A Framework for the Evaluation of Parallel and Distributed Computing Educational Resources

David W. Brown
Computer Science and Information Systems
Elmhurst College
Elmhurst, IL, United States
david.brown@elmhurst.edu

Vitaly Ford
Computer Science and Math
Arcadia University
Glenside, PA, United States
fordv@arcadia.edu

Sheikh K. Ghafoor
Computer Science
Tennessee Tech University
Cookeville, TN, United States
sghafoor@tntech.edu

***Abstract*—This paper proposes a classification scheme for categorization of PDC educational resources. We have also proposed an evaluation framework for assessing the PDC resources. Under the proposed framework, each resource type has a set of criteria and an associated score. A PDC resource will obtain a score if evaluated under our proposed framework that is the sum of the scores of the criteria that the resource satisfies. The evaluation of whether a resource met a criterion is subjective. We have also presented our evaluation of PDC educational resources appropriate for CS1, CS2 (Computer Science 1 and 2), and DS/A (Data Structures and Algorithms) available on the web using our proposed framework.**

***Index Terms*—PDC Educational Resource, Resource Evaluation Framework, CS1, CS2, DS/A**

I. Introduction

Parallel and Distributed Computing (PDC) has become pervasive, from supercomputers and server farms containing multi-core CPUs and GPUs, to individual PCs, laptops, and mobile devices. Even casual users of computers now depend on parallel processing. Therefore, it is important for every computing professional (and especially every programmer) to understand how parallelism and distributed computing affect problem solving. It is essential for educators to impart a range of PDC and HPC knowledge and skills at multiple levels within the educational fabric woven by Computer Science (CS), Computer Engineering (CE), and related computational curricula including data science. The need for including PDC in undergraduate computing curricula has drawn the attention of the CS education community. NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing – Core Topics for Undergraduates [36] released a curriculum recommendation that contains what PDC topics should be included in first two years of computing curriculum. PDC has been included as a required knowledge area in ACM/IEEE 2013 curriculum recommendation [2]. Accreditation Board for Engineering and Technology (ABET) [1] has also required exposure to PDC for accreditation of computing programs. Additionally, federal funding agencies have been funding efforts in developing PDC curriculum and instructional materials in recent years.

There are many challenges in integrating PDC in early CS classes. The major challenges are: 1) lack of trained faculty in PDC, 2) lack of instructional resources, and 3) lack of awareness among CS faculty. A recent survey by the authors of this paper in 2017 among the CS faculty (135 responded out of about 4,000 CS faculty) indicated the lack of proper instructional materials as one of the major challenges of integrating PDC. Also, a recent NSF funded workshop that was participated by thirty thought leaders from Academia, Industry, National Lab, and Government Agencies identified a lack of appropriate instructional resources among one of the major impediments to effectively integrate PDC in undergraduate computing classes.

Many educators individually or in groups have been working in creating curriculum and instructional materials, organizing workshops, symposiums, and integrating PDC in undergraduate CS classes. Lots of PDC educational resources have been developed/created in the last decade by these individuals and groups. The Center for Parallel and Distributed Computing Curriculum Development and Educational Resources (CDER) [36] has been a pioneer in curriculum development and creating a repository for PDC educational resources. There are different types of resources (reading materials, slides, syllabi, lab assignments, etc.) available on the web. These resources differ with respect to quality, maturity, completeness, usability, and adoptability. The community need more instructional resources, but there is also a need for systemic categorization, cataloging, and evaluations of the existing resources. Such categorization and evaluation of the available resources will help the community.

This work is a preliminary attempt of such an effort, and it focuses on investigating what resources are available that will help faculty to integrate PDC in undergraduate curriculum. We want to examine what are the different types of resources available, how easily adaptable they are, what PDC concepts these resources cover at what level, and the quality of the resources. While we were searching for resources, we limited our focus on early CS classes (CS1, CS2, and data structures and algorithms). One of the challenges we faced was that there is no catalog which would help to find such resources with the exception of the commonly known CDER courseware [23], and the Internet search engines do not return any meaningful results. We conducted our search based on our knowledge of the community efforts and looking through the recent publications on PDC education. Due to time limitation, our search was primarily focused on the USA and by no means

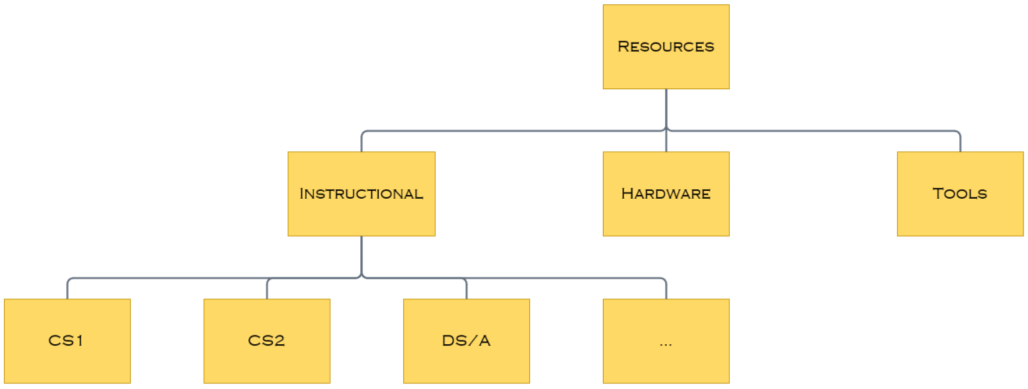


Fig. 1. Resource Classification Tree

it was exhaustive. We are sure that we have missed many resources.

In evaluating the PDC resources, we have proposed a resource classification scheme that can be used to classify a PDC educational resource. We have also proposed a resource evaluation framework that can be used to assess a PDC educational resource and obtain a score indicative of the quality of the resource. The proposed classification and evaluation framework is a preliminary attempt that needs to be examined, deliberated, and improved further in the future.

II. Evaluation Criteria and Classifications

We categorize the available PDC educational resources into three major classes: those designed to aid in the classroom instruction (III-A), those that provide access to hardware platforms for hands-on PDC exercises (III-B), and software tools, libraries and games that can help with the instruction of PDC (III-C).

Instructional: These are resources that can help in teaching PDC concepts. These are materials that can be easily combined with existing current course structures and materials with little or no effort. Example of these types resources are lecture slides, programming assignments, reading handout, course syllabi, questions banks, and unplugged activities. Some of these resources are meant to be used by the instructor only and some may be by both instructor as well students. Some resources may have a different version for instructors and students. These resources can be further categorized based on what level of undergraduate classes (CS1, CS2, networking etc.) they are targeting or what PDC concept(s) they are covering.

Hardware: These are generally parallel distributed computing platforms where students and instructors can perform hands-on PDC exercises. example of these could be a multi-core server, a multi-node cluster with or without GPU capability, Hadoop cluster, etc. The main characteristic of these types of resources is the that they are freely available and easily obtainable to students and instructors.

Tools: These types of resources encompass i) software such as a Java OpenMP thread library that can be used to write parallel programs, ii) games that can be used by the students or instructors to understand or illustrate PDC concepts, iii) libraries that can help to visualize PDC concepts or a parallel program, iv) auto grader and or assessment software that can help instructors and students. We are primarily focusing on tools that are free or easily affordable.

The available PDC educational resources needs to be evaluated based on some criteria that would be an additional help for an instructor to determine whether a particular material is suitable for their class. We have proposed a resource evaluation framework for different resources in our classification scheme. There are several criteria with an assigned score for each type of resource. The total score of a resource is the sum of the scores of all criteria met by the resource. In our proposed framework, instructional resources have 8 criteria, hardware resources have 5 criteria, and tools have 6 criteria. These criteria and the assigned score are our preliminary proposal that requires further deliberation and refinement.

Following are the criteria for different types of resources and their associated score:

Instructional - inspired by the Likert scale, these scores vary on the scale of 5–10: 5 being the least applicable for a specified criteria, and 10 being the most applicable.

1) IC1 - All supplied documentation is complete (**score 10**) or simply in outline form (**score 5**)
2) IC2 - All material is current (**score 10**) or based on older technology (**score 5**)
3) IC3 - Supplied documents are easily editable (**score 10**) or do they exist only in a static form (**score 5**)
4) IC4 - No special requirements are necessary for adoption (**score 10**) or are there significant prerequisites (**score 5**)
5) IC5 - Solutions to any problem sets are supplied (**score 10**) or is it the responsibility of the adopter to develop them (**score 5**)
6) IC6 - Material is self-explanatory (**score 10**) or is specialized knowledge required (**score 5**)
7) IC7 - Separate resources are supplied for student or instructor as appropriate (**score 10**) or are some component resources missing (**score 5**)
8) IC8 - Setup and configuration instructions are complete

and easy to follow (**score 10**) or are they vague or possibly misleading (**score 5**)

In a similar manner, for material that was determined to be *hardware*-related, five criteria were chosen, while for *tools and games* six standards were selected. These criteria are:

Tools and Games - the scores vary on the binary scale of 0–1: 0 being false and 1 being true.

1) TC1 - Detailed installation and configuration instructions are available (**score 1**)
2) TC2 - Program is simple to install and configure (**score 1**)
3) TC3 - Executable is available for Windows Operating Systems (**score 1**)
4) TC4 - Executable is available for Apple OS/X Operating Systems (**score 1**)
5) TC5 - Executable is available for Linux Operating Systems (**score 1**)
6) TC6 - Sample code examples are available (**score 1**)

Hardware - the scores vary on the binary scale of 0–1: 0 being false and 1 being true.

1) HC1 - Detailed instructions on how to request access to the hardware are provided (**score 1**)
2) HC2 - A template for the proposal request is available if needed (**score 1**)
3) HC3 - The resource has an intuitive and easy-to-use interface (**score 1**)
4) HC4 - The resource availability is not limited (**score 1**)
5) HC5 - Student access is allowed (**score 1**)

Each resource was assigned a score based on the summation of the individual points allocated through examination of how well they fulfilled the criteria. Based on this score, we were able to rank the resources below, comparing their ease and effectiveness as a teaching aid. Note, that while many resources may end up with the same score, not all ties are equal. For this reason, a separate evaluation write-up is included for each resource indicating our belief on its efficacy.

III. Resources

Based on the classification and evaluation criteria described in the previous section we have evaluated existing PDC educational resources that are available on the web. As mentioned in the introduction, we have limited our search to resources that are appropriate for CS1, CS2, and DS/A only and our search was not exhaustive. Following are the description of resources that we have found and evaluated for different types of resources.

A. Instructional Resources

High quality classroom instructional materials for PDC education are essential to help educators effectively deliver the major concepts to their students. In this section, we discuss the resources that are available for educators and could aid in the introduction of PDC into the core curriculum of their classes. Figure 2 indicates the scores each received in our classification. In addition, explanations of our evaluation are included below.

1) CMU 15-418 Course – ***Level: DS/A, Score: 42****:* Carnegie Mellon University's course [8] contains lectures and exercises on Parallel Computer Architecture and Programming topics. The lecture material discusses multi-core processors, parallel programming models and basics, GPU and CUDA programming, performance evaluation, cache coherence, memory consistency, synchronization, scheduling, heterogeneous parallelism, parallel computing with Spark, and more. The self-check and programming exercises help students to reinforce the material by analyzing parallel program performance, rendering in CUDA, developing a simple elastic web server, and implementing multiple programs: a task queue on a multi-core CPU, data-parallel histogram, cache coherence, atomic operations, workload balance, fine-grained locking, transactional memory, and routing in an interconnected network. Additionally, they posted all their student final projects with video presentations, which can give ideas for future student projects in parallel computing classes.

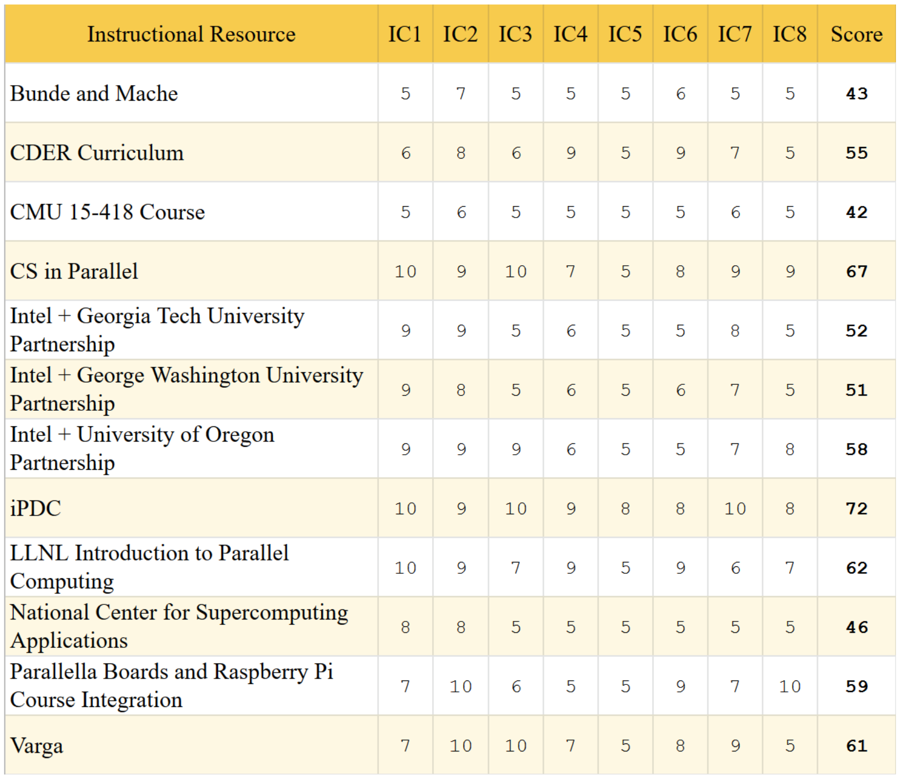
This course's prerequisites include computer systems and strong knowledge of C/C++ and, therefore, this material cannot be used in the introductory courses. However, the slides and assignments are very detailed (with no solutions) and it could be used for developing higher-level PDC coursework and topics.

2) CSinParallel – ***Level: CS1 CS2, DS/A, Score: 67****:* The CSinParallel [33] project is a library of teaching materials for integrating parallel computing into computer science courses. The materials consist of concept-driven teaching materials, readings, homework, lab exercises, evaluation strategies, and supplementary pedagogical advice. The modules are written for a wide range of languages (C, C++, Java, Python, Go, Scheme) and technologies (OpenMP [30], MPI [18], CUDA). In addition, the modules are categorized by the course where they can be incorporated into: from computer architecture and database systems to graphics, algorithms, and intro to computer science for non-majors.

Some notable examples of the exercises include Pandemic Exemplar, Monte Carlo Simulations, UK Traffic Incidents, Flixster Network Data, LastFM Song Dataset Analysis, Drug Design Exemplar, Map-Reduce, and Parallel Sorting. The exercises are self-explanatory but it is difficult to judge how much time to allocate for each one of them as they drastically vary in length, complexity, and special software needed to be installed. Some exercises have lengthy instructions, others are shorter and simpler to follow.

3) Higher-level Languages and Activity-based Laboratories – ***Level: CS2, DS/A, Score: 43****:* Bunde and Mache [7] created a project for teaching parallel programming concepts based on higher-level languages (Chapel [9], Java's Executor framework, Habanero Java, CUDA, and Cilk) and compelling examples. They developed thread modules and wrappers to simplify task parallel programming, educator's toolbox for CUDA, and high-level parallel programming using Chapel and Scratch.

This project only has tutorials for those special languages and frameworks (Chapel, Habanero, Cilk) with CUDA tutorial



Instructional Resource	IC1	IC2	IC3	IC4	IC5	IC6	IC7	IC8	Score
Bunde and Mache	5	7	5	5	5	6	5	5	**43**
CDER Curriculum	6	8	6	9	5	9	7	5	**55**
CMU 15-418 Course	5	6	5	5	5	5	6	5	**42**
CS in Parallel	10	9	10	7	5	8	9	9	**67**
Intel + Georgia Tech University Partnership	9	9	5	6	5	5	8	5	**52**
Intel + George Washington University Partnership	9	8	5	6	5	6	7	5	**51**
Intel + University of Oregon Partnership	9	9	9	6	5	5	7	8	**58**
iPDC	10	9	10	9	8	8	10	8	**72**
LLNL Introduction to Parallel Computing	10	9	7	9	5	9	6	7	**62**
National Center for Supercomputing Applications	8	8	5	5	5	5	5	5	**46**
Parallella Boards and Raspberry Pi Course Integration	7	10	6	5	5	9	7	10	**59**
Varga	7	10	10	7	5	8	9	5	**61**

Fig. 2. Instructional Resources Criteria and Scores

leading to a broken link. The tutorials comprehensively cover the specifics of those languages and they provide short programming exercises with solutions. However, no teaching or student material is present and little information is given about utilization of those resources in the classroom for instructional purposes.

*4) Intel + George Washington University Partnership – **Level: DS/A, Score: 52**:* Intel has established a parallel computing center at the George Washington University [39]. They created a set of lectures (PDF slides) covering a wide range of topics, from architectures and performance to a unified parallel C, Chapel [9], OpenMP [30], and MPI [18]. No other special education material was produced and some of these slides have links to resources that are only available to George Washington University students. However, these lectures could be used as a reference guide to the other resources.

*5) Intel + Georgia Tech University Partnership – **Level: DS/A, Score: 52**:* Another course on Intel processors was offered at Georgia Tech University [38]. They developed undergraduate and graduate presentations as well as undergraduate assignments on C++, including matrix multiplication performance evaluation using OpenMP [30], communication latency and bandwidth measurement using MPI [18], Conway's Game of Life using MPI, simulation of soft particles collections, and movie ratings prediction using Netflix data set and MPI. The slides cover such topics as code optimization (cache, BLAS, SIMD), measuring and reporting performance, interconnects, algorithms for collective communication, parallel data analysis, DGEMV and DGEMM algorithms, and partitioning problems in PDC.

The presentations are made to cover specialized topics and, therefore, are not coherent to be used as a lecture material. However, they can serve as a good reference when those subjects are being covered in the classroom. The assignments are descriptive for students to understand what is required from them but there is no teaching guide and solutions that would help educators prepare.

*6) Intel + University of Oregon Partnership – **Level: DS/A, Score: 58**:* The University of Oregon has partnered with Intel to develop lectures (slides) and hands-on lab modules for the parallel computing course in C/C++ [27]. Their project covers such topics as parallel computer architecture, performance, map, collective, data reorganization, stencil, recurrence, fork-join, pipeline, message passing, parallel algorithms, multicore, and GPU. The labs contain C++ starter code on the topics of a map, collective, data reorganization, stencil, fork/join, and pipeline. They also include a sample proposal for the final group term project.

The lectures comprehensively cover the introduced topics. The lab instructions are presented as a few slides per lab and in most cases, the labs are self-explanatory, given the starter code. It is not clear if the solutions are available. Some of the code is quite complex so students are expected to be comfortable in C++ programming. Only a few labs among those that are available could be used in the introductory courses; others are more suitable for very specialized PDC courses.

*7) iPDC – **Level: CS1, CS2, Score: 72**:* Through their NSF grant on creating a workshop series as part of a Summer Institute for Integrating Parallel and Distributed Computing in Introductory Programming Classes, Ghafoor et al. [14] developed a sequence of PDC educational modules for CS0, CS1, and CS2 courses. Their work contains both *unplugged* and *plugged* modules. The unplugged modules focus on demonstrating PDC concepts without the usage of technology, applicable for any course: finding the youngest student in class, card sorting, M&M sorting, etc. The plugged modules are provided in two languages: C++ and Java. They are categorized as CS1 and/or CS2 and presented as assignments on the topics of parallel min-max, pi estimation, parallel sum, parallel sort, matrix multiplication, parallel image processing, cache awareness, and data races.

iPDC assignments have detailed self-explanatory instructions for students to carry out the tasks at hand. Teacher and student versions of the materials are combined in one document that is available in both PDF and Word formats for editing when needed. The assignments require standard programming software to be installed. Solutions are available upon request via email. Some of the Java activities are based on Pyjama [34] that will require spending some time to get familiar with and learn how to configure it.

*8) Lectures for TCPP Book – **Level: CS1, CS2, DS/A, Score: 61**:* Varga has published a set of lectures online [40] related to topics in parallel and distributed computing as described in the book "Topics in Parallel and Distributed Computing: Introducing Concurrency in Undergraduate Courses" [25]. The lectures include presentations and coding examples. There is no other instructional material available in this repository but it is a good resource to use when following the topics from the above-mentioned book in your classroom.

*9) LLNL Introduction to Parallel Computing – **Level: CS1, CS2, DS/A, Score: 62**:* Barney [5] at the Lawrence Livermore National Laboratory has developed and published online an Introduction to Parallel Computing web-course, covering a variety of PDC topics, such as major concepts (Flynn's taxonomy, general PDC terminology), parallel computer memory architecture (shared and distributed memory), parallel programming models (shared memory model, threads, message passing model, data parallel model, SPMD, MPMP), and designing parallel programs (partitioning, communications, synchronization, data dependencies, load balancing, I/O, performance analysis, and granularity).

Barney created a few examples of parallel array processing, pi calculation, heat equation, and 1D wave equation. A unique characteristic of this resource is that it contains lots of pictures facilitating better understanding of PDC concepts. This material can be used as a guide for educators to prepare for lectures and students to learn PDC concepts through supported visuals.

*10) National Center for Supercomputing Applications – **Level: DS/A, Score: 46**:* The National Center for Supercomputing Applications provides a variety of free courses on PDC topics [22]. The Center also prepares researchers to use XSEDE [11] resources. The courses include but are not limited to simulating PetaFLOPS supercomputers, introduction to MPI [18], introduction to OpenMP [30], introduction to multi-core performance, multilevel parallel programming, performance tuning for clusters, and using the XSEDE user portal.

The materials introduced in these courses can be used as a reference guide to support PDC lectures in the classroom. Besides those courses, this resource does not provide any educational materials and there is more research-oriented than educational-based support.

*11) Parallella Boards and Raspberry Pi Course Integration – **Level: CS2, DS/A, Score: 59**:* Matthews [19], [21] integrated the Parallella [31] (18-core single board computer) into an undergraduate PDC course. Some of the major PDC modules that were covered in the course included `pthreads`, OpenMP, Epiphany, and MPI. A total of four assignments were given to students in C language. The students had to conduct benchmarking tests, comparing parallel and serial programs. The most challenging part of Parallella course integration was working with the Epiphany documentation. As a result of that experience, Matthews decided to replace Parallella boards with Raspberry Pi, OpenMP, and CSinParallel [33] to teach PDC to educators [20] and their students.

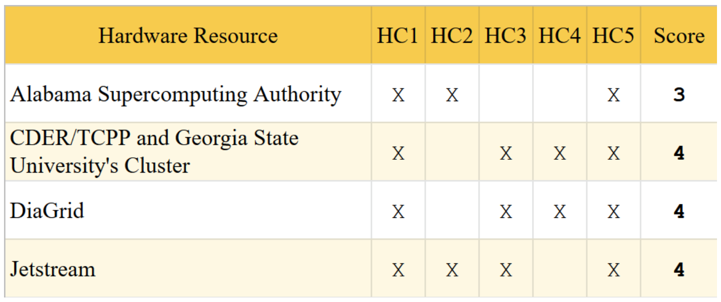
This resource includes handouts, worksheets, and slides on only a few topics, such as integration (computing the area under a curve), drug design exemplar, and Raspberry Pi laptop connection. The available material is concise, editable, and it does not require too much time to learn, making it suitable for introductory courses.

B. Hardware Resources

A real concern for many professors attempting to integrate PDC into the curriculum is the lack of High Performance Computing (HPC) resources that are available to them at their home institutions. While HPC resources are becoming more affordable, they still remain outside the reach of most small institutions. Depending on the researchers' location, a few free options are available, and several others exist at low or no cost thresholds, saving the need to purchase and maintain expensive systems.

In this section, we discuss the hardware resources that are available for educators and could be beneficial to use in the classroom. Figure 3 indicates the scores each received in our classification.

1) Alabama Supercomputing Authority – score 3: On a more limited scope, educators in the state of Alabama can take advantage of resources provided by the Alabama Supercomputing Authority (https://www.asc.edu/). This group was created in 1989 by the state to provide resources to develop and operate supercomputing and telecommunications systems throughout the state. The Dense Memory Cluster (DMC) at the center has 2,360 CPU cores and 14 terabytes of distributed memory. The DMC also contains a high performance GPFS storage cluster, which has 93 terabytes of high performance



Hardware Resource	HC1	HC2	HC3	HC4	HC5	Score
Alabama Supercomputing Authority	X	X			X	**3**
CDER/TCPP and Georgia State University's Cluster	X		X	X	X	**4**
DiaGrid	X		X	X	X	**4**
Jetstream	X	X	X		X	**4**

Fig. 3. Hardware Resources Criteria and Scores

storage accessible from each node. Access to the centers resources are free to all educators throughout the state.

2) DiaGrid – score 4: DiaGrid [10] provides free instant access to high performance, high throughput computing to users via their browsers. It provides "bioinformatics (BLAST, BEAST, etc), simulations (GROMACS, NAMD, CryoEM, etc.), visualization tools (ParaView, PyMol, etc) and general purpose computational tools such as IDEs with plugins for submitting jobs to HPC resources (e.g., RStudio, Spyder for Python, SubmitR for serial, parallel and parameter sweeps jobs, etc.)"

3) Jetstream – score 4: Jetstream project [35] is available for free as part of an NSF ACI-1445604. It provides a user-friendly interface to allocate computing and storage resources via trial, startup, education, and research allocations available through XSEDE [37]. The trial provides 1,000 CPU hours, startup and education 50,000-100,000 CPU hours, and research 1,000,000 CPU hours and more. Educators can use those high performance computing resources for both teaching and researching.

4) CDER cluster at Georgia State University's Cluster – score 4: NSF/IEEE-TCPP allows instructors and students to use a heterogeneous cluster for PDC education maintained at their center at Georgia State University [24]. At the time of writing, the cluster has 28 nodes featuring 656 cores, 1 TB RAM, 4 NVIDIA V100s, SLURM scheduler, Apache Spark. Access to the cluster can be requested directly from the centers website.

C. Tools

As important as instructional resources and hardware are for the education of PDC topics, specialized tools and software as well as simulators can be invaluable in capturing students' attention. The resources in this section may be found in Figure 4 while explanations of our evaluations are included below.

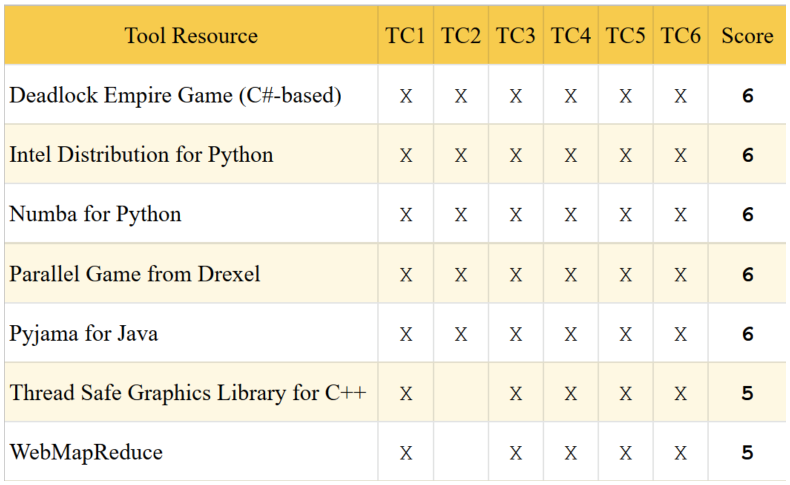
1) Deadlock Empire Game – score 6: Deadlock Empire [15], [32] is a C# based web-game teaching about certain PDC principles including scheduling, concurrency, synchronization, locks, deadlocks, semaphores, producer-consumer, and critical sections. It does not require any installation and can be run from any browser, allowing students to analyze the task-parallel code straight on the website. The game is structured in a unique way: the players act as schedulers whose objective is to exploit flaws in the programs, making them crash or otherwise malfunction.

The game is simple to understand and no interactive coding is required; students would just need to analyze the short pieces of code and attempt to schedule the tasks in a manner that would intentionally crash the program. Even though the game is C#-based, students with knowledge of any other language would be able to quickly understand what is happening in front of them. No teaching instructions are provided but the game is self-explanatory. Being available on GitHub makes this game customizable but for only experienced programmers because there is no instructions on how to do it. However, the authors welcome contributions and suggestions for new levels and challenges that could be requested via email or pull requests on GitHub [16].

2) Intel Distribution for Python – score 6: Intel Distribution for Python [17] accelerates python computing for almost all computational packages out-of-the-box with minimal changes to the original Python code. It can be used on any OS and installed from conda, PIP, APT, YUM, and Docker. This tool accelerates and scales the compute-intensive Python packages NumPy, SciPy, and mpi4py as well as allows for faster machine learning and high performance computing on Python.

3) Numba for Python – score 6: Numba [26] makes python code faster and parallel with just a few directives. Five minutes is enough to learn it. The speed of computations of Python code with Numba is comparable with C and Fortran. It allows for parallelizing loops (as in OpenMP), SIMD vectorization, and GPU acceleration (both NVIDIA's CUDA and AMD's ROCm drivers) straight from Python. It would work in any OS and would be useful for all classes where parallel algorithms make sense to be run.

Numba represents a great solution for the global interpreter lock problem in Python as it compiles the code before running it. As a result, it could be a perfect fit for both introductory as well as more advanced courses like data mining due to Python being a simple to understand and versatile language. A good example of the performance evaluation of Numba is [12] where researchers compare the performance of Go, C++, and Python on N-Queens problem (both sequential and parallel algorithms).



Tool Resource	TC1	TC2	TC3	TC4	TC5	TC6	Score
Deadlock Empire Game (C#-based)	X	X	X	X	X	X	**6**
Intel Distribution for Python	X	X	X	X	X	X	**6**
Numba for Python	X	X	X	X	X	X	**6**
Parallel Game from Drexel	X	X	X	X	X	X	**6**
Pyjama for Java	X	X	X	X	X	X	**6**
Thread Safe Graphics Library for C++	X		X	X	X	X	**5**
WebMapReduce	X		X	X	X	X	**5**

Fig. 4. Tool Resources Criteria and Scores

4) Parallel Game – score 6: Parallel Game is a 2D game developed by Santiago Ontañón et al. [28], [29] where players interactively manipulate arrows by placing semaphores and signals in order to achieve the map objectives. This game visually teaches students about concurrency, deadlocks, multiple tasks, threads, speedup, critical section, signals, and race conditions. This game is constantly updated, open source, and available for any OS. It is also simple to install and has an intuitive interface allowing students to learn the PDC concepts without any special training. There are no special educational materials for teachers besides the topics that students would learn when playing this game.

In this game, players design a synchronization mechanism allowing multiple threads to complete their tasks, avoiding deadlocks and starvation. This resource does not provide any extra educational materials but the game itself is easy to follow and understand thanks to the tooltips and explanations that are available to players.

5) Pyjama for Java – score 6: Pyjama [34] is a parallel computing library that allows utilizing OpenMP-like GUI-aware directives in Java, making it possible to solve problems in parallel on Android or any other Java application. It has not been updated since 2017 and not all OpenMP-like directives are supported. It is simple to use given that the parallel directives are similar to the standard OpenMP and students would not need to change much in their code to make Java applications become parallel. Pyjama installation could take some effort to make sure that it works well with the current version of Java and is included in the IDE or as a command-line tool to compile the programs. It can run on any OS where Java is installed.

6) Thread Safe Graphics Library for C++ – score 5: A Thread Safe Graphics Library (TSGL) in C++ [3] allows drawing on a 2D canvas from multiple threads at near real-time. In [4], Adams et al. demonstrated how visualization can depict the realities of the parallel processing of images. TSGL works straight out of the box with parallel loops and can be easily integrated into the existing code that processes images, visualizing the parallel nature of image processing with just a few extra lines of C++ code. Installation documentation is available for Windows, Linux, and MacOS. TSGL abstracts away the complexities of race conditions, locks, and deadlocks and helps students focus on the visual concepts of parallel processing, making this library suitable for all course levels.

7) WebMapReduce – score 5: WebMapReduce [13] is an easy-to-use Hadoop Map-Reduce algorithm implemented as a web interface where students learn about task and data parallelism, scalability, speedup, and fault-tolerance. It can be deployed as a standalone server on a local machine or in the cloud through their existing AWS EC2 instance. In combination with AWS Educate program, this would be the ideal way to deploy WebMapReduce with minimal technical administration efforts.

WebMapReduce has extensive documentation with examples in such languages as C/C++/C#/Java/Python. It is a part of CSinParallel project [6] with learning goals, teaching materials/notes/tips. Some notable examples of the assignments are poker hands, movie data analysis, flight data analysis, Google N-Grams dataset analysis, and efficiently merging tables. All examples are concise and simple to understand. WebMapReduce contains teaching materials in Word, PDF, and Latex. This tool would be a good fit for the introductory-level courses.

IV. Conclusion

In this paper, we attempted to create instructional, hardware, and tool criteria as well as a classification framework that can be utilized and modified as necessary for evaluating the PDC resources. We did not seek providing an exhaustive listing of available resources but rather looked at multiple materials as an example of how the framework could assist educators considering to integrate PDC concepts into their curriculum as recommended by ACM/IEEE/ABET.

While gathering information for this paper, the researchers were pleased to note that the number of assets available to aid in the inclusion of PDC topics in the CS coursework continues to grow. However, further development of PDC resources is still required to encourage broader adoption. Some examples are:

1) Tools to help debug and score student programs are necessary to help instructors grade student submissions. An automatic grader to compile and test the applications students create has not been developed. Without this grader, instructors who are often unfamiliar with the topics are challenged to successfully gauge how effective the programs are at taking advantage of system resources.
2) Tools to help visualize what is actually occurring in parallel programming will help instructors showcase the material in an easy to understand format for their students. Some visualization suites for PDC do exist, most notably TSGL [3] and the Parallel Game [29]. However, TSGL only provides limited functionality and is not yet intuitive enough for many instructors to create their own visualizations of the material they wish to highlight, whereas the Parallel Game abstractly teaches the PDC concepts without any coding.

V. Acknowledgements

The authors would like to thank the PDC community for developing the resources described in this paper.

References

[1] ABET. Computing accreditation commission, computer science program criteria. https://www.abet.org/accreditation/accreditation-criteria.

[2] ACM-IEEE. Computer science curricula 2013. https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf.

[3] Joel Adams, Elizabeth Koning, and Ian Adams. Github-hosted repository of the thread safe graphics library. https://github.com/Calvin-CS/TSGL.

[4] Joel C Adams, Patrick A Crain, and Christopher P Dilley. Seeing multi-threaded behavior using tsgl. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 972–977. IEEE, 2016.

[5] Blaise Barney. Introduction to parallel computing. https://computing.llnl.gov/tutorials/parallel/_comp.

[6] Richard Brown and Libby Shoop. Map-reduce computing for introductory students using webmapreduce. https://csinparallel.org/csinparallel/modules/IntroWMR.html.

[7] David Bunde and Jens Mache. Teaching parallel computing with higher-level languages and activity-based laboratories. http://faculty.knox.edu/dbunde/parallel.html.

[8] Carnegie Mellon University Course 15-418/618. Parallel computer architecture and programming. http://15418.courses.cs.cmu.edu/spring2015.

[9] Cray. The chapel parallel programming language. https://chapel-lang.org/education.html.

[10] DiaGrid. Get instant access to high throughput, high performance, campus, and national supercomputers via browser. https://diagrid.org/home.

[11] Extreme Science and Engineering Discovery Environment (XSEDE). High performance computing and clusters for research. https://portal.xsede.org/training/overview.

[12] Pascal Fua and Krzysztof Lis. Comparing python, go, and c++ on the n-queens problem. *arXiv preprint arXiv:2001.02491*, 2020.

[13] Patrick Garrity, Timothy Yates, Richard Brown, and Elizabeth Shoop. Webmapreduce: an accessible and adaptable tool for teaching map-reduce computing. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 183–188, 2011.

[14] Sheikh Ghafoor, Michael Rogers, and David Brown. Cybertraining: Cdl: ipdc - summer institute for integrating parallel and distributed computing in introductory programming classes. https://www.csc.tntech.edu/pdcincs/index.php/ipdc-modules/.

[15] Petr Hudeček and Michal Pokorný. The deadlock empire. http://deadlockempire.github.io.

[16] Petr Hudeček and Michal Pokorný. Github-hosted repository of the deadlock empire game. https://github.com/deadlockempire/deadlockempire.github.io.

[17] Intel and Anaconda. Intel distribution for python. https://software.intel.com/en-us/distribution-for-python.

[18] Lawrence Livermore National Laboratory. Message passing interface (mpi). https://computing.llnl.gov/tutorials/mpi.

[19] Suzanne Matthews. Teaching materials by suzanne j. matthews. http://www.suzannejmatthews.com/teaching.html.

[20] Suzanne Matthews, Joel Adams, Richard Brown, and Elizabeth Shoop. Exploring parallel computing with openmp on the raspberry pi. https://csinparallel.org/csinparallel/raspberry_pi.html.

[21] Suzanne J Matthews. Teaching with parallella: A first look in an undergraduate parallel computing course. *Journal of Computing Sciences in Colleges*, 31(3):18–27, 2016.

[22] National Center for Supercomputing Applications. Cyberinfrastructure tutor. https://www.citutor.org.

[23] NSF/IEEE-TCPP. Courseware management. https://tcpp.cs.gsu.edu/curriculum/?q=courseware_management.

[24] NSF/IEEE-TCPP. Heterogeneous cder cluster for pdc education. http://tcpp.cs.gsu.edu/curriculum/?q=node/21615.

[25] NSF/IEEE-TCPP. Topics in parallel and distributed computing: Introducing concurrency in undergraduate courses. https://tcpp.cs.gsu.edu/curriculum/?q=cedr_book.

[26] Numba. Open source jit compiler that translates a subset of python and numpy code into fast machine code. https://numba.pydata.org.

[27] University of Oregon. Intel parallel computing center - parallel curriculum development. http://ipcc.cs.uoregon.edu/curriculum.html.

[28] Santiago Ontañón, Jichen Zhu, Brian K Smith, Bruce Char, Evan Freed, Anushay Furqan, Michael Howard, Anna Nguyen, Justin Patterson, and Josep Valls-Vargas. Designing visual metaphors for an educational game for parallel programming. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 2818–2824, 2017.

[29] Santiago Ontañón, Brian Smith, Jichen Zhu, and Bruce Char. Learning parallel programming concepts through an adaptive game. http://digm.drexel.edu/pxl/parallel-programming.

[30] OpenMP. The openmp api specification for parallel programming. https://www.openmp.org.

[31] Parallella. 18-core credit card sized computer. https://www.parallella.org/.

[32] Michael Pokorny. The deadlock empire, 2016.

[33] Elizabeth Shoop and Richard Brown. Parallel computing in the computer science curriculum. https://csinparallel.org/.

[34] Oliver Sinnen and Nasser Giacaman. Pyjama: an active research project aiming at supporting openmp-like directives for java. http://parallel.auckland.ac.nz/ParallelIT/PJ_About.html.

[35] Craig A Stewart, Timothy M Cockerill, Ian Foster, David Hancock, Nirav Merchant, Edwin Skidmore, Daniel Stanzione, James Taylor, Steven Tuecke, George Turner, et al. Jetstream: a self-provisioned, scalable science and engineering cloud environment. In *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*, page 29. ACM, 2015.

[36] The NSF/IEEE-TCPP Curriculum Committee. Nsf/ieee-tcpp curriculum initiative on parallel and distributed computing - core topics for undergraduates, 2012. http://tcpp.cs.gsu.edu/curriculum/?q=system/files/NSF-TCPP-curriculum-version1.pdf.

[37] John Towns, Timothy Cockerill, Maytal Dahan, Ian Foster, Kelly Gaither, Andrew Grimshaw, Victor Hazlewood, Scott Lathrop, Dave Lifka, Gregory D Peterson, et al. Xsede: Accelerating scientific discovery. *Computing in Science & Engineering*, 16(5):62–74, 2014.

[38] Georgia Tech University. A course on high performance scientific computing on intel processors. https://www.cc.gatech.edu/∼echow/ipcc/hpc-course/.

[39] The George Washington University. Intel parallel computing center - ipcc lectures. https://ipcc.seas.gwu.edu/lectures.

[40] Ervin Varga. Material for lectures related to topics in parallel and distributed computing. https://github.com/evarga/parallel-computing-lectures.