OnRamp to Parallel and Distributed Computing

Samantha S. Foley
University of Wisconsin-La Crosse
La Crosse, WI
sfoley@uwlax.edu

Joshua Hursey
University of Wisconsin-La Crosse
La Crosse, WI
jhursey@uwlax.edu

ABSTRACT

Computer Science students must understand parallel and distributed computing (PDC) concepts to be effective computer scientists in the workforce, as reflected in the ACM Curriculum guidelines [1]. Communities of CS educators are creating educational modules, and making parallel compute environments (PCEs) available to educators who are integrating PDC concepts into their existing curriculum. Even with these resources there is a barrier to entry for students to use PCEs, namely the unfamiliar and complex system software ecosystem of modern PCEs. The OnRamp project lowers that barrier to entry for exploring PDC concepts on a variety of PCEs while also providing a path for students to learn how to be productive on the native PCE. OnRamp is designed to be a general purpose web portal for supporting the exploration of PDC concepts that harnesses the existing educational resources created by the CS education community. It coaches students through interactive modules that teach them about PDC concepts and PCEs while allowing them to launch parallel applications from day one. As students become more comfortable with running parallel applications on PCEs, OnRamp transforms into a reference guide as they graduate to using the native PCE.

1. INTRODUCTION

Computer Science (CS) students must understand parallel and distributed computing (PDC) concepts to be effective computer scientists in the workforce, as reflected in the ACM Computer Science Curriculum guidelines [1]. The CS education community has been creating and distributing educational modules to support educators that wish to incorporate parallel computing concepts into their curriculum. Concurrently, the prevalence of multicore processors in student machines and "Budget Beowulfs" [2] available as classroom resources has nearly eliminated the barrier to access parallel compute environments (PCEs). However, there still exists a barrier to entry for learning how to become productive in most PCEs.

EduHPC2015 November 15-20, 2015, Austin, TX, USA

 ISBN 978-1-4503-3961-2/15/11…$15.00

DOI: http://dx.doi.org/10.1145/2831425.2831426

Students encounter this barrier to entry when they learn how to become productive in their first PCE where they are faced with often unfamiliar and complex system software, programming interfaces, and tools. For example, students must navigate the system via a terminal, handle complex compilation and execution procedures, negotiate resource allocation via a batch submission system, understand proper etiquette for using the shared resource, and utilize parallel programming libraries. The process of overcoming these hurdles distracts from learning parallel computing concepts, and can often intimidate students (and educators), discouraging them from using the PCE. It is important for students exploring curriculum modules to access PCEs that are architecturally different and more capable than their personal machines to understand the relationship between system architecture and parallel computation. The question becomes: how can we lower the barrier to entry for exploring PDC concepts on a variety of PCEs while still providing a path for students to learn how to be productive on the PCE directly (without OnRamp)?

The OnRamp project provides a web portal which coaches users through a collection of interactive curriculum modules teaching them about parallel computing and the software ecosystem while allowing them to launch parallel applications on PCEs from day one. To launch their first parallel application students login to the web interface, choose the PCE and Module (which has been setup by their instructor), set some high-level runtime parameters (e.g., number of nodes/cores/tasks), then launch the job and view their results. They are initially abstracted away from the PCE software ecosystem and can start hands-on exploration of the PDC concepts in that Module in a matter of minutes. This is in contrast to the prevalent model of spending the first day or two of class getting students comfortable with the PCE before hands-on exploration of the PDC concepts.

OnRamp is designed to be a general purpose web portal for supporting the exploration of PDC concepts and PCEs. The portal is not specific to any one programming language, parallel computing library, application, or PCE. This critical design point allows for maximal flexibility in how and where educators can use it with students.

This paper outlines the design of the OnRamp project and how it might be used by CS educators to integrate hands-on PDC modules into the existing curriculum. Section 2 highlights some of the efforts that have been influential to the OnRamp project. Section 3 more completely describes the barrier to entry that motivated the creation of the OnRamp project. Section 4 describes the design of the OnRamp por-

tal. Section 5 describes some use case scenarios where educators might use OnRamp to introduce PDC into existing courses. Finally, Section 6 concludes by highlighting some of the near and long term goals of the project.

2. BACKGROUND

Academic institutions are exploring how to best integrate parallel computing concepts into the CS curriculum as parallel hardware becomes more and more pervasive. The latest NSF/TCPP and ACM Curriculum guidelines [1, 3] contain learning objectives focused on parallelism for many core CS courses. To address this relatively new emphasis on parallel computing throughout the curriculum, there are several efforts targeted at assisting educators in this curriculum integration task.

The spiral approach to exploring parallelism has been a successful approach for educators [4, 5]. This approach introduces hands-on experimental learning activities at several points in the curriculum with increasing depth and complexity. The hands-on exploration component is important as students "feel" the impact of parallelism when applied to programming exercises and exemplar applications.

Another proven approach to teaching parallel computing is to emphasize the parallel programming concepts and patterns, and to present exemplar applications throughout the curriculum in contrast to focusing on specific hardware and software technologies [6]. This approach promotes a general application of "parallel thinking" by the students, so that they are better able to apply these parallel patterns to the ever evolving hardware, software, and programming landscape of PDC.

A growing community of CS educators have been creating and distributing curriculum modules that demonstrate effective techniques and concrete lesson plans for teaching parallel patterns and programming paradigms in a variety of courses [7, 8]. Instructors eager to use these curriculum modules often still need to obtain a PCE for the students to do the hands-on work, and teach them how to use the system software, before they can use the modules.

Historically, gaining access to a PCE or building your own PCE was an expensive and labor intensive task. The prevalence of multicore processors has placed (limited) parallel computers inside the machines of most students. Students need access to more capable, often remote, PCEs to explore parallelism beyond a few concurrent cores in a shared memory system and to explore the relationship between various architectures and parallelism. Recent hardware and software projects targeted at the educational community have made it easier to obtain relatively small physical, virtual, or cloud based PCE at low or no cost [2, 9, 10]. Additionally, parallel computing facilities at the regional or national level (e.g., XSEDE [11], CDER[1], OSCER [12], FutureSystems [13]) can be made available for educational activities. Access to PCEs is no longer the problem that it once was for the broader educational community.

Even though access to PCEs is no longer a significant challenge, using those environments represents a significant problem for educators. If an educator wants students to use a PCE then they must spend time teaching them how to be productive in that specific environment before students can explore parallel computing concepts. This process may take a few hours or days depending upon the level and background of the student – representing a barrier to entry for students and educators.

This barrier to entry to using PCEs directly has spurred the creation of web portals to provide access to PCEs without having to deal with things like the command-line interface, remote login, batch schedulers, and complex build processes.

The WebMapReduce [14] project provides a simplified interface to the Hadoop map-reduce parallel computing model through a web portal. The web portal allows students to write mappers and reducers in a variety of languages, and has been used in both introductory and advanced CS courses. The Wings portal [15] targets big data analytics for non-programmers. It provides a visual workflow composition environment that can take advantage of parallel and distributed computing resources in the backend while highlighting parallel computing concepts. These tools are great for getting students to play with a specific parallel computing paradigm or concept without the student being bogged down in the details of how to use a specific PCE.

Computational scientists have relied on application specific web portals for decades to abstract them away from the complexities of PCEs so they can focus on the science [16]. XSEDE continues this model and provides a long list of application specific web portals for interested users.[2] The DSABR and DP3 web portals [17] enable students from art and archaeology to submit work (animation rendering and image processing, respectively) to a web portal which uses a PCE in the background to perform the designated task.

The web portal described by Lin [18] allows users to submit C++ or Java code to a cluster then view the results. Students need to know exactly how to build and run the code on the system when filling out the web forms, making the interface slightly too complex for introductory students. The Center for Parallel and Distributed Computing Curriculum Development and Educational Resources (CDER) system provides a web portal that allows users to startup interactive sessions (including a full desktop) and launch batch jobs. Similarly, however, students will need to know how to use the system before they can start interacting with parallel applications. The web portal described by Burkhart [19] provides a web service that allows students to submit code (packaged in a workspace) to one or more PCEs. The focus of this web portal is to allow instructors to trust the results submitted by students by controlling the parameterization and submission of jobs to PCEs.

OnRamp is designed to be a general purpose web portal for supporting the exploration of parallel computing concepts and systems at a variety of levels in the curriculum. It is not specific to any one programming language, library, application, or PCE. The phrase *native PCE* refers to the traditional mechanisms for using the PCE without the aid of the OnRamp portal or a gateway service. The portal hides the complexity of interacting with the native PCEs allowing students to focus on parallel computing concepts by running applications contained in curriculum modules, which can be derived from those being developed by the CS educational community. OnRamp can then be used to gradually bring students closer to the native PCE by teaching them the skills needed to "graduate" to using that PCE on their own.

[1] http://cder.gsu.edu

[2] https://www.xsede.org/gateways-listing

3. MOTIVATION FOR ONRAMP

The availability of curriculum modules, PCEs, and domain specific web portals allow educators to start integrating PDC into their existing curriculum. However, there still exists a significant barrier to entry for students and educators desiring to use the curriculum modules on a PCE where a specific web portal is not available.

Consider an instructor that wants to drop-in an existing PDC curriculum module into a mid-level CS class using a modest cluster housed in the department. This instructor can only expect the students to have programming experience with Java and Eclipse.[3] While this is a specific example, it does highlight many of the challenges that are faced by educators in other situations.

Hurdle 1: Creating accounts on a PCE: The instructor must create, possibly temporary, accounts on the PCE for each student so they can edit, build, and run code. This may involve the instructor interacting with the system administrators for the PCE. Depending on bureaucracy and the commitments of each individual involved in the process, this may take some time.

Hurdle 2: Introducing the UNIX shell: The vast majority of the students in the class have never interacted with a computer via the command-line. Interacting with a computer solely through a terminal is a new and often intimidating prospect for these students. The instructor needs to help students become comfortable with basic navigation and file management in the shell as well as instructions on avoiding common pitfalls (e.g., there is no recovering a deleted file).

Hurdle 3: Remotely accessing the PCE: Most PCEs are accessed remotely so students need to become comfortable with remote shell access (via SSH) and secure file transfer (via SCP) between the PCE and their machines. Student may need to install additional software on their systems (e.g., Putty, WinSCP, ssh) before they can access the machines. Alternatively, the instructor can have these tools preinstalled in a computer lab accessible by the students.

Hurdle 4: Editing code remotely: Students are now logged into the remote PCE and ready to edit some code in a module that they have copied to the machine. A student might edit the file(s) on their home machine and transfer the files, but that often leads to inconsistencies. Instead the instructor will want to spend time teaching the students how to use a text-based editor on the PCE (e.g., nano, emacs, vim) as that will serve the students better in the long term.

Hurdle 5: Building software: Now the students have all managed to edit their code, and want to compile and run it. The instructor will have to explain how to invoke the compiler with the necessary libraries specified – a process often encapsulated in a Makefile. An introduction to how a prepared Makefile can and should be used along with common compiler errors is necessary.

Hurdle 6: Running the code: Now that the code is built, the students want to run the code. A short discussion about the difference between head nodes and compute nodes, and how to use a batch scheduler is needed before students are able to run their code without negatively impacting others on the system. The instructor may need to provide a batch submission script or teach students how to create one at this time. Then a discussion of the parallel launch mechanism (e.g., mpirun) and parallel runtime parameters (e.g., number of processes, number of threads), along with a discussion of the parallel architecture should occur before the first execution of the code. Finally, students are running code and you can focus the discussion on how this PCE and the application code relate to the parallel concepts discussed in class.

This example highlights the hurdles that comprise the barrier to entry that still exists even with the expanding set of available curriculum modules and PCEs – noting that some PCEs have different software that students must learn when they switch machines. Reducing these hurdles by introducing some of these topics earlier, maybe in a prerequisite course, would ease the transition to a PCE. However, many educators are trying to fit these concepts into an already overfull curriculum and may only be able to spare a day or two dedicated to technological details when they, instead, wish to focus on hands-on exploration of PDC concepts.

OnRamp addresses this barrier to entry by providing a web portal that hides the complexity of these issues so that students can focus on using the module and learning the concepts. Curriculum modules define which parameters a student should specify (e.g., problem size) along with the runtime options (e.g., number of nodes). The OnRamp web portal allows students to view documentation, launch jobs, view results, and rerun modules. The web portal allows instructors to easily create temporary OnRamp accounts for students, and deploy modules. OnRamp drastically reduces the amount of work an educator needs to do to get students playing with parallel code and concepts. See Section 5 for use cases on how OnRamp might be used in the classroom.

While it is desirable to make this process less painful for instructors and students, the reality is that students *do* need to know all of these details in order to use PCEs effectively. OnRamp is designed to help students become more and more proficient with a PCE through a series of knowledge levels. Each level is designed to support students through learning a set of skills and concepts needed to use a PCE natively. The number of levels and their composition is determined by the educator using the system since every educational environment and situation has their own needs and priorities. To orient the development of OnRamp, we have outlined four knowledge levels into which we have grouped related functionality needed to support learning at that level. These four levels represent one possible logical progression of learning necessary to write parallel software and use a PCE.

The "*Level One*" functionality supports the learning of parallel concepts such as: domain decomposition, Amdahl's law, strong and weak scaling, threads and processes, and parallel architectures. The functionality to support this knowledge level includes the ability to remotely launch a parallel program that has been provided by someone else (the Module writer); to specify parameters for the number of processes, threads, nodes, and application specific values

[3]While this is one possible situation, and is currently true at our university, students at this level may already be familiar with C programming and UNIX environments. This background would reduce some of the barriers. However, hurdles 1 and 6 would still be significant while hurdles 3, 4, and 5 may vary in their significance depending upon the situation.

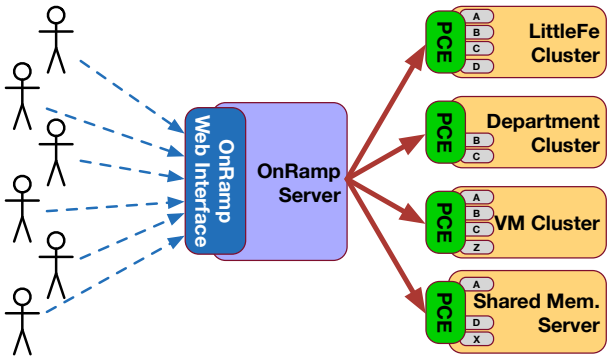

Figure 1: OnRamp architecture diagram showing six users sharing access to four native PCEs.

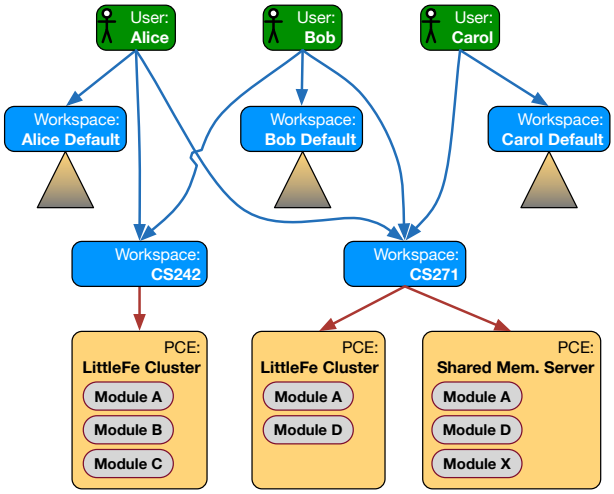

Figure 2: Logical organization of Workspaces, PCEs, Modules and users. Shows one physical LittleFe cluster in two Workspaces.

(e.g., problem size); and to view the results of the job.

The "*Level Two*" functionality supports learning about parallel programming paradigms. Functionality to support this level includes editing of Module defined sections of code, data sets, and scripts. The "*Level Three*" functionality supports learning how to be productive on a PCE in an interactive tutorial manner. Functionality to support this level includes PCE specific tutorials, limited interactive shell tutorial in the browser, exposure to compile and run procedures, and process layout options. In "*Level Four*," OnRamp transforms into a resource while the user migrates to using the machine directly and with confidence. Functionality to support this level includes PCE usage documentation, Module writing and testing documentation, and options to copy OnRamp user directories to the user's own UNIX account including full run history.

4. DESIGN

OnRamp has four major components shown in Figure 1: the OnRamp Web Interface which provides users with a familiar, web-based view of the computing environment; the OnRamp Server acts as a broker between the OnRamp web users and the various connected PCEs; the OnRamp PCE Service which launches jobs and manages activities on a specific PCE on behalf of the OnRamp users and admins; and the OnRamp Module Infrastructure for managing curriculum content in a uniform and portable manner. This section describes how the portal supports the Level One capabilities.

4.1 OnRamp Web Interface

The OnRamp Web Interface is what users see when they interact with OnRamp. There are several logical constructs that are used by the web interface to organize how data is presented to the users and how users navigate resources. First, there are two types of OnRamp user accounts: *users*, and *admins*. *Users* are able to view the Workspaces to which they have been granted access, and the Modules and PCEs associated with this Workspace (Fig. 2). A user may view jobs that they have launched and results for jobs that have completed. *Admins* are able to do everything that a general user can do, as well as manage PCE connections, Modules on PCEs, Workspaces, and OnRamp accounts. The admin role privileges are granted on a per OnRamp Server basis, which means that any admin can manage any thing on that OnRamp Server.

OnRamp has a notion of a *Workspace* (Fig. 2) to organize user activities on the web interface and the corresponding actual activities on the PCEs, as well as provide some structure to help instructors create cohesive lesson plans. Workspaces allow instructors to organize course material into one or more logical groups of related PCEs and Modules, with a different emphasis for each Workspace. All users start with a personal Workspace containing a default set of PCEs and Modules. Instructors (as admin users) can create a Workspace with a set of PCEs and Modules, then add students to the Workspace after testing it. Permissions are managed through Workspaces instead of separately for each PCE, Module and user separately, making it easier to manage larger classes.

The notion of a *PCE* is represented on the web interface as a series of properties of the PCE (name, location, nodes, up/down, etc.) and documentation about the PCE. One of the goals of OnRamp is to not only serve as an educational tool for learning about PDC but to also serve as a reference for those who are using the PCE directly. The information presented on the OnRamp Web Interface is determined by the admins who set up the OnRamp PCE Service.

The notion of a *Module* is represented in the OnRamp Web Interface as a block of descriptive text and a dynamically created form for launching the code associated with the Module. The descriptive text and form are created by the Module writer and can be as complex as the Module writer desires. OnRamp might extend the set of parameters needed to launch the job with PCE-specific parameters since Modules are often written without knowledge of the PCE on which it will be executed. This is all detected and managed by the OnRamp PCE Service and then dynamically rendered by the web interface.

Users of OnRamp start by logging into the web portal with an OnRamp specific username and password to authenticate with the OnRamp Server. This account is separate from

any account the user may have on a target PCE. Once authenticated, they can view their jobs, select a Workspace to start some new jobs, and manage their profile. Admin users can view and manage Workspaces, PCEs, Modules, users and jobs, as well as start new jobs and manage their profile like other users. The current system allows for basic views of the text data. Future development will extend the views to support data visualization capabilities.

4.2 OnRamp Server

The OnRamp Server is a centralized broker for all interactions between users (via the OnRamp Web Interface) and the connected PCEs. The Web Interface interacts with Server through a REST interface. This will allow for a more scalable Server implementation in the future, if load becomes a problem. The OnRamp Server maintains connections with each PCE through another REST interface exposed by the OnRamp PCE Service running on each PCE. This REST interface allows the OnRamp Server to interact with the PCE in a platform-agnostic manner.

The OnRamp Server authenticates users logging into the OnRamp Web Interface and enforces permissions defined by the Workspaces with which they are associated. An admin uses the Web Interface to establish a connection to a new PCE with the OnRamp Server. The admin supplies some basic connectivity information for the PCE, including URL, port, authentication credentials (username, password, PEM file). The credentials are only used to negotiate a secure channel and are not stored on the OnRamp Server or exposed to the users. Once a connection is established the OnRamp Server gathers documentation on the PCE to serve to the Web Interface, and monitors the connection and state of the PCE.

The admin uses the OnRamp Web Interface to install and deploy a Module on a PCE with the help of the OnRamp Server. The Server keeps track of the state of all Modules on each of the connected PCEs. As users request information about a Module (including Module specific parameters) the OnRamp Server pulls that information from the PCE Service and caches it locally.

When jobs are submitted they are given a job identifier by the OnRamp Server that is separate from the one that might be given by the PCE. This allows for unique job identifiers even if interacting with multiple PCEs. The Server provides users with links to access source code and output file(s) generated by completed jobs on the PCEs through the Server so that users do not need to connect directly to the PCE.

4.3 OnRamp PCE Service

The OnRamp PCE Service manages all activities on a specific PCE. This includes discovery of system software, management of available Modules, isolation of per-job files, and job lifecycle management.

The setup and installation of the OnRamp PCE Service is meant to be as turnkey as possible, but does require some configuration. A configuration file might specify the scheduler type (e.g., SLURM, PBS, unmanaged), any scheduler specific parameters (e.g., Accounting, Queue preferences, hostnames), software paths, or environment variables that the Service might need to know to work correctly on that PCE. This is all one time setup that does not require root access on the PCE.

When a Module is deployed by an admin it is installed in a common directory. When a user launches a job (which is always associated with a Module) then the Module directory is copied into a specific location for this user's job. The OnRamp PCE Service then creates a batch script tailored to that platform. The Service is responsible for handling the differences between various scheduling services so that neither the Module writer nor the user have to think about it. The Service then lazily monitors the job in the job queue triggering the necessary OnRamp scripts.

The OnRamp Server coordinates with the OnRamp PCE Service to monitor job status, and relay files to users.

4.4 OnRamp Module Infrastructure

The OnRamp Module Infrastructure provides a flexible framework for instructors to insert curriculum Modules into OnRamp that students can use to explore PDC. The goal is to leverage the curriculum modules being created by the CS education community with minimal overhead in OnRamp. OnRamp provides a platform-agnostic view of the Module to users with the parameter specification and launch being initiated from the OnRamp Web Interface. As such Module writers must wrap their Module code with a few scripts and specify any configuration parameters in a configuration file. Module writers write all of the configuration files and scripts, the users of the Module only see the specified runtime parameters and documentation for a specific Module.

The OnRamp Module Infrastructure defines five (5) stages of the Module workflow:

1. **Installation:** A once-per-PCE step that copies the necessary files to the PCE.
2. **Deployment:** A once-per-PCE step that runs a Module specific "deploy" script to perform any one time setup of the Module on this PCE. Typically this will involve setting any PCE-specific parameters needed by the Module, and building the parallel application(s).
3. **Preprocess:** Upon job launch, a Module specific "preprocess" script performs any actions that are necessary before the job is run on the PCE. If the Module needs to be compiled based upon user specified parameters then this is a good opportunity to do so.
4. **Run:** Inside of the batch script (generated by the OnRamp PCE Service) the Module specific "run" script is executed. This script defines exactly how the software should be run based upon the user specified parameters (e.g., this is where the mpirun command with user specified parameters is constructed and executed).
5. **Postprocess:** Once the job has completed then a Module specific "postprocess" script is run on the PCE. This script is typically used to cleanup the job directory or do any additional data processing required after the job completes.

Each stage of the Module workflow, with the exception of Installation, is represented by a Module specific Python script executed by the PCE Service. If a Module does not need to do anything during that stage then they can leave the file empty. The installation and deployment stages occur once-per-PCE whereas the other stages occur once-per-job.

The installation and deployment stages are meant to be executed without admin intervention by running a Module specific script. However, some software requires detailed

knowledge of the PCE to be deployed correctly (e.g., location of the BLAS libraries). As such there is a mechanism for the Module to request the admin to login to the PCE to complete the installation manually. Future generations of the OnRamp PCE Service will strive to auto-detect many of these PCE-specific parameters so that Modules are more able to self-deploy without admin intervention.

One of the key pieces of a Module is the configuration specification file which describes the parameters that a user can specify (e.g., number of particles in the simulation, number of iterations for the solver) and what constitutes a valid value for that parameter (e.g., positive integer, positive integer that is a power of two). This configuration specification file is used by the OnRamp Web Interface to prompt users to specify values and for the OnRamp Server to validate those values against the specification file. This mechanism gives the Module writer plenty of flexibility to have users specify runtime (or even compile-time) values that are then used by the scripts in the per-job stages, described above.

To allow for different levels of exploration within a Module, it is possible to create a set of Modules that contain the same source code but differ in the explanation, configuration parameters, and per-job scripts. For example, a Module might contain multiple implementations of an application (such as calculating the area under the curve) using MPI, OpenMP, CUDA, and some hybrid combinations. A simplified version of the Module might only expose the MPI version to the user. The configuration file, preprocess script and run script may differ, but the source code will stay the same. Since the Module writer controls what happens at each stage of execution, the Module writer has the flexibility to create Modules with a wide range of capabilities.

5. USE CASES

This section highlights some motivational use cases for OnRamp. These use cases are in contrast to the workflow described in Section 3. Notice how the students are able to explore parallelism more quickly and how the Level One functionality supports a wide range of learning activities.

5.1 CS1/CS2

In an introductory computer science course (i.e., CS1, CS2) students know some programming, but it would take significant time and effort to prepare students to use a PCE directly. However, introducing a few basic PDC concepts at this point in the curriculum can be beneficial and then can be built upon in later courses – following a spiral approach to reinforce and develop these concepts [4, 5]. In this use case, a CS2 instructor is introducing students to data decomposition and multithreading by having the students explore a parallel version of a sequential homework assignment.

After having the students implement a sequential solution, the instructor introduces a data set that is too big to run on a personal computer (i.e., needs more memory, or takes hours/days to run to completion) and asks the students to think about how they would overcome this problem before the next class meeting. The instructor prepares a multithreaded implementation of the same program and packages it as an OnRamp Module. The Module is installed and deployed on a PCE, and made available to the students through a Workspace in the OnRamp portal. The Module allows students to specify the numbers of threads, data sizes, and decomposition patterns. The Workspace contains notes from the instructor on the parallel implementation and the parameters that students can specify when launching a job.

During lecture, the instructor introduces the concept of data decomposition and concurrent execution. The students login to the OnRamp portal to explore running large data sets with various numbers of threads and record the effects on the runtime performance. The instructor then leads the class in a conversation about data decomposition, Amdahl's law, and weak and strong scaling using the results gathered by the students. This could be followed by a conversation about the role of large systems in scientific discovery.

5.2 Computer Architecture

An instructor teaching a Computer Architecture course wants to incorporate a hands-on module where students can explore the concepts of data and task decomposition, scaling, and shared vs. distributed memory architectures with real world applications on a range of PCE architectures. The instructor only has one or two days of class for this activity. The instructor introduces the concepts of scaling, Amdahl's law, data and task decomposition, and shared vs. distributed memory architectures in class prior to the activity. The purpose of the hands-on activity is to reinforce these PDC concepts by enabling students to explore the impact of different PCE architectures on the parallel performance of real world applications.

To prepare for the activity, the instructor identifies a set of four community developed OnRamp Curriculum Modules and deploys them on three different PCEs. The instructor creates a Workspace with these Modules and PCEs adding some notes about the learning objectives to the Workspace. OnRamp accounts are created for each of the students, and the students are added to the Workspace.

On the day of the activity, the instructor spends some time reviewing the parallel concepts, and discussing the scientific applications contained in the Modules highlighting the mathematical and computer science properties of these applications. The students are then asked to run some experiments using the OnRamp portal to explore how shared vs. distributed memory architectures impact the scaling properties of these applications. The Workspace provides guidance on how students should plot the data being gathered, and additional questions for individual or group discussion. The Workspace may ask students to design and run an experiment to study the parallel performance of one particular Module to more throughly understand its scaling properties as part of a homework activity.

5.3 Parallel Computing

An instructor teaching a Parallel Computing course wants to engage students with running code on PCEs from the first day, and then quickly help them transition into using the native PCE.

The instructor creates (or finds) a set of Modules that provide different implementations of the same application each using a different parallel programming paradigm that will be covered in the class: sequential (for reference), Pthreads, OpenMP, MPI, CUDA, and hybrid MPI+OpenMP. These Modules are all deployed on the PCEs that will be used during the course. The Modules are designed to present students with sections of the code that best showcase the programming paradigm. Two Workspaces are setup for the class. The first Workspace guides students through param-

eterizing runs of these preinstalled Modules. The second Workspace provides examples of scientific applications that use a variety of parallel programming paradigms.

In the first week of class, the instructor provides an overview of the different parallel programming paradigms and parallel concepts that will be explored in the course. Later in that week, the instructor introduces the application contained in the Modules along with an outline of the model of parallelism employed by the application. In that same class (or between classes) students use the OnRamp portal to run the Modules on a given data set to gather data for discussion. The first Workspace may highlight some aspects of the Module code to review for discussion. The instructor can then lead a discussion about the similarities and differences in performance and code structure based upon the experiences of the students. The second Workspace can be used in the following classes to demonstrate how real codes may not scale as cleanly as the codes that are studied in class due to the scientific models, data characteristics, and combining multiple parallel programming strategies.

5.4 Interdisciplinary Course

An instructor teaching a non-CS science course wants to use a parallel scientific application in a lab to explore a large data set. This situation is becoming more common as more disciplines add a computational dimension to their curriculum. Students must learn how to use these software applications to analyze complex phenomena. Many of these applications are parallelized in order to manage large problem sizes and long runtimes. Even though some of these applications are accessible though domain specific web portals on specific PCEs, it is difficult (or impossible) to obtain temporary accounts for students, and the portal may be too challenging to use for novice users.

Although OnRamp was designed to help teach PDC concepts to CS students, the flexibility of the infrastructure allows it to be used to create a custom portal for this instructor and tailor it to their students. The instructor, possibly in collaboration with a CS instructor, designs the Module interface and deploys it on a PCE. The Module may only expose a limited set of problem specific parameters to the students, leaving the PCE runtime parameters (e.g., number of tasks or nodes) as optional or hidden options. The instructor then creates a Workspace for the lab that instructs students on what the software is simulating, how to create and upload a data set, then how to analyze the results.

During the lab session, students login to the OnRamp portal, review the Workspace documentation, upload their data set(s), and run those simulations. The students might experiment with a physical system while the simulation is running then compare the results towards the end of the lab. This comparison can foster a discussion about the role of in-silica simulation and in-lab experimentation for exploring complex phenomena in their discipline.

The Workspace and Module may be designed to expose students to why parallelism is needed for this simulation, and a high level view of how parallelism is used in the application. This exposure to the CS mechanics of the simulation may cause students to become more interested in CS and how it relates to their field of study. CS instructors can use the Module creation as an activity for CS students to learn how to package up a complex, real world application, and present it in a meaningful, productive manner to domain scientists. If the PCE runtime parameters (e.g., number of tasks) are automatically set based upon problem size, CS students will need to conduct a scaling study (possibly on a variety of PCEs) to determine how to set those runtime parameters based upon the Module parameters set by the users. This is a concrete instance where the scaling study information can be used to influence how a student might design a piece of code, a Module script in this use case.

6. CONCLUSION AND FUTURE WORK

The high barrier to entry for using a PCE more capable than a laptop can often distract and intimidate CS students away from learning PDC concepts. The OnRamp project provides a web portal which coaches users through a collection of interactive curriculum modules teaching them about parallel computing and the software ecosystem while allowing them to launch parallel applications on PCEs from day one. OnRamp is designed to be a general purpose framework for supporting the exploration of PDC concepts that harnesses the existing educational resources created by the CS education community.

This paper described the design of the OnRamp infrastructure with a focus on supporting Level One functionality along with motivating use cases that ground the project. OnRamp is an open source project available on GitHub.[4]

The first full release of the software is scheduled for March 2016 which will support Level One exploration and a range of PCEs and Modules. Future development efforts will focus on expanding support for PCEs and Modules, and expanding functionality to support the remaining three knowledge levels.

7. ACKNOWLEDGEMENTS

We would like to thank our current students working on the project, Jason Regina, Daniel Koepke, and Justin Ragatz, for their work on the current implementation of the system, as well as Zackory Erickson for his work on an early prototype of this project.

Research supported by the SIGCSE Special Projects grant program, Blue Waters HPC Internship Program, University of Wisconsin–La Crosse (UW-L) Undergraduate Research Grant program, and by the UW-L Computer Science Department.

8. REFERENCES

[1] Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society, *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. New York, NY, USA: ACM, 2013.

[2] J. C. Adams, J. Caswell, S. J. Matthews, C. Peck, E. Shoop, and D. Toth, "Budget Beowulfs: A showcase of inexpensive clusters for teaching PDC," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '15. New York, NY, USA: ACM, 2015, pp. 344–345. [Online]. Available: http://doi.acm.org/10.1145/2676723.2677317

[4] https://github.com/ssfoley/onramp

[3] S. K. Prasad, A. Chtchelkanova, S. Das, F. Dehne, M. Gouda, A. Gupta, J. Jaja, K. Kant, A. La Salle, R. LeBlanc, M. Lumsdaine, D. Padua, M. Parashar, V. Prasanna, Y. Robert, A. Rosenberg, S. Sahni, B. Shirazi, A. Sussman, C. Weems, and J. Wu, "NSF/IEEE-TCPP curriculum initiative on parallel and distributed computing: Core topics for undergraduates," in *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '11. New York, NY, USA: ACM, 2011, pp. 617–618. [Online]. Available: http://doi.acm.org/10.1145/1953163.1953336

[4] R. Brown, E. Shoop, J. Adams, C. Clifton, M. Gardner, M. Haupt, and P. Hinsbeeck, "Strategies for preparing computer science students for the multicore world," in *Proceedings of the 2010 ITiCSE Working Group Reports*, ser. ITiCSE-WGR '10. New York, NY, USA: ACM, 2010, pp. 97–115. [Online]. Available: http://doi.acm.org/10.1145/1971681.1971689

[5] A. Fitz Gibbon, D. A. Joiner, H. Neeman, C. Peck, and S. Thompson, "Teaching high performance computing to undergraduate faculty and undergraduate students," in *Proceedings of the 2010 TeraGrid Conference*, ser. TG '10. New York, NY, USA: ACM, 2010, pp. 7:1–7:7. [Online]. Available: http://doi.acm.org/10.1145/1838574.1838581

[6] J. Adams, R. Brown, and E. Shoop, "Patterns and exemplars: Compelling strategies for teaching parallel and distributed computing to cs undergraduates," in *IEEE International Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW)*, May 2013, pp. 1244–1251.

[7] R. Brown and E. Shoop, "CSinParallel and synergy for rapid incremental addition of PDC into CS curricula," in *IEEE International Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW)*, May 2012, pp. 1329–1334.

[8] S. Foundation. Undergraduate petascale modules. [Online]. Available: http://shodor.org/petascale/materials/modules/

[9] M. Ludin, A. Weeden, J. Houchins, S. Thompson, C. Peck, I. Babic, K. Muterspaw, and E. Sergienko, "LittleFe: The high performance computing education appliance," in *IEEE International Conference on Cluster Computing*, ser. CLUSTER, Sept 2013.

[10] (2013) Parallella university program. [Online]. Available: https://www.parallella.org/pup/

[11] 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. Wilkens-Diehr, "XSEDE: Accelerating scientific discovery," *Computing in Science and Engineering*, vol. 16, no. 5, pp. 62–74, 2014.

[12] C. Carley, L. Sells, B. McKinney, C. Zhao, and H. Neeman, "Using a shared, remote cluster for teaching HPC," in *IEEE International Conference on Cluster Computing (CLUSTER)*, Sept 2013, pp. 1–6.

[13] G. Fox, G. von Laszewski, J. Diaz, K. Keahey, J. Fortes, R. Figueiredo, S. Smallen, W. Smith, and A. Grimshaw, *FutureGrid - a reconfigurable testbed for Cloud, HPC, and Grid Computing*, ser. CRC Computational Science. Chapman & Hall, 04/2013 2013.

[14] P. Garrity, T. Yates, R. Brown, and E. Shoop, "Webmapreduce: An accessible and adaptable tool for teaching map-reduce computing," in *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '11. New York, NY, USA: ACM, 2011, pp. 183–188. [Online]. Available: http://doi.acm.org/10.1145/1953163.1953221

[15] Y. Gil, "Teaching parallelism without programming: A data science curriculum for non-CS students," in *Education for High Performance Computing (EduHPC), 2014 Workshop on*, Nov 2014, pp. 42–48.

[16] M. P. Thomas, J. Burruss, L. Cinquini, G. Fox, D. Gannon, L. Gilbert, G. von Laszewski, K. Jackson, D. Middleton, R. Moore, M. Pierce, B. Plale, A. Rajasekar, R. Regno, E. Roberts, D. Schissel, A. Seth, and W. Schroeder, "Grid portal architectures for scientific applications," *Journal of Physics: Conference Series*, vol. 16, no. 1, p. 596, 2005. [Online]. Available: http://stacks.iop.org/1742-6596/16/i=1/a=083

[17] P. Bui, T. Boettcher, N. Jaeger, and J. Westphal, "Using clusters in undergraduate research: Distributed animation rendering, photo processing, and image transcoding," in *Cluster Computing (CLUSTER), 2013 IEEE International Conference on*, Sept 2013, pp. 1–8.

[18] H. Lin, B. Ganov, J. Kemp, and P. Gilbert, "A grid computing environment for undergraduate research," in *Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on*, June 2012, pp. 16–19.

[19] H. Burkhart, D. Guerrera, and A. Maffia, "Trusted high-performance computing in the classroom," in *Education for High Performance Computing (EduHPC), 2014 Workshop on*, Nov 2014, pp. 27–33.