Teaching Parallel and Distributed Computing Concepts in Simulation with WRENCH

Henri Casanova[a,*], Ryan Tanaka[a,b], William Koch[a], Rafael Ferreira da Silva[b]

[a] *University of Hawai'i at Mānoa, Information and Computer Sciences Dept., Honolulu, HI, USA*
[b] *University of Southern California, Information Sciences Institute, Marina del Rey, CA, USA*

Abstract

Teaching parallel and distributed computing topics in a hands-on manner is challenging, especially at introductory, undergraduate levels. Participation challenges arise due to the need to provide students with an appropriate compute platform, which is not always possible. Even if a platform is provided to students, not all relevant learning objectives can be achieved via hands-on learning on a single platform. In particular, it is typically not feasible to provide students with platform configurations representative of emerging and future cyberinfrastructure scenarios (e.g., highly distributed, heterogeneous platforms with large numbers of high-end compute nodes). To address these challenges, we have developed a set of pedagogic modules that can be integrated piecemeal into university courses. These modules include simulation-driven activities for students to experience relevant application and platform scenarios hands-on. These activities are supported by simulators developed using the WRENCH simulation framework. After motivating and describing our approach, we present and analyze results obtained from evaluations performed in two consecutive offerings of an undergraduate university course.

Keywords: Computer Science Education; Parallel and Distributed Computing

*Corresponding address: University of Hawai'i at Mānoa, Information and Computer Sciences Department, POST Building, Rm 317, 1680 East-West Road, Honolulu, HI, USA, 96822
Email addresses: henric@hawaii.edu (Henri Casanova), tanaka@isi.edu (Ryan Tanaka), kochwill@hawaii.edu (William Koch), rafsilva@isi.edu (Rafael Ferreira da Silva)

Education; Simulation.

1. Introduction

Teaching Parallel and Distributed Computing (PDC) topics is most effective when students have opportunities for hands-on learning. The common approach is to provide students with access to a PDC platform (hardware and software) on which they can develop and/or execute programs 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 a representative platform, which is not feasible at all institutions due to lack of resources. Even when an institution hosts suitable platforms, there may be no straightforward mechanism to use them, or even sizable subsets thereof, for education purposes (e.g., most platforms serve research communities and pedagogic use can disrupt production use).

Even when students are provided with a PDC platform for a course, there are *pedagogic challenges*. First, because students are only exposed to the particular configuration of that platform, many relevant scenarios are out of reach (e.g., different scales, different hardware specifications of compute nodes, different network interconnects, different software stacks). As a result, some PDC SLOs cannot be achieved in a hands-on manner. Second, students must be trained on platform usage mechanisms and policies. Some courses could deliberately devote a large portion of the syllabus to these mechanisms and policies. But in other courses this is pure overhead (e.g., introductory freshmen courses). Third, executing workloads on real-world platforms is not free: it requires time, electricity, and 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. Fourth, instructors typically must devote a significant amount of effort to managing students' use of the platform (e.g., platform monitoring and troubleshooting, interaction with platform administrators, managing scheduled and

unscheduled platform downtimes, managing competition for compute resources among students), which detracts from their pedagogic efforts.

The above challenges become steeper as the SLOs target more heterogeneous, more distributed, and/or large-scale platforms. Some SLOs can be achieved using a single multi-core computer, which typically is available to any student. But other SLOs require a 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, such as a batch scheduler. But even in this "easy" case the participation and pedagogic challenges are significant enough to have prompted the exploration of alternate approaches [1, 2, 3]. At the other end of the spectrum would be courses that attempt to teach principles and practices of cyberinfrastructure computing, i.e., the execution of application workloads via services deployed on large-scale, distributed environments with heterogeneous hardware and software stacks. An example would include the execution of a particular scientific application on a platform that comprises clouds, batch-scheduled HPC clusters, and data hosting services, all distributed over wide-area networks. Such 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. Besides, given the sheer number of relevant software and hardware configurations, it is unclear how a single deployment provided to students could be "representative". We argue that the aforementioned participation and pedagogic challenges are likely insurmountable in most university courses. And yet, it is critical to prepare students that will join the national scientific research and engineering workforce for a world in which cyberinfrastructure computing is the norm.

An alternative to using real-world platforms for teaching 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, such as their personal computer or a server. The pedagogic challenges are also addressed. Simulated executions can target arbitrary hardware and

software stack configurations, from basic (i.e., a single multi-core computer) to complex (i.e, a distributed, heterogeneous cyberinfrastructure deployment) scenarios. One can choose the level of details exposed to students regarding platform usage mechanisms. Finally, simulated executions are repeatable, and, provided an appropriate simulation framework is used, can be executed quickly and at negligible cost.

In this work [1], we present a set of pedagogic modules that target a range of PDC SLOs, and specifically SLOs that focus on quantitative reasoning and performance. Our objective is not to define a PDC curriculum, but rather to target SLOs that: (i) have been outlined by ongoing curriculum development efforts [5]; and (ii) can be achieved better via hands-on learning. Our modules have few pre-requisites and are designed to be integrated piecemeal in university courses, from freshman- to graduate-level courses, so as to complement existing course content. A current design choice is that these modules do not require any programming. But they include simulation-based pedagogic activities through which students acquire knowledge by experimenting with various application and platform scenarios in simulation. The simulations used in these activities provide metrics related to and visualizations of simulated application 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 learn independently, possibly with instructor-provided scaffolding. Finally, several modules include a "capstone" activity. These are case-studies in which students apply what they have learned so far to solve problems inspired by real-world scenarios. Our contributions are as follows:

- We describe and justify the use of the WRENCH [6, 7] simulation framework as a foundation for this work, and how it makes it possible to quickly develop and deliver our simulation-based pedagogic activities;
- We describe a set of pedagogic modules and the PDC SLOs they target; we detail some of the simulation-driven pedagogic activity within one of

[1] A preliminary version of this work appeared in EduHPC'19 [4]

these modules, and highlight how simulation is key to achieving SLOs in a hands-on manner;

- We discuss evaluation results obtained in the classroom for two consecutive offerings of the same 3rd-year undergraduate course at the University of Hawai'i at Mānoa; and
- We discuss and make recommendations for successful adoption of our pedagogic modules in university courses.

This paper is organized as follows. Section 2 discusses related work. Section 3 describes the WRENCH simulation framework and shows how we use it to implement and deliver simulation-driven pedagogic activities. Section 4 describes our pedagogic modules and the SLOs they target. Sections 5 and 6 present evaluation results obtained in the classroom, briefly explaining how these results have led us to evolve the content and structure of our modules. Section 7 discusses integration of our modules in university courses. Finally, Section 8 concludes with a summary of results and perspectives on future work.

2. Related Work

Several options have been proposed to address the challenge of providing students with PDC platforms for teaching purposes when such platforms are not readily available at their institutions. These options, which include building low-cost platforms [8, 9, 10, 11, 12, 13] and emulating platforms using virtualization and/or containers [14, 15, 16, 17], often do not provide platforms with capabilities, scales, and/or performance behaviors representative of production environments. In this work, instead, we rely on simulation to provide students with hands-on experience on arbitrary platform configurations.

Many simulation frameworks have been developed for PDC research and development, several of which could also be used for education. These frameworks span domains such as HPC [18, 19, 20, 21], Grid [22, 23, 24], Cloud [25, 26, 27], Peer-to-peer [28, 29], or Volunteer Computing [30, 31, 32]. Some frameworks have striven to be applicable across some or all or the above domains [33, 34].

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 long-running and/or 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 [35], on cycle-accurate CPU simulation [36] or emulation for computation). 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). Although these models are typically more scalable, they must be developed with care so that they are accurate [37]. One of the key intellectual contributions of the SimGrid project [33] is that it employs simulation models that are scalable (because macroscopic) and yet accurate (as shown in several validation studies [38, 37, 39, 40, 41]). As a result, SimGrid simulations can execute quickly on, say, a student's laptop, and yet yield accurate results even for complex simulation scenarios. SimGrid provides the core simulation technology for the WRENCH simulation framework [7], which is used in this work (see Section 3.

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 networks, since without simulation it can be extremely challenging to create hands-on learning opportunities in these domains. The use of simulation as a pedagogic tool is not as prevalent in the PDC domain, likely because with some effort it is possible to use real-world platforms for teaching purposes (albeit facing the challenges outlined in Section 1). Several works in the early 1990's proposed using simulation for the purpose of parallel computing education [42, 43, 44, 45]. 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 personal computers. Another recent work, Paralab teachware, is presented in [2], which uses the Paralab system [46] to teach parallel computing concepts and algorithms using simulation. From what information is provided in [46], Paralab implements naive simulation models that are not necessarily representative of real-world platforms, which is problematic. Another recent related work is that in [47], which does not point to re-usable pedagogic content but describes the use of a particular simulator for supporting distributed systems education. Unlike the above efforts, our work targets a broader set of PDC SLOs (from multi-core computers to distributed cyberinfrastructures, and for a range of application scenarios) and/or provides pedagogic modules that can be readily integrated into existing university courses.

3. Using WRENCH for Simulation-driven Education

3.1. The WRENCH Simulation Framework

The simulators that support our pedagogic activities must be developed based on some simulation framework. We identify four requirements for this framework:

Requirement #1: *Accuracy* – The framework's simulation models must have been the object of validation/invalidation studies. This is so that simulated executions are realistic when compared to real-world executions.

Requirement #2: *Scalability* – Simulations should have low time and space complexity. This is so that simulations can be executed quickly by students while engaged in hands-on learning.

Requirement #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 vanilla parallel program all the way to multi-site

cyberinfrastructure scenarios with diverse software and hardware stacks. This is so that we can target a broad set of SLOs.

Requirement #4: *Easy development* – The framework must provide high-level APIs that make it possible to implement simulators of complex scenarios with low effort. This is because many simulators must be implemented to support the many simulation-driven pedagogic activities we intend to develop.

As discussed in Section 2, many simulation frameworks have been developed for the PDC domain. One framework that meets the first three requirements above is SimGrid [33, 48]. Its simulation accuracy and scalability have been shown to be significantly better than that of its competitors [37, 33] (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 on wide-area networks (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 [49] 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 distributed cyberinfrastructure scenarios, is labor-intensive.

It is in part to meet requirement #4 that the WRENCH project [6, 7, 50] was initiated. WRENCH builds on SimGrid, so that simulations can be accurate and scalable, but provides high-level simulation abstractions. For instance, it 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 the project's web site [50] (version 1.7 of the software was released in September 2020). As a result, implementing simulators of complex scenarios can be done with minimal software engineering

efforts. This is demonstrated in [7], which describes how simulators of production workflow management systems that execute scientific workflow applications on diverse hardware/software stacks can be implemented with only a few hundred lines of code.

3.2. Implementing Simulation-driven Pedagogic Activities with WRENCH

Given the discussion in the previous section, in this work we use WRENCH for implementing then simulation-driven activities in our pedagogic modules. Each module 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. Many of these questions are to be answered by running simulations. Students answer the questions in two ways: (*i*) they discover a valid answer by running simulations to explore the space of simulated executions; and (*ii*) they come up with an answer through reasoning and validate/invalidate this answer by running simulations. Students invoke simulators through an interactive web interface. They must choose relevant input to the simulator (either guided by particular questions or independently), and inspect visual simulation output. Most visual displays are interactive in that students can click or hover on visual elements to be presented with more detailed information about the simulator's output.

The first version of our pedagogic modules, described in the next section, includes several simulation-driven activities, supported by four distinct WRENCH simulators. Some of these simulators implement relatively complex scenarios an yet were implemented with minimal software engineering effort. Counting C++ header and source files, but ignoring comments, these simulators are implemented in only 152, 226, 257, and 386 SLOCs.

To ensure ease-of-use and portability, all necessary software (web app, simulators, WRENCH and its dependencies) was packaged in a Docker container. When instructed to do so, students simply ran this container on their own computers. The container started a web server on a local port, allowing students to access a web app locally in the browser. The instructions given to students for

running simulators were 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 a particular Docker container. The only software requirement was that Docker be installed.

4. The WRENCH Pedagogic Modules

4.1. Relationship to the DPC Curriculum

The objective of this work is not to define a PDC curriculum. A community-driven, curriculum development effort has been ongoing as part of the NSF/IEEE-TCPP Curriculum Initiative [5]. The latest version of this curriculum was published in November 2020. It includes an extensive set of SLOs, each focused either on knowledge, on quantitative reasoning, or on programming skills. In this work, we develop pedagogic modules that target SLOs that focus on *knowledge* and *quantitative reasoning* as relevant to the performance of PDC applications and platforms. To enable hands-on learning for these SLOs, our modules provide students with the ability to run experiments in simulation. For now, our modules do not target programming skills nor skills pertaining to the practical use of PDC platforms (see further discussion in Section 7).

Table 1 lists the specific SLOs covered by the set of pedagogic modules that was delivered to students in undergraduate courses in Spring 2019 [51] and Fall 2019 [52] (see pedagogic evaluation results in Sections 5 and 6). The SLOs in this list directly map to SLOs in the NSF/IEEE-TCPP curriculum, but do not provide full coverage of the knowledge and quantitative reasoning topics in that curriculum. We selected our initial set of SLOs in a pragmatic, top-down fashion with the overall goal of teaching all necessarily skills to reason about the performance of a workflow application executed on a cluster of homogeneous multi-core compute nodes with both local and remote storage. Our latest set

of pedagogic modules, available at `https://eduwrench.org`, covers many more SLOs in the IEEE-TCPP PDC curriculum, and new modules are still under development. We maintain an up-to-date list of which modules target which SLOs in the IEEE-TCPP curriculum at `http://eduwrench.org/teachers`.

4.2. Initial Modules

Five pedagogic modules were developed and made available on-line at the beginning of 2019, and are still available on-line at [51]. These modules were designed to be performed as a sequence and target the following topics:

(I) Networking – In this module, students acquire essential networking concepts as they relate to the execution of programs on PDC platforms. These concepts include notions of latency, bandwidth, and topology, and how they are used to reason about and compute rough estimates of data transfer times.

(II) Workflows – This module introduces students to concepts necessary to understand the structure of workflow applications; to the steps necessary for executing these applications on a parallel/distributed platform; and to methods for reasoning about the performance of a workflow execution on a particular set of hardware resources. This and the following three modules use workflow applications as a compelling context for achieving fundamental PDC SLOs.

(III) Data Locality – Building on the previous two modules, this module introduces the general concept of data locality in distributed platforms. Students evaluate different workflow execution scenarios and experience first-hand the impact of poor and good data locality on performance.

(IV) Parallelism – Building on the first two modules as well, this module introduces students to notions pertaining to parallel computing (e.g., speedup, efficiency, 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 module, students use the concepts learned in previous modules to solve a problem based on a real-world scenario. Students are presented with a particular workflow application and a set of hardware resources. Given a budget, students must determine the best

Table 1: Curriculum map for our WRENCH pedagogic modules: (I) Networking; (II) workflows; (III) Data locality; (IV) Parallelism; and (V) Resource Provisioning. (E: Emerging, D: Developing, P: Proficient).

Student Learning Objective	**Module**				
	(I)	**(II)**	**(III)**	**(IV)**	**(V)**
Understand notions of network topology, bandwidth, and latency	E	E	D	-	P
Be able to reason about bottleneck links and bandwidth sharing	E	E	D	-	P
Be able to estimate data transfer times given a topology, for both sequential and concurrent data transfers	E	E	D	-	P
Be exposed to CI deployments that consist of storage and compute services deployed on wide-area networks of heterogeneous resources	-	E	E	D	P
Understand the structure of scientific workflow applications and the notion of task dependencies	-	E	D	D	P
Be able to estimate workflow execution time for a given CI deployment including computation and I/O	-	E	D	D	P
Understand notions of data locality and data proximity	E	-	D	-	P
Be able to quantify the impact of data locality on application execution time	E	-	D	-	P
Be familiar with multi-core compute nodes and how they can be aggregated to form compute clusters	-	-	-	E	D
Understand parallelism and parallel speedup	-	E	E	D	P
Understand the tradeoff between parallelism and core utilization	-	-	-	D	P
Understand how memory footprint constraints can limit parallelism	-	-	-	D	P
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
Be able to compare resource provisioning alternatives for a given application	-	-	-	E	D
Be able to make appropriate resource provisioning decisions for a given application given budget constraints	-	-	-	E	D

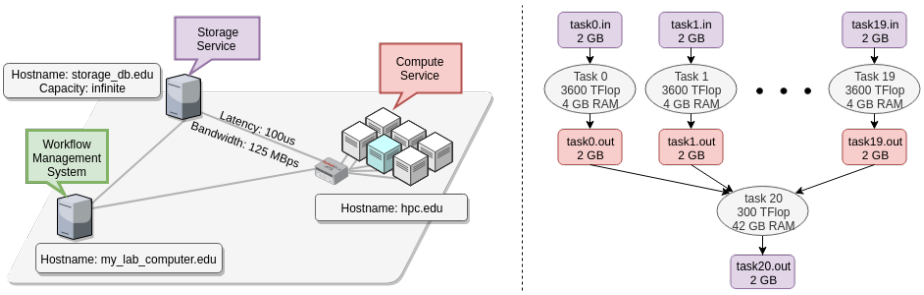

Figure 1: Module (IV): CI deployment (left-hand side) and workflow application (right-hand side).

way to spend this budget on hardware upgrades in order to minimize application execution time.

Due to lack of space we cannot describe all of the above five modules and the simulation-driven pedagogic activities therein. In the next section, instead, we provide an overview of Module (IV).

4.3. Overview of Module (IV)

Module (IV) focuses on parallelism and includes several simulation-driven pedagogic activities. The module first presents students with the platform 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 software agent (Workflow Management System) to orchestrate application executions using the hardware resources provided by the Storage and Compute Services.

The application to be executed on the above platform 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 a 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 module presents the above scenario to the students in the form of a narrative, with the depictions in Figure 1, and some review of the SLOs achieved in previous modules. 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? Use the simulator to check your answer". The module then evolves the scenario to cases in which the compute node at the Compute Service has 10, 15, or 20 cores. In each case, students quantify the performance gain, or lack thereof, and experience first-hand the interplay between core utilization and application performance using the simulation.

In a second phase of this module, 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 (manual) binary search. Other questions pertain to core utilization, getting to the notion of which hardware investments are worthwhile (this notion is fully explored in Module (V), which focuses on resource provisioning).

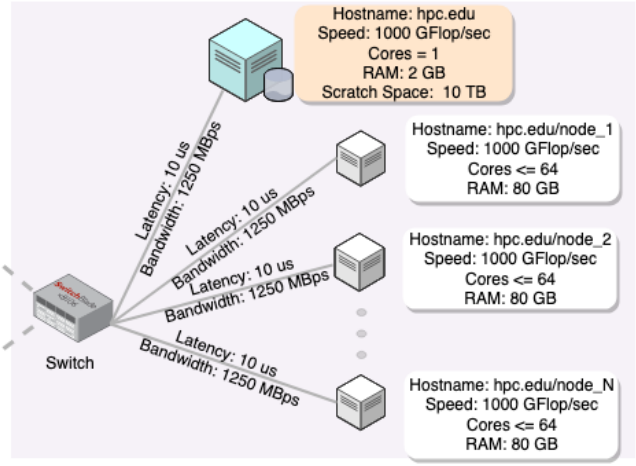

Figure 2: Module (IV): Simulated hardware specification of the cluster used by the Compute Service.

As described in Section 3.2, students invoke the simulator via a web interface, entering the number of compute nodes and the number of cores per compute node, specifying whether tasks require extra RAM, and then clicking the "Run Simulation" button. The simulation takes less than 1 second and updates the web page with several visualizations of the simulated execution. Figure 3 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 above 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 4. 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

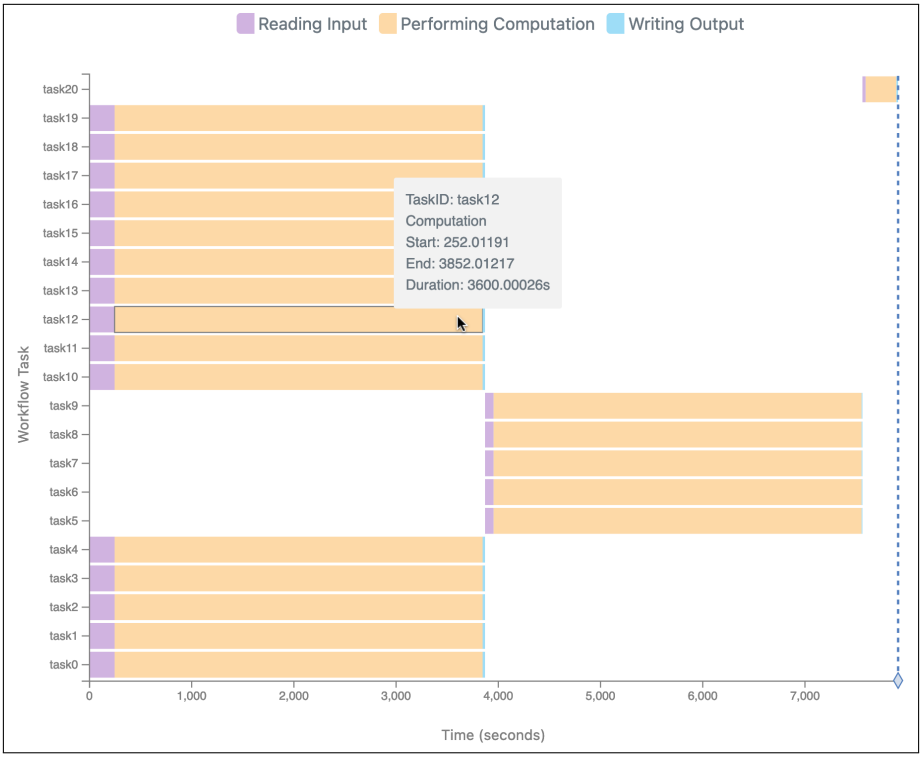

Figure 3: Module (IV): Sample Gantt chart of task executions for a platform with 3 6-core compute nodes.

again, a student can hover over any component of this visualization to gain more detailed information.

5. Spring 2019 Evaluation

5.1. Preliminary Evaluation

In February 2019, we performed a preliminary evaluation of our modules with three student participants, seniors in the B.S. in Computer Science program at the University of Hawai‘i at Mānoa (UHM) who volunteered due to their desire to learn more about PDC. They were given an assignment in which they completed modules (I) and (II) on their own. A week later, in a 1-hour session, the pedagogic team (the first and second author) together with the students

Figure 4: Module (IV): Sample core utilization time-line for a platform with 3 6-core compute nodes.

completed module (III), fielding questions. A week later, in another 1-hour session, the participants completed as many questions as possible in module (IV), with the pedagogic team providing scaffolding. Detailed feedback from these students led us to make many improvements to the pedagogic narrative and the simulation visualizations.

5.2. Classroom Evaluation

The first author regularly teaches the undergraduate Operating Systems course at UHM. The PDC topic was added to the course syllabus for the Spring 2019 semester as follows:

1. A 30-minute lecture on PDC to motivate the topic;
2. A reading assignment in which students complete modules (I) and (II) on

their own;

3. A 75-minute in-class interactive session during which the pedagogic team goes through module (III), soliciting participation from students and fielding questions;
4. A 75-minute in-class interactive session during which students, either individually or in groups of up to 3, start on module (IV) with scaffolding provided by the pedagogic team;
5. A homework assignment in which students answer four of the last (and most challenging) questions of module (IV), with solutions to preceding questions provided to them; and
6. Three problems on the final exam (covering SLOs #1 to #13 in Table 1), worth 10% of the final exam grade.

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;
- Anonymized time-stamped trace data of student interactions with the simulation web app;
- Anonymized correlation between number of simulator invocations and grade for relevant final exam questions.

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 PDC platform. However, as explained in Section 1, this is difficult and typically not done (especially at the undergraduate level). This is why we advocate the use of simulation in the first place. Besides, given the steep pedagogic challenges of using a real-world platform, the

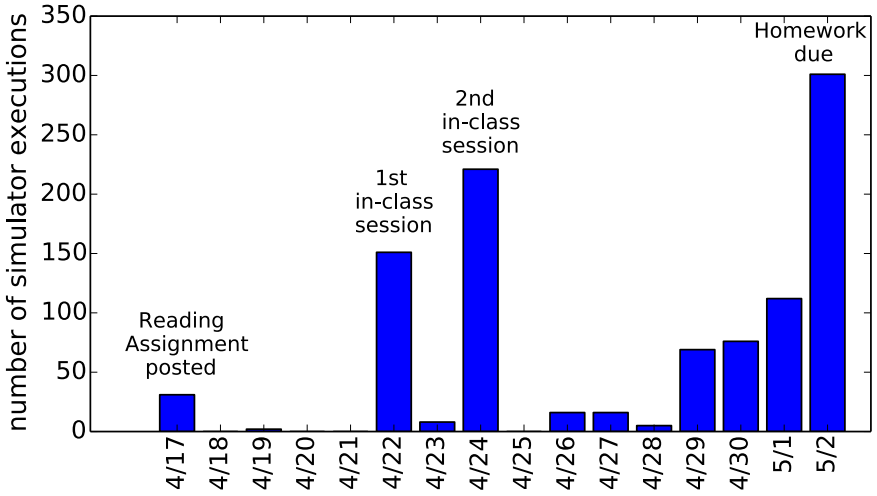

Figure 5: Daily numbers of simulations executed by students (Spring 2019).

control group students would receive a vastly inferior educational experience as part of their undergraduate education (which, from a human subject research perspective, goes against the principle of justice).

5.3. Results

In this section, we analyze the collected data to answer three key questions.

Question #1: Are students using the simulation?

45 of the 55 students who took the final exam ran simulations. Daily simulation execution totals are shown in Figure 5. Overall 1,008 simulations were executed in a 16-day period, with expected peak days when the reading assignment was posted, during the two in-class sessions, and leading up the homework assignment's due date. Note that during the first 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 2: Correlation between numbers of simulations executed and average grades on PDC-focused final exam questions (Spring 2019).

# of simulations	# of students	grade average	grade coeff. var.
0	10	67.6	14.8%
1-10	14	88.8	27.6%
11-20	13	99.8	18.1%
21-30	6	81.0	21.7%
31+	12	75.5	37.0%

Module (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 out of curiosity. However, students who struggle with the material could also be running many simulations haphazardly.

Question #2: Are students learning the material?

Table 2 shows how grades obtained by students on PDC-related final exam questions correlate with the number of simulations that students have executed. For each range of number of simulations executed the table shows the corresponding number of students, the average grade, and the coefficient of variance (i.e., the standard deviation divided by the mean) as a percentage. During in-class sessions students often worked in groups, meaning that for some students lower numbers of simulations were recorded. We find that students who have executed no simulations perform worse than other students. Interestingly, students who ran a lot of simulations also do not perform well, scoring a C on average (although a few performed well, hence the higher coefficient of variance).

This may be because most of these students struggled with the material. We find that students who ran between 11 and 20 simulations performed best, with almost a perfect score on average. 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 Module (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. Based on our own observations and experience, we believe that these good results are due to our simulation-driven approach. But due to the lack a control group, we cannot draw a definite conclusion regarding the reason for these results.

Question #3: Are students having a positive experience?
Student feedback on their experience collected via anonymous questionnaires was unanimously positive, with written-in comments such as "I've really enjoyed the simulations and feel like they were a nice simplified introduction to some complicated ideas" or "It was engaging and educational". In terms of perceived difficult of the material, 60% of students deemed it "just right", 23% "too hard but useful", 10% "too hard to be useful", and 7% answered "too easy but useful".

5.4. Updates to the Pedagogic Modules

Anonymous questionnaires asked students for feedback on the pedagogic content. Based on this feedback we re-structured some of the longer modules, and added more examples and explanatory figures throughout. We also made one significant structural change. The pedagogic modules used workflow applications as a motivating context. Because workflows are ubiquitous and compelling (i.e., "workflows were used for computations that led to the discovery of gravitational waves") they are a good class of application to use for motivating the

material. But for teaching basic PDC concepts the use of workflows adds complexity to the pedagogic content, which can be overwhelming for some students. We thus added three preliminary modules, with three new simulators, for targeting basic PDC SLOs without mentioning workflows. Finally, to our surprise, several students had technical difficulties running Docker on their Windows 10 Home computers. So instead of enforcing the use of Docker, we opted for also hosting our web app on a static server to which students can authenticate for running simulations remotely. These updated pedagogic modules, which were used in the subsequent semester, are available at [52].

6. Fall 2019 Evaluation

We used our (updated) pedagogic modules in the Fall 2019 offering of the Operating Systems course at UHM, and performed another evaluation.

6.1. Classroom Evaluation

The PDC topic added to the course syllabus was identical to that in the previous semester (see Section 5.2), and the pedagogic team (the first and third authors of this paper) performed the evaluation by collecting the following qualitative and quantitative data:

- One anonymous post questionnaire about experience and perceived learning;
- Anonymized time-stamped trace data of student interactions with the simulation web app;
- Anonymized correlation between number of simulator invocations and grade for relevant final exam questions.

6.2. Results

In this section, we revisit the same questions as in Section 5.3.

Question #1: Are students using the simulation?

58 students took the final exam and 52 of these students ran simulations. This

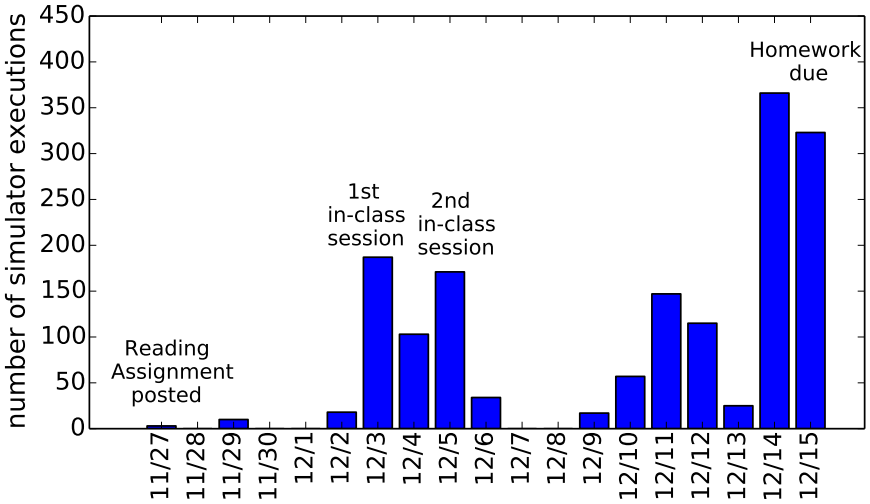

Figure 6: Daily numbers of simulations executed by students.

Table 3: Correlation between numbers of simulations executed and average grades on PDC-focused final exam questions (Fall 2019).

# of simulations	# of students	grade average	grade coeff. var.
0	6	47.5	18.4%
1-10	11	69.4	15.3%
11-20	16	86.5	19.4%
21-30	4	75.3	20.4%
31+	21	81.1	48.3%

represents 89.6% of students, vs. 81% in Spring 2019. Daily simulation execution totals are shown in Figure 6. Overall 1,576 simulations were executed by 52 students in a 19-day period. On average, each student executed 30 simulations, vs. 22 in Spring 2019 (the updated version of the pedagogic modules did not explicitly require that students run more simulations than in their Spring 2019 versions. The pattern in Figure 6 is similar to that in Figure 5. One difference is that more students ran simulations in between the two in-class interactive sessions, which may indicate a higher level of engagement.

Question #2: Are students learning the material?

Table 3, like Table 2 shows final exam grade results. As for Spring 2019, we find that students who ran few simulations did not do well, and that students who ran between 11 and 20 simulations did best on average. A different pattern is seen for students who ran large number of simulations. Examining the data, we find that, unlike in Spring 2019, about half of the students who ran 31+ simulations score 100% on the final exam but the other half scored relatively poorly, leading to a good average overall (but a high coefficient of variance). So although we again see students who might have run many simulations haphazardly, we now see students who have mastered the material and still ran many simulations (perhaps out of curiosity).

On average, students did not do as well as in the previous semester. In the previous semester the grade point average for the PDC questions on the final exam is 83.8%, but for this semester it is 76.5%. To determine whether the reason for this difference was due to the updates to our pedagogic modules, we computed average grades for the final exam **excluding PDC questions**. We find that the Spring 2019 students overall also did much better than the Fall 2019 students (82.4% vs. 75.1% average grade), just as they did for the PDC questions. Such differences in levels of performance between classes in subsequent semesters is not uncommon.

Question #3: Are students having a positive experience?

Students were asked for feedback on their experience via an anonymous questionnaire. 31 of the 58 students in the class completed the questionnaire. Table 4 shows student answers to key questions. The main take-away is that most students found the material useful and are interested in learning more about PDC. Importantly, 80% of students felt that using simulation was very useful for learning, and the remaining 20% felt that it was somewhat useful. Just like in the previous semester, written-in comments were unanimously positive, such as "I think that the simulations were really helpful in understanding the concepts and fundamentals in action" or "I liked the simulation. It was a nice

Table 4: Highlighted post questionnaire questions and student answers (Fall 2019).

How easy/difficult is the material?	very easy:	**4**
	somewhat easy:	**5**
	neither:	**8**
	somewhat difficult:	**12**
	very difficult:	**2**
How useful/not useful is the material?	very useful:	**22**
	somewhat useful:	**9**
	not useful at all:	**0**
To what extent did the module help you learn new things?	to a great extent:	**22**
	to some extent:	**9**
	not at all:	**0**
Are you interested in learning more about Parallel and Distributed Computing?	yes:	**27**
	no:	**4**
How useful is simulation to learn about Parallel and Distributed computing?	very useful:	**25**
	somewhat useful:	**6**
	not at all useful:	**0**
	cannot tell:	**0**
How valuable is the overall learning experience in the module?	very valuable:	**24**
	somewhat valuable:	**7**
	not valuable at all:	**0**

addition to visually see as well as check my work"

6.3. Updates to the Pedagogic Modules

Based on feedback from the Fall 2019 student, we integrated the simulation app in the same web page as the pedagogic narrative and we have added practice questions throughout the pedagogic narratives. We have also created new modules to cover more PDC SLOS (e.g., Amdahl's law, disk I/O, client-server, coordinator-worker), raising the number of simulators to 10. Given the large increase in pedagogic content, we have also added an easily accessible glossary

of terms.

This improved and expanded set of pedagogic modules is available on-line at `https://eduwrench.org`. For reference purposes we still maintain the sites that contain the versions of our pedagogic modules used in Spring 2019 [51] and Fall 2019 [52].

7. Adoption in University Courses

Adopting our pedagogic modules in existing university courses requires no technology buy-in since all pedagogic material, including the simulation-driven activities, are available in the browser. The instructor can simply point students to the pages hosted at `https://eduwrench.org`. Students can then go through the material, simulation-driven activities, and practice questions on the page, whether on their own or during a lecture with instructor-provided scaffolding. Finally, the instructor can include non-practice questions provided on that site in homework assignments and exams (solutions to these questions are provided to instructors upon request). Instructors can select which modules, or subset thereof, to include in their courses based on their listed SLOs. Up-to-date coverage of the NSF/IEEE-TCPP curriculum's SLOs is listed at `http://eduwrench.org/teachers`.

Our current modules focus on knowledge and quantitative reasoning SLOs, and do not target PDC programming SLOs. Furthermore, because they require no PDC platforms, they do not target SLOs that focus on the practical skills that are necessary for using these platforms (note that the NSF/IEEE-TCPP curriculum does not include such SLOs). As a result, our modules are not sufficient for full coverage of PDC topics as there are PDC skills that can only be acquired by developing programs and running them on real-world platforms. This said, our modules can be adopted by instructors for (i) injecting PDC content into courses without requiring programming; or (ii) enhancing teaching in courses that already include PDC content and in particular that include PDC programming. We discuss both adoption paths hereafter.

The undergraduate Operating Systems course for which we performed our pedagogic evaluations originally had no PDC content besides a mere introduction to the concepts of threads and mutual exclusion. Our modules made it straightforward to inject PDC content into the course. We included coverage of many of our modules, but it should be straightforward to include only a few or even a single module into an existing course. Some modules are prerequisite to others, but backward references are included so that students who need to review prerequisite material can do so (also, our web site includes an easily accessible glossary of terms). We are currently working with instructors at our and other institutions to include one or two of our modules in lower division courses. A challenge may be to motivate the new PDC content, especially since our modules only focus on knowledge and quantitative SLOs, albeit with hands-on experience with simulation. In our case, we provided such motivation via a 1/2-hour lecture as a preamble to going through our pedagogic modules. This lecture introduced PDC and motivated it in the overall context of the course. Based on student feedback, we found that this was amply sufficient. In addition, many students were aware that PDC is a crucial topic that is not sufficiently covered in the standard curriculum, which provided inherent motivation.

Our pedagogic modules can also be used to enhance teaching in courses that already include coverage of PDC, including courses in which students engage in PDC programming. Rather than replacing hands-on learning via programming, our modules provide complementary hands-on learning opportunities via simulation. One option is for the instructor to design programming activities related to our pedagogic modules, something we did not do in the Operating Systems course due to lack of time in the semester. For instance, our second module focuses on multi-core architectures, and one of the topics it covers is load-balancing of independent tasks onto cores. It would be straightforward to design a relevant programming assignment in which students develop a multi-threaded program, using whatever programming language, and run it on their own multi-core computers. The material in our module would (i) provide the initial concepts necessary for students to embark on the programming assign-

ments for a particular application on their computers; and (ii) allow students to experiment, in simulation, with a broader range of application and computer scenarios.

Given the above, we hypothesize that our modules can both facilitate and reinforce learning achieved via programming activities. To obtain some preliminary insight into whether this hypothesis holds, we have used several of our modules in an upper-division Concurrent and High-Performance Programming elective course at UHM (Fall 2020, 30 students). This course is programming-heavy as students develop many multi-threaded programs (in Java and C), as part of a semester-long project. Students learn about fundamental parallel computing concepts such as Amdahl's law, overlap of I/O and computation in multi-threaded programs, task graphs, mixed task- and data-parallelism, etc. The programming assignments, because of the time constraints and because they were designed around a specific real-world application, do not allow students to experience hands-on all the ways in which the above concepts impact parallel program executions. To complement students' experience we have used our pedagogic modules, and in particular their simulation-driven activities. Although we did not perform a systematic evaluation, informal student feedback as well as the level of achievement of SLOs indicate that combining PDC programming with in-simulation experiments was very effective. Based on this successful, if preliminary, experience, we have started adding suggested programming activities to our pedagogic modules.

8. Conclusion

We have presented pedagogic modules for teaching PDC concepts and practices. The key aspect of this work is that these modules include simulation-based pedagogic activities through which students can learn in a hands-on manner by experimenting with various application and platform scenarios. These modules have few pre-requisites and, in their current form, do not require any programming. These modules can be used to inject PDC content into university

courses. We have done so ourselves in two consecutive offerings of an undergraduate Operating Systems course, for which we have conducted pedagogic valuations. Evaluation results show that students were engaged and actively used the simulation, learned the material, and had an overall very positive experience. Our modules can also be used to enhance coverage of PDC content in university courses. We have done so for an undergraduate elective Concurrent and High-Performance Programming course, in which the modules' simulation-driven activities were used to complement PDC programming assignments.

A clear future direction is to develop more pedagogic modules, and in particular modules that target more advanced PDC topics. These include modules for teaching students cyberinfrastructure concepts and their underlying technologies. For instance, we have begun developing a "batch scheduler module" in which students interact with a simulated batch-scheduled cluster subject to simulated background load, using a subset of the Slurm [53] interface. WRENCH already provides the simulation capabilities necessary for easily implementing this and many other kinds of more advanced simulation scenarios. Broader future plans include conducting user studies, with small groups of students, in order to quantify the extent to which knowledge acquired through our simulation-driven pedagogic activities translates to proficiency when using real-world systems.

We encourage instructors to browse the growing set of available pedagogic modules at `https://eduwrench.org`, provide feedback, and consider adoption of these modules in their courses.

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

References

[1] G. Zarza, D. Lugones, D. Franco, 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, 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] R. Tanaka, R. Ferreira da Silva, H. Casanova, Teaching parallel and distributed computing concepts in simulation with wrench, in: Workshop on Education for High-Performance Computing (EduHPC), 2019. doi:10.1109/EduHPC49559.2019.00006.

[5] NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing, http://www.cs.gsu.edu/~tcpp/curriculum/ (2020).

[6] H. Casanova, S. Pandey, J. Oeth, R. Tanaka, F. Suter, R. Ferreira da Silva, WRENCH: A Framework for Simulating Workflow Management Systems, in: 13th Workshop on Workflows in Support of Large-Scale Science (WORKS'18), 2018, pp. 74–85. doi:10.1109/WORKS.2018.00013.

[7] H. Casanova, R. Ferreira da Silva, R. Tanaka, S. Pandey, G. Jethwani, W. Koch, S. Albrecht, J. Oeth, F. Suter, Developing accurate and scalable simulators of production workflow management systems with wrench, Future Generation Computer Systems 112 (2020) 162–175. doi:10.1016/j.future.2020.05.030.

[8] M. Ludin, A. Weeden, J. Houchins, S. Thompson, C. Peck, I. Babic, K. Muterspaw, E. Sergienko, LittleFe: The high performance computing education appliance, in: Proc. of the International Conference on Cluster Computing, 2013.

[9] S. Holt, A. Meaux, J. Roth, D. Toth, Making the One Cluster Per Student Method of Teaching Parallel Computing Financially Practical, Journal of Computing Sciences in Colleges 33 (4) (2018) 106–113.

[10] R. Brown, J. Adams, S. Matthews, 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.

[11] A. M. Pfalzgraf, 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.

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

[13] O. Abuzaghleh, K. Goldschmidt, Y. Elleithy, J. Lee, Implementing an Affordable High-performance Computing for Teaching-oriented Computer Science Curriculum, ACM Transactions on Computing Education 13 (1) (2013) 3:1–3:14.

[14] C. Ivica, J. T. Riley, 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.

[15] P. Marshall, M. Oberg, N. Rini, T. Voran, 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.

[16] N. A. Robison, 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.

[17] D. Johnson, S. Mason, 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.

[18] M. Tikir, M. Laurenzano, L. Carrington, 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, no. 5704 in LNCS, Springer, 2009, pp. 135–148.

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

[20] G. Zheng, G. Kakulapati, 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), 2004.

[21] R. Bagrodia, E. Deelman, T. Phan, Parallel Simulation of Large-Scale Parallel Applications, IJHPCA 15 (1) (2001) 3–12.

[22] W. H. Bell, D. G. Cameron, A. P. Millar, L. Capozza, K. Stockinger, F. Zini, OptorSim - A Grid Simulator for Studying Dynamic Data Replication Strategies, IJHPCA 17 (4) (2003) 403–416.

[23] R. Buyya, 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 14 (13-15) (2002) 1175–1220.

[24] S. Ostermann, R. Prodan, 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.

[25] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, R. Buyya, CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, Software: Practice and Experience 41 (1) (2011) 23–50.

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

[27] G. Kecskemeti, DISSECT-CF: A simulator to foster energy-aware scheduling in infrastructure clouds, Simulation Modelling Practice and Theory 58 (2) (2015) 188–218. doi:10.1016/j.simpat.2015.05.009.

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

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

[30] M. Taufer, A. Kerstens, T. Estrada, D. Flores, 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.

[31] T. Estrada, M. Taufer, K. Reed, 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.

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

[33] H. Casanova, A. Giersch, A. Legrand, M. Quinson, F. Suter, Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms, Journal of Parallel and Distributed Computing 74 (10) (2014) 2899–2917.

[34] C. D. Carothers, D. Bauer, 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.

[35] The ns-3 Network Simulator, http://www.nsnam.org.

[36] E. León, R. Riesen, A. Maccabe, P. Bridges, Instruction-Level Simulation of a Cluster at Scale, in: Proc. of the Intl. Conf. for High Performance Computing and Communications (SC), IEEE, 2009, pp. 1–12. doi:10.1145/1654059.1654063.

[37] P. Velho, L. Mello Schnorr, H. Casanova, A. Legrand, On the Validity of Flow-level TCP Network Models for Grid and Cloud Simulations, ACM Transactions on Modeling and Computer Simulation 23 (4) (2013).

[38] P. Bedaride, A. Degomme, S. Genaud, A. Legrand, G. Markomanolis, M. Quinson, M. Stillwell, F. Suter, 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, Springer, 2013, pp. 158–181. doi:10.1007/978-3-319-10214-6_8.

[39] P. Velho, 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, pp. 1–10. doi:10.4108/ICST.SIMUTOOLS2009.5592.

[40] K. Fujiwara, H. Casanova, Speed and Accuracy of Network Simulation in the SimGrid Framework, in: Proc. of the 1st Intl. Workshop on Network Simulation Tools, 2007, pp. 1–10.

[41] A. Lèbre, A. Legrand, F. Suter, 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, IEEE, 2015, pp. 251–260. doi:10.1109/CCGrid.2015.134.

[42] E. Luque, R. Suppi, 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.

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

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

[45] J. Hartman, D. Sanders, Teaching parallel processing using free resources, in: Proc. 26th IEEE Frontiers in Education Conference, Vol. 3, 1996, pp. 1483–1486.

[46] V. Gergel, 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.

[47] A. Nunez, C. Manoso, A. P. de Madrid, S. Pickin, SIMCAN: A simulator to improve the learning of distributed and high-performance computing systems in engineering degrees, Computer Applications in Engineering Education 27 (5) (2019) 1126–1138.

[48] The SimGrid Project, https://simgrid.org (2021).

[49] G. Kecskemeti, S. Ostermann, 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.

[50] The WRENCH Project, `http://wrench-project.org` (2021).

[51] Spring 2019 WRENCH Pedagogic Activities, `https://henricasanova.github.io/wrench-pedagogic-modules/` (2021).

[52] Fall 2019 WRENCH Pedagogic Activities, `http://wrench-project.org/wrench-pedagogic-modules/` (2021).

[53] A. B. Yoo, M. A. Jette, M. Grondona, Slurm: Simple Linux Utility for Resource Management, in: Proc. 9th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP), 2003, pp. 44–60.