PDCunplugged: A Free Repository of Unplugged Parallel & Distributed Computing Activities

Suzanne J. Matthews
Department of Electrical Engineering & Computer Science
United States Military Academy
West Point, USA
suzanne.matthews@westpoint.edu

***Abstract*—Integrating parallel and distributed computing (PDC) topics in core computing courses is a topic of increasing interest for educators. However, there is a question of how best to introduce PDC to undergraduates. Several educators have proposed the use of "unplugged activities", such as role-playing dramatizations and analogies, to introduce PDC concepts. Yet, unplugged activities for PDC are widely-scattered and often difficult to find, making it challenging for educators to create and incorporate unplugged interventions in their classrooms. The *PDCunplugged* project seeks to rectify these issues by providing a free repository where educators can find and share unplugged activities related to PDC. The existing curation contains nearly forty unique unplugged activities collected from thirty years of the PDC literature and from all over the Internet, and maps each activity to relevant CS2013 PDC knowledge units and TCPP PDC topic areas. Learn more about the project at pdcunplugged.org.**

***Index Terms*—parallel and distributed computing, education, unplugged, activity, repository**

I. Introduction

Computer science educators increasingly look to integrate parallel and distributed computing (PDC) topics into their undergraduate computing courses. In 2012, the NSF/IEEE TCPP Initiative on Parallel and Distributed Computing identified [1] and recommended over a hundred PDC topics with course mappings to help CS undergraduates develop a capacity for parallel thinking [1]. The ACM/IEEE Joint Task Force on Computing Curricula supported TCPP's findings, recommending in their 2013 Computing Curricula (CS2013) that every CS program cover at least 15 hours of PDC. More recently, ABET's Computing Accreditation Commission required that all undergraduate computer science students learn parallel and distributed computing [2].

As faculty move toward integrating PDC topics into their courses, many are unsure where to begin. Teaching PDC can be challenging for those without prior knowledge. There are also questions on the best way to introduce concepts, especially to an increasingly diverse student population. "Unplugged" PDC activities – or dramatizations and analogies that teach PDC without computers – are one potential solution. The unplugged movement for teaching computing gained signficant traction with the release of CS Unplugged [3], a collection of free unplugged activities for teaching computing concepts to K-12 students. Unplugged activities are typically easy to implement, inexpensive, require few materials, encourage collaboration and often represent a "welcome break" for teachers used to teaching in front of computer screens [3], [4]. They are also a great way to give a high-level overview to a topic before getting into technical details. More importantly, unplugged activities can promote equity in computing by removing the expense and requirement of computers and by catering to individuals with unique needs. Specifically, unplugged activities enable individuals who are disabled or who speak a foreign language to engage with computing concepts with their other senses [5], [6]. Faculty who employ unplugged activities to teach computing concepts to college students generally agree that the activities aid in student understanding (e.g. [7]–[9]).

While several educators (e.g [10]–[14]) have developed unplugged activities for teaching PDC, their contributions are widely scattered and generally hard to find. Curating unplugged materials into a centralized repository makes it easier for educators to identify and adopt activities for their classrooms. CS Unplugged is not an optimal choice for this curation for several reasons. First, CS Unplugged focuses on introducing general computing concepts to K-12 students. Educators introducing PDC concepts to older populations may therefore find the activities in CS Unplugged too "childish" for their classrooms. Second, a repository designed specifically for unplugged PDC activities can cater directly to the needs of computing educators by mapping activities to well-established PDC topic areas and learning outcomes. Lastly, a repository of unplugged PDC activities allows activity creators to identify existing activities and potential gap areas, preventing them from inadvertently reinventing the wheel.

This paper presents *PDCunplugged*, a collection of unplugged PDC activities curated from the existing PDC literature and from across the Web. The curation enables educators to quickly find existing unplugged activities to try out in their classes, and allows them to augment activity entries with assessments and experiences of their own. In creating *PDCunplugged*, we set out to answer the following questions:

- What unplugged activities currently exist for PDC?
- How do existing activities cover TCPP Topic Areas and CS2013 Knowledge Units?
- Where should educators concentrate on developing new content?

National Science Foundation (NSF) Grant DUE-1855761

As of writing, the curation identifies nearly forty unique activities that span all the CS2013 knowledge units, the TCPP topic areas, and core computing courses. Each activity contains links to external materials (if available), summaries of known assessment, and citations to source papers. Detailed instructions are provided to allow future contributors to add to the repository by initiating pull requests through GitHub or (if they prefer) through e-mail. Lastly, we identify several "holes" in the curation and identify opportunities for future development of unplugged activities.

The rest of this paper is organized as follows. Section II discusses the *PDCunplugged* website, its features, use cases, and how users can contribute. Section III covers the curation process, summarizes the collection of identified PDC unplugged activities, and discusses lessons learned. We conclude and discuss future work in Section IV.

II. Overview of *PDCunplugged*

The *PDCunplugged* website (`pdcunplugged.org`) is built using the Hugo Static Site Generator [15] version 0.59.1, an open-source website creation framework written in the Go language. Hugo was chosen as the underlying web framework due to its sophisticated taxonomy system (see Section II-B), fast build times, and seamless integration with GitHub.

We anticipate three main classes of users of *PDCunplugged*: 1.) *Activity Authors* who create and curate unplugged activities into the *PDCunplugged* repository; 2.) *Educators* who implement *PDCunplugged* activities in their classrooms; and 3.) *Assessors* who evaluate the efficacy of particular activities in a classroom. We anticipate visitors to *PDCunplugged* to span several user categories, with some activity authors or educators augmenting existing activities with variations and assessments based on their own classroom experiences.

Contributors to *PDCunplugged* write activities in Markdown [16], a lightweight text markup language that easily renders to HTML and other formats. Unlike HTML, Markdown requires relatively little knowledge and enables contributors to write in near plain-text. While installing Hugo is an optional step, it is recommended for users who wish to locally view how an activity renders on the *PDCunplugged* website.

A. Activity structure

Activities form the heart of *PDCunplugged*. In this context, the term "activity" refers to a variety of interventions, including kinesthetic learning activities, role-playing, and even analogies. We choose to group all these interventions under the common term "activity", as several analogies can be dramatized, and vice versa. Each activity exists in *PDCunplugged* as a separate Markdown file containing all the information that defines it. Suppose an contributor wishes to create a new unplugged activity called `example`. The contributor first copies the template shown in Fig. 1 into a text file called `example.md` and either e-mails the curator with the contents of this file, or initiates a GitHub pull request into the `content/activities` folder. If the contributor has a local installation of Hugo, they can instantiate a pre-populated `example.md` file containing the contents of Fig. 1 by running the command `hugo new activities/example.md`.

```
---
title:
date:
tags:
---

## Original Author/link

---

## CS2013 Knowledge Unit Coverage

---

## TCPP Topics Coverage

---

## Recommended Courses

---

## Accessibility

---

## Assessment

---

## Citations
```

Fig. 1. Activity Markdown Template

The first three lines in the Markdown file form the header of the document and represents the activity title, date, and associated tags (see Section II-B). There are seven sections that form the body of the activity, with each section separated by a horizontal rule (`---`):

a) Original Author/Link: The name of the activity author along with any available on-line resources are listed first. If for whatever reason the author does not have a public-facing website containing the activity's details, the note "No external resources found. See details below" is included, and a "Details" section (`## Details`) appears next.

b) Details: This optional section describes the activity and all the relevant details needed for someone to adopt it in a classroom. If the Details section is populated using information from a proceeding or article, citations must be included to ensure that the original source is properly attributed. The Details section often takes the majority of the work in creating an activity.

c) CS2013 Knowledge Unit Coverage / TCPP Topics Coverage: The next two sections detail the CS2013 knowledge units and TCPP topics coverage respectively. The CS2013 section enumerates each relevant knowledge unit and lists

the relevant learning outcomes. The TCPP section lists the relevant topic areas and itemizes the associated topics covered by the activity. Critically, the information in these two sections are used to populate the tags and taxonomies described in Section II-B.

d) Recommended Courses: This section lists any recommended courses for the activity. We expect authors to populate this section based on their own experiences or recommendations for using the activity. For additional ideas, activity authors should look at the TCPP recommendations (available through the TCPP view, see below), which include a list of recommended courses for each topic area.

e) Accessibility: This section discusses how the activity can be presented to different audiences, and mentions if the material may be challenging to certain groups. For example, if an activity requires a lot of movement, a note is made that the activity may be inappropriate for students with mobility issues. Suggested variations to make an activity more accessible are usually included in this section. Lastly, this section acts as a gentle nudge to activity authors to think about inclusion when designing activities.

f) Assessment: This section lists what (if any) assessment exists for the activity in question. Educators who use particular activities in their classroom are encouraged to augment this section with their classroom experiences. Note that most activities in the literature do not include assessment. This section also gently nudges activity authors to think about evaluation when presenting activities.

g) Citations: The citations section lists full citations to all papers that reference the activity, and includes links to websites containing supporting materials when applicable.

B. Taxonomies and Tagging

We chose Hugo as the platform for *PDCunplugged* primarily due to its sophisticated support for taxonomies. Each taxonomy consists of a series of terms, a subset of which are listed on each entry. Hugo then automatically groups entries together by their listed terms, making it possible to view all the entries that share a common term. *PDCunplugged* leverages Hugo's taxonomy system to provide users with a number of custom taxonomies to view and filter unplugged activities:

a) CS2013: The CS2013 taxonomy (`cs2013`) classifies unplugged activities by their associated knowledge units and learning outcomes as specified by CS2013. The terms associated with this taxonomy are knowledge units. For example, an activity with learning outcomes that match the Parallel Decomposition and Parallel Algorithms knowledge units would list the terms `PD_ParallelDecomposition` and `PD_ParallelAlgorithms`.

b) TCPP: The TCPP taxonomy (`tcpp`) classifies unplugged activities by the topic areas outlined by the TCPP Curriculum Initiative. The terms associated with this taxonomy are the general topic areas outlined by the 2012 TCPP report. For example, an activity that covers topics associated with the TCPP Algorithms and Programming topic areas would list the terms `TCPP_Algorithms` and `TCPP_Programming`.

c) Courses: The courses taxonomy (`courses`) classifies unplugged activities by courses recommended for introducing the activity. College-level courses have separate terms (e.g `CS0`, `CS1`, `DSA`) while K-12 activities are labeled with the `K_12` term.

d) Senses: The senses taxonomy (`senses`) classifies activities by the sensory mediums primarily engaged by learners. The senses taxonomy attempts to improve accessibility by aiding educators in identifying activities that best match their particular classrooms. An activity that is primarily visual and tactile would contain the terms `visual` and `touch`. A general `accessible` term is included to denote activities judged to be accessible to a diverse range of populations with minimal modification.

As an example, consider the "FindSmallestCard" activity that was proposed by Bachelis *et al.* [10]. The activity contains learning outcomes belonging to the Parallel Algorithms and Parallel Decomposition CS2013 knowledge units, and topics belonging to the TCPP Programming and TCPP Algorithms topic areas. The activity is recommended for undergraduate students in CS1, CS2, and DSA, and contains tactile and visual elements. Therefore, the `tags` field in the header of the activity is replaced with the following lines (Fig. 2):

```
---
title: "FindSmallestCard"
cs2013: ["PD_ParallelDecomposition", \
         "PD_ParallelAlgorithms"]
tcpp: ["TCPP_Algorithms", "TCPP_Programming"]
courses: ["CS1", "CS2", "DSA"]
senses: ["touch", "visual"]
---
```

Fig. 2. Header for FindSmallestCard activity

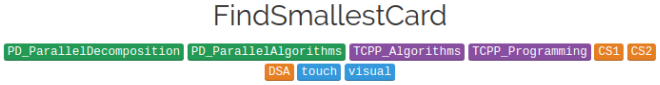

Fig. 3. Rendered Header for FindSmallestCard activity

The rendered version of this content is shown in Fig. 3 and is available at `https://www.pdcunplugged.org/activities/findsmallestcard/`. Each taxonomy is assigned a different color, and the terms associated with each taxonomy are listed under the activity title at the top of the page. Furthermore, each term links to a separate page that contains all the activities that share that term. The activity header enables educators to quickly get a sense of what the activity offers prior to reading the details, and helps them identify activities with similar features.

e) Hidden Taxonomies: Not all available taxonomies are visible in the activity header. *PDCunplugged* employs three

hidden taxonomies (`cs2013details`, `tcppdetails` and `medium`), that enable an activity author a finer granularity of classification. The `cs2013details` taxonomy enables an author to specify the learning outcomes that are associated with an activity. Likewise, the `tcppdetails` taxonomy allows the specification of the Bloom taxonomy topics associated with an activity. The `medium` taxonomy allows an activity author to indicate the communication medium (e.g. analogy, paper, role-play) used in the activity.

Typically, the terms associated with the `cs2013details` taxonomy consist of an abbreviation of the knowledge unit followed by the corresponding learning outcome's numeric listing. For example, an activity that covers learning outcomes 1 and 3 of the Parallel Decomposition knowledge unit would have the terms `PD_1` and `PD_3` listed under `cs2013details`. For the TCPP topics, each term lists the Bloom taxonomy classification ("K" for "Know", "C" for "Comprehend" and "A" for "Apply") followed by a word that succinctly describes the topic area. Thus, an activity that covers the TCPP programming topic "Comprehend Speedup" will have the term `C_Speedup` listed under `tcppdetails`.

C. Activity Views

PDCunplugged uses the aforementioned taxonomy system to create several "views" for browsing unplugged activities. In addition to viewing a listing of all activities, visitors to *PDCunplugged* can also browse activities by separate CS2013, TCPP, Courses, and Accessibility views. The different views allow visitors to quickly narrow in on unplugged activities that meet their needs, and (in the case of activity authors) identify gaps in coverage.

For instance, *PDCunplugged* uses the `cs2013details` and `tcppdetails` hidden taxonomies to enumerate the set of activities associated with particular CS2013 learning outcomes and TCPP topics in the CS2013 and TCPP views respectively. We anticipate that activity authors will use these views to gauge the level of potential impact for their proposed activity. For example, a new activity that covers learning outcomes or topic areas not covered by existing activities in the curation may be judged to have a larger impact than one whose learning outcomes are well covered by other activities. Likewise, educators looking for activities to match a particular learning outcome or topic area can quickly focus on the activities that best fit their needs.

The `medium` hidden taxonomy is used in tandem with the `senses` taxonomy to build the Accessibility view. This view enables visitors to *PDCunplugged* to search for activities based on sensory perception or medium of communication. For example, an educator wondering how to teach parallelism with a deck of cards could select the "cards" term to view the list of related activities. Likewise, someone looking to incorporate tactile activities into their course can select the "touch" term to view all activities that heavily involve touch. The Course view is self-explanatory; activities are organized by the courses recommended for their adoption. This last view is especially useful for educators teaching a particular course who want to see what unplugged activities are recommended for it.

III. Curation and Results

One of the goals of this paper is to curate existing unplugged activities into the *PDCunplugged* website. Note that while a serious attempt was made to aggregate as many unplugged activities as possible, we do not claim to have identified *all* existing unplugged PDC activities. Readers cognizant of any missing PDC activities are welcome to contact us directly with the details or add the activity to the GitHub repository themselves.

The ACM Digital Library, IEEE Xplore and Google Scholar databases were primarily used to build the curation. We used various search keywords, including "unplugged", "analogy", "game", "metaphor", "parallel" and "concurrency". Each relevant paper's list of references were searched to identify earlier activities and build accurate citations profiles. Google Scholar was also used to identify citing publications to further extend the search. In some cases, full-texts of earlier papers were not available. In other cases, several distinct papers described a single activity (or an existing similar activity), sometimes without referencing each other. In those cases, the descriptions were listed as "variations" of a single activity, and collapsed together under a single activity heading. We also note that several papers listed multiple activities.

A. Existing Unplugged Activities for PDC

As of writing, the curation has identified nearly forty unique activities gathered from the literature over the last thirty years. The earliest paper to advocate for the use of unplugged activities for teaching PDC concepts is a tutorial written by Bachelis, James, Maxim and Stout in 1990 [17]. While the 1990 tutorial write-up does not extrapolate the specific activities, a follow-up paper in 1994 [10] gives a detailed listing, and a separate paper [11] by Kitchen, Schaller and Tymann references the earlier tutorial and describes two of the activities (which are also described in [10]).

Sorting algorithms represent the most common set of unplugged PDC activities described in the literature. Bachelis *et al.* described a card sorting activity [10] that later researchers [14], [18] adapted. Adam Rifkin [12] discussed activities that dramatize odd-even transposition sort (parallel bubble sort) and parallel radix sort. Both activities were incorporated into a larger workshop by Sivilotti and Demirbas [19] and partially assessed. Sivilotti [20] helpfully provides a one-page instructor write-up on the activities.

Several papers also present anologies for teaching PDC concepts. For example, the Oklahoma Supercomputing Center for Education and Research (OSCER) released a workshop series entitled "Supercomputing in Plain English" [21] that utilizes several analogies to introduce PDC concepts to non-computing students and practitioners [13], including those for load balancing, resource contention, shared memory, distributed memory, communication overhead and race conditions. Giacaman [22] developed several analogies to introduce parallel computing

concepts to sophomore undergraduates. Bogaerts [23], [24] also developed a series of analogies to introduce parallelism in a CS1 course.

Many early papers discussing PDC unplugged activities did not include assessment or only provided qualitative feedback from students. However, recent research efforts [9], [14], [25], [26] attempt to not only develop unplugged activities but also assess their efficacy. For example, Ghafoor, Brown, Rogers and Hines described two unplugged activities [14] (with an additional three listed at the iPDC modules website [27]) that were evaluated in a CS1 and CS2 course. Their preliminary assesment suggested that the activities aided students in learning PDC concepts. A separate paper by Chitra and Ghafoor [9] incorporated an unplugged activity in a graduate PDC course, as part of a larger effort to incorporate active learning in the course. Their assessment revealed that students who were taught with the active learning methodology earned higher grades than students taught the material in a traditional lecture-style format [9].

The curation also reveals differences in opinion about pedagogy and how students best learn. Early papers by Ben-Ari and Kolikant [28], [29] make the argument for constructivism, which states that students learn by refining and extending the knowledge that they already know. In their first paper [28], Ben-Ari and Kolikant use a scenario of robots concurrently trying to sweeten a glass of juice to illustrate race conditions and the need for mutual exclusion. In a later paper, Kolikant [29] presents two additional activities to illustrate distributed systems, one involving concert tickets, and a second involving gardening. The concert tickets activity was further refined [30], [31] by Lewandowski *et al.* in the development of their "Commonsense Computing" program.

Most of the unplugged activities in the literature follow an operational view of computing, where people act as processes or processors (in the course of dramatazing algorithms) or as memory (when illustrating the workings of data structures). Sivilotti [32], [33] presents an alternative set of activities that follow an assertional view of computing, where students focus on what is true for *all* execution sequences through the identification of invariants. Sivilotti argues [32] that approaching algorithms with assertional reasoning leads to less ambiguity about how a concurrent algorithm works, and increases a student's ability to prove an algorithm's correctness. Sivilotti and Pike [32] developed three assertional PDC unplugged activities, including a non-determinstic sorting activity, parallel garbage collection, and stable leader election, all which were used to introduce upper-level students to concurrent algorithms. Sivilotti and Demirbas [19] also developed an unplugged activity that introduced middle school girls to self-stabilizing token rings for mutual exclusion. It remains to be seen how effective assertional unplugged activities are for introducing novices to PDC topics; this is perhaps an area of interesting future research.

Less than half (41%) of the materials have some sort of external resource (slides, handouts, etc.) associated with them; the quality and utility of the external resources also varies greatly. Older activities in the literature were less likely to have associated external resources. In terms of course coverage, there are 15 activities listed on *PDCunplugged* recommended for K-12, 8 for CS0, 17 for CS1, 25 for CS2, 27 for DSA, and 22 for Systems courses.

B. Coverage of CS2013 Topic Areas

Table I shows how the current set of unplugged activities cover the various knowledge units and learning outcomes of the PDC knowledge area described in CS2013 [34]. For each knowledge unit, CS2013 recommends coverage of all Tier 1 learning outcomes, at least 80% of Tier 2 learning outcomes, and a "significant" amount of elective material [34]. Thus, Table I contains all the knowledge units, with purely elective knowledge units marked with an E. For each knowledge unit, we list the set of associated learning outcomes, the number of learning outcomes with corresponding activities, and the total number of activities associated with the knowledge unit.

TABLE I
CS2013 COVERAGE

Knowledge Unit	**Num. Learning Outcomes**	**Num. Covered Outcomes**	**Percent Coverage**	**Total Activities**
Parallel Fundamentals	3	2	66.67%	2
Parallel Decomposition	6	5	83.33%	21
Parallel Communication and Coordination	12	6	50.00%	9
Parallel Algorithms, Analysis, and Programming	11	6	54.54%	12
Parallel Architecture	8	7	87.50%	9
Parallel Performance (E)	7	6	85.71%	10
Distributed Systems (E)	9	1	11.11%	2
Cloud Computing (E)	5	1	20.00%	3
Formal Models and Semantics (E)	6	1	16.66%	1

The Parallel Decomposition knowledge unit has the largest number of unplugged activities (21), followed by the Parallel Algorithms (12) and the Parallel Performance (10) knowledge units. Curiously, the Parallel Fundamentals knowledge unit has among the least activities (2), despite having the smallest number of learning outcomes. The reason becomes clear upon closer inspection. The learning outcomes under the Parallel Fundamentals knowledge unit ask students to distinguish between two competing concepts. For example, while there are several unplugged activities that discuss what data races are, none distinguish them from "higher level races". Likewise, while there are several unplugged activities that discuss synchronization, only one [35] compares multiple

methods for synchronization. Likewise, only one unplugged activity [25], [26] distinguishes between "using computational resources for a faster answer from managing efficient access to a shared resource".

In general, learning outcomes that ask students to distinguish between competing concepts, define metrics, or implement code had the lowest number of corresponding unplugged activities. However, certain knowledge units are noticebly sparse in the number of available activities. For example, in the Cloud Computing knowledge unit, only three unplugged activities exist (by Lloyd [36] and Kolikant [29] respectively), and both cover the same learning outcome. A similar story is told in the Distributed Computing knowledge unit, with two activities covering the same outcome. The Formal Models and Semantics knowledge unit has the least coverage, with only one activity covering a single outcome.

C. Coverage of TCPP Topic Areas

Table II shows the coverage of the collected unplugged activities over the PDC topic areas and topics listed in the 2012 TCPP PDC Curricula report [1]. For each topic area, we enumerate the total number of corresponding topics, the number of topics covered by unplugged materials, the percent coverage, and the total unplugged activities corresponding to that topic area. The TCPP report emphasizes the need for parallelism covered in "core courses": CS1, CS2, DSA or Systems. Therefore, we focus specifically on the topics TCPP suggests for core courses and exclude any topics that were solely associated with advanced courses.

TABLE II
TCPP COVERAGE

Topic Area	Num. Topics	Num. Covered Topics	Percent Coverage	Total Activities
Architecture	22	10	45.45%	9
Programming	37	19	51.35%	24
Algorithms	26	13	50.00%	22
Crosscutting and Advanced Topics	12	7	58.33%	8

The topic area with the lowest level of coverage is Architecture at 45.45%. The Architecture topic area is subdivided into the Classes, Memory Hierarchy, Floating-point representation and Performance Metrics categories; of these, the Floating-point Representation and Performance Metric categories have no corresponding unplugged activities. The Algorithms topic area has the next lowest level of coverage at 50%. The Algorithms topic area can be further sub-divided into PD Models/Complexity, Algorithmic Paradigms, and Algorithmic Problems. Of these, the PD Models/Complexity topics have the lowest coverage at 36.36%. We suspect that this is due to the large number of topic areas that are very specific to a particular model (e.g. PRAM) or involve theoretical definitions (e.g. make/span, work, asymptotics). The Algorithmic Paradigms category, despite its low coverage, has much promise for future contribution. Specifically, there are activities missing for the parallel aspects of recursion, reduction and barrier synchronizations. The Algorithmic Problems category is fairly well covered, though there are opportunities to add activities that discuss communication constructs (e.g. scatter/gather, broadcast and multicast).

The Programming topics area has the most number of topics, and a coverage of 51.35%. The set of Programming topics is further subdivided into three categories: Paradigms and Notations, Correctness, and Performance. The Paradigms and Notations category has the lowest level of coverage (35.71%), largely due to the high specificity of some of the topics for particular languages or libraries (process vector extensions, OpenMP, TBB, etc.).

The curation of the literature also identified 8 activities that correspond to Cross Cutting and Advanced topics. Together, these activities cover 7 topics of the topics in the area (58.33% coverage). Surprisingly, we were unable to identify any unplugged activities that explain how web-searches or peer-to-peer computing work, or that discuss cloud/grid computing or the concept of locality. Readers may also be surprised to see the absence of any unplugged activity that corresponds to the "know why and what is parallel/distributed computing" PDC topic. However, this topic is overly broad, incorporating history, different levels of parallelism, and common issues.

D. Accessibility

The curation includes 11 analogies and 11 role-playing activities, and 4 activities that are labeled as "games". Popular activity mediums include paper (8), chalk-/white-board (6), and cards (6). Other activities involve various categories of objects, include pens (4), coins (2), food (4) and musical instruments (1). Most activity authors did not appear to be explicitly cognizant on how their activities or analogies would appeal to a diverse array of students. The vast majority (71.05%) of the identified unplugged materials have a strong visual component. Approximately 38.84% involve movement. Surprisingly, the sense of touch is only dominant in 26.32% of the activities. Only two identified unplugged activities incorporate sound. We note that 9 of the curated activities appear generally accessible; that is, with minor modification they can be presented to a wide variety of audiences.

For some students, analogies are a more preferable way of communication than role-playing or other activities that require motion or a strong visual component. However, it is possible for analogies to become out of date. For example, the "Long Distance Phone Call" analogy presented by Neeman *et al.* [13] is likely incomprehensible to younger audiences with unlimited cell phone plans, where the concepts of "connection charges" and "per-minute charges" may be foreign. Likewise, analogies that have culturally specific references may be inaccessible to students from other cultural groups.

On the other hand, visual, tactile and kinesthetic activities are preferable for specific populations. A highly-cited study [6] on non-native English speakers in university-based ESL programs found that ESL students preferred tactile and kinesthetic

communication methods. We therefore recommend that educators with a higher percentage of non-native speakers in their classrooms consider incorporating activities that engage their students' others senses when introducing new concepts.

E. Lessons Learned

The curation reveals a rich assortment of unplugged activities for assisting students learning PDC concepts and topics. Analogies are a quick and cost-free way to relate parallel computing concepts in class. Kinesthetic, visual, and tactile activities help students with unique needs interact and synthesize various PDC concepts. Several researchers have successfully used role-playing and other kinesthetic activities to introduce PDC concepts in programming-intensive courses [9], [18], [30], [33]. Unplugged activities are also a useful way to introduce parallelism in outreach or workshop settings.

While the current curation of unplugged activities spans all the TCPP topic areas and CS2013 PD knowledge units, the distribution is not uniform, and there are several gaps in the coverage. Perhaps most glaring is the lack of unplugged materials that cover concepts related to distributed systems, cloud computing, and power consumption. There is also a distinct lack of activities that engage learners in a tactile or auditory fashion. Developing activities in these areas may help engage a more diverse array of students. Lastly, assessing unplugged activities appears to be a relatively recent trend. There is value in assessing even well-established unplugged activities, as assessments can guide educators who are interested in incorporating specific materials into their own classrooms. We challenge the PDC community to focus on these areas as they continue to develop new unplugged materials, and welcome their contributions to *PDCunplugged*.

IV. Conclusions

This paper introduces *PDCunplugged*, a free on-line repository for unplugged activities that focuses on parallel and distributing computing topics. The repository, built using the Hugo static site generator and hosted on GitHub, enables activity authors to easily share and upload descriptions of activities and classroom experiences, along with links to external resources. Entries are written using Markdown, making it easy for people unfamiliar with HTML to easily create content. A key feature of *PDCunplugged* is its use of taxonomies to classify unplugged activities and analogies over a variety of areas, including how they cover TCPP topics and CS2013 learning outcomes related to PDC. *PDCunplugged* also enables activity authors to make note of recommended courses, the senses engaged, and the medium of communication employed.

The current curation consists of nearly forty unique activities gathered from thirty years of PDC literature. Each activity lists original authors, links to available materials, known variations, available assessment, notes on accessibility and recommended course, citations to external work, and tags that map the activity to different TCPP topic areas, CS2013 knowledge units, particular courses, senses, and mediums. The curation allows us to quantify the distribution of activities over different PDC learning outcomes and topics, and identify gaps in coverage. We believe that our analysis helps activity authors focus on more impactful areas when creating new unplugged activities. We also believe that *PDCunplugged* will be an invaluable resource for educators looking for unplugged interventions for their classrooms, or for designing PDC-related workshops.

There are many avenues for future work. First, we plan on continuing to round out the curation by looking at the literature outside of PDC for related unplugged activities. Other disciplines in computing, such as Architecture and Networking, may have relevant unplugged activities that apply to TCPP topic areas and CS2013 knowledge units. Secondly, we plan to reach out to identified unplugged activity authors to see if they would be willing to store related external materials directly on the *PDCunplugged* website; currently, *PDCunplugged* only includes *links* to external resources, and not the external resources themselves. The inherent risk is that external links can expire; several authors [12], [35], [37] cite external activities in their papers, but those links have since been de-activated. Listing activity materials directly on *PDCunplugged* ensures that a copy of the materials exist at an independent location.

We anticipate that *PDCunplugged* will be an invaluable resource for the PDC educational community; we encourage everyone employing unplugged activities in their classrooms to use and contribute to the repository. One way to ensure that the repository remains updated is for journals and proceedings in the PDC community to recommend or require that authors submit educational materials to select repositories as part of the regular publication process. Lastly, we believe the existence and proliferation of specialized repositories like *PDCunplugged* will ultimately make it easier for educators to integrate PDC topics in their curricula.

Acknowledgment

Funding for this work is provided by the National Science Foundation (NSF) Collaborative Research Grant DUE-1855761. Special thanks to Joel Adams, Elizabeth Shoop and Richard Brown of *CSinParallel* for offering advice as this work was put together, and to Kevin Barlett for Hugo debugging assistance. The views expressed in this article are those of the author and do not reflect the official policy or position of the Department of the Army, Department of Defense or the U.S. Government.

References

[1] S. K. Prasad, A. Y. Chtchelkanova, S. K. Das, F. Dehne, M. G. Gouda, A. Gupta, J. Jaja, K. Kant, A. La Salle, R. LeBlanc *et al.*, "Nsf/ieee-tcpp curriculum initiative on parallel and distributed computing: core topics for undergraduates," in *SIGCSE*, vol. 11, 2011, pp. 617–618.

[2] A. C. A. Commission, *Criteria For Accrediting Computing Programs (2018-2019) version 2.0*. Baltimore, MD, USA: ABET, 2017.

[3] T. Bell, J. Alexander, I. Freeman, and M. Grimley, "Computer science unplugged: School students doing real computing without computers," *The New Zealand Journal of Applied Computing and Information Technology*, vol. 13, no. 1, pp. 20–29, 2009.

[4] T. J. Cortina, "Reaching a broader population of students through "unplugged" activities," *Commun. ACM*, vol. 58, no. 3, pp. 25–27, Feb. 2015. [Online]. Available: http://doi.acm.org/10.1145/2723671

[5] H. Manabe, S. Kanemune, M. Namiki, and Y. Nakano, "Cs unplugged assisted by digital materials for handicapped people at schools," in *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*. Springer, 2011, pp. 82–93.

[6] J. M. Reid, "The learning style preferences of esl students," *TESOL quarterly*, vol. 21, no. 1, pp. 87–111, 1987.

[7] A. Fleury, "Acting out algorithms: how and why it works," *The Journal of Computing in Small Colleges*, vol. 13, no. 2, pp. 83–90, 1997.

[8] J. Eum and S. Sethumadhavan, "Teaching microarchitecture through metaphors," Columbia University, Tech. Rep. CUCS-006-14, June 2014.

[9] P. Chitra and S. K. Ghafoor, "Activity based approach for teaching parallel computing: An indian experience," in *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2019, pp. 290–295.

[10] G. F. Bachelis, B. R. Maxim, D. A. James, and Q. F. Stout, "Bringing algorithms to life: Cooperative computing activities using students as processors," *School Science and Mathematics*, vol. 94, no. 4, pp. 176–186, 1994.

[11] A. T. Kitchen, N. C. Schaller, and P. T. Tymann, "Game playing as a technique for teaching parallel computing concepts," *SIGCSE Bull.*, vol. 24, no. 3, pp. 35–38, Sep. 1992. [Online]. Available: http://doi.acm.org/10.1145/142040.142064

[12] A. Rifkin, "Teaching parallel programming and software engineering concepts to high school students," *SIGCSE Bull.*, vol. 26, no. 1, pp. 26–30, Mar. 1994. [Online]. Available: http://doi.acm.org/10.1145/191033.191044

[13] H. Neeman, L. Lee, J. Mullen, and G. Newman, "Analogies for teaching parallel computing to inexperienced programmers," in *Working Group Reports on ITiCSE on Innovation and Technology in Computer Science Education*, ser. ITiCSE-WGR '06. New York, NY, USA: ACM, 2006, pp. 64–67. [Online]. Available: http://doi.acm.org/10.1145/1189215.1189172

[14] S. K. Ghafoor, D. W. Brown, M. Rogers, and T. Hines, "Unplugged activities to introduce parallel computing in introductory programming classes: An experience report," in *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '19. New York, NY, USA: ACM, 2019, pp. 309–309. [Online]. Available: http://doi.acm.org/10.1145/3304221.3325573

[15] S. Francia, B. E. Pedersen *et al.*, "Hugo: The worlds fastest framework for building websites," internet Website, last accessed December 2, 2019. [Online]. Available: https://gohugo.io/

[16] J. Gruber, "Markdown," internet Website, last accessed December 2, 2019. [Online]. Available: https://daringfireball.net/projects/markdown/

[17] B. R. Maxim, G. Bachelis, D. James, and Q. Stout, "Introducing parallel algorithms in undergraduate computer science courses (tutorial session)," in *Proceedings of the Twenty-first SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '90. New York, NY, USA: ACM, 1990, pp. 255–. [Online]. Available: http://doi.acm.org/10.1145/323410.323415

[18] M. Moore, "Introducing parallel processing concepts," *J. Comput. Sci. Coll.*, vol. 15, no. 3, pp. 173–180, Mar. 2000. [Online]. Available: http://dl.acm.org/citation.cfm?id=1852563.1852589

[19] P. A. G. Sivilotti and M. Demirbas, "Introducing middle school girls to fault tolerant computing," in *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '03. New York, NY, USA: ACM, 2003, pp. 327–331, http://web.cse.ohio-state.edu/ sivilotti.1/outreach/FESC02/. [Online]. Available: http://doi.acm.org/10.1145/611892.611999

[20] P. A. Sivilotti, "Parallel programming: Parallel programs are fast," last accessed Oct 16, 2019. [Online]. Available: http://web.cse.ohio-state.edu/ sivilotti.1/outreach/FESC02/parallel.pdf

[21] H. Neeman, H. Severini, and D. Wu, "Supercomputing in plain english: Teaching cyberinfrastructure to computing novices," *SIGCSE Bull.*, vol. 40, no. 2, pp. 27–30, Jun. 2008. [Online]. Available: http://doi.acm.org/10.1145/1383602.1383628

[22] N. Giacaman, "Teaching by example: Using analogies and live coding demonstrations to teach parallel computing concepts to undergraduate students," in *Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, ser. IPDPSW '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 1295–1298. [Online]. Available: http://dx.doi.org/10.1109/IPDPSW.2012.158

[23] S. A. Bogaerts, "Limited time and experience: Parallelism in cs1," in *2014 IEEE International Parallel Distributed Processing Symposium Workshops*, May 2014, pp. 1071–1078.

[24] S. A. Bogaerts, "One step at a time: Parallelism in an introductory programming course," *Journal of Parallel and Distributed Computing*, vol. 105, pp. 4 – 17, 2017, keeping up with Technology: Teaching Parallel, Distributed and High-Performance Computing. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731517300023

[25] M. Smith and S. Srivastava, "Evaluating student engagement towards integrating parallel and distributed computing (pdc) topics in undergraduate level computer science curriculum," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '19. New York, NY, USA: ACM, 2019, pp. 1269–1269. [Online]. Available: http://doi.acm.org/10.1145/3287324.3293854

[26] S. Srivastava, M. Smith, A. Ghimire, and S. Gao, "Assessing the integration of parallel and distributed computing in early undergraduate computer science curriculum using unplugged activities," in *Proceedings of the Workshop on Education for High Performance Computing*, ser. EduHPC '19, 2019, to appear. [Online]. Available: https://tcpp.cs.gsu.edu/curriculum/sites/default/files/ws_eduhpcp110s2-file1.pdf

[27] S. K. Ghafoor, M. Rogers, D. Brown, and A. Haynes, "ipdc modules (unplugged)," last accessed Oct 16, 2019. [Online]. Available: csc.tntech.edu/pdcincs/index.php/ipdc-modules/

[28] M. Ben-Ari and Y. B.-D. Kolikant, "Thinking parallel: The process of learning concurrency," in *Proceedings of the 4th Annual SIGCSE/SIGCUE ITiCSE Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '99. New York, NY, USA: ACM, 1999, pp. 13–16. [Online]. Available: http://doi.acm.org/10.1145/305786.305831

[29] Y. B.-D. Kolikant, "Gardeners and cinema tickets: High school students' preconceptions of concurrency," *Computer Science Education*, vol. 11, no. 3, pp. 221–245, 2001. [Online]. Available: https://doi.org/10.1076/csed.11.3.221.3831

[30] G. Lewandowski, D. J. Bouvier, R. McCartney, K. Sanders, and B. Simon, "Commonsense computing (episode 3): Concurrency and concert tickets," in *Proceedings of the Third International Workshop on Computing Education Research*, ser. ICER '07. New York, NY, USA: ACM, 2007, pp. 133–144. [Online]. Available: http://doi.acm.org/10.1145/1288580.1288598

[31] G. Lewandowski, D. J. Bouvier, T.-Y. Chen, R. McCartney, K. Sanders, B. Simon, and T. VanDeGrift, "Commonsense understanding of concurrency: Computing students and concert tickets," *Commun. ACM*, vol. 53, no. 7, pp. 60–70, Jul. 2010. [Online]. Available: http://doi.acm.org/10.1145/1785414.1785438

[32] P. A. G. Sivilotti and S. M. Pike, "The suitability of kinesthetic learning activities for teaching distributed algorithms," in *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '07. New York, NY, USA: ACM, 2007, pp. 362–366. [Online]. Available: http://doi.acm.org/10.1145/1227310.1227438

[33] P. A. G. Sivilotti, "Kinesthetic learning activities in an upper-division computer science course," in *National Academy of Engineering Frontiers of Engineering Education symposium*, 2010, poster; http://web.cse.ohio-state.edu/ sivilotti.1/research/publications/nae-fee2010_poster.pdf.

[34] A. f. C. M. A. Joint Task Force on Computing Curricula and I. C. Society, *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. New York, NY, USA: Association for Computing Machinery, 2013.

[35] R. A. Chesebrough and I. Turner, "Parallel computing: At the interface of high school and industry," in *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '10. New York, NY, USA: ACM, 2010, pp. 280–284. [Online]. Available: http://doi.acm.org/10.1145/1734263.1734361

[36] W. S. Lloyd, "Exploring the byzantine generals problem with beginning computer science students," *SIGCSE Bull.*, vol. 26, no. 4, pp. 21–24, Dec. 1994. [Online]. Available: http://doi.acm.org/10.1145/190650.190656

[37] S. K. Andrianoff and D. B. Levine, "Role playing in an object-oriented world," in *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE 02. New York, NY, USA: Association for Computing Machinery, 2002, p. 121125. [Online]. Available: https://doi.org/10.1145/563340.563386