Software Foundation. 2009. GNU Autoconf. (2009). Retrieved September 28, 2018 from https://www.gnu.org/software/autoconf/autoconf.html

[2] Free Software Foundation. 2012. GNU Autoconf - Creating Automatic Configuration Scripts. (2012). Retrieved September 28, 2018 from https://www.gnu.org/software/autoconf/manual/index.html

[3] Free Software Foundation. 2016. GNU Make. (May 2016). Retrieved September 28, 2018 from https://www.gnu.org/software/make/

[4] Free Software Foundation. 2018. GNU Coding Standards: Configuration. (Aug. 2018). Retrieved September 28, 2018 from https://www.gnu.org/prep/standards/

[5] SCons Foundation. 2018. SCons: A software construction tool. (2018). Retrieved September 28, 2018 from https://socns.org/

[6] Matteo Frigo and Steven G. Johnson. 2005. The Design and Implementation of FFTW3. *Proc. IEEE* 93, 2 (2005), 216–231. Special issue on "Program Generation, Optimization, and Platform Adaptation".

[7] Brian Gough. 2004. An Introduction to GCC. (Feb. 2004). Retrieved September 28, 2018 from http://www.network-theory.co.uk/docs/gccintro/

[8] https://hpc carpentry.github.io. 2017. About HPC Carpentry. (2017). Retrieved September 28, 2018 from https://hpc-carpentry.github.io/about

[9] https://sns.ias.edu. 2009. Prospects in Theoretical Physics: Computational Astrophysics. (July 2009). Retrieved September 28, 2018 from http://www.sns.ias.edu/pitp2/index2009final.html

[10] Kitware. 2018. CMake. (2018). Retrieved September 28, 2018 from https://cmake.org/

[11] Daniel Quinlan Rusty Russell and Christopher Yeoh. 2004. Filesystem Hierarchy Standard. (Jan. 2004). Retrieved September 28, 2018 from http://www.pathname.com/fhs/pub/fhs-2.3.pdf

[12] Trace K. Teal, Karen A. Cranston, Hilmar Lapp, Ethan White, Greg Wilson, Karthik Ram, and Aleksandra Pawlik. 2015. Data Carpentry: Workshops to Increase Data Literacy for Researchers. *International Journal of Digital Curation* 10 (02 2015).

[13] Greg Wilson. 2006. Software Carpentry: Getting Scientists to Write Better Code by Making Them More Productive. *Computing in Science & Engineering* 8, 6 (November–December 2006), 66–69. https://doi.org/10.1109/MCSE.2006.122 Summarizes the what and why of Version 3 of the course.

[14] Greg Wilson, Jennifer Bryan, Karen Cranston, Justin Kitzes, Lex Nederbragt, and Tracy K. Teal. 1996. Good enough practices in scientific computing. *IEEE Computational Science and Engineering* 3, 2 (Summer 1996), 46–65. https://doi.org/10.1109/99.503313

[15] G.V. Wilson, R.H. Landau, and S.McConnell. 1996. What Should Computer Scientists Teach to Physical Scientists and Engineers? *IEEE Computational Science and Engineering* 3, 2 (Summer 1996), 46–65. https://doi.org/10.1109/99.503313

Using Virtual Reality to Enforce Principles of Cybersecurity

Jinsil Hwaryoung Seo
Department of Visualization
Texas A&M University
College Station, Texas
hwaryoung@tamu.edu

Michael Bruner
Department of Visualization
Texas A&M University
College Station, Texas
michael.bruner3@tamu.edu

Austin Payne
Department of Visualization
Texas A&M University
College Station, Texas
payn4478@tamu.edu

Nathan Gober
Department of Electrical and Computer Engineering
Texas A&M University
College Station, Texas
ngober@tamu.edu

Donald "Rick" McMullen
High Performance Research Computing
Texas A&M University
College Station, Texas
mcmullen@tamu.edu

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

ABSTRACT

The Cyberinfrastructure Security Education for Professionals and Students (CiSE-ProS) virtual reality environment is an exploratory project that uses engaging approaches to evaluate the impact of learning environments produced by augmented reality (AR) and virtual reality (VR) technologies for teaching cybersecurity concepts. 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. In its current implementation, the program supports undergraduate student education. The overarching goal is to develop the CiSE-ProS VR program for implementation at institutions with low cyberinfrastructure adoption where students may not have access to a physical data center to learn about the physical aspects of cybersecurity.

KEYWORDS

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

1 INTRODUCTION

Cybersecurity is a constantly evolving landscape. It is critical to raise awareness of disruptive computing technologies that result in new threats that appear on extremely short timescales. A recent report on the state of CS curricula in Texas, entitled "Building the Texas Computer Science Pipeline" recommended that students be aware of prevalent threats to personal information, prevalence of cyber-bullying, the increasing need for confidentiality [9]. In an increasingly technological era, students must learn to be conscious of digital citizenship and cybersecurity at an early age. While the software aspects of cybersecurity get prominent attention, physical access control to a cybersystems remains a high-priority requirement. Poor physical security may be out of compliance with government regulations, but also presents an incredible risk to the integrity of the machines in question. For this reason, software cybersecurity is built on a foundation of good physical access control.

As such, any cybersecurity training program must have a cybersecurity training program that emphasizes the physical aspects of cybersecurity. However, existing cybersecurity programs do not offer much training in this area. Further, 2- and 4-year universities alike have struggled to support sufficient faculty in computer education [13, 17]. As a result, certifications have become a favored source of cybersecurity education. We have developed a system that utilizes the emerging technology of virtual reality to enhance cybersecurity education at the university level, particularly in the area of physical security. In this paper, we present the development of the CiSE-ProS virtual reality (VR) program and a pilot study with high school students at the Summer Computing Academy at Texas A&M University.

The rest of the paper is organized as follows. A brief survey of government standards and regulations, as well as existing training solutions, is presented in Section 2. The benefits and limitations regarding the use of advanced technologies, including virtual reality, is discussed in Section 3. We discuss the importance of advanced computing education in Section 4, and the program to facilitate learning is outlined in Section 5. The design of the CiSE-ProS system is discussed in Section 6, followed by an evaluation of the system's effectiveness, in Section 7.

2 PREVIOUS WORK

The National Security Administration and Department of Homeland Security consider system administration and, by extension, access control to be a core knowledge unit, essential to any 2- or 4-year cybersecurity education program [14]. The National Institute of Standards and Technology publishes standards for security of data centers in the United States, including security of physical access controls. Its National Initiative for Cybersecurity Education emphasizes that "a knowledgeable and skilled cybersecurity workforce is needed to address cybersecurity risks within an organization's overall risk management process" [16]. The latest standards require organizations to verify individual authorizations for access to the data center, to maintain audit logs of access, to escort visitors on the premises, and to change combinations and locks when they are

https://doi.org/10.22369/issn.2153-4136/10/1/13

compromised, among other standards. High-sensitivity locations are further required to demand physical access authorization to the data center that is different than access to the surrounding facilities [11]. However, these standards, while essential for organizations, do not inform the physical security aspects of our practitioner training. These standards establish organization-level expectations, which is useful for management, rather than individual-level expectations, which is useful for practitioners.

The Computer Technology Industry Association (CompTIA) is a vendor-neutral certification provider that offers many ANSI-accredited security practitioner certification programs that include topics on physical access. The CompTIA A+ exam, an entry-level exam, requires its participants to compare and contrast a variety of physical security methods [5]. The CompTIA Network+ and Security+ exams, both considered intermediate-level exams, both require a summarial knowledge of physical security controls [7, 8]. Also, the CompTIA Advanced Security Practitioner (CASP) exam requires participants to analyze security components, including physical access control systems [6]. This suggests that familiarity with physical access control is a desired skill in the IT industry. These exams range from 90 to 165 minutes in length and cost between $211 and $439.

Texas A&M University offers a 16-hour minor field of study for undergraduates in cybersecurity [20]. Students may select from courses such as "Advanced Network Systems and Security" and "Cybersecurity and Digital Ethics". Other institutions have also implemented programs in cybersecurity, with some emphasis made on physical security. One such program is the Master of Science in Information Assurance and Security program at Sam Houston State University in Huntsville, Texas [19]. The 36-hour degree requires coursework in principles of access control and physical approaches to data protection. Graduate-level programs, while comprehensive, require previous undergraduate work, as well as significant investments of time and money. These barriers to entry reduce the accessibility of the field of cybersecurity, and do little to resolve the diversity gap which exists in the professional computing environment.

The National Security Agency and Department of Homeland Security designate over 200 degree-granting institutions across the country as National Centers of Academic Excellence in Cyber Defense Education, of which Texas A&M University is one [15]. Institutions designated as CAE-CDE must provide degree programs that equip students in a specified selection of essential cybersecurity topics. Of these, physical security is required to be addressed in introductory IT Systems Components courses, a foundational knowledge unit, and physical security is an important aspect in Security Program Management courses, a non-technical core knowledge unit. Hardware and Firmware Security, which focuses primarily on physical threats, is an optional knowledge unit [14].

All fields of study evolve over time, but cybersecurity chiefly is marked by a rapidly changing nature, underscoring the need for young cybersecurity professionals to be prepared to identify and mitigate new threats in their daily routine. These educational programs and government standards demonstrate a desire amidst society at large to have a well-trained cohort of computing technology and cybersecurity professionals.

3 ADOPTING EMERGING TECHNOLOGIES

Virtual reality and augmented reality are participatory technologies that provide the means to achieve engagement while underscoring the role of computing in scientific discovery and research. VR systems have seen increased adoption across industries since 1999, when researcher Fred Brooks, having taken a survey of VR technologies, concluded that it "barely works" [4]. Recently, VR systems are used to replace expensive rapid prototyping systems, perform research on ergonomics, and communicate ideas (including the playing of video games) [1]. More than a decade and a half has passed since students were first described as "digital natives," [18] and technology has advanced even more rapidly in the 21st century than before. Indeed, the students of today are the children of the students that Prensky studied. Students today are not merely proficient in emerging technologies, it is indigenous to them.

One ongoing challenge with the emerging technology is locomotion within the simulation. The method of control and locomotion can have a significant impact on the immersion and positive affect on the part of the user [2]. Free movement in the VR environment is restricted by the detection range of the motion sensors and the range of the communication between the wearable technology and the simulator, even if this communication is wireless. Additionally, the simulation may include obstacles to movement that are difficult to replicate physically in any dynamic way. VR systems must overcome the dual challenges of preserving immersion while allowing the user to move through an environment larger than the dynamic range of the VR system. Any data center environment is likely to be larger than the free movement restrictions of a VR environment, so in our CiSE-ProS system there must be some capability for locomotion.

Existing solutions for VR locomotion are varied, and few have seen widespread adoption. Some solutions place the user within a large external motion capture system and allow the floor to move beneath them while they walk. These solutions cannot currently create an accurate feel of a floor space and do not recreate the internal feeling of walking [3, 12]. Other solutions implement motion in software, either by teleporting the user on their command or by guided motion, as if fixed on a track. Of these, the teleportation, both free and to fixed points, imparts significantly fewer feelings of motion sickness on the user, in addition to being fast and easy to operate [10].

4 ADVANCING KNOWLEDGE AND UNDERSTANDING IN COMPUTING

Computing is a constantly evolving landscape. Disruptive computing technologies result in new threats that appear on extremely short timescales. Today's computer education curriculum must equip students to use existing technology while preparing them to use emerging technologies. VR and AR are likely to play important roles in computing technology in the future, given their current trajectory of development, so using them as part of other training programs will accomplish both ends.

Advanced computing resources, including high performance computing, high capacity storage, and enterprise-scale network devices, have become common across all industries. The concept of a data center is increasingly familiar, so training on the needs of

a data center benefits all professions that may use these advanced computing resources.

Physical access control to a data center is a high-priority requirement. The National Institute of Standards and Technology Special publication 800-53 requires that any system of moderate-to-high security implement an access control scheme that "allow[s] only authorized accesses for users... which are necessary to accomplish assigned tasks" [11]. In addition to compliance to regulations, limiting physical access to data center devices reduces the number of possible attack vectors on those machines.

If physical access to a data center is compromised, many assumptions that software cybersecurity systems make may be violated, including the topology of the network, the presence of any given machine in the data center, or even that a peripheral device is not malicious. A physical attacker can introduce devices that deceive the system, delivering maliciously incorrect information to other parts of the system, causing undesired behavior. For this reason, software cybersecurity is built on a foundation of good physical access control. Future practitioners in STEM fields will require exposure to advanced computing resources in order to be effective.

To that end, our CiSE-ProS system will be used at Texas A&M University in education programs targeted at students who are on educational tracks towards careers in STEM fields. The purpose of these programs are to usher in the next generations of cyber-practitioners in the country through hands-on exercises and active learning opportunities. Modern computing and cybersecurity education, and indeed modern STEM education, cannot simply impart programming knowledge, but must seek to develop in its students a well-rounded view of the computing fields.

5 PEDAGOGY

The underpinning 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. Part of these guided activities includes the use of our CiSE-ProS system. The use of the VR system is an opportunity for the student to engage in the material being taught in an experiential way. 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 surrounding program incorporates several elements of the experiential learning cycle in which:

- *New experiences:* Students are introduced to several aspects of cybersecurity
- *Processing ideas and taking ownership of ideas:* Skills developed in earlier guided practice are later revisited in exercises where students have opportunities to integrate and apply these skills to specific problems.
- *Opportunities to develop hypotheses to solve problems, and validate them:* Students must decide on which of their repertoire of skills to select and apply to problems.

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 [21]. In this approach, the learning objectives are defined in advance, and techniques are designed to support them. Such an approach to learning mirrors typical engineering design processes, where the goals and parameters of a product are established prior to the design phase. This parallel marks our pedagogical process already as a proven and effective one. In addition, STEM students seeing the backward design approach in pedagogy may be further encouraged to use it in their own design decisions.

We first identified the learning objectives and competencies that participants were expected to learn and built each exercise around them. Throughout the development process of CiSE-ProS, the focus is on developing scenarios that facilitate learning in the user. To develop desired capabilities, the CiSE-ProS program focuses on the described attributes:

- *Defining desired capabilities:* Trainees using the CiSE-ProS system should be able to describe physical access security measures and use them easily.
- *Operationalizing learning outcomes:* The CiSE-ProS system should provide an immersive environment where the trainee can experience these security measures and respond effectively to them. Additionally, the trainees can be observed by other students, leading to collaborative discussions wherein the instructor can guide students towards a better solution than the one demonstrated.
- *Evaluating learner development:* The learner is evaluated at all times while in the simulations by an operator. The learner also completes a survey following the experience to determine short-term retention of the material.

6 CISE-PROS VR

The CiSE-ProS VR seeks to enable aspiring computer scientists, developers, and engineers in their pursuit of computing fields by offering cybertraining programs that focus on cybersecurity. This effort seeks to help prepare the next generation of cyber-practitioners in the United States. The overarching objectives of the program are to:

- Use research-based methods to develop a high-impact, high-immersion opportunity that introduces participants to concepts in computing including software, hardware, networking, cybersecurity and data management practices,
- Reinforce and develop further knowledge of cyber skill sets through exercises,
- Retain participant interest after the camp by offering access to series of free in-person and online cybertraining-themed short courses and seminars at Texas A&M.

The CiSE-ProS VR program was developed to support users to learn cybersecurity principles through immersive and embodied tasks in the virtual data center environment. It offers a blend of cybersecurity and interactive visualization technologies to students in an innovative learning environment. The CiSE-ProS VR program is designed on the principles of engagement, training, retention, and sustainability to promote cyberinfrastructure as a professional career path. Virtual reality technology enables embodied/interactive learning that allows high engagement while underscoring the role of computing in scientific discovery and research. Simultaneously,

Figure 1: The HTC Vive system, which contains a headset, two handheld controllers, and two motion tracking sensors.

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.

6.1 Hardware

To fully utilize the potential of virtual reality, we chose an HTC Vive (Figure 1) that allows embodied actions including selection, manipulation and navigation within the virtual data center in the program. The HTC Vive was chosen as the primary hardware target for running this application due to its popular use in both the consumer market and development space for virtual reality applications. For instance, Unity 2017, the game engine used for developing this program, offers a great amount of support for initializing and running the HTC Vive hardware in a short amount of time. As a result, more time was allocated in developing the user interaction and implementing crucial elements for simulating the data center seen here. While the hardware was able to ease development constraints, the HTC Vive has also proven to be a popular product within a good portion of the target audience, making the system more accessible and easy to use.

6.2 User Training Scenario in CiSE-ProS VR

The main user experience consists of four activities in the virtual data center: 1) tutorial, 2) entering/exiting the data center, 3) inspecting the data center, and 4) replacing hardware.

Tutorial Room. Although the users within the target audience have shown to have moderately high technology literacy in using this hardware, it is still crucial to provide guidelines and rules so that new users are able to use the application by themselves. To compensate for this, the program offers a quick tutorial for the users to learn about the functionality provided and the expectations of them to complete the provided simulation later in the program. For instance, the user learns about how objects can be picked up with the controllers, by holding the triggers on the back of the controllers. To help introduce the concept, the tutorial stage provides rubber balls for them to pick up and throw within the environment (Figure 2). While this may seem fairly straightforward to some users within this program, it is important for the tutorial stage to also

Figure 2: The first instruction monitor in the tutorial room. The red balls on the table are used as demonstrations to familiarize the user with object manipulation in the virtual environment.

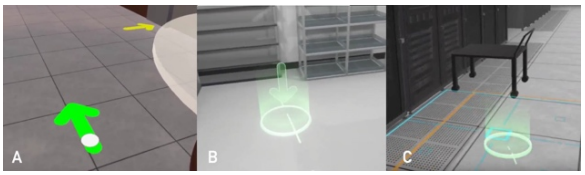


Figure 3: Iterations in the development of the locomotion tool, from the first iteration (A) to the current iteration (C).

account for those who may not be comfortable with the hardware just yet. As a result, this allows the user to interact with the given objects as much time as needed, before proceeding to the next on their own pace.

After completing the object interaction portion of the tutorial stage, users will learn how to navigate around the given space through the use of teleportation. This technique has already proven to be a popular navigational tool, and many iterations were created to achieve a better user experience as well as minimizing visual distractions throughout the data center (Figure 3). For instance, we used 2D arrows and 3D arrows as a navigational tool in the environment (Figure 3A). In this iteration, the user would simply point and click towards these objects above the ground and teleport the user to these points in space. When presenting this to users, there seemed to be frustration of not being able to see them clearly in the space, as well as not having enough feedback of where the controller was being pointed to. From these initial observations, it was decided that utilizing the SteamVR's teleportation tool was the better direction, in order to reduce these issues in a timely manner (Figure 3B). The next iteration of the teleportation tool used a grid on the ground for each teleportation point, allowing the user a larger degree of freedom of teleporting around the space (Figure 3C). In addition, the tool provided better visual feedback for the pointing and clicking portion of the program, as colors and trajectory of the cursor on the controllers were introduced. Although it showed

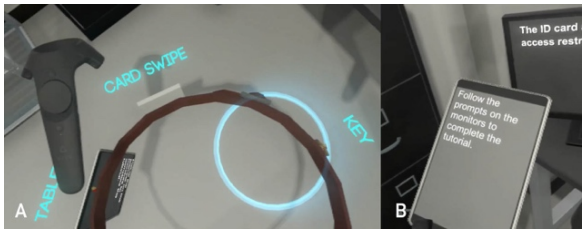


Figure 4: The user's tool belt in the VR simulation, containing (A, from left to right) a tablet, an ID card, and a rack key. (B) The tablet contains the last instructions given, so that the user can refer to it.

promises in improvement, the rendered grid for each teleportation point created more visual clutter in the data center. This also proved to be somewhat counter-intuitive for setting teleportation points, as tight corridor spaces created overlaps of these points, leading to less efficiency in navigating around the environment. As a result, the current iteration of this tool uses a similar approach with the arrows, but with better visual feedback through the use of colors and additional objects, such as having a circle below the arrow to show where exactly the point is located in the environment. In addition, the visual elements of the cursor was kept from the previous iteration, to prevent any recreation of the initial problems found earlier in the development process. With the combination of these elements in the tool, teleportation around the environment have shown to have better ease of navigation, based on a set of preliminary observations.

After users have learned how to use the locomotion tools, they make their way towards the other side of the room, where they pick up a tool belt before proceeding further into the level (Figure 4A). The tool belt is the most important aspect of the program, as users will need to be versatile and resourceful in completing tasks with the given items. The belt contains a card swipe and a regular key in this program, which are typical elements found in this type of profession. Users will need to use these items to gain access to secured areas, in a manner very much like how the average data center secures their resources and implements access control. In addition, the tool belt also features a tablet for the user to interact with. Since this type of tool is often used by those in this profession to understand the protocols of a particular data center, the tablet is used a guide for the users, in case they ever feel lost or confused about what needs to be completed next (Figure 4B). After the user has learned about the functionalities of all of the given items on their tool belt, they have successfully completed the tutorial stage and can proceed to the main areas of the data center to test their abilities.

Entering/Exiting Data Center. Users will next be brought into a lobby area, from where they can explore the data center. While in the lobby, users will receive instructions about the the main task at hand. The user will first need to navigate from the lobby area to the server room on the second floor of the building. To successfully accomplish this, they will need to gain control access of the elevator. In this virtual data center, access is granted by the use of a key card reader. The user can get access by swiping the card provided in the

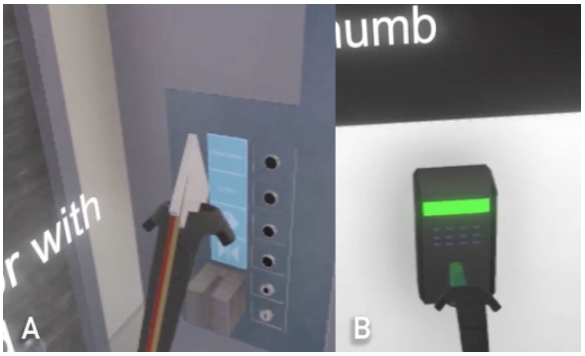


Figure 5: Tools to enter and exit the data center. (A) Using the ID card to operate the elevator. (B) Using the thumb scanner in the mantrap room.

Figure 6: Inspecting the data center. The rows of racks are lettered, and the positions within the rows are numbered. The user is instructed to navigate to rack B5.

tool belt (Figure 5A). Once the user has successfully passed the key card reader, he/she will be able to select buttons to move between different floors.

After selecting the "Data Center" button in the elevator, the elevator opens up to a "mantrap" room, a security clearance checkpoint before reaching the main server room. Here, users will approach the security guard behind the glass and scan his/her thumb on the thumb scanner in order to proceed to the main data center (Figure 5B).

Inspecting Racks. The user now can navigate the data center by teleporting to different areas of the room (Figure 6). While navigating, the user can find problematic nodes on the racks.

Replacing Hardware. One of the main tasks in this application is replacing a RAM on one of the server nodes within the area of the data center, while also following security protocols. To replace a RAM module, they will reach the main area of the program, where the broken node is located. Once they figure out its location, based on its given row number, they will start the process of removing and repairing a node.

First, they will need to remove the two cable attached in the front of the node (Figure 7). While it may vary on the amount of cables for certain data center, for the sake of this simulation, this

Figure 7: Removing Cables from the Node

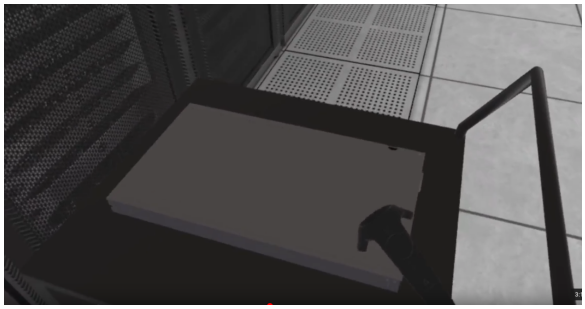


Figure 8: Placing the node on a cart for transport.

node only features two cables with their own distinct colors. Next, the node can be taken out, but it must be placed on the cart near the node (Figure 8). The reason for this is that these types of nodes are known to be extremely heavy, which is both difficult and not desirable to simulate accurately for this program. To help reinforce this idea, the program forces the user to place the node on a cart to transport it around the server room. After the node has been placed on the cart, they are able to move the cart into the workplace area to begin the repair process.

The repairment of the node begins with the user removing the top cover to view the computer parts and seeking for the broken RAM. For this program, the broken component is marked as red, while the rest of the computer parts are marked as green. The users will need to use their controllers to detach this part and replace it with the new RAM, which is provided on their workstation (Figure 9). Once they successfully replace the RAM, they will need to put the cover back on the node and return it to its server through cart transportation once more. From there, users will place the node back to its original spot, and reconnect the cables that were detached from earlier. After accomplishing this task, they will be able to exit the data center.

7 EVALUATIONS AND ASSESSMENTS

The earlier prototype of the CiSE-ProS VR simulator was demonstrated at the TAMU HPRC booth SuperComputing 17 Conference

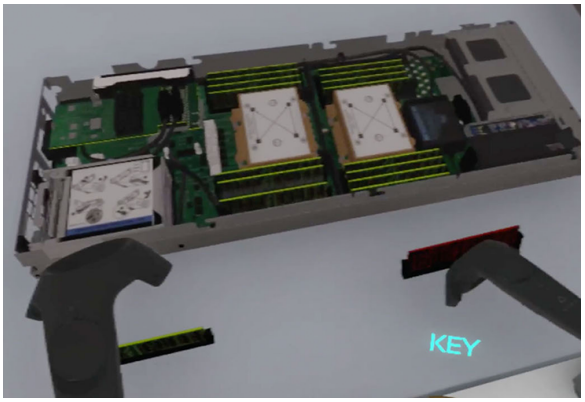


Figure 9: Replacing a defective RAM card.

in Denver, CO. The current version was tested with students who participate in the Summer Computing Academy at Texas A&M University. We collected participants' feedback using questionnaires. In addition, we collected background information to identify high-achieving or highly knowledgeable students who might need additional frameworks or scaffolds of instruction to be available. Evaluations and assessments are critical aspects of program refinement. Twenty-five students' virtual reality experiences were assessed using a post-experience survey. 80% of the participants had prior virtual reality experience: Google cardboard, HTC Vive, or Oculus Rift. They have used VR for mostly games and science education programs. They experienced the CiSE-ProS VR in a typical HTC Vive setting. Each student spent about 5 minutes in the program and filled out the survey afterwards. Their experiences with the CiSE-ProS VR application were very positive. 90% of students remembered the layers of data center physical security and the procedure of fixing a node with a broken RAM in the VR application. In addition, students acknowledged that virtual reality technology would be beneficial to education and were fascinated by interactive and immersive qualities of the virtual reality technology. The students' experiences were also assessed one week after the initial VR session. 80% of the participants still remembered the details of the data center security checkpoints and the procedure of replacing hardware in the node. Students acknowledged that virtual reality technology would be beneficial to education and were fascinated by interactivity, realistic simulation, and immersion. Some of their written responses include:

"VR is very interactive so it would be a good way to teach students that will keep them engaged," (ID03)

"It is very easy and fun to use and make, it's very easy to remember information." (ID05)

"You can explore places rather than read them in the text book." (ID18)

"Great for visual learners" (ID22)

8 CONCLUSION AND FUTURE DEVELOPMENT

Based on observations and feedback given on the current version of CiSE-ProS VR, we learn that embodied interaction in a virtual reality

training program can benefit students with short term and long term memories in engaging and playful ways. We also learn that there are a few elements that can be improved upon to help further build a more effective and engaging version of this application. For instance, during the tutorial stage, it appeared that some users were confused about how to use the controls, despite our efforts of using audio, text, and even visual cues on the controller to teach them this information. As a result, it is expected that visual graphics that vividly depict the motion and button presses in front of the user will have better results for users retaining and understanding this information. It also appeared that users weren't able to differentiate when to use the card swipe and key, based on the wording of the directions given to them. By establishing clear definitions and wording of these item's uses, this should be able to help mediate this problem.

In addition to improving these elements, the inclusion of new areas or components of this program is also being heavily considered as long-term goals. For instance, the current iteration of this program only focuses on one particular scenario. In the future, the implementation of more scenarios to choose from that tests either or both routine maintenance or emergency situations would make the program more applicable for others to utilize its potential in learning. To further build upon this mentality, it has also been suggested to allow users to customize the layout of a data center, security levels, and scenarios to mimic closely to a particular data center, to help better retain information in a similar environment. While this direction is tailored for those interested in expanding the learning potential, the cognitive effects of using this application is also being considered as well, as interests has been shown in analyzing the effectiveness and the outcomes of using this program, in comparison to other learning methods offered.

ACKNOWLEDGMENTS

The authors would like to thank staff, student workers and researchers at Texas A&M HPRC, Department of Visualization, the Laboratory for Molecular Simulation, TexGen, Division of Research and Provost IT for supporting the HPRC short course program at Texas A&M University. Portions of this research were conducted on the Ada and Terra clusters, and virtual machines provided by TAMU HPRC. We gratefully acknowledge support from the National Science Foundation Abstract 1730695 (https://www.nsf.gov/awardsearch/showAward?AWD_ID=1730695) "CyberTraining: CIP: CiSE-ProS: Cyberinfrastructure Security Education for Professionals and Students." We also appreciate Dell for providing VR laptops. Special thanks to the instructors of each course: Dylan Rodriguez, Michael Dickens, Rick McMullen, Lisa Perez, Mark Huang, Ping Luo, Jian Tao, Yang Liu, Marinus Pennings, Keith Jackson, Noushin Ghaffari, and Shichen Wang. We also gratefully acknowledge support from Francis Dang, Mark Huang and Jack Perdue for maintaining the clusters and virtual machines used in these efforts.

REFERENCES

[1] Leif P. Berg and Judy M. Vance. 2017. Industry Use of Virtual Reality in Product Design and Manufacturing: A Survey. *Virtual Reality* 21, 1 (2017), 1–17.

[2] Max Birk and Regan L. Mandryk. 2013. Control Your Game-self: Effects of Controller Type on Enjoyment, Motivation, and Personality in Game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*, Wendy E. Mackay, Patrick Baudisch, and Michel Beaudouin-Lafon (Eds.). ACM, Paris, France, 685–694. https://doi.org/10.1145/2470654.2470752

[3] Ian Bishop and Muhammad Rizwan Abid. 2018. Survey of Locomotion Systems in Virtual Reality. In *Proceedings of the 2nd International Conference on Information System and Data Mining (ICISDM '18)*. ACM, Lakeland, FL, USA, 151–154. https://doi.org/10.1145/3206098.3206108

[4] Frederick P. Brooks. 1999. What's Real About Virtual Reality? *IEEE Computer Graphics and Applications* 19, 6 (1999), 16–27.

[5] Computer Technology Industry Association. 2018. CompTIA A+ Certification Exam Objectives. https://certification.comptia.org/docs/default-source/exam-objectives/comptia-a-220-902-exam-objectives.pdf

[6] Computer Technology Industry Association. 2018. CompTIA Advanced Security Practitioner (CASP) Certification Exam Objectives. https://certification.comptia.org/docs/default-source/exam-objectives/comptia-casp-objectives-(cas-002).pdf

[7] Computer Technology Industry Association. 2018. CompTIA Network+ Certification Exam Objectives. https://certification.comptia.org/docs/default-source/exam-objectives/comptia-network-n10-007-v-3-0-exam-objectives.pdf

[8] Computer Technology Industry Association. 2018. CompTIA Security+ Certification Exam Objectives. https://certification.comptia.org/docs/default-source/exam-objectives/comptia-security-sy0-501-exam-objectives.pdf

[9] Carol L. Fletcher. 2014. *Building the Texas Computer Science Pipeline: Strategic Recommendations for Success.* Technical Report. Texas Regional Collaboratives. https://www.thetrc.org/web/assets/files/pdfs/Building-the-Texas-CS-Pipeline_Fletcher.pdf

[10] Julian Frommel, Sven Sonntag, and Michael Weber. 2017. Effects of Controller-Based Locomotion on Player Experience in a Virtual Reality Exploration Game. In *Proceedings of the 12th International Conference on the Foundations of Digital Games (FDG '17)*. ACM, Hyannis, Massachusetts, USA, 30.

[11] Joint Task Force Information Initiative. 2013. *Security and Privacy Controls for Federal Information Systems and Organizations.* Technical Report. National Institute of Standards and Technology. https://doi.org/10.6028/nist.sp.800-53r4

[12] William E. Marsh, Tim Hantel, Christoph Zetzsche, and Kerstin Schill. 2013. Is the User Trained? Assessing Performance and Cognitive Resource Demands in the Virtusphere. In *Proceedings of the 2013 IEEE Symposium on 3D User Interfaces (3DUI)*, Anatole Lécuyer, Frank Steinicke, and Mark Billinghurst (Eds.). IEEE Computer Society, Orlando, Florida, USA, 15–22. https://doi.org/10.1109/3DUI.2013.6550191

[13] National Academies of Sciences, Engineering, and Medicine. 2018. *Assessing and Responding to the Growth of Computer Science Undergraduate Enrollments.* The National Academies Press, Washington, DC. https://doi.org/10.17226/24926

[14] National IA Education and Training Programs. 2018. *Centers of Acadcemic Excellence in Cyber Defense 2019 Knowledge Units.* Technical Report. National Security Agency and Department of Homeland Security. https://www.iad.gov/NIETP/documents/Requirements/CAE-CD_2019_Knowledge_Units.pdf

[15] National Security Administration. 2018. Information Assurance Directorate at the NSA. https://www.iad.gov/NIETP/reports/cae_designated_institutions.cfm Retrieved December 12, 2018.

[16] William Newhouse, Stephanie Keith, Benjamin Scribner, and Greg Witte. 2017. *National Initiative for Cybersecurity Education (NICE) Cybersecurity Workforce Framework.* Technical Report. National Institute of Standards and Technology. https://doi.org/10.6028/nist.sp.800-181

[17] One Hundred Fifteenth Congress of the United States of America, First Session 2017. *Public-Private Solutions to Educating a Cyber Workforce: Joint Hearing Before the Subcommittee on Cybersecurity and Infrastructure Protection of the Committee on Homeland Security, House of Representatives and the Subcommittee on Higher Education and Workforce Development of the Committee on Education and the Workforce, House of Representatives.* One Hundred Fifteenth Congress of the United States of America, First Session, U.S. Government Publishing Office, Washington. http://purl.fdlp.gov/GPO/gpo90199 pp. 42-43,66.

[18] Marc Prensky. 2001. Digital Natives, Digital Immigrants Part 1. *On the Horizon* 9, 5 (2001), 1–6.

[19] Sam Houston State University. 2018. Academic Catalog 2018-2019. https://catalog.shsu.edu/

[20] Texas A&M University. 2018. Cybersecurity–Minor. http://catalog.tamu.edu/undergraduate/engineering/cybersecurity-minor/cybersecurity-minor.pdf

[21] Grant P. Wiggins and Jay McTighe. 2005. *Understanding by Design* (expanded 2nd ed.). ASCD, Alexandria, Virginia, USA.

Towards an HPC Certification Program

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

Kai Himstedt
Nathanael Hübbe
Hinnerk Stüben
Sandra Schröder
Michael Kuhn
Matthias Riebisch
Stephan Olbrich
Thomas Ludwig
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 HPC community has always considered the training of new and existing HPC practitioners to be of high importance to its growth. This diversification of HPC practitioners challenges the traditional training approaches, which are not able to satisfy the specific needs of users, often coming from non-traditionally HPC disciplines, and only interested in learning a particular set of competences. Challenges for HPC centres are to identify and overcome the gaps in users' knowledge, while users struggle to identify relevant skills.

We have developed a first version of an HPC certification program that would clearly categorize, define, and examine competences. Making clear what skills are required of or recommended for a competent HPC user would benefit both the HPC service providers and practitioners. Moreover, it would allow centres to bundle together skills that are most beneficial for specific user roles and scientific domains. From the perspective of content providers, existing training material can be mapped to competences allowing users to quickly identify and learn the skills they require. Finally, the certificates recognized by the whole HPC community simplify inter-comparison of independently offered courses and provide additional incentive for participation.

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 center 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. Understanding SLURM is a good example of a very fine-grained skill, indeed we can identify "resource management" as a generic skill that illustrates concepts across the rich variety of available resource managers. Institutions which operate HPC systems typically offer regularly teaching events about general aspects of their supercomputer's hard- and software architecture and about the software environment around parallel programming and optimization. The learning material provided by an HPC center, however, is geared to the special demands of the institutions they support and its specific HPC environment. This content typically covers a small part of basic HPC skills which are necessary to use other HPC systems. Moreover, they do not attest the users to have a certain competence. Certificates are widely used in industry to attest certain knowledge but, so far, there is no similar approach for HPC training.

This lightning talk describes the current status of the certification program and the effort to create an independent body that curate the competences and issue certificates for the users.

The foundation for this work was laid with the Performance Conscious HPC (PeCoH) project, which among other goals, aims to establish an HPC certification program. A white paper about the approach containing technical details is provided in [1]. Within the project it become quickly apparent that the local effort is useful for the wider HPC community and could be extended to a global effort.

2 RELATED WORK

Relevant work can be classified into approaches to establish a curriculum or the creation of teaching material. In academia, individual universities offer their own curriculum around scientific computing and HPC, covering theoretical aspects like the software development of numerical applications. They are not tailored to the needs of a practitioner to actually use HPC systems effectively. Data centers offer their own material and courses to support their own users. Several projects address the generation and sharing of teaching material for HPC. The EuroLab-4-HPC project establishes training

https://doi.org/10.22369/issn.2153-4136/x/x/x

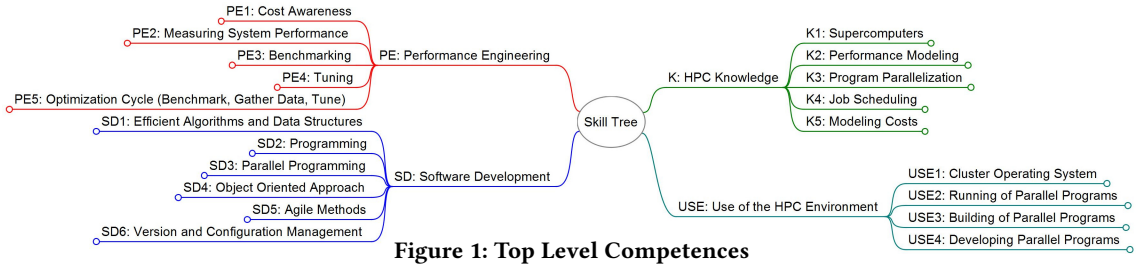


Figure 1: Top Level Competences

in form of (online) courses[1]. The Barcelona Supercomputing Centre (BSC) aims to develop a professional training curriculum [2]. The virtual organization XSEDE[2] provides an online system to train the usage of an HPC system, structuring the corresponding information on their website into major topics like "Getting Started". The user can navigate the topics and receive further information.

3 CERTIFICATION PROGRAM

The certification program serves two purposes: 1) the definition and organization of fine-grained competences (skills); 2) the establishment of certificates and (online) exams that confirm that users possess a certain skill. Note that the certification program does not regulate the content – the definition of skills and certificates is separated from content creation. This allows the re-use of existing content but also allows to create a new ecosystem in which HPC centers or commercial companies could offer the best teaching material. Teaching material should be marked to indicated which skills it covers. 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. We assume that the collaboration of scientific institutions will complement each other in producing a rich variety of content for the different learning styles.

3.1 Skills

The skills represent competences in a fine-grained fashion. A skill is defined by a unique key, short name, background, level, and a description of what it encompasses. This model can be compared to the classification of school knowledge, for example, the skill with the short name "addition" could describe the math skill of being able to add numbers successfully. The level serves the purpose of distinguishing the expertise further, a *basic level*, for example, may mean to be able to add two numbers between 1-20 while the *expert level* of that skill could indicate to be able to add any numbers.

The individual skills are organized into a tree that shows generic competences close to the root and refined skills on the leafs. The two top-levels of the skill tree are shown in Figure 1; we have identified more than 35 skills. In respect to the granularity, we expect the basic level of a leaf like "Bash programming" can be acquired by novel users in a workshop day. The tree serves the purpose of navigating across the skills, and at the same time a parent node defines the scope of its children. For example, the *USE* competence provides means of using the HPC environment to perform various tasks, such as running parallel applications, using core services of the operating system. While the tree structure has been chosen as a graphical representation, some competences are cross-referenced, particularly in the skills of the USE branch. The tree is managed in an XML file and we offer tools to visualize the skills as mindmap or embed them into a webpage – more information are found in [1].

3.2 Certificates

Certifying users with a certain competency is a core element of the program. Since we identified so many skills, it is not useful to perform an exam on a single competence like "SLURM". Therefore, initially, we aim to group sets of skills into certificates and establish an online examination that attests users that they mastered a certain skill. To be meaningful these tests must prevent cheating to some extent. However, as any examination can be cheated with sufficient effort, we focus on practical aspects like a huge corpus of questions and some instantiated questions like *how would you start ProgramX with 4 MPI processes?*

4 CONCLUSIONS

The HPC Certification program offers a strategy to classify and organize HPC competences. While the idea started with the PeCoH project, we are supporting the HPC Certification Forum[3] which is an independent international body that aims to sustain the work. The certification forum has the role of a (virtual) central authority to curate and maintain the skill tree and certificates. Moreover, the forum supports tools and an ecosystem around the competences.

ACKNOWLEDGMENTS

This work was supported by the German Research Foundation (DFG) under grants LU 1353/12-1, OL 241/2-1, and RI 1068/7-1. The authors acknowledge the discussion with Hendryk Bockelmann and Markus Stammberger.

REFERENCES

[1] Kai Himstedt, Nathanael Hubbe, Julian Kunkel, and Hinnerk Stuben. 2018. An HPC Certification Program Proposal Meeting HPC Users' Varied Backgrounds. https://www.hpc-certification.org/downloads/hpccp-concept-paper-180201.pdf
[2] Maria-Ribera Sancho. 2016. BSC best practices in professional training and teaching for the HPC ecosystem. *Journal of Computational Science* 14 (2016), 74–77.

[1] https://www.eurolab4hpc.eu/
[2] https://portal.xsede.org/web/xup/training/overview
[3] https://hpc-certification.org

Potential Influence of Prior Experience in an Undergraduate-Graduate Level HPC Course

Chris Fietkiewicz
Electrical Engineering and Computer Science Department
Case Western Reserve University
Cleveland, OH 44106 USA
001-216-368-8829
chris.fietkiewicz@case.edu

ABSTRACT
A course on high performance computing (HPC) at Case Western Reserve University included students with a range of technical and academic experience. We consider these experiential differences with regard to student performance and perceptions. The course relied heavily on C programming and multithreading, but one third of the students had no prior experience with these techniques. Academic experience also varied, as the class included 3rd and 4th year undergraduates, master's students, PhD students, and a non-degree student. Results indicate that student performance did not depend on technical experience. However, average overall performance was slightly higher for graduate students. Additionally, we report on students' perceptions of the course and the assigned work.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education - Computer Science Education, Curriculum

General Terms
Education.

Keywords
Undergraduate, graduate, education, high performance computing.

1. INTRODUCTION
Graduate level courses at universities are typically open to undergraduate students with significantly less academic experience. Additionally, such courses can attract students from multiple disciplines and departments due to a shared interest in a particular topic. The potential for a high diversity in backgrounds and experience levels poses challenges for instructors. Previously, we investigated the potential influence of technical and academic experience levels for a single homework assignment in a class on high performance computing (HPC) at Case Western Reserve University in Cleveland, Ohio [1]. In that study, it was found that prior experience was not a significant predictor of a student's performance with regard to implementing a successful programming solution. In the present study, we look at the student outcomes for the course as a whole, and we consider how students' backgrounds may influence perceptions of the course.

DOI: https://doi.org/10.22369/issn.2153-4136/10/1/15

2. METHODS
The class was taught during the Spring semester of 2018 at Case Western Reserve University in Cleveland, Ohio. The total enrollment was 23 students, including undergraduate, graduate, and non-degree students. The course had been offered twice before, and course evaluation statistics were available to prospective enrollees. At the beginning of the course, survey data was collected to determine whether students had prior experience with C programming and multithreading. Six main HPC techniques were covered in the course and are listed below:

- Batch job processing
- General optimization for sequential programming
- Parallel programming using spawned (forked) processes
- Parallel programming using OpenMP and multithreading
- Parallel programming using OpenACC and GPUs
- Parallel programming using message passing and MPI

All students were graded using the same criteria and rubrics. Assignments consisted of seven programming projects on required topics and a three-week course project that focused on an application of the student's choice. The seven programming assignments were designed to apply the above HPC techniques to four different applications. Assignments generally focused on either introducing an application or comparing different HPC techniques. The four applications covered in the programming assignments are listed below:

- Sorting algorithms (e.g. merge sort)
- Matrix multiplication (iterative and recursive)
- Prime number discovery
- Numerical integration of Laplace's equation

Assignments generally included 3 or more separate problems to be solved. Below is an example of a typical problem statement that requires parallel processes for the discovery of prime numbers:

Count the number of prime numbers up to two different maxima N_1 and N_2. Choose maxima such that the serial-version run time for N_1 is at least 5 seconds and for N_2 is at least 10 seconds. For parallel versions, using 2 and 4 processes respectively, each process should do an approximately equal amount of work (same approximate run time). For parallel versions, report the speedup as a ratio of the serial-version run time to the parallel-version run time. In your report, explain how you equalized the work, and briefly discuss how the speedup compares to the number of processes.

Prior to each assignment, lectures were provided on the requisite material, including discussion of all sample programs. For the example problem statement above, sample C programs were

provided to demonstrate the use of the fork() instruction and a serial algorithm for discovering prime numbers (see APPENDIX for sample programs).

Assignments were designed to provide explicit instructions that would be understandable to typical undergraduates. The instructions had stringent reporting requirements that included a thorough explanation of methods, highly detailed timing results, and a careful discussion of results. The primary requirement of the discussion section of each report was a textual observance of any trends in the results and whether the student found the results to be as expected. Students were not required to accurately explain any anomalies.

3. RESULTS

Table 1 shows the distribution of students by level, including subcategories for undergraduate and graduate students. Graduate students include Doctoral and Master's. Undergraduates include juniors (3rd year) and seniors (4th year). Results also include one student who was of non-degree status but had bachelor's degrees in two related fields.

Table 1. Distribution of students by level.

Level	*Number*	*Portion*
Doctoral	4	17%
Master's	6	26%
Senior	10	44%
Junior	2	9%
Non-degree	1	4%
Total	23	100%

In the following analyses, we organized students into three categories: doctoral, master's + non-degree, and undergraduate. The non-degree student is included in the same group as the master's students because their academic backgrounds were equivalent. For the undergraduate category, we combined the seniors and juniors because there were only 2 juniors, and their performances fall within the bounds of the distribution for the seniors.

Figure 1. Course scores by academic experience. Left: Doctoral. Middle: Master's + non-degree. Right: undergraduate. In this box-and-whisker plot, horizontal bars indicate quartiles, and the X indicates the mean.

We used the final score for the course (maximum of 100) and compared the three categories of students. The scores for the three categories are analyzed in Figure 1. The mean scores are 96.5 for Doctoral students, 94.9 for Master's students, and 93.7 for undergraduates. It can be seen in Figure 1 that the mean score decreases as the level of academic experience decreases.

All students had significant programming experience, but 35% (n = 8) reported having no significant experience with C programming or multithreading. We analyzed the course scores (maximum of 100) based on whether or not students had this prior technical experience. The results are shown in Figure 2.

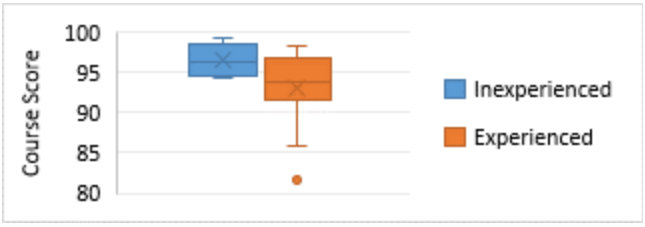


Figure 2. Course scores by technical experience. Left: No prior experience with C programming or multithreading. Right : Prior experience. In this box-and-whisker plot, horizontal bars indicate quartiles, X indicates the mean, and the circle indicates an outlier.

The mean scores are 96.5 for technically inexperienced students and 93.0 for technically experienced students. It can be seen in Figure 2 that the mean score for the inexperienced students was higher than that of the experienced students. These results can be understood by looking at the levels of academic experience within these groups. The graduate students were more likely to lack technical experience, having come from other programs at other institutions. In fact, the inexperienced students were comprised of 87.5% graduate students, while the experienced students were comprised of only 33.3% graduate students. Because graduate students generally had higher scores (see Figure 1), this accounts for the negative correlation with technical experience, indicating that academic experience is more important in predicting success in the course.

We also considered students' perceptions of the course in an effort to characterize the appropriateness of graded work. Anonymous course evaluations were submitted by 11 students. As the evaluations were entirely anonymous, it is not possible to separate them according to academic experience. Overall, students gave the course a rating of 4.09 on a scale of 1 – 5. Students were asked to provide anonymous comments on the assigned work, and all comments were positive in this regard. We provide only one example below that was similar to the other student comments:

"The assignments he gave really helped me understand the content of this course and help me to understand how to implement it to any other algorithm out there. He also tells you what he expects to see in the report for each assignment."

All comments regarding graded work indicated that the problems were relevant and instructions were clear.

4. DISCUSSION

We have presented a course comprised of both graduate and undergraduate students. Because the course required a high degree of technical competence, we expected that technical experience might be an advantage to students and be reflected in student performance. To the contrary, however, we found that academic experience was correlated to performance, and technical experience may have no correlation at all, assuming adequate coverage in class is provided.

Different reasons are possible for the correlation between performance and academic experience. In the most general sense, graduate students may simply be more capable of working with larger projects and report writing, as compared to undergraduates.

Though we did not track requests for help from the instructor, we did perceive that graduate students appeared more likely to seek help and request clarifications regarding the instructions.

In the future, we will consider two changes to our course design to improve relative performances of graduate and undergraduate students. First, we will consider requiring graduate students to do additional project work and reporting, as compared to undergraduates. This is a well known practice, and it is clearly appropriate in our course. A second consideration in the future will be to administer post-assignment surveys that allow students to reflect on their performance and possible influences. Survey results could be used to identify challenges common to undergraduates.

5. ACKNOWLEDGMENTS

This work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University.

6. REFERENCES

[1] Fietkiewicz, C. (*in press*) Student Outcomes in Parallelizing Recursive Matrix Multiply. *Journal of Computational Science Education.*

7. APPENDIX: Sample Code

Figure 3: C program that demonstrates fork instruction:

```
pid_t pid;
/* fork a child process */
pid = fork();
if (pid == 0) { /* child process */
  printf("Child pid = %d\n", pid);
}
else { /* parent process */
  printf("Parent pid = %d\n", pid);
  /* wait for the child to complete */
  pid = wait(NULL);
  printf("Child %d is done.\n", pid);
}
```

Figure 4: C program for discovering prime numbers:

```
int nMax = 100; // Upper limit
int n, d, isPrime;
for (n = 2; n <= nMax; n++) {
  isPrime = 1;
  for (d = 2; d < n; d++){
    if (n % d == 0){
         isPrime = 0;
         break;
    }
  }
  // Print each prime number
  if (isPrime == 1)
    printf("%d ", n);
  }
}
```

Deep Learning by Doing: The NVIDIA Deep Learning Institute and University Ambassador Program

Xi Chen
University of Kentucky
Lexington, Kentucky
billchenxi@gmail.com

Gregory S. Gutmann
Tokyo Institute of Technology
Tokyo, Japan
gutmann@c.titech.ac.jp

Joe Bungo
Deep Learning Institute, NVIDIA Corporation
Austin, Texas
jbungo@nvidia.com

ABSTRACT

Over the past two decades, High-Performance Computing (HPC) communities have developed many models for delivering education aiming to help students understand and harness the power of parallel and distributed computing. Most of these courses either lack a hands-on component or heavily focus on theoretical characterization behind complex algorithms. To bridge the gap between application and scientific theory, NVIDIA Deep Learning Institute (DLI) (nvidia.com/dli) has designed an on-line education and training platform that helps students, developers, and engineers solve real-world problems in a wide range of domains using deep learning and accelerated computing. DLI's accelerated computing course content starts with the fundamentals of accelerating applications with CUDA and OpenACC in addition to other courses in training and deploying neural networks for deep learning. Advanced and domain-specific courses in deep learning are also available. The online platform enables students to use the latest AI frameworks, SDKs, and GPU-accelerated technologies on fully-configured GPU servers in the cloud so the focus is more on learning and less on environment setup. Students are offered project-based assessment and certification at the end of some courses. To support academics and university researchers teaching accelerated computing and deep learning, the DLI University Ambassador Program enables educators to teach free DLI courses to university students, faculty, and researchers.

KEYWORDS

Hands-on learning, HPC Education, Open edX, Deep learning, Professional education

1 INTRODUCTION

With an increasing emphasis on using computing as the new scientific experimentation method, computer science has become an interdisciplinary academic area. Deep Learning (DL), an emerging and powerful tool for machine learning, has gained attention in recent years due to its potential to reshape the future of computational research. Many ground-breaking results rely on DL, such as image classification[11], Atari game bots[10], and medical diagnosis[9], and have achieved superhuman levels of performance. One reason for this recent quantum leap in research has its roots in the growing prevalence of High-Performance Computing (HPC).

HPC provides the parallel and distributed computing power that allows deep learning to excel. NVDIA CUDA-powered GPUs have several times enabled the title of the fastest supercomputer in the world since 2010[2]. Since then, the new setup for most HPC research has been parallel systems incorporating CPUs with GPUs[7].

Despite decades-long efforts in HPC education, HPC and DL are still understood and used by only a small portion of the scientific and engineering community[6]. There two main problems contributing to the low rates of adoption: 1. the material requires a certain amount of computational and statistical literacy; and 2. HPC programming environments are difficult to set up due to the variety of hardware and OS system types. Overcoming these two concerns requires curating learning material for a broad range of audiences that includes practical cases, as well as abstracting the system implementation to simplify the use of HPC resources.

Combing current online learning andragogy with a cloud computing platform consisting of a VM and Docker containers, we illustrate a new educational platform designed by the NVIDIA Deep Learning Institute as showing in Figure 1. This platform provides hands-on experience with DL and facilitates practical DL and HPC education.

1.1 High Performance and GPU Accelerated Computing Education

High Performance Computing and GPU acceleration used to be accessible only to scientists and engineers to fulfill their desire to better model realistic physical systems. As computing hardware becomes cheaper and its performance improves, these resources are no longer found only in mega research centers. Major tech companies such as Microsoft, Google, and Amazon now provide low-cost HPC instances in the cloud, and major academic programs have gradually embraced the convenience and efficiency of HPC because of it. Despite these efforts there is still a large unmet need for new, powerful, HPC-enabled tools and curricula for education.

To extend exposure and build a larger base within the HPC research community, many government agencies have initiated programs aimed to develop awareness at all stages of the education pipeline. These include the Education, Outreach and Training (EOT) program from the National Science Foundation (NSF), the Advanced Scientific Computing Initiative (ASCI) funding program, and the High-Performance Computing Modernization Program (HPCMP) from the U.S. Department of Energy (DOE), among others. Such

https://doi.org/10.22369/issn.2153-4136/10/1/16

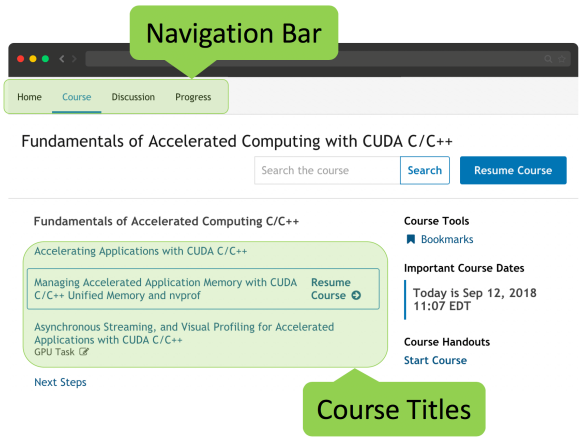
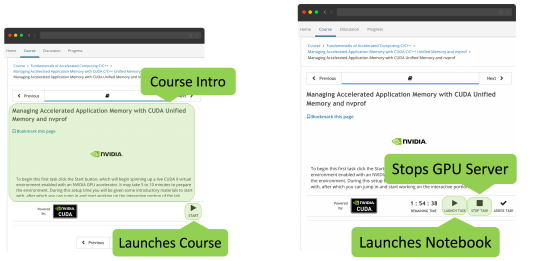


Figure 1: CUDA C/C++ course on DLI online platform. The Navigation Bar allows students navigate across different sections of the platform. Courses that contain multiple subsections will list all course titles as url links to navigate to the course content.



(a) To start a course, click the "Start" button.

(b) How to access Jupyter Notebook. To stop GPU server and end a course, click "Stop Task."

Figure 2: Individual course landing page.

initiatives have a huge effect in developing awareness of HPC at all stages of the education system.

While waiting for these efforts to pay off, and in response to the immediate need for HPC, hundreds of modules and online training sessions have been created for users from the world's supercomputing centers. While these approaches are widespread and effective, they cannot be scaled to reach a larger audience as they are centered around a set of one-to-one tutorials and are only available for certain infrastructures. In addition, due to the high demand of research needs, only limited HPC resources from these computing centers are allocated for education and experimentation by the beginning learners and students.

1.2 Abstractions through CUDA

CUDA, in essence, is a minimal extension of the C and C++ programming language. As a scalable parallel programming platform, it allows sophisticated HPC programing to be expressed in an easily understood abstraction. After its release in 2007, CUDA rapidly evolved into a popular Application Programming Interface (API) model that facilitates a wide range of research and applications[5].

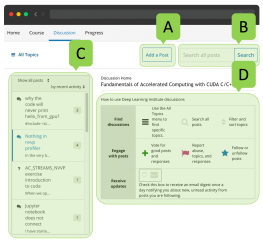
CUDA provides three key abstractions: a hierarchy of thread groups, shared memories, and barrier synchronization. The CUDA paradigm allows programmers to partition a "problem into coarse subproblems that can be solved independently in parallel, and then into finer pieces that can be solved cooperatively in parallel"[7] while hiding memory management and thread synchronization behind the scenes. In addition, NVIDIA provides a CUDA profiler to help programmers further understand and debug each parallel process to achieve the best possible performance (Figure 7).

1.3 NVIDIA's Deep Learning Institute for HPC and DL Education

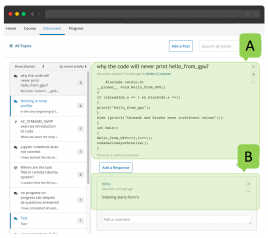
NVIDIA's Deep Learning Institute (DLI) aims to lower the barrier of entry for HPC education and develop novel ways of enabling scientists and engineers in their own research through the use of HPC resources [www.nvidia.com/dli]. Based on the experience from previous instructor-led courses via DLI's University Ambassador Program, cloud-based computing solutions for training and education reduces the learning curve and allows students to gain hands-on experience with parallel computing systems without getting bogged down setting up programming environments. Participants can earn certification to prove subject matter competency and support professional career growth. Certification is offered for select online courses and instructor-led workshops. [www.nvidia.com/dli] All students need to participate in the hands-on training is a computer with a modern browser installed and a reliable internet connection. DLI is currently using GPU virtual instances in combination with Jupyter Notebooks to implement a hands-on, project-based experience in DL and accelerated computing. Participants can easily click a "Launch Task" or "Stop Task" button to begin and end their learning tasks as part of a full course, as showing in Figure 2. If a student has issues or questions about the course material at any point, they can participate in a MOOC-style discussion session to post their questions or search for solutions, as showing in Figure 3. For programming questions beyond the scope of the course, NVIDIA provides online forums and documentation that allow students to find answers to more complex questions.

DLI offers five full-day, hands-on "Fundamentals" courses for those who may be new to accelerated computing or deep learning. Two of DLI's most popular courses with project-based certification are Fundamentals of Deep Learning for Computer Vision and Fundamentals of Accelerated Computing with CUDA C/C++. To prepare learners with more practical knowledge, DLI also provides application-specific content in the following disciplines:

- Deep Learning for Autonomous Vehicles
- Deep Learning for Healthcare
- Deep Learning for Digital Content Creation
- Deep Learning for Finance



(a) Users can post a topic (A), search existing topics (B), and browse all topics (C). The instruction of how to use Discussion (D).



(b) Individual topic layout: (A) Title and content of the topic, and (B) responses from members or course staff.

Figure 3: Discussion panel layout.

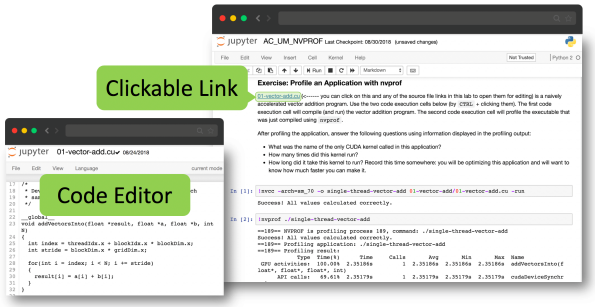


Figure 4: Inside the DLI CUDA C/C++ course - Managing Accelerated Application Memory with CUDA C/C++ Unified Memory and nvprof. Course content including links that allow students open browser-based code editors.

In this article, we present a hands-on, online course provided by DLI: Fundamentals of Accelerated Computing with CUDA C/C++ (Figure 4). This course provides a path that allows learners with little or no knowledge of HPC GPU acceleration to harness the power of parallel computing and possibly achieve an official CUDA certification based on project-based assessments.

The rest of the article is organized as follows: In Section 2, we present the technologies used in the design of the course. Section 3 highlights the University Ambassador Program, and Section 4 demonstrates the results of teaching events across the globe. We conclude with a summary of the work.

2 METHODS

When developing courseware, DLI combines the primary design elements of HPC abstraction and hands-on experience with deep learning. Before DLI courses, the vast majority of learners were using personal computers or relying on traditional batch-processing application programming interfaces (APIs) to manage a HPC environment. Instead of convincing participants to follow the traditional HPC education paradigm, DLI focused on providing access to fully-configured GPU servers in the cloud that abstract complicated environment setup, HPC project-based courses and tasks available

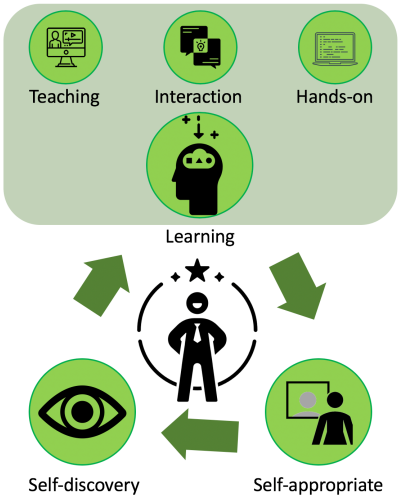


Figure 5: By combining teaching and interaction with instructor and hands-on practice, learning promotes self-discovery and self-appropriate. which leads to future application.

to students perpetually after training for students to refer back to course materials at any time.

Since NVIDIA's 2007 introduction of CUDA, a parallel computing platform API, and the emergence of many open source software libraries extended from CUDA such as TensorFlow, TensorRT, Torch, Caffe, CNTK, etc., access to high-performance computation has given researchers the scientific and parallel computing abstractions they need to utilize HPC resources. Therefore, all DLI courses and labs provide CUDA-powered GPU acceleration. NVIDIA also provides cloud-computing support through the NVIDIA GPU Cloud (NGC): GPU-accelerated containers available on-demand on all major cloud platforms for accelerating deep learning and scientific research.

2.1 Reflection-in-action Model

From the very beginning of DLI's course development, the emphasis has always been on hands-on experience and a journey of reflection-in-action. For participants of all backgrounds, learning by doing provides the ultimate method to master a skill, especially in the case of continuing learning throughout a professional career. For live, instructor-led courses, students receive active coaching and participate in a transparent teaching experience with a instructor which helps students understand how and why they are learning course content in particular ways. A similar experience is provided by DLI online self-paced courses. By presenting students with real-world problems, feedback and project-based assessment, students develop a much better grasp of the course material. A trial-and-error approach to solve in-lecture tasks provides students with opportunities for self-discovery and self-appropriate learning [Schon 1987] (Figure 5). We believe a hands-on andragogy encourages students to go beyond what the course work provides and makes it more likely that they will later apply what they have learned in their own disciplines.

If students are enthusiastic and want to immediately apply what they have learned, NVIDIA also provides documentation webpages, FAQs, and expert support to facilitate the efforts of course participants.

2.2 OpenEdX MOOC and Cloud Delivery Method

Massive Open Online Courses (MOOCs) are no longer a new concept in the educational landscape. Open edX, a non-profit MOOC platform, emerged in 2012 [Porter et al. n.d.]. The main aim behind it is to provide essential education for a particular topic to anyone, and sometimes at any time,either toward a degree or just to satisfy one's own personal learning desires. One advantage of this approach is the ability to reach many attendees across geographical location and technology barriers. The Open edX platform includes both a Content Management System (CMS) and a Learning Management System (LMS)[8]. DLI has adopted this platform and delivers courses to help address the pressing educational needs in the HPC and communities.

Unlike traditional education, MOOC-style platforms provide an immediate feedback assessment technique (IF-AT) that allows students to receive immediate results that assess their knowledge[4]. IF-AT encourages students to self-discover and self-explore new knowledge, provides an enjoyable learning experience, and evidently improves students' retention of the course content[3].

To date, DLI have planned and delivered hundreds of instructor-led trainings, created more than 50 courses and labs across the globe, and have fully embraced the MOOC paradigm.

One obstacle that many HPC and parallel computing beginners face is the environment setup. To learn from a hands-on programming project, students need a clean and functional setup so the debugging process can focus on coding problems instead of HPC environment issues. DLI uses a cloud container solution to allow students to focus more on learning and less on environment setup. DLI uses fully-configured GPU servers in the cloud to provide this immersive programming environment. Cloud containers are a lightweight technology to virtualize applications in the cloud. They allow elastic and rapid resource pooling to provide a fully functional parallel CPU and GPU programming environment[1]. The average setup time is below 5 minutes for most DLI courses. To further simplify the process for students, DLI uses an automated script to abstract the whole process behind the scenes as showing in Figure 2.

2.3 Interactive Course Interface

Here we illustrate the DLI interface using the Fundamentals of Accelerated Computing C/C++ course as an example. Students navigate through the course via a series of links. Each course page links to areas within four subgroups: Home (the landing page), Course, Discussion, and Progress (Figure 1). The actual training content is located in the course section and developed into modules using flecture videos and practical/programming sessions to reinforce the key concepts, as shown in Figure 1. Navigating between units and modules is as simple as clicking on the links, completing assessments, and leaving the interactive assessment lab by clicking the start/launch and stop buttons (Figure 2).

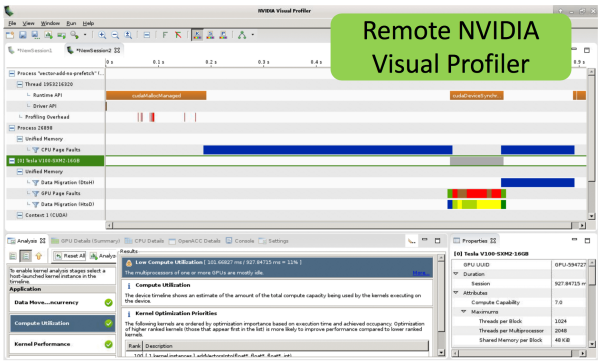


Figure 6: Remote Nvidia Visual Profiler captures all of the GPU metrics.

As part of the LMS, a progress page keeps track of all the questions and projects that a student has completed as well as the corresponding assessment results for each module. This allows both students and instructors have an at-a-glance grasp of the students performance to date. The hands-on assignments (Figures 4) were developed using Jupyter notebooks allowing students to receive immediate feedback while working though the content. In the case of the CUDA C/C++ course, the remote NVIDIA Visual Profiler (Figure 6) enabled students student a deep understanding from the GPU visual metrics allowing them to better determine the bottlenecks in a GPU function.

3 DLI UNIVERSITY AMBASSADOR PROGRAM

3.1 DLI Mission

The mission of the DLI is to help engineers and researchers solve extremely challenging problems using AI and deep learning. To achieve this end, it is necessary to utilize HPC platforms. DLI provides the education essential for using such platforms leveraging massively parallel GPUs. DLI helps "developers, data scientists and engineers to get started in architecting, optimizing, and deploying neural networks to solve real-world problems in diverse industries such as autonomous vehicles, healthcare, robotics, media & entertainment and game development." [nvidia.com/dli]

As stated above, one of the most important elements of the DLI course design is the hands-on experience, which is integrated into both the self-paced and instructor-led versions of the courses. Course materials are currently available across many disciplines ranging from finance to biology, from computer vision to autonomous vehicles, and from game development to accelerated computing [nvidia.com/dli]. More industry-specific content is coming soon.

3.2 Bringing DLI to Campus: University Ambassador Program

DLI recognizes and awards qualified academics as applied deep learning experts. "DLI University Ambassador" is an additional status academics and researchers can achieve on top of a DLI instructor certification. The University Ambassador Program enables

educators to teach free instructor-led DLI courses exclusively to university students and staff. This program is free to join and provides a wealth of benefits to academic communities looking to bring AI and deep learning to their campuses. The Ambassador Program offer free DLI instructor certification. provides ready-made, world-class educational content to universities, and offers expense reimbursement for travel and catering expenses for instructor-led workshops. Ambassadors leverage DLI content in their university curriculum courses, campus-wide workshops, and to satisfy workshop and tutorial submissions at academic conferences around the world. For more information about this program visit www.nvidia.com/dli.

3.3 DLI Teaching Kits

In an effort to extend the HPC community and encourage students to harness the power of DL, DLI offers downloadable Teaching Kits co-developed with well-known academic experts and universities. Each kit includes curriculum materials for a semester-long university course. Complementing DLI's application and hands-on approach, Teaching Kit content integrates more academic theory to satisfy the needs of traditional university coursework. Albeit, in addition to lecture slides, video, and textbook materials, there is at least one accompanying hands-on lab with full source code solutions found in private Git repositories. Many Teaching Kit modules offer multiple labs and solutions. The Teaching Kit program also enables educators to give free access to online, self-paced DLI courses and student certification by way of promotional codes on the MOOC-style platform. To learn more about these resources please visit developer.nvidia.com/teaching-kits. The current Teaching Kits and academic co-authors includes:

- Machine/Deep Learning (NYU/Yann LeCun)
- Accelerated/Parallel Computing (UIUC/Wen-Mei Hwu)
- Robotics (CalPoly)

Most Teaching Kits contain:

- Lecture slides
- Lecture videos
- Hands-on labs with solutions
- Larger coding projects with solutions
- Quiz/Exam questions with solutions
- Electronic textbooks
- DLI online self-paced promotional codes
- Syllabus with specific DLI online labs interleaved

3.4 Fundamentals of Accelerated Computing with CUDA C/C++

In the Fundamentals of Accelerated Computing courses, DLI introduced CUDA parallel computing platform that aims to accelerating computing in terms of impressive performance and ease of use. CUDA supports many popular programming languages such as C, C++, Fortran, Python and MATLAB and expresses parallelism through extensions in the form of basic keywords. CUDA has an ecosystem of highly optimized libraries for DNN, BLAS, graph analytics, FFT, and more, and also ships with powerful command line and visual profilers. Here we present a glimpse of the course to help readers have a better understand of the structure and contents of the course on how to use CUDA to accelerate computing in C/C++.

Figure 7: Slides on differentiating GPU-accelerated vs. CPU-only applications.

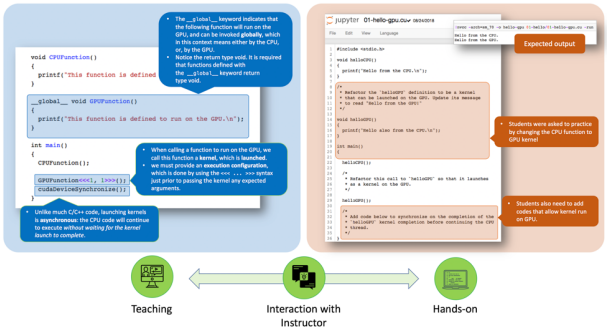


Figure 8: Hands-on experience reinforces learning. Through hands-on coding practice, students have chance to learn though doing. It also opens opportunities to interact with instructor.

To help students ease in the parallel programming paradigm, the CUDA course first differentiates the GPU-accelerated vs. CPU-only applications through a series of animation as showing in Figure 7. Then after CUDA syntax and keywords introduction, a hands-on task follows: students need to modify a CPU C/C++ function into a GPU kernel as showing in Figure 8. (Due to the space limitation, we will not further include detailed and exciting course contents.) For every hands-on task, comments in the code will assist student work and the solutions are available in case they get stuck.

Following the CUDA design cycle: Assess Parallelize, Optimize, Deploy (APOD), in the second and third labs of the course, students will learn how to use command line and visual profilers to further optimize their CUDA code. Along the way, CUDA course will also introduce concepts such as Unified Memory, Streaming Multiprocessors, asynchronous memory prefetching, manual memory allocation and copying, Streams and so on. All in the efforts to help students to have a better grasp of the HPC programming bottle-necks in real-world scenario and to offer solutions on how to toggle them. In Figure 9 we showed that using the command-line and visual profilers one can quickly and qualitatively measure the performance of a application, and to identify opportunities for optimization.

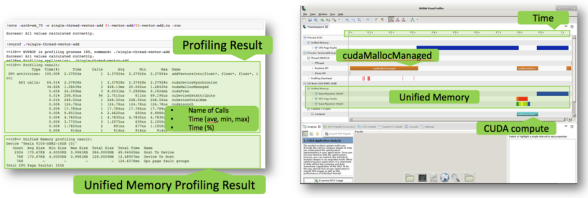


Figure 9: Command line and Visual Profilers. In second and third lab, students will learn how to use profiler to measure the performance and identify opportunities for optimization.

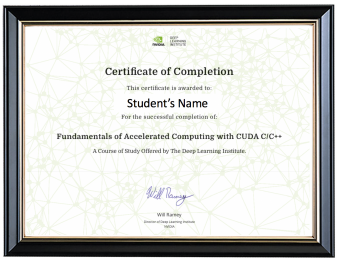


Figure 10: Certificate of Competency. After completing the course and passing the project-based assessment, students receive a certificate of course competency.

After students finish the course and pass the assessments, they will receive a certificate of course completion, as shown in Figure 10. Every certification has a unique identification number and online hyperlink, and is tied directly to the student. Students are actively linking their certifications from their resumes and LinkedIn profiles. At the time of this writing, there are no other hands-on, project-based assessment and certification programs in applied deep learning.

4 RESULTS

DLI has awarded over 150 certified University Ambassadors across the globe, and they have delivered over 150 instructor-led workshops in less than a year's time since 2017. Many academics have included DLI course materials in their own syllabi to prepare their students to solve real-world problems using AI and parallel computing.

4.1 Reaching Students Globally

At UC Berkeley, Prof. Rekesha and Prof. Arias have modified the content of their courses to include a full life-cycle example from the DLI of how Deep Learning can be used, and to illustrate the relationship between Deep Learning and CPU/GPU-powered hardware. At National Tsing Hua University, Prof. Lee has integrated DLI course materials into teaching activities. His team, NVISION, won the 2016 NVIDIA Intelligent Robotics challenge.

In India, Prof. Chickerur of KLE Technology University has conducted several DLI training events training thousands of students. The feedback from students continues to be positive and favorable, especially due to the fact that some courses offer certification they can use to bolster their professional career in the future.

4.2 Feedback from University Ambassador Instructor

DLI content primarily focuses on the hands-on application of deep learning and parallel computing. This creates a good complement to the theoretical approaches of traditional academic curricula. DLI courses enable students who are new to deep learning quickly jump-start their journey of solving real world problems. The content is not designed to provide a low-level explanation on the core components of deep learning such as mathematics and statistics.

At Tokyo Institute of Technology, Prof. Gutmann (author of the paper) conducted a pilot workshop using the CUDA course and online platform described in this paper, and there were no reported issues of students using the online learning platform despite the use of various host machines and operating system types. At the start of each topic within the course Prof. Gutmann would give a very brief introduction to the concepts, then the students were able to go through the material and solve the exercises that were corresponding to each topic. Unlike traditional lectures where students have little interactivity with the content and tend to easily lose interest and not be actively engaged, Prof. Gutmann's students remained enthusiastic with the hands-on materials and teaching instructions. At the end of the course, many students were able to use what they learned from the course and their hands-on exercises to solve the assessment task of accelerating a particle system code, resulting in their individual certificate of competency. Students who did not finish on-sight were able to finish over the next couple days on their own, as the course platform conveniently provides students perpetual access to the online version of the course.

At University of Kentucky (UK), students from the Association for Computing Machinery organization also conducted the DLI CUDA C/C++. DLI Certified University Ambassador Xi Chen (author of the paper) taught the workshop paving the way for students to have a better understanding of how to accelerate and parallelize applications using GPU computing. UK students were very excited about learning the concepts from the course materials, and also from each other, since Xi Chen himself happens to be a PhD student. The CUDA course demystified GPU programming for the students, and the hands-on exercises provide a platform that allows students to challenge each other with positive reinforcement. Many students expressed how the course made it simple to quickly learn how to harness the power of CUDA and GPUs, and they were interested in future DLI courses.

4.3 Feedback from Students

The feedback from students who attended the DLI CUDA C/C++ course was quite positive and encouraging, shown in Figure 11. None of the students had difficulty navigating the course platform, and all of them expressed interest in future DLI workshops in AI and accelerated computing. The feedback survey showed that almost all of the students thought the instructors were helpful during the learning path, with 86% of them indicating the instructors were extremely helpful. Part of this can be attributed to DLI's rigorous instructor certification process consisting of a collection of instructor

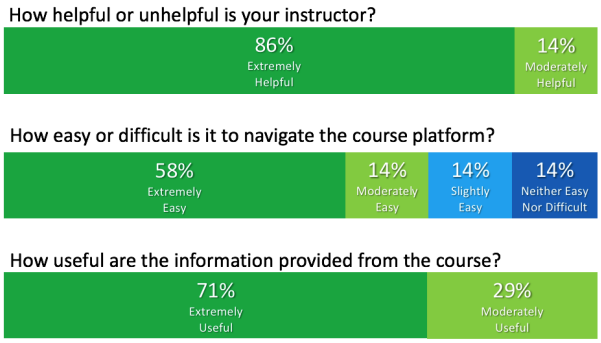


Figure 11: Summary of the Student Feedback Survey for Fundamentals of Accelerated Computing with CUDA C/C++.

assessments and in-person interviews with DLI master instructors. When asking how useful the information provided by the course might be for solving real-world problems and science, 71% of the students suggested the information is extremely useful, while 29% indicated moderately. Regarding the question about how helpful the course was for sharpening their HPC programming knowledge in general, 72% suggested it was extremely helpful, 14% expressed moderately helpful, and 14% indicated slightly helpful.

We also asked what percentage of the course should be instructor-led vs. self-paced, and students suggested they should be roughly 50% versus 50%. There were a few suggestions on how to improve the course delivery: 66% expressed a desire to add more low-level details and theory in the course, and 17% wish the course content was easier to understand. Other free-form comment responses from the survey included more time was needed for the exercises and more practical examples and less number crunching would be beneficial.

5 CONCLUSIONS

Like the discovery of the electricity, Deep Learning is fundamentally changing the world using scalable HPC platforms as the medium of education for this technology. Major breakthroughs in computer vision, Natural Language Processing (NLP) and autonomous driving go hand-in-hand with the growth of HPC. Three related goals form the foundation of DLI course design: (1) to provide training to assist the AI and HPC communities in fulfilling its educational needs; (2) to help researchers transition to using the technologies future HPC will build upon; and (3) to explore novel learning and application-based teaching paradigms. We present the platform, content and programs from NVIDIA's Deep Learning Institute, which aims to train researchers, scientists and professional engineers how to solve real-world problems with AI and accelerated computing. We believe that deeper hands-on experiences provide the most rapid and effective style of technology education.

ACKNOWLEDGMENTS

The authors would like to thank NVIDIA Deep Learning Institute for providing the figures, documents and online course platform, and Tokyo Institute of Technology and ACM University of Kentucky chapter for their helps organizing the course workshops.

The authors would also like to thank the anonymous reviewers for their valuable comments and helpful suggestions.

REFERENCES

[1] David Bernstein. 2014. Containers and Cloud: From LXC to Docker to Kubernetes. *IEEE Cloud Computing* 1, 3 (Sept. 2014), 81–84.
[2] Shane Cook. 2013. *CUDA Hardware Overview.* Morgan Kaufmann.
[3] Michael L Epstein, Amber D Lazarus, Tammy B Calvano, Kelly A Matthews, Rachel A Hendel, Beth B Epstein, and Gary M Brosvic. 2017. Immediate Feedback Assessment Technique Promotes Learning and Corrects Inaccurate first Responses. *The Psychological Record* 52, 2 (May 2017), 187–201.
[4] Andrew Ho, Isaac Chuang, Justin Reich, Cody Coleman, Jacob Whitehill, Curtis Northcutt, Joseph Williams, John Hansen, Glenn Lopez, and Rebecca Petersen. 2015. HarvardX and MITx: Two Years of Open Online Courses Fall 2012-Summer 2014. *SSRN Electronic Journal* (March 2015).
[5] David Luebke and 2008. [n. d.]. CUDA: Scalable parallel programming for high-performance scientific computing. In *2008 5th IEEE International Symposium on Biomedical Imaging (ISBI 2008).* IEEE, 836–838.
[6] Julia Mullen, Chansup Byun, Vijay Gadepally, Siddharth Samsi, Albert Reuther, and Jeremy Kepner. 2017. Learning by doing, High Performance Computing education in the MOOC era. *J. Parallel and Distrib. Comput.* 105 (July 2017), 105–115.
[7] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. 2008. Scalable parallel programming with CUDA. In *ACM SIGGRAPH 2008 classes.* ACM Press, New York, New York, USA, 1.
[8] B Porter, M Haseltine, N Batchelder The Open edX Conference, and 2015. [n. d.]. The state of Open edX.
[9] Ravi, Daniele, Wong, Charence, Deligianni, Fani, Berthelot, Melissa, Andreu-Perez, Javier, Lo, Benny, and Yang, Guang-Zhong. [n. d.]. Deep Learning for Health Informatics. *IEEE journal of biomedical and health informatics* 21, 1 ([n. d.]), 4–21.
[10] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (Jan. 2016), 484–489.
[11] C Szegedy, W Liu, Y Jia, P Sermanet, and S Reed. 2015. Going deeper with convolutions. (2015).

Using CloudLab as a Scalable Platform for Teaching Cluster Computing

Linh B. Ngo
West Chester University
West Chester, Pennsylvania
lngo@wcupa.edu

Jeff Denton
Clemson University
Clemson, South Carolina
denton@clemson.edu

ABSTRACT

A significant challenge in teaching cluster computing, an advanced topic in the parallel and distributed computing body of knowledge, is to provide students with an adequate environment where they can become familiar with real-world infrastructures that embody the conceptual principles taught in lectures. In this paper, we describe our experience setting up such an environment by leveraging CloudLab, a national experimentation platform for advanced computing research. We explored two approaches in using CloudLab to teach advanced concepts in cluster computing: direct deployment of virtual machines (VMs) on bare-metal nodes and indirect deployment of VMs inside a CloudLab-based cloud.

CCS CONCEPTS

• **Applied computing → Education**; • **Computer systems organization → Cloud computing**; • **Software and its engineering → Distributed systems organizing principles**;

KEYWORDS

experimental platform, distributed computing, hands-on learning

1 INTRODUCTION

Within a parallel and distributed computing undergraduate curriculum, advanced topics are defined as those carrying significant current or emerging interests [15]. Among these topics, cluster computing remains an advanced yet fundamental subject that provides the background information for other topics such as cloud/grid computing and big data computing. Since the early adaptation of cluster computing into the undergraduate curriculum [2], a large body of materials has been developed to support in-class lectures. The challenge is providing a hands-on environment for students to connect these materials to real-world technical problems and skill sets.

To address this challenge, institutions have turned to virtual technology as an economical and scalable platform. Previous work has shown the applicability of virtual machines in teaching computer networking, operating systems, security, and databases [4, 19]. More recent work has seen virtual machines used in courses for big data [7] and cloud computing [8]. Even with virtual technology, many technical hurdles remain. Most notable is the necessary investment of time, money, and effort from both the instructors and the institutions into setting up and maintaining the virtual machine images and hardware infrastructure as well as the accompanying lecture materials.

In this paper, we present our approach in leveraging CloudLab [16], a publicly available computing resource as a platform to develop and host materials to support teaching topics in cluster computing. Using CloudLab, we demonstrate how instructors can develop scalable, maintainable, and shareable contents that minimize technical hurdles while still exposing students to critical concepts in cluster computing. The remainder of this paper is organized as follows. Section 2 provides an overview about the CloudLab platform. Next, we describes in details our gradual integration of CloudLab into materials for a distributed and cluster computing course over several semesters in Section 3. Lessons learned are discussed in Section 4. We examine related work in Section 5, including those that leverage CloudLab in general computer science education and those that leverage other resources besides CloudLab specifically for high performance computing education. Section 6 concludes the paper and discusses future work.

2 CLOUDLAB

Funded by the National Science Foundation in 2014, CloudLab has been built on the successes of the Global Environment for Network Innovations (GENI) [3] in order to provide researchers with a robust cloud-based environment for next generation computing research [16]. These resources are distributed across several U.S. institutions. As of Summer 2018, CloudLab boasts an impressive collection of hardware. At the Utah site, there is a total of 785 nodes, including 315 with ARMv8, 270 with Intel Xeon-D, and 200 with Intel Broadwell. The compute nodes at Wisconsin include 270 Intel Haswell nodes with memory ranging between 120GB and 160GB and 260 Intel Skylake nodes with memory ranging between 128GB and 192GB. At Clemson University, there are 100 nodes running Intel Ivy Bridges, 88 nodes running Intel Haswell, and 72 nodes running Intel Skylake. All of Clemson's compute nodes have large memory (between 256GB and 384GB), and there are also two additional storage-intensive nodes that have a total of 270TB of storage available. CloudLab is currently expanding under a follow award from the National Science Foundation which will include more network interfaces, new CPU architectures, and the ability to interface with other cloud services including Amazon Web Services and the Massachusetts Open Cloud [17].

In order to provision resources using CloudLab, a researcher needs to describe the necessary computers, network topologies, startup commands, and how they all fit together in a resource

https://doi.org/10.22369/issn.2153-4136/10/1/17

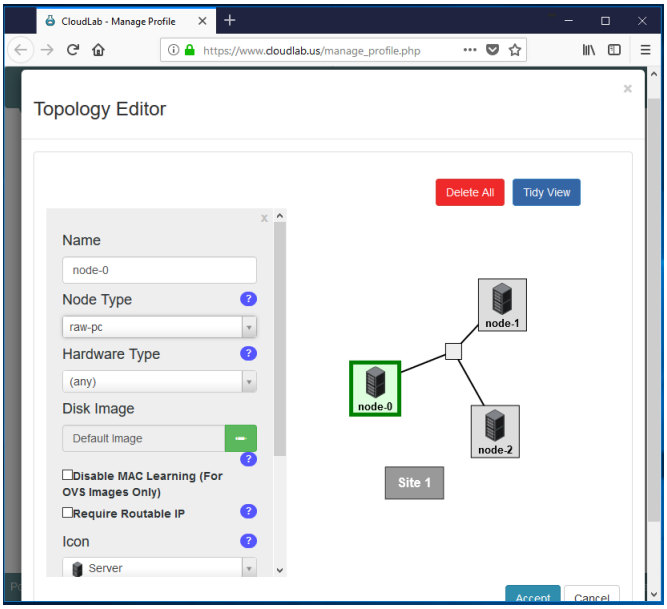


Figure 1: CloudLab's GUI for designing experimental profiles

description document. CloudLab provides a graphical interface, as shown in Figure 1, inside a web browser that allows users to visually design this document through drag-and-drop actions. For large and complex profiles, this document can also be automatically generated via Python in a programmatic manner as demonstrated in Listing 1. Starting Fall 2017, CloudLab supports a direct integration between publicly readable git repositories and their profile storage infrastructure. This significantly minimizes the effort needed to modify existing profile while still maintaining a complete history of previous changes.

Listing 1: A CloudLab profile written in Python to describe a 6-node experimental profile

```
import geni.portal as portal
import geni.rspec.pg as pg
import geni.rspec.igext as IG

pc = portal.Context()
request = pc.makeRequestRSpec()

link = request.LAN("lan")
for i in range(6):
  if i == 0:
    node = request.XenVM("head")
    node.routable_control_ip = "true"
  elif i == 1:
    node = request.XenVM("metadata")
  elif i == 2:
    node = request.XenVM("storage")
  else:
    node = request.XenVM("compute-" + str(i))
    node.cores = 2
    node.ram = 4096

  node.disk_image = "urn:publicid:IDN+emulab.net+
      ↪ image+emulab-ops:CENTOS7-64-STD"

  iface = node.addInterface("if" + str(i-3))
  iface.component_id = "eth1"
  iface.addAddress(pg.IPv4Address("192.168.1." + str(
      ↪ i + 1), "255.255.255.0"))
  link.addInterface(iface)
pc.printRequestRSpec(request)
```

The resource description document provides the blueprints for CloudLab to provision resources and instantiate the experiments. Once the resources are allocated and images for the computing components are booted on top of bare metal infrastructure, CloudLab users are granted complete administrative privilege over the provisioned infrastructure. Like XSEDE, CloudLab allows instructors to apply for educational projects and to add students to these projects.

3 USING CLOUDLAB TO TEACH CLUSTER COMPUTING CONCEPTS

At Clemson University, the distributed and cluster computing course is intended to present students with a broad overview of key components and concepts in distributed and cluster computing. The topics covered include the Beowulf model of networked computers, distributed file systems, the message-passing programming paradigm, scheduling on a cluster of computers, big data and data-intensive computing, and the map-reduce programming paradigm. The goal of this course is to provide students with the fundamental concepts in distributed and cluster computing and hands-on exposure to latest real world technologies and platforms built and expanded on these concepts.

3.1 Course History

The distributed and cluster computing course is at junior level, but most students in the class wait until the first or second semester of their senior year before registering. Typically, the enrollment ranges between 35 and 40 students. With the availability of Clemson University's centralized supercomputer with more than 2000 compute nodes, early offerings of the course in 2012 through 2014 had focused primarily on the large-scale programming aspects for both high performance and big data/data-intensive computing. This includes MPI-based concepts such as pleasantly parallel, divide-and-conquer, and synchronous computing and the MapReduce programming paradigms for big data. While the course covered distributed infrastructure knowledge such as the Beowulf cluster architecture, parallel and distributed file systems, and the Hadoop Big Data infrastructure, students' understanding of these concepts were assessed only through in-class examination. All take-home assignments were designed to be programming-based.

Over time, we had begun to realize several short-comings of the course regarding its practicality for students. There exists other courses that focus primarily on parallel programming in MPI and

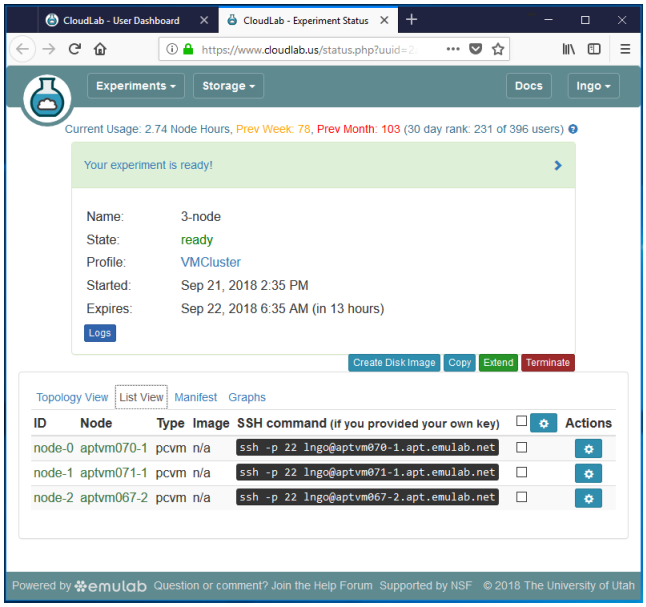


Figure 2: A 3-node cluster using XenVM on CloudLab for Assignment 2, Fall 2015

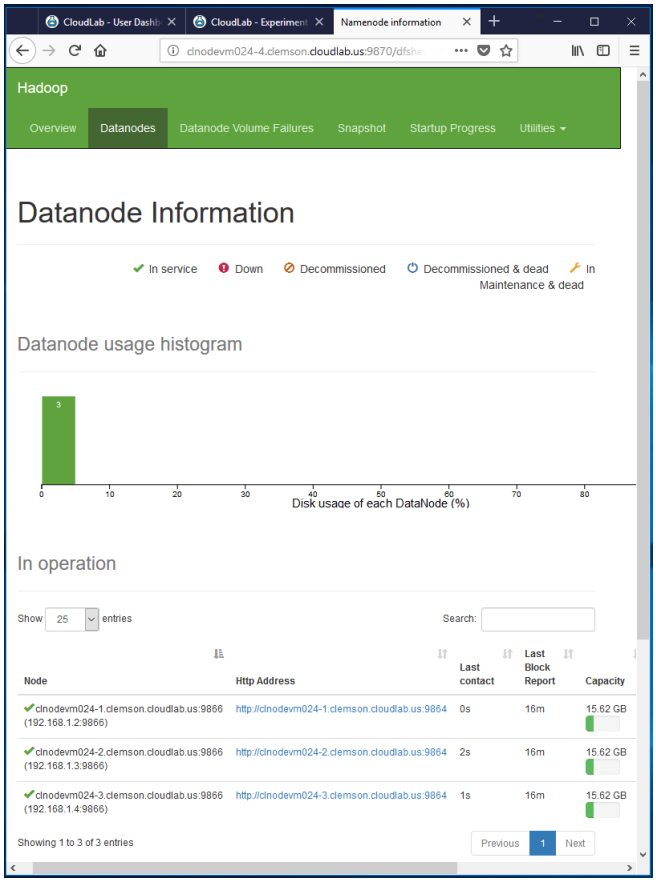


Figure 3: Public Web-UI of the Hadoop cluster deployed inside CloudLab

CUDA that can provide students with a more in-depth programming knowledge. While students can pick up some Linux skills by working with the supercomputer for programming assignments, these interactions are limited in user space and are barely on-par, if not less than, the amount of knowledge gained from a Linux administration course. To improve the course, we need to find a way to enable students to gain hands-on experience for the core system concepts regarding distributed infrastructures. Prior to CloudLab, several approaches had been explored including accessing virtual machines and creating a special dedicated cluster from a subset of the supercomputer. While they provided students with more access to the system aspects of distributed infrastructure, significant technical overhead remains [13].

3.2 First CloudLab Interaction: Fall 2015

CloudLab was first incorporated into the course's materials in Fall 2015, immediately after it became publicly available. For lecture materials, CloudLab was first used as a direct example of how a network of computers can be deployed such that it presents a transparent and unified interface to the users, yet is consisted of compute nodes placed at geographically distributed locations. This is to emphasize the development of distributed computing infrastructures, starting from on-site computer clusters, moving to grid computing, and then to cloud computing.

CloudLab was also incorporated in the second individual assignment whose purpose is to help students to become more familiar with working in distributed environments and moving between computing sites through a command-line terminal. The assignment was divided into two parts. In the first part, students learned how to log into the local supercomputer, write a job script, and submit this job to the scheduler for execution. The job requests three nodes from the supercomputer and then runs the *pbsdsh* command which would in turn execute the *hostname* command on each of these nodes and return the domain names. In the second part, students first deployed a 3-node cluster on CloudLab and acquired their IP addresses manually. Next, they modify their job script from part 1 so that each local compute node would contact a CloudLab node via SSH and request the CloudLab node's host name. The deployment and instruction to access these nodes are displayed on CloudLab's web interface as shown in Figure 2.

Later in the semester, CloudLab was utilized once again as part of a hands-on learning lecture about Hadoop Distributed File System [18]. After students learned about the principles behind Hadoop's architectural design and implementation, they spent one class to practice setting up and deploying a 3-node Hadoop cluster on CloudLab and also observing how data placement and data movement processes happened. This process is performed as a team exercise. Each member of the team will be responsible to log in and set up appropriate Hadoop components (NameNode or DataNode) on each CloudLab node. In Figure 3, the Web-UI of a Hadoop cluster is shown accessible via a normal browser. This is due to CloudLab's availability of public IP addresses that can be requested and attached to nodes in an experiment.

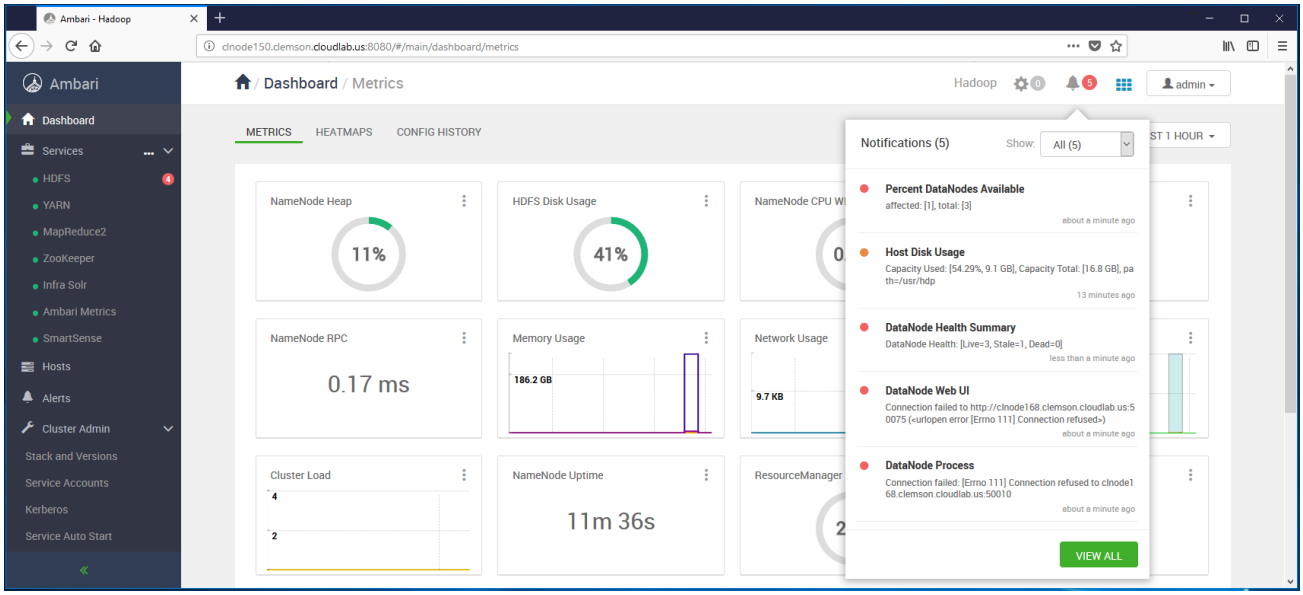


Figure 4: Public interface using Apache Ambari of the Hortonworks Hadoop Distribution cluster deployed inside CloudLab

3.3 CloudLab and On-site Computing Resource: Fall 2016

By Fall 2016, a new dedicated Hadoop cluster was deployed on site at Clemson University. To further expose students to the difference between the core open-source version of Hadoop and its industrial counterpart, another assignment has been added to the course. In this assignment, students were asked to once again deploy Hadoop on CloudLab. However, they were to use an open-source variety of Hadoop that was offered through Hortonworks [11], whose Hadoop distribution is used in the deployment Clemson University's Hadoop cluster. This distribution of Hortonworks is open source, and the complex deployment and management process is performed through Apache Ambari [21], another open source product from Hortonworks. Once the Hadoop cluster had been deployed, students were to deactivate one of the DataNodes and observe the behavior of the Hadoop Distributed File System in the event of failure. Through the process, the students would also have a chance to see how Apache Ambari helps providing a user-friendly interface to help managing all components of a Hadoop infrastructure, as shown in Figure 4.

3.4 CloudLab Bare-metal: Spring 2017

Starting in Spring 2017, efforts have been made to introduce project-based learning to the course, with the goals of tying all knowledge units taught in the course to a comprehensive picture. To this end, most of the previous semesters' assignments were taken a way and replaced with a semester-long, team-based project, in which students are required to develop a mini-cluster that has all the relevant components including a head node, compute nodes, a scheduler, and a parallel file system. Three individual assignments preceded the project to prepare students for the upcoming work in Linux environment. The CloudLab assignment using XenVM in Fall 2015 was among these assignments. Programming assignments were no longer assigned individually for grading purposes but offered first as in-class activities and later as validation mechanisms for various stages of the project.

The project was divided into five stages, built upon one another and scattered throughout the semester. In the first stage, students created the architecture design for their CloudLab-based research cluster. In the remainder of the stages, they implemented and deployed various cluster components, which correspond to the progression of the lectures [12]. The detailed descriptions for the stages are as follows:

- Stage 1 (2 weeks): Each team was tasked with first researching the various XSEDE sites to find a computing cluster site to be used as a motivation for their projects. To avoid conflict, each team must report on three different sites, ranked based on the team's preference. Subjects to be investigated including topological design, hardware configuration, selection of scheduler and file systems, and selection of open source scientific software.
- Stage 2 (2 weeks): Once a motivation site was selected, each team would first develop a similar (does not have to be exact) topological design and map this design to the corresponding CloudLab resources. An example of such design is shown in Figure 5. Based on this design, each team deployed their computing component, which is consisted of a network of three to four computers. OpenMPI was installed on each node and a demonstration using in-class programming materials was required to show that the nodes were properly connected. Unlike the XenVM-CloudLab assignment, the deployment process was completely done using CloudLab's scripting approach. The selection, configuration, and connection of the compute nodes as well as the installation of software on

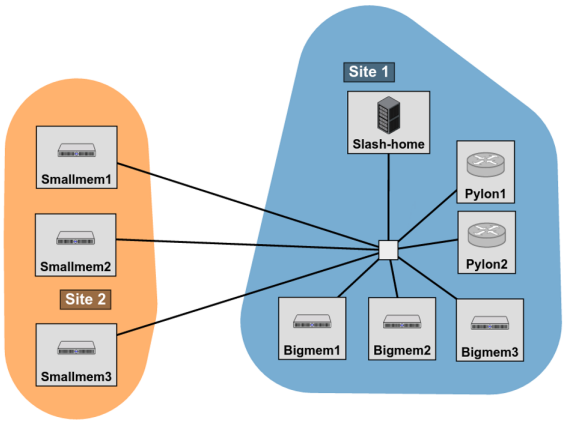


Figure 5: Example of a topological design based on the Bridges Supercomputer from Pittsburgh Supercomputing Center

each node were expressed using the Python programming language.

- Stage 3 (1.5 weeks): In this stage, the computing cluster was to be augmented with a network file system. A new node was to be added to the cluster, and the deployment code was also modified so that OpenMPI was reinstalled on this node and mounted across the compute nodes. In addition, this node also hosted a shared scratch file system. To validate this stage, a MPI heat-exchange programming example that write (in a non-parallel manner) to this scratch location was used.
- Stage 4 (3.5 weeks): Several additional nodes hosting a parallel file system were added to the cluster. This stage was when the teams began to pursue different technical paths due to the variety in parallel file system selections at the XSEDE sites. The options we had included LustreFS, CephFS, and OrangeFS (PVFS2). In-class MPI-IO demos from the lecture was used to validate this stage.
- Stage 5 (2 weeks): In the last stage, a scheduler was added to the node hosting the networked file system. Like the previous stage, different teams would work on different software, which could be one of PBS' varieties, Moab, or SLURM. For validation, each team had to provide a submission script that run one of the examples from the previous stages and can be submitted to their cluster's scheduler.

After the project was completed, the final cluster had an average of six to eight nodes, and each node was deployed directly on a bare-metal CloudLab node. Students were to submit a project report and their final CloudLab deployment script and any other installation and configuration scripts that were needed.

3.5 CloudLab's Openstack Profile: Spring 2018

The contents and target platform for the project was further modified for Spring 2018. In this semester, CloudLab's default experimentation cloud profile which uses Openstack [5] was leveraged. This allowed each team to launch a cloud environment using only three physical CloudLab nodes. Given the relatively large amount of resources available on physical CloudLab nodes, it was possible to launch smaller virtual machines inside this cloud environment, enabling the deployment of a full-scale cluster using fewer physical nodes. Furthermore, an update during Fall 2017 to CloudLab's user interface enabled direct linking between a Github repository containing the CloudLab resource description document and CloudLab's experimentation deployment interface. This significantly reduced the turn-around time for students' projects.

Instead of having each group model their cluster after a distinct XSEDE site, the Spring 2018 project used a single model, Clemson University's Palmetto Cluster, which used Oracle Linux 7 (OL7). This enabled a common required knowledge base, allowing various groups to collaborate and share their expertise. Another pre-project individual assignment was added, in which students first tasked with developing a baseline virtual image running OL7 inside VirtualBox on their personal computers. This assignment served as a preliminary training stage to further prepare students for the necessary Linux administrative tasks for the project, in addition to the XenVM/CloudLab assignment. From a architectural perspective, this did not significantly change how the computing cluster for the project was designed. If anything, restricting the model cluster to Clemson's supercomputer provided a uniform set of software selection, including OrangeFS for the parallel file system and PBS Pro for the scheduler.

On the other hand, the transition to using an Openstack profile on CloudLab to deploy the nodes significantly changed the technical approaches to this project. With Openstack, the entire project became more modularized, with the two processes, installing and configuring individual nodes and deploying them on CloudLab, are clearly separated. These changes are can be captured as follows:

- From the baseline (OL7) image from the initial assignment, students were to develop different images: a) a compute image for the computing nodes, b) a metadata server for OrangeFS, c) an I/O server for OrangeFS, and d) a login node that also server as the metadata server for the PBS Pro scheduler.
- Network topologies were determined prior to the installation and configuration of the images. Domain names and IP addresses were hard-coded into the images.
- Configured images were uploaded to file sharing servers (Dropbox, Box, or Google Drive). The default Openstack profile from CloudLab were modified to enable the downloading and instantiating of these images. For cluster components that had multiple nodes (computing or parallel file system), the same base image was reused for additional instantiation.

Two in-class technical sessions were hosted after the fourth and final stages, in which groups who passed specific technical hurdles shared their experience with others. The amount of computing nodes for the final cluster were six nodes, but most project submission required only two bare-metal CloudLab nodes to run. Figure 6 shows example of a successful project submission.

4 DISCUSSION

While CloudLab has been used in the course described in this paper since 2015, the extensive usage of the computing environment in

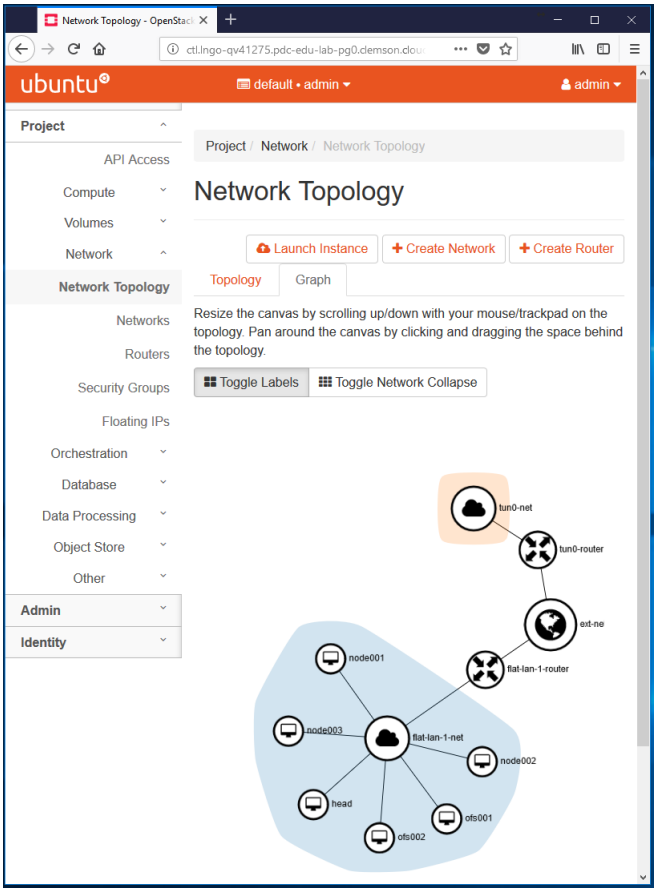


Figure 6: Example of a topological design deployed through CloudLab's default Openstack profile

a semester-long class project only started in Spring 2017. For this semester, there were 60 students in the class, divided into 15 teams. The group number decreased to 14 later during the semester due to students dropping from the course. In the next offering of the course and the class project, there were 33 students, divided into 11 teams. The nature of the project enables students to have extensive experience interacting with the Linux environment. This has led to overwhelmingly positive feedback from students, particularly regarding the hands-on and collaborative nature of the project, as well as the applicability of the course's content. However, there remains several challenges in using CloudLab in a project-based learning environment.

The largest issue that we noticed in using CloudLab, as with any remote and shared resources, was the potential competition with researchers and students from other institutions for computing hours. We experienced this issue in Spring 2017, since each team deployed directly on bare-metal CloudLab nodes. As a result, provisioning enough resources for the entire class became a challenge, and this hindered students' ability to complete some stages of the project in a timely fashion. The changes to the project made in Spring 2018 had been done partly to address this issue. By deploying virtual nodes inside fewer physical nodes, students could request adequate resources for their work. Furthermore, they could work offline to develop the baseline images and only used CloudLab for large-scale testing and validation. A minor issue is the amount of time it took to fully instantiate a complex CloudLab experiment. In both versions of the project (Spring 2017 and Spring 2018), the average time required to have the project ready ranged between 45 minutes to one and a half hours. This reduced the effectiveness of in-class sessions.

In our first version of the project in Spring 2017, we observed several pedagogical challenges to our learning approach. Firstly, a significant portion of students lacked the fundamental Linux skills to get started. A single XenVM CloudLab assignment was not enough to equip students with the necessary Linux administration skill to meaningfully contribute to the team. Secondly, since each group used a different XSEDE site as a model for their mini-cluster, this led to different software stacks and network topologies. As a result, in-class collaboration between the groups was not fruitful. The second version of the project in Spring 2018 had been modified to address these challenges.

5 RELATED WORK

There exists previous work describing the usage of local computing resources to teach distributed and cluster computing. In this section, we focus on related work that uses publicly available resources. The Extreme Science and Engineering Discovery Environment (XSEDE) is the largest and most popular resource for computationally-enabled sciences [20]. In addition to supporting scientific computing, XSEDE also makes resources available for computer science education. As XSEDE is a research production environment, the majority of its computing sites support programming topics in distributed and parallel computing rather than system-related topics [6, 10, 22]. Jetstream is another large-scale resource that is part of XSEDE. However, unlike the traditional HPC model of XSEDE, Jetstream offers users the ability to launch custom virtual machines images on bare-metal nodes for extended amount of time. Educational workshops and courses have leverage this capability of Jetstream to support topics in large-scale data-enabled sciences such as genomics, neuroimaging, and big data management [9]. CloudLab has also been used in a number of computer science education courses, but they primarily focus on areas such as computer and network security, networking, and cloud computing [1, 14].

6 CONCLUSIONS

In this paper, we have discussed our experience in using CloudLab to teach cluster computing topics. Between the two modes of deployment, bare-metal and cloud-based, we were able to provide students with a computing environment that enabled both hands-on and project-based learning. The flexibility, availability, and scale of CloudLab bring significant applicability to other topics in computer science, including operating system, networking, and cyber-security.

REFERENCES

[1] Jay Aikat, Michael K Reiter, and Kevin Jeffay. 2016. Education Modules for Networking, Cloud Computing, and Security in Systems Courses. In *Proceedings*

of the 47th ACM Technical Symposium on Computing Science Education. ACM, 400–400.

[2] Amy Apon, Rajkumar Buyya, Hai Jin, and Jens Mache. 2001. Cluster computing in the classroom: Topics, guidelines, and experiences. In *Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on*. IEEE, 476–483.

[3] Mark Berman, Jeffrey S Chase, Lawrence Landweber, Akihiro Nakao, Max Ott, Dipankar Raychaudhuri, Robert Ricci, and Ivan Seskar. 2014. GENI: A federated testbed for innovative network experiments. *Computer Networks* 61 (2014), 5–23.

[4] William I Bullers Jr, Stephen Burd, and Alessandro F Seazzu. 2006. Virtual machines-an idea whose time has returned: application to network, security, and database courses. In *ACM SIGCSE Bulletin*, Vol. 38. ACM, 102–106.

[5] Antonio Corradi, Mario Fanelli, and Luca Foschini. 2014. VM consolidation: A real case based on OpenStack Cloud. *Future Generation Computer Systems* 32 (2014), 118–127.

[6] Andrew Danner and Tia Newhall. 2013. Integrating parallel and distributed computing topics into an undergraduate CS curriculum. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International*. IEEE, 1237–1243.

[7] Joshua Eckroth. 2016. Teaching big data with a virtual cluster. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, 175–180.

[8] Jesse Eickholt and Sharad Shrestha. 2017. Teaching big data and cloud computing with a physical cluster. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, 177–181.

[9] Jeremy Fischer, David Y Hancock, John Michael Lowe, George Turner, Winona Snapp-Childs, and Craig A Stewart. 2017. Jetstream: A cloud system enabling learning in higher education communities. In *Proceedings of the 2017 ACM Annual Conference on SIGUCCS*. ACM, 67–72.

[10] Steven I Gordon. 2013. Advancing computational science education through xsede. *Computing in science & Engineering* 15, 1 (2013), 90–92.

[11] Hortonworks. 2018. Hortonworks Data Platform. https://hortonworks.com/products/data-platforms/hdp/.

[12] Linh Bao Ngo. 2018. CPSC 4770-6770: Distributed and Cluster Computing. https://github.com/linhbngo/cpsc-4770_6770.

[13] Linh Bao Ngo, Edward B Duffy, and Amy W Apon. 2014. Teaching HDFS/MapReduce systems concepts to undergraduates. In *Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International*. IEEE, 1114–1121.

[14] Younghee Park, Hongxin Hu, Xiaohong Yuan, and Hongda Li. 2018. Enhancing Security Education Through Designing SDN Security Labs in CloudLab. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, 185–190.

[15] Sushil K. Prasad, Almadena Chtchelkanova, Frank Dehne, Mohamed Gouda, Anshul Gupta, Joseph Jaja, Krishna Kant, Anita La Salle, Richard LeBlanc, Andrew Lumsnaide, David Padua, Manish Parashar, Viktor Prasanna, Yves Robert, Arnold Rosenberg, Sartaj Sahni, Behrooz Shirazi, Alan Sussman, Chip Weems, and Jie Wu. 2017. NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing - Core Topics for Undergraduates. http://www.cs.gsu.edu/ tcpp/curriculum/index.php.

[16] Robert Ricci, Eric Eide, and CloudLab Team. 2014. Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications. *; login:: the magazine of USENIX & SAGE* 39, 6 (2014), 36–38.

[17] Robert Ricci, Kuang-Ching Wang, Srinivasa Akella, Michael Zink, and Glenn Ricart. 2017. CloudLab Phase II: Community Infrastructure To Expand the Frontiers of Cloud Computing Research. https://www.nsf.gov/awardsearch/showAward?AWD_ID=1743363.

[18] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. 2010. The hadoop distributed file system. In *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on*. Ieee, 1–10.

[19] Mark Stockman. 2003. Creating remotely accessible virtual networks on a single PC to teach computer networking and operating systems. In *Proceedings of the 4th conference on Information technology curriculum*. ACM, 67–71.

[20] John Towns, Timothy Cockerill, Maytal Dahan, Ian Foster, Kelly Gaither, Andrew Grimshaw, Victor Hazlewood, Scott Lathrop, Dave Lifka, Gregory D Peterson, et al. 2014. XSEDE: accelerating scientific discovery. *Computing in Science & Engineering* 16, 5 (2014), 62–74.

[21] Sameer Wadkar and Madhu Siddalingaiah. 2014. Apache ambari. In *Pro Apache Hadoop*. Springer, 399–401.

[22] XSEDE. 2018. Course Catalog. https://portal.xsede.org/training/course-catalog.

Programmable Education Infrastructure: Cloud resources as HPC Education Environments

Eric Coulter
Indiana University
Bloomington, Indiana
jecoulte@iu.edu

Richard Knepper
Indiana University
Bloomington, Indiana
rknepper@iu.edu

Jeremy Fischer
Indiana University
Bloomington, Indiana
jeremy@iu.edu

KEYWORDS

Clusters, Cloud, Jetstream, Openstack, OpenHPC, XSEDE, XCRI

ABSTRACT

Cloud computing is growing area for educating students and performing meaningful scientific research. The challenge for many educators and researchers is knowing how to use some of the unique aspects of computing in the cloud. One key feature is true elastic computing - resources on demand. The elasticity and programmability of cloud resources make them an excellent tool for educators who require access to a wide range of computing environments. In the field of HPC education, such environments are an absolute necessity, and getting access to them can create a large burden on the educators above and beyond designing content.

While cloud resources won't replace traditional HPC environments for large research projects, they are an excellent option for providing both user and administrator education on HPC environments. The highly configurable nature of cloud environments allows educators to tailor the educational resource to the needs of their attendees, and provide a wide range of hands-on experiences. In this demo, we'll show how the Jetstream cloud environment can be used to provide training for both new HPC administrators and users, by showing a ground-up build of a simple HPC system. While this approach uses the Jetstream cloud, it is generalizable across any cloud provider. We will show how this allows an educator to tackle everything from basic command-line concepts and scheduler use to advanced cluster-management concepts such as elasticity and management of scientific software.

https://doi.org/10.22369/issn.2153-4136/10/1/9

The HPC Best Practices Webinar Series

Osni A. Marques*
Lawrence Berkeley National Laboratory
oamarques@lbl.gov

David E. Bernholdt†
Oak Ridge National Laboratory
bernholdtde@ornl.gov

Elaine M. Raybourn‡
Sandia National Laboratories
emraybo@sandia.gov

Ashley D. Barker†
Oak Ridge National Laboratory
ashley@ornl.gov

Rebecca J. Hartman–Baker*
Lawrence Berkeley National Laboratory
rjhartmanbaker@lbl.gov

ABSTRACT

In this contribution, we discuss our experiences organizing the Best Practices for HPC Software Developers (HPC-BP) webinar series, an effort for the dissemination of software development methodologies, tools and experiences to improve developer productivity and software sustainability. HPC-BP is an outreach component of the IDEAS Productivity Project [4] and has been designed to support the IDEAS mission to work with scientific software development teams to enhance their productivity and the sustainability of their codes. The series, which was launched in 2016, has just presented its 22nd webinar. We summarize and distill our experiences with these webinars, including what we consider to be "best practices" in the execution of both individual webinars and a long-running series like HPC-BP. We also discuss future opportunities and challenges in continuing the series.

1 INTRODUCTION

The Best Practices for HPC Software Developers (HPC-BP) webinar series is a major component of the outreach efforts of the IDEAS Productivity Project[1]. Since its inception, the project has been addressing the confluence of trends in hardware and increasing demands for predictive multiscale, multiphysics simulations. The project has also been responding to trends for continuous refactoring with efficient agile software engineering methodologies and improved software design. The webinar series began when the IDEAS Project was particularly focused on working with the terrestrial ecosystem modeling community within the U.S. Department of Energy (DOE), and the series has been adapted as the focus (and sponsorship) of the IDEAS Project shifted to DOE's Exascale Computing Project (ECP) [5], wherein the focus is on helping teams in both scientific applications and software technologies to be more productive in their software development efforts and to produce more sustainable results.

To achieve its goals, IDEAS-ECP (as we refer to this phase of the IDEAS Productivity Project) has implemented an agenda for training and outreach, including HPC-BP. The target audience for the series largely overlaps with the community of users of the computer facilities under DOE's Office of Science: the Argonne Leadership Class Facility (ALCF) [2], the Oak Ridge Leadership Class Facility (OLCF) [3], and the National Energy Research Scientific Computing Center (NERSC) [1]. However, the webinars can be (and typically are) attended by a much broader community: announcements are done through various email lists, participation is free, and only a simple registration is required for each event. The series may have similarities with other efforts (e.g., [6, 7]), but it is also distinct in many ways. For example, when the webinar focuses on tools, it is often necessary to address differences in the corresponding installations at the computing facilities.

In this contribution, we discuss the process, i.e., the *HowTo*, that we have adopted for HPC-BP, and we describe some webinars that we have organized and delivered in the series. We provide an overview of the process we follow for the selection of topics, how the webinars are executed, and the future we foresee for the series.

2 SELECTION OF TOPICS

The webinars in the HPC-BP series occur approximately once a month and last about one hour each. Earlier in the series, the topics were set well in advance. Currently, we maintain a dynamic pool of potential topics (and speakers), from which we pull the ones that we consider to be timely. This decision is based on interactions with ECP application teams; interactions with staff at ALCF, OLCF and NERSC; perceived trends in hardware evolution; and discussions

*This manuscript has been authored by an author at Lawrence Berkeley National Laboratory under Contract No. DE-AC02-05CH11231 with the U.S. Department of Energy. The U.S. Government retains, and the publisher, by accepting the article for publication, acknowledges, that the U.S. Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for U.S. Government purposes.

†This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

‡Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

[1]IDEAS stands for *Interoperable Design of Extreme-scale Application Software*.

https://doi.org/10.22369/issn.2153-4136/10/1/19

among members of IDEAS-ECP. In addition, on several occasions the series has featured presentations by "volunteers," i.e., members of the HPC community at large who learned about HPC-BP and contacted us about the suitability of their topics for the series.

At the time of this writing, we have delivered 22 webinars in the HPC-BP series. Seven webinars in 2016 were in conjunction with an earlier phase of the IDEAS project. We relaunched the webinar series in June 2017 as part of the IDEAS-ECP project, and have so far delivered 15 webinars on an approximately monthly cadence. The following is a sample of topics that have been covered: intermediate Git (presenting Git in terms of a data structure and a set of algorithms to manipulate that data structure); the Roofline Model (introducing the concepts of data locality and arithmetic intensity, and the impact of their interplay in computational performance); the management of defects in HPC software development (discussing the relevance of software verification as a method for removing defects at and earlier development phase, and its impact on productivity); scientific software development with Eclipse (focusing on the latest features that are mostly useful for scientific applications); on-demand learning for better software development (demonstrating how a personalized transmedia learning framework based on on-line resources can be applied towards learning more productively); the importance of proper citation mechanisms for software (so software developers and maintainers can get academic credit for their work); and software sustainability seen from different angles (cultural issues, software sharing, adoption of best practices, etc.)

Usually, we select a topic at least two months in advance, marking the beginning of our "formal" interactions with the presenter(s). Speakers are asked to prepare a 45-minute presentation and include breaks to answer potential questions from the participants. About six weeks before the webinar we ask the presenter for a title, abstract and a short bio. About a month before the webinar we prepare a registration page and start an announcement campaign. We also schedule a dry-run, which is typically conducted the week prior to the webinar. The dry-run is usually attended by a few members of the IDEAS-ECP project, and has proven to be essential for fine-tuning the presentation. The day before the webinar we send a link to the event to the registrants, together with a copy of the slides to be presented.

3 MISE EN SCÈNE

Based on our experience, we have developed a model for a number of distinct roles and steps in the process of developing and delivering a specific webinar event. Figure 1 provides an illustration. The letters to the right of the figure indicate the steps, the corresponding "actors" and their (leading or supporting) roles:

(a) Actor: meeting manager. Roles: displays introductory slide (typically prepared by the moderator), which contains information about how participants can ask questions and provide feedback; monitors connections and checks whether participants are having problems with video/audio; starts recording.
(b) Actor: moderator. Role: introduces speaker(s) and other people that will help answer participants' questions.

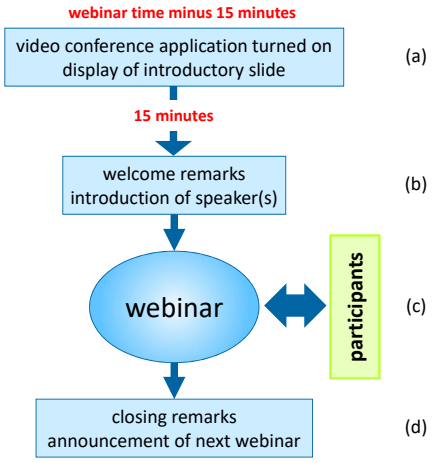


Figure 1: Steps and actors in HPC-BP webinar series.

(c) Actor: speaker. The meeting manager and the moderator check questions submitted through chat and Google doc and relay them to the speaker at specific breaks.
(d) Actors: moderator and meeting manager. Roles: ask audience whether there are additional questions; display slide announcing the next webinar; end the recording and video service.

The number of registrants for the webinars has ranged from 69 to 207, and our experience is that roughly 50% of those who register actually attend the event. Table 1 gives the total number of registrations, registrations affiliated with ECP, and number of participants in the webinars of the HPC-BP series. We observe that the number of registrations varies significantly across the 15 webinars. We believe this can be attributed to a variety of sources. Although we haven't attempted to analyze the variation, we believe that the webinar topic is likely to be the most significant driver for registrations. Speakers who are well-known in the community may also attract more registrations. The detailed data also suggests the possibility of seasonal variation, but we do not yet have sufficient experience to consider this conclusive. There are also variations in how far in advance we are able to get our announcements out, which may impact registration rates. In terms of timing, we have a specific time slot (1:00–2:00pm U.S. Eastern Time, on a Wednesday) that we have successfully held to for the vast majority of our webinars. However the week in the month will vary, so the schedule is not completely regular.

We note that the level of interest in the series, as measured by the number of people registering, and the number of participants, often significantly exceeds the typical experience that the ASCR facilities have with HPC-oriented training. We also note that participation of the audience has motivated us to include topics in our pool, or to consider offering webinar topics in different formats such as workshops or tutorials (e.g., for Eclipse and CMake).

Questions from the participants are highly encouraged, and are accepted through the chat capability in the webinar tool and also a shared Google Doc. While participants are muted by default, in

some cases we allowed participants to directly ask specific questions (by unmuting) to the presenter(s) at the end of the webinar.

Recordings of the webinars, along with the corresponding slides, are posted about a week after the webinar, together with a revised and curated Q&A (i.e., answers to the questions from the participants). An announcement of the "archival" copy of the webinar is distributed to those who registered for and/or attended the webinar, in case they wish to refer back to it or share with colleagues. All the relevant information is available under the "Events" tab in the IDEAS web site[4].

Table 1: Registrations and attendees in the 15 webinars of the HPC-BP Series (June 2017 - September 2018).

	Total Registrations	ECP Affiliated Registrations	Attendees
Minimum	69	22	22
Average	134	42	75
Maximum	210	71	127
Std. Deviation	50	15	35

4 OVERALL CONSIDERATIONS AND NEXT STEPS

One of the unique features of HPC-BP is that its agenda is implemented in concert with three major computer facilities (i.e. ALCF, OLCF and NERSC). This feature also poses challenges: for example, one of our webinars focused on Python in HPC. Maximizing performance from Python applications can be demanding on supercomputing architectures, even more so for different installations. The approach we used for the webinar was to have it presented by three staff, one from each facility. This exercise also presented an opportunity for an exchange of knowledge among the facilities.

One challenge is to capture meaningful data for success metrics. Although each workshop is accompanied by a survey, the number of respondents is typically low. Tracking the number of registrants and participants seems more appropriate and straightforward. (Sometimes, one registration corresponds to multiple people viewing the event from a single location.) We take the number of questions in the Q&A into consideration for potential follow-ups on the topic, e.g., further training through tutorials.

As mentioned before, IDEAS-ECP is particularly (though not exclusively) focused on topics relating to software productivity and sustainability, and also works in collaboration with the Training component of the ECP project. The training team is in the process of compiling a list of tools, programming languages, programming models and runtimes, etc., which will be used to construct a survey for ECP developers of applications and software technology to gauge their needs for training, plus the optimal format(s) (webinar, tutorial, etc.) and level(s) (novice, etc.). We anticipate that this survey will enable us to identify opportunities to further enrich the pool of topics related to software productivity and sustainability and further expand our outreach activities.

Looking into the future, once the early exascale computers are in production and the ECP has completed its mission, the IDEAS Productivity Project will need to find new ways to support and sustain the HPC-BP webinar series.

ACKNOWLEDGMENTS

This research was supported by the Exascale Computing Project (ECP 17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

This work was carried out in part at Lawrence Berkeley National Laboratory, managed by the University of California for the U.S. Department of Energy under contract number DE-AC02-05CH11231.

This work was carried out in part at Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U.S. Department of Energy under contract number DE-AC05-00OR22725.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

REFERENCES

[1] National Energy Research Scientific Computing Center. 2018. http://www.nersc.gov. (2018).
[2] Argonne Leadership Computing Facility. 2018. https://www.alcf.anl.gov. (2018).
[3] Oak Ridge Leadership Computing Facility. 2018. https://www.olcf.ornl.gov. (2018).
[4] IDEAS. 2018. https://ideas-productivity.org. (2018).
[5] The Exascale Computing Project. 2018. https://www.exascaleproject.org. (2018).
[6] Blue Waters Webinars Series. 2018. https://bluewaters.ncsa.illinois.edu/webinars. (2018).
[7] ACM Learning Webinars. 2018. https://learning.acm.org/webinars. (2018).

January 2019

Volume 10 Issue 1

ISSN 2153-4136 (online)