Teaching Parallel and Distributed Computing Concepts in Simulation with WRENCH

Ryan Tanaka*, Rafael Ferreira da Silva†, Henri Casanova*
*Information and Computer Sciences, University of Hawaii, Honolulu, HI, USA
†Information Sciences Institute, University of Southern California, Marina Del Rey, CA, USA
{henric,ryanyt}@hawaii.edu, rafsilva@isi.edu

***Abstract*—Teaching topics related to high performance computing and parallel and distributed computing in a hands-on manner is challenging, especially at introductory, undergraduate levels. There is a participation challenge due to the need to secure access to a platform on which students can learn via hands-on activities, which is not always possible. There are also pedagogic challenges. For instance, any particular platform provided to students imposes constraints on which learning objectives can be achieved. These challenges become steeper as the topics being taught target more heterogeneous, more distributed, and/or larger platforms, as needed to prepare students for using and developing Cyberinfrastructure.**

To address the above challenges, we have developed a set of pedagogic activities that can be integrated piecemeal in university courses, starting at freshman levels. These activities use simulation so that students can experience hands-on any relevant application and platform scenarios. This is achieved by capitalizing on the capabilities of the WRENCH and SimGrid simulation frameworks. After describing our approach and the pedagogic activities currently available, we present results from an evaluation performed in an undergraduate university course.

***Index Terms*—Computer Science Education, High Performance Computing, Parallel and Distributed Computing, Cyberinfrastructure, Simulation**

I. Introduction

Teaching topics related to High Performance Computing (HPC), including many topics in Parallel and Distributed Computing (PDC), is most effective when students have opportunities for hands-on learning. The common approach is to provide students with access to a platform on which they can develop and/or execute applications so as to achieve various Student Learning Objectives (SLOs). Unfortunately, instructors using this approach face several challenges.

A *participation challenge* stems from the need to provide students with access to HPC platforms, which is not feasible at all institutions. Even when an institution hosts such platforms, there may be no straightforward mechanism to use them, or even sizable subsets thereof, for education purposes (e.g., the platform serves a research community and pedagogic use would disrupt production use). Also, a popular way to increase participation is to develop Massive Open Online Courses (MOOCs). However, developing a hands-on MOOC would require that access to a representative platform be guaranteed for a large, diverse, and distributed student population, which is at best feasible only at a few institutions.

Even when students are provided with an HPC platform for a course, there are *pedagogic challenges*. This is because students are only exposed to the particular configuration of that platform, leaving many relevant scenarios out of reach (e.g., different scales, different hardware specifications of compute nodes, different network topologies and fabrics, different software stacks) and thus precluding achieving some SLOs in a hands-on manner. Also, students must be trained on platform usage mechanisms and policies. While some courses could deliberately devote a large portion of the syllabus to these mechanisms and policies, in other courses this would be pure overhead (e.g., introductory freshmen courses). Also, executing workloads on real-world platforms is not free: it requires time, electricity, in some cases funds. There are thus practical bounds on the number and/or scale of the executions available to students, which can impede hands-on learning. An additional cost is that instructors typically devote a significant amount of effort to managing students' use of the platform (e.g., platform monitoring and troubleshooting, interaction with platform administrators, managing competition for resources among students).

The above challenges become steeper as the SLOs target more heterogeneous, more distributed, and/or larger platforms. At one end of the spectrum would be courses for which SLOs can be achieved using a single, moderate-scale, commodity cluster. Some institutions (e.g., Ph.D.-granting institutions) can provide students access to a representative cluster with relatively straightforward mechanisms and policies e.g., a batch scheduler). But even in this "easy" case the participation and pedagogic challenges are significant and have prompted the exploration of alternate approaches [1]–[3]. At the other end of the spectrum would be courses that attempt to teach principles and practices of CyberInfrastructure (CI) computing, i.e., the execution of application workloads via services deployed on distributed environments with heterogeneous hardware and software stacks. An example would include the execution of a particular scientific application on a platform that comprises cloud platforms, batch-scheduled HPC clusters, and data hosting services, all distributed over wide-area networks. Such CI deployments would be available for teaching purposes only at a few institutions, and would require that students learn a large set of usage policies and mechanisms, likely precluding hands-on teaching of CI-related concepts in introductory courses.

Furthermore, such platforms are subject to many effects that can make application executions not completely repeatable, which impedes learning. Finally, given the sheer number of relevant software and hardware configurations, it is unclear how a single CI deployment provided to students could be "representative". We argue that the aforementioned participation and pedagogic challenges are likely insurmountable in most university courses as far as CI education is concerned. And yet, it is critical to prepare students that will join the national scientific research and engineering workforce for a world in which CI computing is the norm.

An alternative to using real-world platforms for teaching purposes is to use *simulation*, i.e., simulate executions using a software artifact that mimics real-world executions. The participation challenge is obviated as the only requirement is that students have access to a computer on which the simulation software is installed (e.g., a student laptop). The pedagogic challenges are also addressed: simulated executions can target arbitrary hardware and software stack configurations, including those encountered in complex CI scenarios; one can pick the level of details exposed to students regarding platform usage mechanisms; simulated executions are repeatable and can be executed quickly and at negligible cost.

In this work, we present a set of pedagogic activities that target HPC and PDC SLOs, in particular as relevant to CI computing. The intent is for these activities to be integrated piecemeal in university courses, from freshman- to graduate-level courses. These activities allow students to acquire knowledge by experimenting with various application and platform scenarios. The simulations used in these activities provide both metrics and visualizations of executions through which students can empirically verify their answers to relevant questions. Students can also use simulations to explore complex design spaces so as to acquire knowledge independently, possibly with instructor-provided scaffolding. Finally, some "capstone" activities consist of case-studies in which students apply what they have learned in previous activities to solve real-world problems. Our contributions are as follows:

- We describe and justify the use of the WRENCH [4] simulation framework as a foundation for this work, and how it makes it possible to quickly develop and release our pedagogic activities;
- We describe the currently available pedagogic activities and the SLOs they target, highlighting how simulation is key to achieving these SLOs;
- We present evaluation results obtained in the classroom during a pilot study conducted in a 3rd-year undergraduate course at the University of Hawai'i at Mānoa.

II. Related Work

Several options have been proposed to address the challenge of providing students with HPC platforms for teaching purposes when such platforms are not readily available at their institutions. These options, which include building low-cost platforms [5]–[10] and emulating platforms using virtualization and/or containers [11]–[14], often do not provide platforms with capabilities, scales, and/or performance behaviors representative of production environments. In this work, we rely instead on simulation to provide students with hands-on experience for arbitrary platform configurations.

Many simulation frameworks have been developed for parallel and distributed computing research and development, several of which could also be used for education. These frameworks span domains such as HPC [15]–[18], Grid [19]–[21], Cloud [22]–[24], Peer-to-peer [25], [26], or Volunteer Computing [27]–[29]. Some frameworks have striven to be applicable across some or all or the above domains [30], [31]. Two conflicting concerns are *accuracy* (the ability to capture the behavior of a real-world system with as little bias as possible) and *scalability* (the ability to simulate large systems with as few CPU cycles and bytes of RAM as possible). The aforementioned simulation frameworks achieve different compromises between these two concerns by using various simulation models. At one extreme are discrete event models that simulate the "microscopic" behavior of hardware/-software systems (e.g., by relying on packet-level network simulation for communication [32], on cycle-accurate CPU simulation [33] or emulation for computation). In this case, the scalability challenge can be handled by using Parallel Discrete Event Simulation [34], i.e., the simulation itself is a parallel application that requires a parallel platform whose scale is at least commensurate to that of the simulated platform. At the other extreme are analytical models that capture "macroscopic" behaviors (e.g., transfer times as data sizes divided by bottleneck bandwidths, compute times as numbers of operations divided by compute speeds). While these models are typically more scalable, they must be developed with care so that they are accurate. In previous work, it has been shown that several popular simulation frameworks use macroscopic models that can exhibit high inaccuracy [35]. One of the key intellectual contributions of the SimGrid project [30] is that it employs simulation models that are scalable (because macroscopic) and yet accurate (as shown in several validation studies [35]–[39]). As a result, SimGrid simulation can execute quickly on, say, a student's laptop, and yet yield accurate results even for complex simulation scenario. SimGrid provides the core simulation technology for the WRENCH simulation framework, which is used in this work.

Simulation is used routinely as a pedagogic tool in many areas of the computer science curriculum. For instance, it is traditional to use simulation frameworks for teaching computer architecture and network concepts, since without simulation it can be extremely challenging to create hands-on learning opportunities in these domains. By comparison, the use of simulation as a pedagogic tool is relatively rare in the parallel, distributed, and/or HPC domains, likely because in some cases it is possible to use some HPC platform for teaching purposes, albeit facing the challenges outlined in Section I. Several works in the early 1990's proposed using simulation for the purpose of parallel computing education [40]–[43]. More recently, the authors in [1] describe the parallel computing module of a M.S. degree in HPC, which relies on simulation

as a foundational technology. The simulation in use employs microscopic simulation models, thus mandating that students be provided by an HPC platform to run simulations. In this work instead, because we rely on WRENCH for simulations, students can easily run simulations on their own computers. Another recent work, Paralab teachware, is presented in [2], which uses the Paralab system [44] to teach parallel computing concepts and algorithms using simulation. From what information is provided in [44], Paralab implements naive simulation models that are not necessarily representative of real-world platforms, which is problematic when teaching advanced topics and concepts. Finally, a major difference between this work and all the above is that, instead of targeting SLOs that pertain only to parallel computing on homogeneous platforms, instead we target the much broader set of SLOs relevant to PDC/CI concepts and practices.

III. Using WRENCH for Education

A. The WRENCH Simulation Framework

The simulators in our proposed pedagogic activities must be developed on top of a simulation framework. We identify four requirements for this framework:

- #1 Accuracy: the framework's simulation models must have been the object of validation/invalidation studies;
- #2 Scalability: it should be possible to run simulations quickly on a single computer (e.g., a student laptop) with low RAM footprint;
- #3 Versatility: the framework must be expressive enough to allow the simulation of a wide range of scenarios, from a single homogeneous cluster running a standard HPC application all the way to complex multi-site CI scenarios with diverse software and hardware stacks;
- #4 Easy development: the framework must provide APIs that make it possible to implement simulations of complex scenarios in, say, at most a few hundred lines of code. This is so that developing a wealth of pedagogic activities that students can use and benefit from is not labor-intensive.

The last requirement is key as each pedagogic activity can comprise several simulators, and as we ultimately intend to develop dozens of such activities.

As discussed in Section II, many simulation frameworks have been developed for the PDC domain. One framework that meets the first three requirements above is SimGrid [30], [45]. Its simulation accuracy and scalability have been shown to be significantly better than that of its competitors [30], [35] (requirements #1 and #2). It is applicable to and has been utilized for scenarios ranging from the simulation of MPI applications on clusters to the simulation of peer-to-peer applications (requirement #3). It has also been actively developed for almost two decades, with a regular release schedule, a large team of developers, and a vibrant user community. Finally, it has already been used successfully for teaching purposes [3]. Unfortunately, SimGrid does not meet requirement #4 above because its simulation abstractions are low-level. The critical analysis in [46] recognizes that SimGrid provides superior accuracy and scalable simulation capabilities, but also observes that using it to implement a simulator of a complex system, such as CI scenarios, is labor-intensive.

It is in part to meet requirement #4 that the WRENCH project [4], [47] was initiated. WRENCH builds on SimGrid, so that simulations can be accurate and scalable, to expose reusable high-level simulation abstractions, so that implementing simulators of CI scenarios can be done with minimal software engineering efforts. For instance, WRENCH provides several simulated implementations of "compute services" for bare-metal hardware resources, virtualized hardware resources, cloud platforms, batch-scheduled clusters, or HTCondor pools. The whole set of WRENCH-provided simulation abstractions is listed on WRENCH's Web site [47] (version 1.4 of the WRENCH software was released in April 2019). Demonstrating WRENCH's ease of use, the work in [4] describes how a simulator of a production Workflow Management System that executes scientific workflow applications on several CI hardware/software stacks can be implemented with only a few hundred lines of code.

B. Pedagogic Activities with WRENCH

Given the discussion in the previous section, we use WRENCH to implement our pedagogic activities. Each activity is a Web page that introduces concepts to students through a narrative. For each concept students are asked questions that they must answer before proceeding forward. Students answer questions in two ways: (*i*) they discover a valid answer by running simulations to explore the space of simulated executions; (*ii*) they come up with an answer through analysis and validate/invalidate this answer by running simulations. Students invoke simulators provided to them, using particular input. All such invocations are made through an interactive Web graphical interface. This interface displays simulation output as both non-interactive and interactive visualizations.

Each simulator consists of a main program that sets up the environment to simulate; and of a simulated application execution management system, which orchestrates a particular application workload on available resources. At the time of writing, 4 different simulators are provided as part of our pedagogic activities and consist of only 362, 516, 224, and 442 lines of C++ including comments.

To ensure ease-of-use and portability, we package all necessary software (Web app, simulators, WRENCH and its dependencies, SimGrid and its depencies) in a Docker container. When instructed to do so students simply run this container on their own machines. The container starts a Web server on a local port and hosts a Web app that students access by navigating to a local URL in a browser. The instructions given to students for running a simulator are straightforward:

1) Copy-and-paste in the terminal: `docker pull <name>`
2) Copy-and-paste in the terminal: `docker container run -p 3000:3000 -d <name>`
3) Visit `http://localhost:3000/`

where <name> is the name of the Docker container we provide. The only software requirement is that Docker be installed.

IV. The WRENCH Pedagogic Activities

A. Overview

To date we have released 5 activities that are all available on-line [48]. These activities are designed to be performed as a sequence and target the following topics:

(I) Networking – In this activity, students acquire essential network concepts as they relate to the execution of applications on distributed CI deployments. These concepts include notions of latency, bandwidth, and topology, and how they are used to reason about data transfer times.

(II) Workflows – This activity introduces students to concepts necessary to understand how a workflow application can execute on a distributed CI deployment, and how application execution time can be estimated given application specification and available hardware resources.

(III) Data Locality – Building on the previous two activities, this activity introduces the concept of data locality. Students evaluate different application execution scenarios and experiment first-hand the impact of data locality on performance.

(IV) Parallelism – Building on the first two activities as well, this activity introduces students to notions pertaining to parallel computing (e.g., speedup, dependent vs. independent tasks) and parallel computing platforms (e.g., multi-core nodes with limited RAM, multi-node clusters).

(V) Resource Provisioning – In this capstone activity, students use the concepts acquired in previous activities to solve a real-world problem. Students are presented with an application and a set of hardware resources. Given a budget, students must determine the best way to spend this budget on hardware upgrades in order to minimize application execution time.

Table I shows a curriculum map for the above activities, indicating coverage of a set of 15 SLOs. These SLOs in no way constitute a full curriculum. We are currently designing and implementing other activities to target missing essential SLOs (e.g., Amdahl's law, overhead, strong scaling vs. weak scaling) and more advanced SLOs (e.g., scheduling, fault-tolerance). Due to lack of space we cannot describe all the above activities and refer the reader to our Web site for full details [48]. In the next section, we provide an overview of one of our activities. Although (purposely) at an introductory level, this activity showcases our overall approach for delivering pedagogic content using simulation.

B. Sample Activity: Parallelism

Activity (IV) focuses on parallelism and presents students with the CI deployment depicted in the left-hand side of Figure 1. The deployment consists of three sites. The site on the top left (`storage_db.edu`) hosts a Storage Service with infinite storage capacity. The site on the right hosts a Compute Service (accessible via `hpc.edu`). These two sites are connected via a network link with 100μs latency and 125 MB/sec bandwidth. The third site (`my_lab_computer.edu`) is where the user resides and runs a Workflow Management System software that will orchestrate application executions using the hardware resources provided by the Storage and Compute Services.

The application to be executed on the above CI deployment is provided to students as depicted on the right-hand side of Figure 1. It consists of 20 identical and independent single-threaded tasks, each taking in a 2GB input file, computing 3600 TFlop using 4GB of RAM, and producing a 2GB output file. A last, also single-threaded, task takes in all these output files, computes 300 TFlop using 42GB of RAM, and produces a final 2GB output file. The first 20 input files, are initially stored on the Storage Service at host `storage_db.edu`. All tasks must be executed on the Compute Service. This service has local scratch space (with fast bandwidth of 1250 MB/sec) to which intermediate files (i.e., the output files of the first 20 tasks) are written. Finally, the last output file must be written back to the Storage Service.

The above scenario is presented to the students in the form of a narrative, with the depictions in Figure 1, and some review of the SLOs achieved in previous activities. Using an example, students are introduced to the concept of core utilization and core idle time. Students are then asked to answer a series of questions, assuming the Compute Service only hosts one single-core node. An example question is: "What do you expect the overall execution time to be? Write a simple formula, and use the simulator to check your answer". The activity then evolves the scenario to cases in which the compute node at the Compute Service has 10, 15, or 20 cores. In each cases, students quantify the performance gain, or lack thereof, and experience first-hand the interplay between core utilization and application performance.

In a second phase of this activity, the Compute Service hosts multiple multi-core nodes, which is depicted to students as in Figure 2. At the same time, each of the first 20 tasks uses 12GB more RAM, thus limiting the utilization of each compute node (whose RAM capacity is 80GB). The narrative explains to students how RAM constraints impact parallelism on a compute node, and then presents students again with a series of questions. An example question is: "What is the minimum number of 3-core nodes that achieves the previously determined fastest possible execution time?" Students can answer this question analytically and check their answer in simulation, or empirically by running several simulations to do a binary search. Other questions pertain to core utilization, getting to the notion of which hardware investments are "worth it". This notion is fully explored in Activity (V), which focuses on Resource Provisioning.

Students invoke the provided simulator via a Web app packaged as a Docker container, as described in Section III-B. Figure 3 shows the portion of the Web app through which students configure and run the simulator. In the example, the simulator is configured for a Compute Service with three 6-core compute nodes, and for the tasks to require extra RAM. Clicking the "Run Simulation" button invokes the simulator (which takes less than 1 second), and updates the Web page

TABLE I
CURRICULUM MAP FOR 5 WRENCH PEDAGOGIC ACTIVITIES: (I) NETWORKING; (II) SCIENTIFIC WORKFLOWS AND THEIR EXECUTIONS ON CI DEPLOYMENTS; (III) DATA LOCALITY; (IV) PARALLELISM; (V) RESOURCE PROVISIONING. (E: EMERGING, D: DEVELOPING, P: PROFICIENT).

Student Learning Objective		Activity				
		(I)	(II)	(III)	(IV)	(V)
#1	Understand notions of network topology, bandwidth, and latency	E	E	D	-	P
#2	Be able to reason about bottleneck links and bandwidth sharing	E	E	D	-	P
#3	Be able to estimate data transfer times given a topology, including for concurrent data transfers	E	E	D	-	P
#4	Be exposed to CI deployments that consist of storage and compute services deployed on wide-area networks of heterogeneous resources	-	E	E	D	P
#5	Understand the structure of scientific workflow applications and the notion of task depencies	-	E	D	D	P
#6	Be able to estimate workflow execution time for a given CI deployment including computation and I/O	-	E	D	D	P
#7	Understand notions of data locality and data proximity	E	-	D	-	P
#8	Be able to quantify the impact of data locality on application execution time	E	-	D	-	P
#9	Be familiar with multi-core compute nodes and how they can be aggregated to form compute clusters	-	-	-	E	D
#10	Understand parallelisms and parallel speedup	-	E	E	D	P
#11	Understand the tradeoff between parallelism and core utilization	-	-	-	D	P
#12	Understand how memory footprint constraints can limit parallelism	-	-	-	D	P
#13	Be able to estimate application execution times when tasks are executed in parallel on a cluster of multi-core compute nodes with limited RAM capacity	-	-	-	D	P
#14	Be able to compare resource provisioning alternatives for a given application	-	-	-	E	D
#15	Be able to make appropriate resource provisioning decisions for a given application given budget constraints	-	-	-	E	D

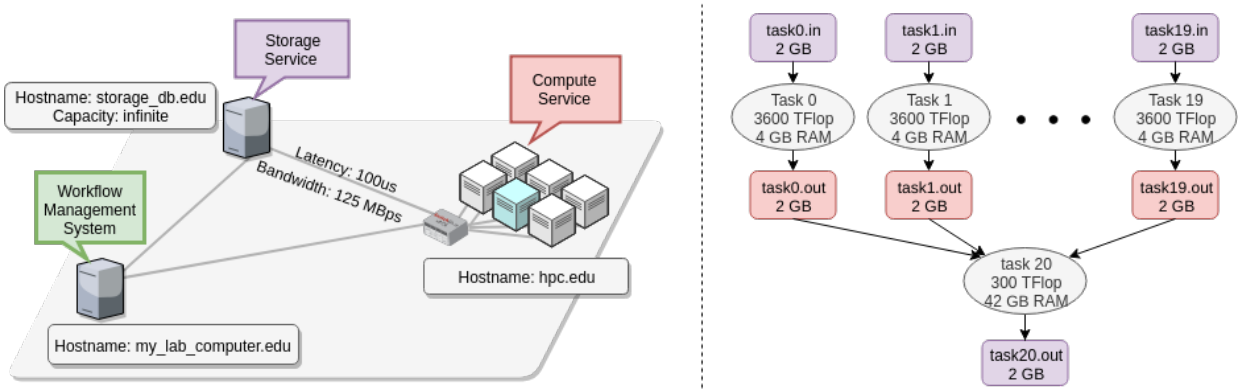

Fig. 1. Activity (IV) simulated CI deployment (left-hand side) and workflow application (right-hand side).

with several visualizations of the simulated execution. Figure 4 shows a Gantt chart visualization, which displays, for each task on the vertical axis, the task execution timeline with input read, computation, and output write. Tasks that read/write input concurrently split the bandwidth, which is why I/O times for the 5 tasks that start around time 3,900 are lower than that of the 15 tasks that start at time 0. Hovering about any component in this visualization pops up a tooltip with qualitative and quantitative information (e.g., task and file names, durations, start and end times). Another provided visualization is the core utilization time-line shown in Figure 5. This visualization shows idle core time due to I/O operations (e.g., the gap at time 0), due to RAM limitation (e.g., the 6th core on a node is never used), due to imperfect load-balancing (e.g., the 2nd and 3rd nodes are fully idle after time 3,900), and lack of parallelism (e.g., all cores but one are idle while the last task executes). Here again, a student can hover over any component of this visualization to gain more detailed information.

V. EVALUATION

A. Preliminary Evaluation

In February 2019, we performed a preliminary pedagogic evaluation of our activities with three student participants. These participants were seniors in the B.S. in Computer Science program at the University of Hawai'i at Mānoa (UHM), and were recruited on a volunteer basis. Their motivation

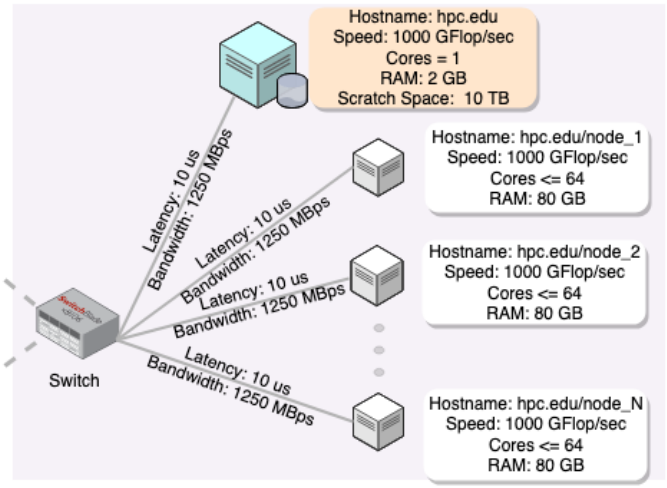

Fig. 2. Simulated hardware specification of the cluster hosted by the Compute Service for Activity (IV).

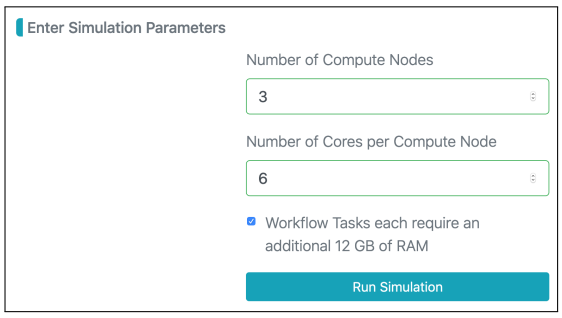

Fig. 3. Simulator input panel for Activity (IV).

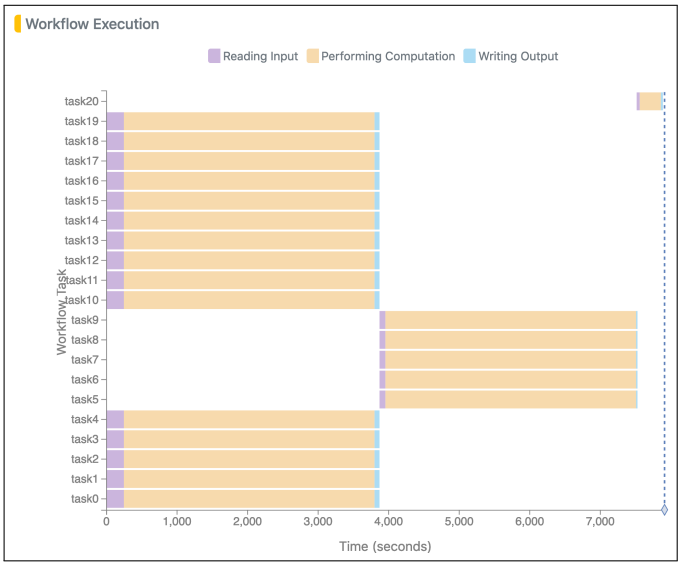

Fig. 4. Sample Gantt chart of task executions for Activity (IV) given the input shown in Figure 3.

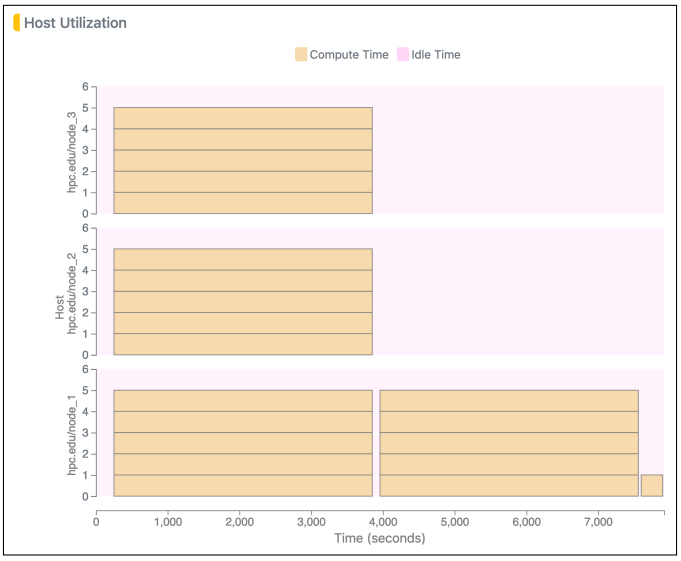

Fig. 5. Sample core utilization time-line for Activity (IV) given the input shown in Figure 3.

for participating was a desire to learn more about a crucial topic that is often not (sufficiently) taught in the standard curriculum. Students were given a reading assignment in which they had to perform activities (I) and (II) on their own. A week later, in a 1-hour session, the pedagogic team (the first and last author of this paper) together with the students did activity (III), answering questions based on group discussion. A week later, in another 1-hour session, the participants did as many questions as possible in activity (IV), with the pedagogic team providing necessary scaffolding.

Students were given pre and post knowledge tests, as well as questionnaires about their experience. The purpose was mostly to identify potential improvements in the tests and questionnaires themselves, and in the pedagogic content. Based on participant feedback we made several content changes: (*i*) we removed content that proved too detailed and detrimental to learning; (*ii*) we added text and figures to clarify points of confusion; (*iii*) we improved simulated execution visualizations to include more information. In spite of these students being presented with a preliminary version of our activities, feedback was overall very positive and students felt that they acquired new knowledge easily.

B. Classroom Evaluation

The last author of this paper regularly teaches the undergraduate Operating Systems (ICS 332) course at UHM. A PDC module was added to the course syllabus for the Spring 2019 semester, which consists of the following steps:

1) A 30-minute lecture on PDC to motivate the topic;
2) A reading assignment in which students performed activities (I) and (II) on their own;
3) A 75-minute in-class interactive session during which the pedagogic team did activity (III), soliciting participation from students and fielding questions;

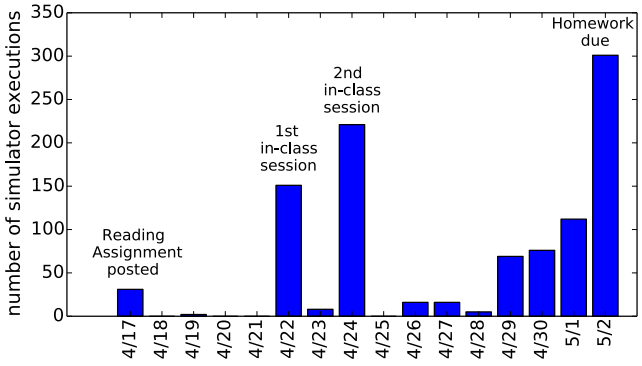

Fig. 6. Daily numbers of simulations executed by students.

4) A 75-minute in-class interactive session during which students, either individually or in groups of up to 3, started on activity (III) with scaffolding provided by the pedagogic team;
5) A homework assignment in which students answered 4 late questions of activity (III), with solutions to preceding questions provided to them; and
6) Three problems on the final exam (covering SLOs #1 to #13 in Table I). These problems were worth 10% of the final exam, which was itself worth 30% of the overall grade for the course.

We gathered qualitative and quantitative data:

- Anonymous post questionnaires about experience and perceived learning in step 3 and 4 above;
- Anonymous pre and post knowledge tests in step 3 and 4 above;
- Informal feedback volunteered by students directly and/or entered in UHM's course evaluation system;
- Non-anonymous grades for the homework assignment;
- Non-anonymous grades for relevant final exam questions;
- Non-anonymous time-stamped trace data from activity Web apps for each simulator execution.

What is missing from our evaluation data is a control group, i.e., another group of students that are taught the same material but without using simulation. A comparison to a version of the module taught without any hands-on experience for student would only evaluate the benefit of hands-on education, which is already well documented. What is needed instead is a comparison to teaching this module using a real-world infrastructure. However, as explained in Section I, using a real-world infrastructure for this purpose is extremely difficult, and is typically not done (especially at the undergraduate level). This is precisely why we advocate the use of simulation in the first place. But as a result, we cannot provide comparisons with results obtained with a sensible control group.

C. Results

Question #1: Are students using the simulation?

55 students took the final exam in the course, and 45 of these students ran simulations. Daily totals are shown in Figure 6. Overall 1008 simulations were executed by 45 students in a 16-day period, with expected peak days when the reading assignment was posted, during the two in-class activities, and leading up the homework assignment's due date. Note that during the 1st in-class session it was not a requirement for students to run simulations, but many of them opted to "follow along" on their own computers. For the 16-day period, and only considering students who ran at least one simulation, each student executed 22 simulations on average (with a maximum of 61 simulations for one student).

TABLE II
CORRELATION BETWEEN NUMBER OF SIMULATIONS EXECUTED AND AVERAGE GRADE ON PDC-FOCUSED FINAL EXAM QUESTIONS.

# of simulations	# of students	grade average
0	10	67.6
1-10	14	88.8
11-20	13	99.8
21-30	6	81.0
31+	12	75.5

Activity (IV), which was started in the 2nd in-class session and completed in the homework assignment, explicitly asks students to run the simulator 7 times for particular input settings. But we find that 82% of students ran more than 7 simulations (with the average at 21). Furthermore, we find that 40% of simulation runs were for input settings that were not suggested to students. We conclude that most students used simulation independently for better learning the material and completing their assignments. And indeed, in the classroom the pedagogic team observed groups of students "trying out" various simulation configurations to check whether they understood the material, often just out of curiosity. However, students who struggle with the material could also be running many simulations, perhaps even haphazardly.

Question #2: Are students learning the material?

Table II shows how grades obtained by students on the final exam, counting only those questions that pertain to PDC SLOs, correlate with the number of simulations that students have executed. Recall that during in-class sessions students often worked in groups, meaning that for some students lower numbers of simulations were recorded because they worked with another student who ran the simulations using their computer. We find that students who have executed no simulations perform much worse than other students, to the point of not getting a passing grade on average. Interestingly, students who ran a lot of simulations also do not perform well, scoring a C on average. These are likely students who struggled with the material (recall that Activity (IV) asks students to run, at a minimum, 7 simulations). We find that students who ran between 11 and 20 simulations performed best, with almost a perfect score on average. Note that there was a difficult extra credit PDC question on the exam, which allows some students to score above 100%. Overall, out of the 55 students who took the final exam, the grade distribution for the PDC-related questions was as follows: A: 25; B: 10; C: 5; D: 6; F: 9. In Activity (IV)'s pre knowledge test, only 23% of students answered correctly the multiple-choice question: "You have two 4-core machines, and your application has 10 independent tasks. Each task runs in 1 minute on a single

core and uses only a few bytes of RAM. How fast can you run your application?". Scores on the final exam show that close to 82% of students answered similar but more difficult questions correctly.

It is difficult to generalize conclusions from a single group of students, albeit 55 of them. Nevertheless, we feel that the above results indicate that running simulations enhances learning. This result, however, could be merely about the benefits of hands-on learning in general. But, as explained earlier, without simulation it is extremely difficult to teach this material hands-on in the first place.

Question #3: Are students having a positive experience? Students were asked for feedback on their experience via anonymous questionnaires. Overall, when accounting for all answers to a multiple-choice question about the level of difficulty of the material, 60% of students answered "just right", 23% answered "too hard but useful", 10% answered "too hard to be useful", 7% answered "too easy but useful". 95% of students answered "yes" to the question "have you learned something new?" During the in-class sessions the pedagogic team observed that many students were engaged in the material, with many groups having animated discussions throughout the whole sessions. Written-in comments in course evaluations were all positive (e.g., "I've really enjoyed the simulations and feel like they were a nice simplified introduction to some complicated ideas", "Nice addition to the course especially for someone who's never seen Distributed Computing before," "I would rate this experience 100/100," "It was fun but also challenging," "It was engaging and educational"). In fact, 3 students from this course approached the third author of this paper asking whether they could volunteer to developed simulation software for future pedagogic activities. All three students are currently working on this project (for credit and/or as research assistants), each developing various simulators and helping the authors develop pedagogic content.

The questionnaires also asked students for feedback on the pedagogic content, which was overall very positive. For the more challenging Activity (IV), 83% of students would have liked to see more of a step-by-step structure in which steps are revealed after completion as opposed to a sequence of questions on a single page. We plan to split this activity into two separate activities and, for all activities, provide more dynamic Web pages with portions that can be collapsed or expanded by students.

Finally, about 20% of students had technical difficulties running our simulator, which was a surprise. This was caused by Windows 10 Home laptops on which virtualization was disabled and/or not accessible to Docker. Several of these problems were addressed, but we had to loan a few laptops to some students. In the future, we will likely host our Web app on a public server.

VI. Conclusion

We have presented pedagogic activities to teach HPC and PDC concepts and practices, in particular as relevant to CI computing. The key aspect of this work is that, using simulation, students can learn in a hands-on manner by experimenting with various application and platform scenarios. Our activities can be integrated in university courses as we have done ourselves in a 300-level undergraduate course. Results obtained via a pedagogic evaluation in that course, to be confirmed in subsequent evaluations, indicate that students used simulation effectively to achieve SLOs in a hands-on manner.

Ongoing work includes using student feedback collected during the above study to improve both the content and the organization of our pedagogic activities. A clear future direction is to develop new activities. First, we will develop activities similar to the ones currently available but for other SLOs that target fundamental topics (e.g., Amdahl's law). Second, still following the same model, we will develop activities for more advanced topics (e.g., scheduling, fault-tolerance, cloud computing). Third, we will develop activities that train students for particular technologies. For instance, we are planning a "batch scheduler activity" through which students interact with a simulated batch-scheduled cluster subject to simulated workloads, using, e.g., a subset of the SLURM interface. The simulation capabilities for all the above are already available in WRENCH. We encourage instructors to browse the (ever evolving) set of available pedagogic activities (available on the project's Web site [48]), and consider including some of these modules into actual courses.

A broader future plan includes conducting user studies, with small groups of students, that quantify the extent to which knowledge acquired through our simulation-driven pedagogic activities translates to proficiency when using a real-world infrastructures. Results from such studies will provide invaluable information to improve the content of our pedagogic activities.

Acknowledgments. This work is funded by NSF contracts #1642369 and #1642335, and partly funded by NSF contracts #1923539 and #1923621: "CyberTraining: Implementation: Small: Integrating core CI literacy and skills into university curricula via simulation-driven activities".

References

[1] G. Zarza, D. Lugones, D. Franco, and E. Luque, "An Innovative Teaching Strategy to Understand High-Performance Systems through Performance Evaluation," in *Proc. of International Comference on Computational Science*, 2012.

[2] A. Kozinov, E.and Shtanyuk, "Learning Parallel Computations with ParaLab," in *Proc. of the 1st Ural Workshop on Parallel, Distributed, and Cloud Computing for Young Scientists*, 2015, pp. 11–20.

[3] H. Casanova, M. Quinson, A. Legrand, and F. Suter, "SMPI Courseware: Teaching Distributed-Memory Computing with MPI in Simulation," in *Proc. of the Workshop on Education for High-Performance Computing (EduHPC)*, 2018.

[4] H. Casanova, S. Pandey, J. Oeth, R. Tanaka, F. Suter, and R. Ferreira da Silva, "WRENCH: A Framework for Simulating Workflow Management Systems," in *13th Workshop on Workflows in Support of Large-Scale Science (WORKS)*, 2018.

[5] 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 *Proc. of the International Conference on Cluster Computing*, 2013.

[6] S. Holt, A. Meaux, J. Roth, and D. Toth, "Making the One Cluster Per Student Method of Teaching Parallel Computing Financially Practical," *Journal of Computing Sciences in Colleges*, vol. 33, no. 4, pp. 106–113, 2018.

[7] R. Brown, J. Adams, S. Matthews, and E. Shoop, "Teaching Parallel and Distributed Computing with MPI on Raspberry Pi Clusters," in *Proc. of the 49th ACM Technical Symposium on Computer Science Education*, 2018, pp. 1054–1054.

[8] A. M. Pfalzgraf and J. A. Driscoll, "A low-cost computer cluster for high-performance computing education," in *Proc. of the International Conference on Electro/Information Technology*, 2014, pp. 362–366.

[9] K. Doucet and J. Zhang, "Learning Cluster Computing by Creating a Raspberry Pi Cluster," in *Proc. of the SouthEast Conference*, 2017, pp. 191–194.

[10] O. Abuzaghleh, K. Goldschmidt, Y. Elleithy, and J. Lee, "Implementing an Affordable High-performance Computing for Teaching-oriented Computer Science Curriculum," *ACM Transactions on Computing Education*, vol. 13, no. 1, pp. 3:1–3:14, 2013.

[11] C. Ivica, J. T. Riley, and C. Shubert, "StarHPC – Teaching parallel programming within elastic compute cloud," in *Proc. if the 31st International Conference on Information Technology Interfaces*, 2009, pp. 353–356.

[12] P. Marshall, M. Oberg, N. Rini, T. Voran, and M. Woitaszek, "Virtual Clusters for Hands-on Linux Cluster Construction Education," in *Proc. of the 11th LCI International Conference on High-Performance Clustered Computing*, 2010.

[13] N. A. Robison and T. J. Hacker, "Comparison of VM Deployment Methods for HPC Education," in *Proc. of the 1st Annual Conference on Research in Information Technology*, 2012, pp. 43–48.

[14] D. Johnson, S. Mason, and B. Hartpence, "Designing, Constructing and Implementing a Low-Cost Virtualization Cluster for Education," in *Proc. of International Multi-Conference on Society, Cybernetics and Informatics*, 2013.

[15] M. Tikir, M. Laurenzano, L. Carrington, and A. Snavely, "PSINS: An Open Source Event Tracer and Execution Simulator for MPI Applications," in *Proc. of the 15th International Euro-Par Conference on Parallel Processing*, ser. LNCS, no. 5704. Springer, Aug. 2009, pp. 135–148.

[16] T. Hoefler, T. Schneider, and A. Lumsdaine, "LogGOPSim - Simulating Large-Scale Applications in the LogGOPS Model," in *Proc. of the ACM Workshop on Large-Scale System and Application Performance*, Jun. 2010, pp. 597–604.

[17] G. Zheng, G. Kakulapati, and L. Kalé, "BigSim: A Parallel Simulator for Performance Prediction of Extremely Large Parallel Machines," in *Proc. of the 18th International Parallel and Distributed Processing Symposium (IPDPS)*, Apr. 2004.

[18] R. Bagrodia, E. Deelman, and T. Phan, "Parallel Simulation of Large-Scale Parallel Applications," *IJHPCA*, vol. 15, no. 1, pp. 3–12, 2001.

[19] W. H. Bell, D. G. Cameron, A. P. Millar, L. Capozza, K. Stockinger, and F. Zini, "OptorSim - A Grid Simulator for Studying Dynamic Data Replication Strategies," *IJHPCA*, vol. 17, no. 4, pp. 403–416, 2003.

[20] R. Buyya and M. Murshed, "GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing," *Concurrency and Computation: Practice and Experience*, vol. 14, no. 13-15, pp. 1175–1220, Dec. 2002.

[21] S. Ostermann, R. Prodan, and T. Fahringer, "Dynamic Cloud Provisioning for Scientific Grid Workflows," in *Proc. of the 11th ACM/IEEE International Conference on Grid Computing (Grid)*, 2010, pp. 97–104.

[22] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[23] A. Núñez, J. Vázquez-Poletti, A. Caminero, J. Carretero, and I. M. Llorente, "Design of a New Cloud Computing Simulation Platform," in *Proc. of the 11th International Conference on Computational Science and its Applications*, June 2011, pp. 582–593.

[24] G. Kecskemeti, "DISSECT-CF: A simulator to foster energy-aware scheduling in infrastructure clouds," *Simulation Modelling Practice and Theory*, vol. 58, no. 2, pp. 188–218, 2015.

[25] A. Montresor and M. Jelasity, "PeerSim: A Scalable P2P Simulator," in *Proc. of the 9th International Conference on Peer-to-Peer*, Sep. 2009, pp. 99–100.

[26] I. Baumgart, B. Heep, and S. Krause, "OverSim: A Flexible Overlay Network Simulation Framework," in *Proc. of the 10th IEEE Global Internet Symposium*. IEEE, May 2007, pp. 79–84.

[27] M. Taufer, A. Kerstens, T. Estrada, D. Flores, and P. J. Teller, "SimBA: A Discrete Event Simulator for Performance Prediction of Volunteer Computing Projects," in *Proc. of the 21st International Workshop on Principles of Advanced and Distributed Simulation*, 2007, pp. 189–197.

[28] T. Estrada, M. Taufer, K. Reed, and D. P. Anderson, "EmBOINC: An Emulator for Performance Analysis of BOINC Projects," in *Proc. of the Workshop on Large-Scale and Volatile Desktop Grids (PCGrid)*, 2009.

[29] D. Kondo, "SimBOINC: A Simulator for Desktop Grids and Volunteer Computing Systems," Available at http://simboinc.gforge.inria.fr/, 2007.

[30] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2899–2917, 2014.

[31] C. D. Carothers, D. Bauer, and S. Pearce, "ROSS: A High-Performance, Low Memory, Modular Time Warp System," in *Proc. of the 14th ACM/IEEE/SCS Workshop of Parallel on Distributed Simulation*, 2000, pp. 53–60.

[32] "The ns-3 Network Simulator," Available at http://www.nsnam.org.

[33] E. León, R. Riesen, A. Maccabe, and P. Bridges, "Instruction-Level Simulation of a Cluster at Scale," in *Proc. of the Intl. Conf. for High Performance Computing and Communications (SC)*, Nov. 2009.

[34] R. Fujimoto, "Parallel Discrete Event Simulation," *Commun. ACM*, vol. 33, no. 10, pp. 30–53, 1990.

[35] P. Velho, L. Mello Schnorr, H. Casanova, and A. Legrand, "On the Validity of Flow-level TCP Network Models for Grid and Cloud Simulations," *ACM Transactions on Modeling and Computer Simulation*, vol. 23, no. 4, 2013.

[36] P. Bedaride, A. Degomme, S. Genaud, A. Legrand, G. Markomanolis, M. Quinson, M. Stillwell, F. Suter, and B. Videau, "Toward Better Simulation of MPI Applications on Ethernet/TCP Networks," in *Prod. of the 4th Intl. Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*, 2013.

[37] P. Velho and A. Legrand, "Accuracy Study and Improvement of Network Simulation in the SimGrid Framework," in *Proc. of the 2nd Intl. Conf. on Simulation Tools and Techniques*, 2009.

[38] K. Fujiwara and H. Casanova, "Speed and Accuracy of Network Simulation in the SimGrid Framework," in *Proc. of the 1st Intl. Workshop on Network Simulation Tools*, 2007.

[39] A. Lèbre, A. Legrand, F. Suter, and P. Veyre, "Adding Storage Simulation Capacities to the SimGrid Toolkit: Concepts, Models, and API," in *Proc. of the 8th IEEE Intl. Symp. on Cluster Computing and the Grid*, 2015.

[40] E. Luque, R. Suppi, and J. Sorribes, "A Quantitative Approach for Teaching Parallel Computing," in *Proc. of the 23rd SIGCSE Technical Symposium on Computer Science Education*, 1992, pp. 286–298.

[41] A. N. Pears, "Using the DiST Simulator to Teach Parallel Computing Concepts," in *Proc. of the 1st International Forum on Parallel Computing Curricula*, 1995.

[42] B. Lester, *The Art of Parallel Programming*. Prentice Hall, 1993.

[43] J. Hartman and D. Sanders, "Teaching parallel processing using free resources," in *Proc. 26th IEEE Frontiers in Education Conference*, vol. 3, 1996, pp. 1483–1486.

[44] V. Gergel and A. Labutina, "ParaLab System for Investigating the Parallel Algorithms," in *Proc. of the Russia-Taiwan Symposium on Methods and Tools of Parallel Processing*, 2010, pp. 95–104.

[45] "The SimGrid Project," Available at http://simgrid.gforge.inria.fr/, 2015.

[46] G. Kecskemeti, S. Ostermann, and R. Prodan, "Fostering Energy-Awareness in Simulations Behind Scientific Workflow Management Systems," in *Proc. of the 7th IEEE/ACM Intl. Conf. on Utility and Cloud Computing*, 2014, pp. 29–38.

[47] "The WRENCH Project," http://wrench-project.org, 2019.

[48] "The WRENCH Pedagogic Modules," http://wrench-project.org/wrench-pedagogic-modules/, 2019.